Click & Eat Application

Technical Document

30th April 2021

Bachelor Of Science (Honours) Software Development

(Student) - Ana Griga
(Student Number) - C00231441
(Tutor) - Dr. Chris Meudec

Institiúid Teicneolaíochta Cheatharlach
INSTITUTE of TECHNOLOGY CARLOW
At the Heart of South Leinster

Declaration

I declare that this technical document titled “Click & Eat” has been written by me under the supervision of Dr Christophe Meudec.

This work was not presented in any previous research for the award of a bachelor degree to the best of my knowledge.

The work is entirely mine and I accept full responsibility for any errors that might be found in the work, while the reference to publish materials have been duly acknowledged.

I have provided a complete table of reference of all works and sources used in the preparation of this document.

I understand that failure to conform with the Institute’s regulations governing plagiarism represents a serious offence.

Signature: *Ana Griga* Date: 25.04.2021

Ana Griga (Student)
C00231441 (Student Number)

Table of Contents

Table of Figures

1. Introduction

Click & Eat is an idea born in this times of pandemic, which has impacted all aspects of people's life, therefore since the interactions between people were restricted, customers are migrating more into the virtual world including smart working, smart shopping, smart entertainments and even digital education.

Click & Eat came as help for restaurants and their customer, the restaurateurs will keep overheads and labour costs low and have diners, while the customers can still enjoy a dinner out with small interaction by using this application to order and pay for their meal with minimum interaction simply and safe from their devices.

This technical document aims to outline the Click & Eat application's implementation details, related code and deployment process.

2. Technologies Used

The Click & Eat application was developed in Visual Studio using Asp.Net Core and C# for the API server and ReactJS for the user interface. In the Research Report, all of these technologies were researched and documented, and they will be briefly listed in the sections below.

2.1 Visual Studio

Visual Studio was chosen as it was previously used and was preferable to other IDE as it has great helpful features and navigating the interface is smooth which enables writing the code speedily and accurately. Visual Studio allows simple access to a variety of debugging tool which assists in quickly diagnose bugs or problems. IntelliSense is a great feature of Visual Studio which allows getting specific suggestions. Another useful characteristic offered by Visual Studio is the ability to see the number of references to a method and clicking on it gives access to the reference. Publishing the application was also very intuitive with Visual Studio.

2.2 Asp.Net Core and C#

Microsoft mature technologies were chosen for the server-side development of the application because they integrate well together, they are reliable, widely used and tested in the industry. Asp.Net Core was used in creating the web APIs using C#.

2.3 ReactJS

After extensive research, it was found that ReactJS is an excellent library that allows creating interactive single-page applications which will be fast, simple and scalable.

The productivity of the development increased when using ReactJs due to its reusable components and Click & Eat contains a considerably good number of components.

ReactJS was chosen also because of its popularity being trusted by successful companies like Facebook, PayPal, DropBox, Tesla Motors, Netflix and also because of its great documentation and community.

3. Code Structure

The architecture of the code follows an MVC design with the database tables contained in the Model Folder, the Controllers in the Controllers folder and the Views, using ReactJs will be in the components subfolder of the ClientApp folder.

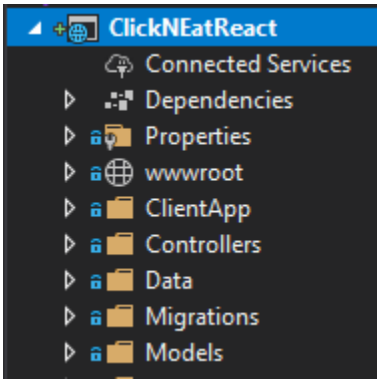

Figure 1- Application Code Structure
Source: Ana Griga, 2021

4. Project Code

The coding started by creating a new project as an ASP.NET Core Web Application. Then by selecting the ASP.NET Core with React.js option the ASP.NET Core React Single Page Application was created.

Visual Studio created the important files for the application to first run without adding any code.

4.1 Program.cs

The Program.cs file in the root of the project contains the Main method which represents the entry point for the application.

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ClickNEatReact
{
    /// <summary>
    /// Program.cs is the entry point for an application.
    /// See <a
href="https://www.tutorialspoint.com/what-is-the-purpose-of-program-cs-file-i
n-chash-asp-net-core-project"></a>
    /// </summary>

    public class Program
    {
        /// <summary>
        /// An Asp.Net Core web application is basically a console project
that runs from the public static void Main()
        /// method in this Program.cs class, where a web application host is
built.
        /// See <a
href="https://www.tutorialspoint.com/what-is-the-purpose-of-program-cs-file-i
n-chash-asp-net-core-project"></a>
```

```
        /// </summary>
        /// <param name="args">args parameter stores all command line
arguments which are given by the user when a program is ran.</param>

        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        /// <summary>
        /// Creates a non-HTTP workload with an IHostedService implementation
added to the DI container.
        /// See <a
href="https://www.tutorialspoint.com/what-is-the-purpose-of-program-cs-file-i
n-chash-asp-net-core-project"></a>
        /// </summary>
        /// <param name="args">args parameter stores all command line
arguments which are given by the user when a program is ran</param>

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

4.2 Startup.cs

The Startup class, which can be found in Startup.cs, is responsible for configuring the app's services as well as the request/response pipeline.

```
using ClickNEatReact.Data;
using ClickNEatReact.Models;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.SpaServices.ReactDevelopmentServer;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
```

```
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi.Models;
using System;
using System.IO;
using System.Reflection;
using System.Text;
using Newtonsoft.Json;

namespace ClickNEatReact
{
    public class Startup
    {
        /// <summary>
        /// Startup.cs configures the application's services which provides
the application functionality.
        /// Referenced from <a
href="http://docs.microsoft.com/en-us/aspnet/core/fundamentals/startup?view=a
spnetcore-5.0"></a>
        /// </summary>
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        ///<summary>
        ///This method gets called by the runtime and is used to configure
the HTTP request pipeline.
        ///</summary>

        ///<summary>
        ///UseSwaggerUI - Enable middleware to serve swagger-ui (HTML, JS,
CSS, etc.), specifying the Swagger JSON endpoint.
        /// See <a
href="https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-
with-swashbuckle?view=aspnetcore-5.0&tabs=visual-studio"></a>
        ///</summary>
        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllersWithViews();

            services.AddDbContext<ClickEatContext>(options =>

options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"))
);
```

```
            services.AddIdentity<ApplicationUser, IdentityRole>()
                .AddEntityFrameworkStores<ClickEatContext>()
                .AddDefaultTokenProviders();

            services.AddAuthentication(option =>
            {
                option.DefaultAuthenticateScheme =
JwtBearerDefaults.AuthenticationScheme;
                option.DefaultChallengeScheme =
JwtBearerDefaults.AuthenticationScheme;
                option.DefaultScheme =
JwtBearerDefaults.AuthenticationScheme;
            }).AddJwtBearer(option =>
            {
                option.SaveToken = true;
                option.RequireHttpsMetadata = false;
                option.TokenValidationParameters = new
TokenValidationParameters()
                {
                    ValidateIssuer = true,
                    ValidateAudience = true,
                    ValidAudience = Configuration["JWT:ValidAudience"],
                    ValidIssuer = Configuration["JWT:ValidIssuer"],
                    IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(Configuration["JWT:Secrete"])),
                };

            });

            services.AddSwaggerGen(options =>
            {
                options.SwaggerDoc("v1", new
Microsoft.OpenApi.Models.OpenApiInfo
                {
                    Title = "Click & Eat API",
                    Version = "v1",
                    Description = "Api for test purpose",
                });

                options.AddSecurityDefinition("Bearer", new
Microsoft.OpenApi.Models.OpenApiSecurityScheme
                {
                    Description = "JWT Authorization header using the Bearer
scheme (Example: 'Bearer 12345abcdef')",
                    Name = "Authorization",
                    In = Microsoft.OpenApi.Models.ParameterLocation.Header,
                    Type = Microsoft.OpenApi.Models.SecuritySchemeType.Http,
```

```
                    Scheme = "Bearer"
                });

                options.AddSecurityRequirement(new
Microsoft.OpenApi.Models.OpenApiSecurityRequirement
                {
                    {
                        new OpenApiSecurityScheme
                        {
                            Reference = new OpenApiReference
                            {
                                Type = ReferenceType.SecurityScheme,
                                Id = "Bearer"
                            }
                        },
                        Array.Empty<string>()
                    }
                });

            });
            // In production, the React files will be served from this
directory
            services.AddSpaStaticFiles(configuration =>
            {
                configuration.RootPath = "ClientApp/build";
            });
        }

        // This method gets called by the runtime. Use this method to
configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment
env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Error");
                // The default HSTS value is 30 days. You may want to change
this for production scenarios, see https://aka.ms/aspnetcore-hsts.
                app.UseHsts();
            }

            app.UseHttpsRedirection();
            app.UseStaticFiles();
            app.UseSpaStaticFiles();
```

```
            app.UseRouting();
            app.UseAuthentication();
            app.UseAuthorization();


            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller}/{action=Index}/{id?}");
            });
            app.UseSwagger();

            ///<summary>
            ///AddSwaggerGen - Swagger generator added to the services
collection. Used for API documentation.
            ///Inspired from <a
href="https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-
with-swashbuckle?view=aspnetcore-5.0&tabs=visual-studio"></a>
            ///</summary>
            app.UseSwaggerUI(c =>
            {
                c.SwaggerEndpoint("/swagger/v1/swagger.json", "Click & Eat
API V1");
                c.RoutePrefix = "docs";
            });

            app.UseSpa(spa =>
            {
                spa.Options.SourcePath = "ClientApp";

                if (env.IsDevelopment())
                {
                    spa.UseReactDevelopmentServer(npmScript: "start");
                }
            });
        }
    }
}
```

4.3 appsettings.json

The appsettings.json file contains all the application's settings.

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=clickeat.database.windows.net;Initial
Catalog=ClickEatDB;User ID=AnaGriga;
  },
  "JWT": {
    "ValidIssuer": "https://localhost:44395",
    "ValidAudience": "User",
    "Secrete":
"IQ0471PXqFE3VJZHwS96OmGBxc5aNgTRCstLMdplnvfyheDobjuWKiY2UkzAr8"
  },
  "Paypal": {
    "clientId":
"AQUkPTWDhuc2xS2lLW4Mn7p0MZutB1UHU6plWduwOmUCsuxHigkHW1E-YDGaNqVRyLWbHxwmJ8B
konHM",
    "secret":
"EOaljQadvp5dzAC8rw70ws8LAVmcMOqEcJO1ySU7hAS0sb8dX66z9shLtwMv6gIBOJ-P42MRVUe
9EOdZ"
  },
  "Braintree": {
    "MerchantId": "8wyh46fzsqkcgydk",
    "PublicKey": "hyp4hnjh74gd3qr2",
    "PrivateKey": "3314e44a63fc1057271315f63f2c21a6"
  }
}
```

4.4 Dependencies/Packages

All the necessary packages were installed using the NuGet Package Manager

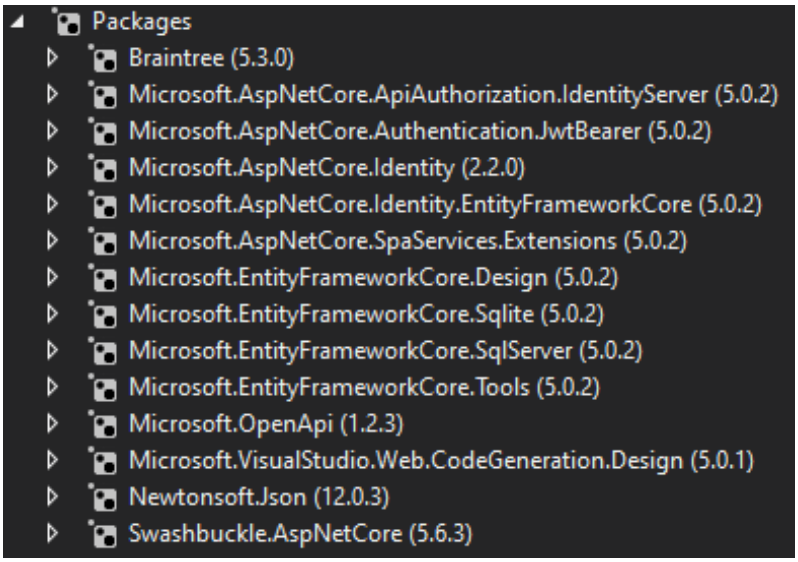

Figure 2 - Packages
Source: Ana Griga, 2021

4.5 Models

A model is a class that comprises the application's business logic. Models are often referred to as objects that are used to execute the application's conceptual logic. A controller interacts with the model by gaining access to data, performing logic, and passing the results to the view. The model is in charge of dealing with database changes. Each model reflects a table in the database and contains properties that match the columns in the table.

4.5.1 MenuCategory.cs Model

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text.Json.Serialization;
using System.Threading.Tasks;
using System.Xml.Serialization;

namespace ClickNEatReact.Models
{
    /// <summary>
    /// This model represents the attributes used for storing and
    /// retrieving MenuCategory data from and in the database.
    /// </summary>
    public class MenuCategory
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int CategoryId { get; set; }
        public string Name { get; set; }
        [XmlIgnore,JsonIgnore]
        public virtual ICollection<MenuItem> menuItems { set; get; }
    }
}
```

4.5.2 MenuItem.cs Model

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;
using System.Xml.Serialization;

namespace ClickNEatReact.Models
{
    /// <summary>
    /// This class models the MenuItem object in the database with all the
necessary attributes.
    /// </summary>
    public class MenuItem
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int MenuItemId { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public double Price { get; set; }
        public double AvgRate { set; get; }
        public int ReviewCount { set; get; }
        public string ImgPath { set; get; }
        public bool Availability { set; get; }
        public string Allergens { set; get; }
        public bool isVegan { set; get; }
        public int CategoryId { get; set; }
        [ForeignKey("CategoryId")]
        [XmlIgnore, JsonIgnore]
        public virtual MenuCategory MenuCategory {set;get;}
        [XmlIgnore, System.Text.Json.Serialization.JsonIgnore]
        public virtual ICollection<OrderItem> OrderItems { set; get; }
        [XmlIgnore, System.Text.Json.Serialization.JsonIgnore]
        public virtual ICollection<ItemReview> ItemReviews { set; get; }
    }
}
```

4.5.3 Order.cs Model

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;
using System.Xml.Serialization;

namespace ClickNEatReact.Models
{
    /// <summary>
    /// This class models the data for the Order object will all necessary
attributes.
    /// </summary>
    public class Order
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

        public int OrderId { set; get; }
        public double Total { set; get; }
        public string Instruction { set; get; }
        public string TableIdentity { set; get; }
        public string Status { set; get; }
        [XmlIgnore, JsonIgnore]
        public virtual ICollection<OrderItem> OrderItems { set; get; }
    }
}
```

4.5.4 OrderItem.cs Model

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text.Json.Serialization;
using System.Threading.Tasks;
using System.Xml.Serialization;

namespace ClickNEatReact.Models
{
    /// <summary>
    /// This class models the OrderItem object in the database.
    /// </summary>
    public class OrderItem
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]

        public int OrderItemId { set; get; }
        public int ItemAmmount { set; get; }
        public double price { set; get; }
        public string OrderItemStatus { set; get; }
        public bool IsReviewed { set; get; }
        public string Instruction { set; get; }
        public int MenuItemId { set; get; }
        [ForeignKey("MenuItemId")]

        [XmlIgnore, Newtonsoft.Json.JsonIgnore]
        public virtual MenuItem MenuItem { set; get; }

        [XmlIgnore, JsonIgnore]
        public virtual Order Order { set; get; }
    }
}
```

4.5.5 ItemReview.cs Model

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;
using System.Xml.Serialization;

namespace ClickNEatReact.Models
{
    /// <summary>
    /// This is the model class which models the data for the
    /// review feature in the application. It mirrors a database table.
    /// </summary>
    public class ItemReview
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int ReviewId { set; get; }
        public float Rate { set; get; }
        public string Comment { set; get; }
        public string Reviewer { set; get; }
        public int OrderItemId { set; get; }

        [DatabaseGenerated(DatabaseGeneratedOption.Computed)]
        public DateTime CreatedAt { set; get; }
        public int MenuItemId { set; get; }
        [ForeignKey("MenuItemId")]
        [XmlIgnore,JsonIgnore]
        public virtual MenuItem MenuItem { set; get; }
    }
}
```

4.5.6 Payment.cs Model

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;
using System.Xml.Serialization;

namespace ClickNEatReact.Models
{
    /// <summary>
    /// This class models the Payment object.
    /// </summary>
    public class Payment
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int PaymentId { set; get; }
        public string CardHolderName { set; get; }
        public double Amount { set; get; }
        public string CardNumber { set; get; }
        public string ExpireDate { set; get; }
        public string CVV { set; get; }
        [DatabaseGenerated(DatabaseGeneratedOption.Computed)]
        public DateTime CreatedAt { set; get; }
        public int OrderId { set; get; }
        [ForeignKey("OrderId")]
        [XmlIgnore, JsonIgnore]
        public virtual Order order { set; get; }
    }
}
```

4.5.7 TransactionData.cs Model

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ClickNEatReact.Models
{
    /// <summary>
    /// This class models the attributes needed for
    /// the Braintree payment service.
    /// </summary>
    public class TransactionData
    {
        public string Correlation_id { set; get; }
        public string Nonce { set; get; }
        public decimal Amount { set; get; }
    }
}
```

4.5.8 Response.cs Model

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ClickNEatReact.Models
{
    /// <summary>
    /// This class models the Response status codes
    /// and the title of the responses for the authentication purposes.
    /// </summary>
    public class Response
    {
        public string status { set; get; }
        public string title { set; get; }
    }
}
```

4.5.9 UserRole.cs Model

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ClickNEatReact.Models
{
    /// <summary>
    /// This model class sets up the role attributes
    /// for authorization purposes.
    /// </summary>
    public static class UserRole
    {
        public const string User = "User";
        public const string Admin = "Admin";
        public const string Waiter = "Waiter";
    }
}
```

4.5.10 ApplicationUser.cs Model

```
using Microsoft.AspNetCore.Identity;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ClickNEatReact.Models
{
    /// <summary>
    /// User data inherits from IdentityUser and thus is supported.
    /// </summary>
    public class ApplicationUser:IdentityUser
    {
        [PersonalData]
        public string FirstName { set; get; }
        [PersonalData]
        public string LastName { set; get; }
    }
}
```

4.5.11 RegistrationModel.cs Model

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace ClickNEatReact.Models
{
    /// <summary>
    /// This class model the Registration object for the Register feature.
    /// </summary>
    public class RegistrationModel
    {
        [Required(ErrorMessage = "Username is Required")]
        public string Username { set; get; }
        [Required(ErrorMessage = "Firstname is Required")]
        public string Firstname { set; get; }
        [Required(ErrorMessage = "Lastname is Required")]
        public string Lastname { set; get; }

        [Required(ErrorMessage = "Password is Required")]
        public string Password { set; get; }
        [Required(ErrorMessage = "Phone number is Required")]
        public string Phone { set; get; }
        [Required(ErrorMessage = "Email address is Required")]
        public string Email { set; get; }
    }
}
```

4.5.12 LoginModel.cs Model

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace ClickNEatReact.Models
{
    /// <summary>
    /// This class models the Login in the database with the required
```

```
attributes.
    /// </summary>
    public class LoginModel
    {
        [Required(ErrorMessage = "Username is Required")]
        public string Username { set; get; }
        [Required(ErrorMessage = "Password is Required")]
        public string Password { set; get; }
    }
}
```

4.5.13 PasswordData.cs Model

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ClickNEatReact.Models
{
    /// <summary>
    /// This class models the Password data in the database with
    /// the required attributes.
    /// </summary>
    public class PasswordData
    {
        public string OldPassword { set; get; }
        public string NewPassword { set; get; }
        public string ConfirmPassword { set; get; }
    }
}
```

4.6 ClickEatContext.cs Data

```
using ClickNEatReact.Models;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ClickNEatReact.Data
{
    /// <summary>
    ///  A single Context class is used that inherits from
IdentityDbContext.This way the context is aware of any
    ///  relations between the existing classes and the IdentityUser and
Roles of the IdentityDbContext.
    /// This way the overheads are reduced. This concept is basically a
DbContext with a set data for
    /// the users and one for the roles.
    /// </summary>
    public class ClickEatContext:IdentityDbContext<ApplicationUser>
    {
        public ClickEatContext(DbContextOptions<ClickEatContext>
options):base(options)
        {

        }
        /// <summary>
        /// This method is called when the context is first created to build
the model and the mapping in the memory.
        /// </summary>
        /// <param name="builder">OnModelCreating method takes and instance
of ModelBuilder as a parameter</param>
        protected override void OnModelCreating(ModelBuilder builder)
        {
            base.OnModelCreating(builder);
            builder.Entity<ItemReview>()
            .Property(b => b.CreatedAt)
            .HasDefaultValueSql("getdate()");
            builder.Entity<Payment>()
            .Property(p => p.CreatedAt)
            .HasDefaultValueSql("getdate()");
        }
        public DbSet<Order> orders { set; get; }
        public DbSet<MenuCategory> menuCategories { set; get; }
        public DbSet<MenuItem> menuItems { set; get; }
        public DbSet<Payment> Payment { get; set; }
```

```
        public DbSet<OrderItem> OrderItems { set; get; }
        public DbSet<ItemReview> ItemReviews { set; get; }
    }
}
```

4.7 Controllers

The application's flow control logic is contained in a controller. When a user makes a browser request, the controller decides what answer to send back to the user.

4.7.1 AuthenticationController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.IdentityModel.Tokens;
using ClickNEatReact.Models;
using System;
using System.Collections.Generic;
using System.IdentityModel.Tokens.Jwt;
using System.Linq;
using System.Security.Claims;
using System.Text;
using System.Threading.Tasks;

namespace ClickNEatReact.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AuthenticationController : ControllerBase
    {
        private readonly UserManager<ApplicationUser> userManager;
        private readonly RoleManager<IdentityRole> roleManager;
        private readonly IConfiguration _configuration;

        public AuthenticationController(UserManager<ApplicationUser>
userManager, RoleManager<IdentityRole> roleManager, IConfiguration
configuration)
        {
            this.userManager = userManager;
```

```
            this.roleManager = roleManager;
            _configuration = configuration;
        }

        [HttpPost]
        [Route("register/user")]
        public async Task<IActionResult> Register([FromBody]
RegistrationModel user)
        {
            var userExists = await
userManager.FindByNameAsync(user.Username);
            if (userExists != null)
            {
                return StatusCode(StatusCodes.Status400BadRequest, new
Response { status="Error", title="User already exists" });
            }
            else
            {
                ApplicationUser appUser = new ApplicationUser()
                {
                    Email = user.Email,
                    SecurityStamp = Guid.NewGuid().ToString(),
                    UserName = user.Username,
                    PhoneNumber = user.Phone,
                    FirstName = user.Firstname,
                    LastName = user.Lastname
                };

                var result = await userManager.CreateAsync(appUser,
user.Password);

                if (!result.Succeeded)
                {
                    var errorString = "";
                    foreach(var err in result.Errors)
                    {
                        errorString += err.Description + "\r\n";

                    }
                    return StatusCode(StatusCodes.Status400BadRequest, new
Response { status = "Error", title = errorString });
                }
                else
                {
                    if (!await roleManager.RoleExistsAsync(UserRole.User))
                    {
                        await roleManager.CreateAsync(new
IdentityRole(UserRole.User));
```

```
                    }
                    if (await roleManager.RoleExistsAsync(UserRole.User))
                    {
                        await userManager.AddToRoleAsync(appUser,
UserRole.User);
                    }
                    return StatusCode(StatusCodes.Status201Created, new
Response { status = "Success", title = "User Created Successfully" });
                }
            }
        }

        [Authorize(Roles =UserRole.Admin)]
        [HttpPost]
        [Route("register/waiter")]
        public async Task<IActionResult> RegisterWaiter([FromBody]
RegistrationModel user)
        {
            var userExists = await
userManager.FindByNameAsync(user.Username);
            if (userExists != null)
            {
                return StatusCode(StatusCodes.Status400BadRequest, new
Response { status = "Error", title = "User already exists" });
            }
            else
            {
                ApplicationUser appUser = new ApplicationUser()
                {
                    Email = user.Email,
                    SecurityStamp = Guid.NewGuid().ToString(),
                    UserName = user.Username,
                    PhoneNumber = user.Phone,
                    FirstName = user.Firstname,
                    LastName = user.Lastname
                };

                var result = await userManager.CreateAsync(appUser,
user.Password);

                if (!result.Succeeded)
                {
                    var errorString = "";
                    foreach (var err in result.Errors)
                    {
                        errorString += err.Description + "\r\n";

                    }
```

```
                    return StatusCode(StatusCodes.Status400BadRequest, new
Response { status = "Error", title = errorString });
                }
                else
                {
                    if (!await roleManager.RoleExistsAsync(UserRole.Waiter))
                    {
                        await roleManager.CreateAsync(new
IdentityRole(UserRole.Waiter));
                    }
                    if (await roleManager.RoleExistsAsync(UserRole.Waiter))
                    {
                        await userManager.AddToRoleAsync(appUser,
UserRole.Waiter);
                    }
                    return StatusCode(StatusCodes.Status201Created, new
Response { status = "Success", title = "Waiter Created Successfully" });
                }
            }
        }

        [HttpPost]
        [Route("login")]
        public async Task<IActionResult> Login([FromBody] LoginModel user)
        {
            var userExists = await
userManager.FindByNameAsync(user.Username);
            if (userExists != null)
            {
                if (await userManager.CheckPasswordAsync(userExists,
user.Password))
                {
                    var userRoles = await
userManager.GetRolesAsync(userExists);
                    var authClaims = new List<Claim>
                    {
                        new Claim(ClaimTypes.Name,userExists.UserName),
                        new
Claim(JwtRegisteredClaimNames.Jti,Guid.NewGuid().ToString()),
                    };

                    foreach (var userRole in userRoles)
                    {
                        authClaims.Add(new Claim(ClaimTypes.Role,
userRole));
                    }

                    var authSignInKey = new
```

```
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["JWT:Secrete"]));
                    var token = new JwtSecurityToken(
                        issuer: _configuration["JWT:ValidIssuer"],
                        audience: _configuration["JWT:ValidAudience"],
                        expires: DateTime.Now.AddHours(24),
                        claims: authClaims,
                        signingCredentials: new
SigningCredentials(authSignInKey, SecurityAlgorithms.HmacSha256)
                        );
                    return StatusCode(StatusCodes.Status200OK, new
                    {

                        username=userExists.UserName,
                        userId=userExists.Id,
                        type=userRoles.Count>1?UserRole.Admin:userRoles[0],
                        token = new
JwtSecurityTokenHandler().WriteToken(token)
                    });;
                }
                else
                    return Unauthorized();
            }
            else
            {
                return Unauthorized();
            }
        }

        [HttpPost]
        [Route("register/admin")]
        public async Task<IActionResult> RegisterAdmin([FromBody]
RegistrationModel user)
        {
            var userExists = await
userManager.FindByNameAsync(user.Username);
            if (userExists != null)
            {
                return StatusCode(StatusCodes.Status400BadRequest, new
Response { status = "Error", title = "User already exists" });
            }
            else
            {
                ApplicationUser appUser = new ApplicationUser()
                {
                    Email = user.Email,
                    SecurityStamp = Guid.NewGuid().ToString(),
                    UserName = user.Username,
                    PhoneNumber = user.Phone,
```

```
                    FirstName = user.Firstname,
                    LastName = user.Lastname
                };

                var result = await userManager.CreateAsync(appUser,
user.Password);

                if (!result.Succeeded)
                {
                    var errorString = "";
                    foreach (var err in result.Errors)
                    {
                        errorString += err.Description + "\r\n";

                    }
                    return StatusCode(StatusCodes.Status400BadRequest, new
Response { status = "Error", title = errorString });
                }
                else
                {
                    if (!await roleManager.RoleExistsAsync(UserRole.Admin))
                    {
                        await roleManager.CreateAsync(new
IdentityRole(UserRole.Admin));
                    }
                    if (!await roleManager.RoleExistsAsync(UserRole.Waiter))
                    {
                        await roleManager.CreateAsync(new
IdentityRole(UserRole.Waiter));
                    }

                    if (!await roleManager.RoleExistsAsync(UserRole.User))
                    {
                        await roleManager.CreateAsync(new
IdentityRole(UserRole.User));
                    }
                    if (await roleManager.RoleExistsAsync(UserRole.Admin))
                    {
                        await userManager.AddToRolesAsync(appUser,new
List<string>() { UserRole.Admin, UserRole.User, UserRole.Waiter });
                    }
                    return StatusCode(StatusCodes.Status201Created, new
Response { status = "Success", title = "Admin Created Successfully" });
                }
            }
        }

        [Authorize(Roles = UserRole.Admin)]
```

```
        [HttpGet]
        [Route("users")]
        public async Task<IActionResult> GetUsers()
        {
            var users = await userManager.FindByNameAsync("admin");
            return Ok(new { user = users });
        }

        [Authorize(Roles = UserRole.Admin)]
        [HttpGet]
        [Route("users/waiters")]
        public async Task<IActionResult> GetWaiters()
        {
            var users = await
userManager.GetUsersInRoleAsync(UserRole.Waiter);
            return Ok(new { user = users });
        }

        [Authorize(Roles = UserRole.Admin)]
        [HttpDelete]
        [Route("user/{username}")]
        public async Task<IActionResult> DeleteUser(string username)
        {
            var userExists = await userManager.FindByNameAsync(username);
            if (userExists != null)
            {
                var users = await userManager.DeleteAsync(userExists);
            }
            else
            {
                return StatusCode(StatusCodes.Status400BadRequest, new
Response { status = "Error", title = "User doesn't exists" });
            }

            return NoContent();
        }

        [Authorize(Roles = UserRole.Admin)]
        [HttpGet]
        [Route("users/customers")]
        public async Task<IActionResult> GetCustomers()
        {
            var users  = await
userManager.GetUsersInRoleAsync(UserRole.User);
            var admins = await
userManager.GetUsersInRoleAsync(UserRole.Admin);
            foreach(ApplicationUser admin in admins)
            {
```

```
                    users = users.Where<ApplicationUser>(U=>U.UserName !=
admin.UserName).ToList<ApplicationUser>();
            }
            //Console.WriteLine(JsonConvert.SerializeObject(users));
            return Ok(new { user = users });
        }

        [Authorize]
        [HttpPost]
        [Route("changePassword/{username}")]
        public async Task<IActionResult> ChangePassword([FromBody]
PasswordData passwords, string username)
        {
            var user = await userManager.FindByNameAsync(username);
            if (user == null)
            {
                return StatusCode(StatusCodes.Status400BadRequest, new
Response { status = "Error", title = "User does not exist" });
            }
            else
            {

                if (passwords.NewPassword.Equals(passwords.ConfirmPassword))
                {
                    var result = await userManager.ChangePasswordAsync(user,
passwords.OldPassword, passwords.NewPassword);
                    if (!result.Succeeded)
                    {
                        var errorString = "";
                        foreach (var err in result.Errors)
                        {
                            errorString += err.Description + "\r\n";

                        }
                        return StatusCode(StatusCodes.Status400BadRequest,
new Response { status = "Error", title = errorString });
                    }
                    else
                    {
                        return StatusCode(StatusCodes.Status200OK, new
Response { status = "Success", title = "Password changed successfully" });
                    }
                }
                else
                {
                    return StatusCode(StatusCodes.Status400BadRequest, new
Response { status = "Error", title = "Confirm password doesn't match with
new password"});
```

4.7.2 BraintreeController.cs

```
using Braintree;
using ClickNEatReact.Models;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;

namespace ClickNEatReact.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class BraintreeController : ControllerBase
    {
        private readonly IConfiguration _configuration;
        private readonly IBraintreeGateway gateway;
        public BraintreeController(IConfiguration configuration)
        {
            _configuration = configuration;
            gateway = new BraintreeGateway
            {
                Environment = Braintree.Environment.SANDBOX,
                MerchantId = _configuration["Braintree:MerchantId"],
                PublicKey = _configuration["Braintree:PublicKey"],
                PrivateKey = _configuration["Braintree:PrivateKey"]
            };
        }

        [HttpGet]
        public ObjectResult GetClientToken()
        {
            var token = gateway.ClientToken.Generate();

            return Ok(new{ token = token });
        }

        [HttpPost]
        public ObjectResult Transaction([FromBody] TransactionData data)
        {
            var req = new TransactionRequest
            {
```

```
                Amount = data.Amount,
                PaymentMethodNonce = data.Nonce,
                DeviceData =
"{\"correlation_id\":\""+data.Correlation_id+"\"}",
                CurrencyIsoCode="EUR",
                Options=new TransactionOptionsRequest
                {
                    SubmitForSettlement=true
                }

            };

            Result<Transaction> result = gateway.Transaction.Sale(req);

            return Ok(new { result=result });
        }
    }
}
```

4.7.3 ItemReviewsController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ClickNEatReact.Data;
using ClickNEatReact.Models;

namespace ClickNEatReact.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ItemReviewsController : ControllerBase
    {
        private readonly ClickEatContext _context;

        public ItemReviewsController(ClickEatContext context)
        {
            _context = context;
        }

        // GET: api/ItemReviews
        [HttpGet]
        public async Task<ActionResult<IEnumerable<ItemReview>>>
GetItemReviews()
        {
            return await _context.ItemReviews.ToListAsync();
        }

        [HttpGet("MenuItem/{menuItemId}")]
        public async Task<ActionResult<IEnumerable<ItemReview>>>
GetItemReviewsByMenuItem(int menuItemId)
        {
            return await
_context.ItemReviews.Where(m=>m.MenuItemId==menuItemId).OrderByDescending(m=
>m.CreatedAt).ToListAsync();
        }

        // GET: api/ItemReviews/5
        [HttpGet("{id}")]
        public async Task<ActionResult<ItemReview>> GetItemReview(int id)
        {
            var itemReview = await _context.ItemReviews.FindAsync(id);
```

```
            if (itemReview == null)
            {
                return NotFound();
            }

            return itemReview;
        }


        // PUT: api/ItemReviews/5
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPut("{id}")]
        public async Task<IActionResult> PutItemReview(int id, ItemReview
itemReview)
        {
            if (id != itemReview.ReviewId)
            {
                return BadRequest();
            }

            _context.Entry(itemReview).State = EntityState.Modified;

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!ItemReviewExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return NoContent();
        }

        // POST: api/ItemReviews
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPost]
        public async Task<ActionResult<ItemReview>>
```

```
PostItemReview(ItemReview itemReview)
        {

            _context.ItemReviews.Add(itemReview);

            MenuItem menuItem = await
_context.menuItems.Where(m=>m.MenuItemId==itemReview.MenuItemId).FirstOrDefa
ultAsync();
            menuItem.ReviewCount += 1;
            menuItem.AvgRate = (menuItem.AvgRate *
(double)(menuItem.ReviewCount - 1) + (double)itemReview.Rate) /
(double)menuItem.ReviewCount;
            _context.Entry(menuItem).State = EntityState.Modified;
            await _context.SaveChangesAsync();
            return CreatedAtAction("GetItemReview", new { id =
itemReview.ReviewId }, itemReview);
        }

        // DELETE: api/ItemReviews/5
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteItemReview(int id)
        {
            var itemReview = await _context.ItemReviews.FindAsync(id);
            if (itemReview == null)
            {
                return NotFound();
            }

            _context.ItemReviews.Remove(itemReview);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool ItemReviewExists(int id)
        {
            return _context.ItemReviews.Any(e => e.ReviewId == id);
        }
    }
}
```

4.7.4 MenuCategoriesController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ClickNEatReact.Data;
using ClickNEatReact.Models;
using Microsoft.AspNetCore.Authorization;

namespace ClickNEatReact.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class MenuCategoriesController : ControllerBase
    {
        private readonly ClickEatContext _context;

        public MenuCategoriesController(ClickEatContext context)
        {
            _context = context;
        }

        // GET: api/MenuCategories
        [Authorize(Roles = UserRole.Admin)]
        [HttpGet]
        public async Task<ActionResult<IEnumerable<MenuCategory>>>
GetmenuCategories()
        {
            return await _context.menuCategories.Include(c =>
c.menuItems).ToListAsync();
        }

        // GET: api/MenuCategories/5
        [Authorize(Roles = UserRole.Admin)]
        [HttpGet("{id}")]
        public async Task<ActionResult<MenuCategory>> GetMenuCategory(int
id)
        {
            var menuCategory = await
_context.menuCategories.Include(c=>c.menuItems).Where(c=>c.CategoryId==id).F
irstOrDefaultAsync();

            if (menuCategory == null)
```

```
            {
                return NotFound();
            }

            return menuCategory;
        }

        // PUT: api/MenuCategories/5
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [Authorize(Roles = UserRole.Admin)]
        [HttpPut("{id}")]
        public async Task<IActionResult> PutMenuCategory(int id,
MenuCategory menuCategory)
        {
            if (id != menuCategory.CategoryId)
            {
                return BadRequest();
            }

            _context.Entry(menuCategory).State = EntityState.Modified;

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!MenuCategoryExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return NoContent();
        }

        // POST: api/MenuCategories
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [Authorize(Roles = UserRole.Admin)]
        [HttpPost]
        public async Task<ActionResult<MenuCategory>>
PostMenuCategory(MenuCategory menuCategory)
```

```
        {
            _context.menuCategories.Add(menuCategory);
            await _context.SaveChangesAsync();

            return CreatedAtAction("GetMenuCategory", new { id =
menuCategory.CategoryId }, menuCategory);
        }

        // DELETE: api/MenuCategories/5
        [Authorize(Roles = UserRole.Admin)]
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteMenuCategory(int id)
        {
            var menuCategory = await _context.menuCategories.FindAsync(id);
            if (menuCategory == null)
            {
                return NotFound();
            }

            _context.menuCategories.Remove(menuCategory);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool MenuCategoryExists(int id)
        {
            return _context.menuCategories.Any(e => e.CategoryId == id);
        }
    }
}
```

4.7.5 MenuItemsControllers.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ClickNEatReact.Data;
using ClickNEatReact.Models;
using Microsoft.AspNetCore.Authorization;
using System.IO;

namespace ClickNEatReact.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class MenuItemsController : ControllerBase
    {
        private readonly ClickEatContext _context;

        public MenuItemsController(ClickEatContext context)
        {
            _context = context;
        }

        // GET: api/MenuItems
        [HttpGet]
        public async Task<ActionResult<IEnumerable<MenuItem>>>
GetmenuItems()
        {
            //await PayPalClient.captureOrder();
            return await
_context.menuItems.Include(m=>m.MenuCategory).Include(m =>
m.ItemReviews).ToListAsync();
        }

        // GET: api/MenuItems/5

        [HttpGet("{id}")]
        public async Task<ActionResult<MenuItem>> GetMenuItem(int id)
        {
            var menuItem = await _context.menuItems.Include(m =>
m.MenuCategory).Include(m=>m.ItemReviews).Where(m => m.MenuItemId ==
id).FirstOrDefaultAsync() ;
```

```
            if (menuItem == null)
            {
                return NotFound();
            }

            return menuItem;
        }

        // PUT: api/MenuItems/5
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [Authorize(Roles = UserRole.Admin)]
        [HttpPut("{id}")]
        public async Task<IActionResult> PutMenuItem(int id, MenuItem
menuItem)
        {
            if (id != menuItem.MenuItemId)
            {
                return BadRequest();
            }

            _context.Entry(menuItem).State = EntityState.Modified;

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!MenuItemExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return NoContent();
        }

        // POST: api/MenuItems
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [Authorize(Roles = UserRole.Admin)]
        [HttpPost]
        public async Task<ActionResult<MenuItem>> PostMenuItem(MenuItem
```

```
menuItem)
        {
            _context.menuItems.Add(menuItem);
            await _context.SaveChangesAsync();

            return CreatedAtAction("GetMenuItem", new { id =
menuItem.MenuItemId }, menuItem);
        }

        // DELETE: api/MenuItems/5
        [Authorize(Roles = UserRole.Admin)]
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteMenuItem(int id)
        {
            var menuItem = await _context.menuItems.FindAsync(id);
            if (menuItem == null)
            {
                return NotFound();
            }

            _context.menuItems.Remove(menuItem);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        [HttpPost]
        [Route("/api/fileupload")]
        public async Task<IActionResult> PostUploadAsync(IFormFile file)
        {
            long size = file.Length;
            var filePath = Path.Combine("wwwroot/img", Guid.NewGuid() +
Path.GetExtension(file.FileName));


            if (file.Length > 0)
            {

                Console.WriteLine(filePath);
                using (var stream = System.IO.File.Create(filePath))
                {
                    await file.CopyToAsync(stream);
                }
            }

            return Ok(new { size, url = filePath.Replace("wwwroot", "") });
        }
```

```
        private bool MenuItemExists(int id)
        {
            return _context.menuItems.Any(e => e.MenuItemId == id);
        }
    }
}
```

4.7.6 OrderItemsController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ClickNEatReact.Data;
using ClickNEatReact.Models;
using Microsoft.AspNetCore.Authorization;

namespace ClickNEatReact.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class OrderItemsController : ControllerBase
    {
        private readonly ClickEatContext _context;

        public OrderItemsController(ClickEatContext context)
        {
            _context = context;
        }

        // GET: api/OrderItems
        [Authorize]
        [HttpGet]
        public async Task<ActionResult<IEnumerable<OrderItem>>>
GetOrderItems()
        {
            var orderItems=await
_context.OrderItems.Include(o=>o.MenuItem).ThenInclude(m=>m.MenuCategory).In
clude(o=>o.Order).ToListAsync();

            return orderItems;
```

```
        }

        // GET: api/OrderItems/5
        [Authorize]
        [HttpGet("{id}")]
        public async Task<ActionResult<OrderItem>> GetOrderItem(int id)
        {
            var orderItem = await _context.OrderItems.Include(o =>
o.MenuItem).ThenInclude(m=>m.MenuCategory).Include(o=>o.Order).Where(o =>
o.OrderItemId == id).FirstOrDefaultAsync() ;

            if (orderItem == null)
            {
                return NotFound();
            }

            return orderItem;
        }

        // PUT: api/OrderItems/5
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754

        [HttpPut("{id}")]
        public async Task<IActionResult> PutOrderItem(int id, OrderItem
orderItem)
        {
            if (id != orderItem.OrderItemId)
            {
                return BadRequest();
            }

            _context.Entry(orderItem).State = EntityState.Modified;

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!OrderItemExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
```

```
            }

            return NoContent();
        }

        // POST: api/OrderItems
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPost]
        public async Task<ActionResult<OrderItem>> PostOrderItem(OrderItem
orderItem)
        {
            _context.OrderItems.Add(orderItem);
            await _context.SaveChangesAsync();

            return CreatedAtAction("GetOrderItem", new { id =
orderItem.OrderItemId }, orderItem);
        }

        // DELETE: api/OrderItems/5
        [Authorize]
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteOrderItem(int id)
        {
            var orderItem = await _context.OrderItems.FindAsync(id);
            if (orderItem == null)
            {
                return NotFound();
            }

            _context.OrderItems.Remove(orderItem);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool OrderItemExists(int id)
        {
            return _context.OrderItems.Any(e => e.OrderItemId == id);
        }
    }
}
```

4.7.7 OrdersController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ClickNEatReact.Data;
using ClickNEatReact.Models;
using Microsoft.AspNetCore.Authorization;
using Newtonsoft.Json;

namespace ClickNEatReact.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class OrdersController : ControllerBase
    {
        private readonly ClickEatContext _context;

        public OrdersController(ClickEatContext context)
        {
            _context = context;
        }

        // GET: api/Orders
        [Authorize]
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Order>>> Getorders()
        {
            return await
_context.orders.Where(o=>o.Status!="Served").Include(o =>
o.OrderItems).ThenInclude(o=>o.MenuItem).ThenInclude(m =>
m.MenuCategory).OrderByDescending(o=>o.OrderId).ToListAsync();
        }


        [HttpPost("userOrders/ids")]
        public async Task<ActionResult<IEnumerable<Order>>>
GetordersByIds([FromBody] string orderIds)
        {

            //orderIds = JsonConvert.DeserializeObject<string>(orderIds);
```

```
            orderIds = orderIds.Trim().Replace(' ',',');

            var orders= await _context.orders.FromSqlRaw("select * from
orders where orderId in ("+ orderIds + ")").Include(o =>
o.OrderItems).ThenInclude(o => o.MenuItem).ThenInclude(m =>
m.MenuCategory).OrderByDescending(o => o.OrderId).ToListAsync();

            return orders;
        }

        // GET: api/Orders/5

        [HttpGet("{id}")]
        public async Task<ActionResult<Order>> GetOrder(int id)
        {
            var order = await _context.orders.Include(o =>
o.OrderItems).ThenInclude(o=>o.MenuItem).ThenInclude(m=>m.MenuCategory).Wher
e(o => o.OrderId == id).FirstOrDefaultAsync();


            if (order == null)
            {
                return NotFound();
            }

            return order;
        }

        // PUT: api/Orders/5
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [Authorize]
        [HttpPut("{id}")]
        public async Task<IActionResult> PutOrder(int id, Order order)
        {
            if (id != order.OrderId)
            {
                return BadRequest();
            }

            _context.Entry(order).State = EntityState.Modified;

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
```

```
                if (!OrderExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return NoContent();
        }
        [HttpPut("{id}/addItems")]
        public async Task<IActionResult> PutOrderToAddItem(int id, Order
order)
        {
            var prevOrder = await _context.orders.Include(o =>
o.OrderItems).ThenInclude(o => o.MenuItem).ThenInclude(m =>
m.MenuCategory).Where(o => o.OrderId == id).FirstOrDefaultAsync();
            foreach(OrderItem item in order.OrderItems)
            {
                prevOrder.OrderItems.Add(item);
            }
            if (id != prevOrder.OrderId)
            {
                return BadRequest();
            }
            prevOrder.TableIdentity = order.TableIdentity;
            prevOrder.Total += order.Total;

            _context.Entry(prevOrder).State = EntityState.Modified;

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!OrderExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
```

```
            return Ok(new { prevOrder });
        }

        // POST: api/Orders
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPost]
        public async Task<ActionResult<Order>> PostOrder(Order order)
        {
            _context.orders.Add(order);
            await _context.SaveChangesAsync();

            return CreatedAtAction("GetOrder", new { id = order.OrderId },
order);
        }

        // DELETE: api/Orders/5
        [Authorize]
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteOrder(int id)
        {
            var order = await _context.orders.FindAsync(id);
            if (order == null)
            {
                return NotFound();
            }

            _context.orders.Remove(order);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool OrderExists(int id)
        {
            return _context.orders.Any(e => e.OrderId == id);
        }
    }
}
```

4.7.8 PaymentsController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ClickNEatReact.Data;
using ClickNEatReact.Models;
using Microsoft.AspNetCore.Authorization;
using System.Diagnostics;

namespace ClickNEatReact.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class PaymentsController : ControllerBase
    {
        private readonly ClickEatContext _context;

        public PaymentsController(ClickEatContext context)
        {
            _context = context;
        }

        [Authorize(Roles = UserRole.Admin + ","+UserRole.Waiter)]
        // GET: api/Payments
        [HttpGet("period/{period}")]
        public async Task<ActionResult<IEnumerable<Payment>>>
GetPaymentByPeriod(string period)
        {
            if(period.Equals("1"))
                return await _context.Payment.Include(p =>
p.order).ThenInclude(o => o.OrderItems).ThenInclude(o =>
o.MenuItem).ThenInclude(m => m.MenuCategory).Where(p => p.CreatedAt.Date ==
DateTime.Now.Date).ToListAsync();
            else if (period.Equals("7"))
                return await _context.Payment.Include(p =>
p.order).ThenInclude(o => o.OrderItems).ThenInclude(o =>
o.MenuItem).ThenInclude(m => m.MenuCategory).Where(p => (p.CreatedAt.Date <=
DateTime.Now.Date) && (p.CreatedAt.Date >
DateTime.Now.AddDays(-7))).ToListAsync();
            else if (period.Equals("30"))
                return await _context.Payment.Include(p =>
p.order).ThenInclude(o => o.OrderItems).ThenInclude(o =>
```

```
o.MenuItem).ThenInclude(m => m.MenuCategory).Where(p => (p.CreatedAt.Date <=
DateTime.Now.Date) && (p.CreatedAt.Date >
DateTime.Now.AddDays(-30))).ToListAsync();
            else
                return await _context.Payment.Include(p =>
p.order).ThenInclude(o => o.OrderItems).ThenInclude(o =>
o.MenuItem).ThenInclude(m => m.MenuCategory).Where(p => (p.CreatedAt.Year ==
DateTime.Now.Year)).ToListAsync();

        }

        [Authorize(Roles =UserRole.Admin)]
        // GET: api/Payments
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Payment>>> GetPayment()
        {
            return await _context.Payment.Include(p=>p.order).ThenInclude(o
=> o.OrderItems).ThenInclude(o => o.MenuItem).ThenInclude(m =>
m.MenuCategory).ToListAsync();
        }

        // GET: api/Payments/5
        [Authorize(Roles = UserRole.Admin)]
        [HttpGet("{id}")]
        public async Task<ActionResult<Payment>> GetPayment(int id)
        {
            var payment = await _context.Payment.Include(p =>
p.order).ThenInclude(o => o.OrderItems).ThenInclude(o =>
o.MenuItem).ThenInclude(m =>
m.MenuCategory).Where(p=>p.PaymentId==id).FirstOrDefaultAsync();

            if (payment == null)
            {
                return NotFound();
            }

            return payment;
        }

        // PUT: api/Payments/5
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [Authorize(Roles = UserRole.Admin)]
        [HttpPut("{id}")]
        public async Task<IActionResult> PutPayment(int id, Payment payment)
        {
            if (id != payment.PaymentId)
            {
```

```
                return BadRequest();
            }

            _context.Entry(payment).State = EntityState.Modified;

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!PaymentExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return NoContent();
        }

        // POST: api/Payments
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPost]
        public async Task<ActionResult<Payment>> PostPayment(Payment
payment)
        {
            var amount = 0.0;

            foreach(var item in payment.order.OrderItems)
            {
                MenuItem menuItem = await
_context.menuItems.FindAsync(item.MenuItemId);
                amount += menuItem.Price * item.ItemAmount;
            }

            payment.Amount = amount;

            _context.Payment.Add(payment);
            await _context.SaveChangesAsync();

            return CreatedAtAction("GetPayment", new { id =
payment.PaymentId }, payment);
        }
```

```
        [HttpPost("DuePayment")]
        public async Task<ActionResult<Payment>> PostDuePayment(Payment
payment)
        {
            var amount = 0.0;

            foreach (var item in payment.order.OrderItems)
            {
                MenuItem menuItem = await
_context.menuItems.FindAsync(item.MenuItemId);
                amount += menuItem.Price * item.Item Amount;
            }

            payment.Amount = amount;
            var order = payment.order;
            payment.order = null;

            _context.Payment.Add(payment);
            await _context.SaveChangesAsync();

            order.Status = "Paid";
            _context.Entry(order).State = EntityState.Modified;
            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException ex)
            {
                Console.WriteLine(ex.Message);
            }

            return CreatedAtAction("GetPayment", new { id =
payment.PaymentId }, payment);
        }

        // DELETE: api/Payments/5
        [Authorize(Roles = UserRole.Admin)]
        [HttpDelete("{id}")]

        public async Task<IActionResult> DeletePayment(int id)
        {
            var payment = await _context.Payment.FindAsync(id);
            if (payment == null)
            {
                return NotFound();
            }
```

```
            _context.Payment.Remove(payment);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool PaymentExists(int id)
        {
            return _context.Payment.Any(e => e.PaymentId == id);
        }
    }
}
```

4.8 ClientApp/Views/Components

The ClientApp folder contains all the ReactJS components and static files such as css files.

4.8.1 Customer.js Component

```
import React, { Component } from 'react';
import { Card, CardBody, Col, Row } from 'reactstrap';
import UserService from '../../services/UserService';


export default class Customers extends Component {
    constructor(props) {
        super(props)
        this.state = {
            customers: []
        }

        this.getCustomers = this.getCustomers.bind(this);
    }

    componentDidMount() {
        this.getCustomers();
    }

    getCustomers() {
        let self = this;
        UserService.getCustomers().then(function (resp) {
            console.log(resp.data.user);
            self.setState({ customers: resp.data.user });
        }).catch(function (error) {
            console.log(error.response);
        });
    }

    render() {
        return (
            <>
                {
                    this.state.customers.length > 0 ?
                        <Row className="mt-4">
                            <Col>
                                <h4>Customer Accounts</h4>
                                <Row>
                                    {
                                        this.state.customers.map(customer =>
```

```
                                            <Col
key={Math.random().toString(36).substring(0)} xs={12} xs={12} sm={6} lg={4}
className=" mb-4">
                                                <Card
className="menuItemCard" >
                                                    <CardBody
className="text-purple">
                                                        <h6><em
className="font-weight-bold">Name:</em> {customer.firstName + " " +
customer.lastName}</h6>
                                                        <h6><em
className="font-weight-bold">Username:</em> {customer.userName}</h6>
                                                        <h6><em
className="font-weight-bold">Phone:</em> {customer.phoneNumber}</h6>
                                                        <h6><em
className="font-weight-bold">Email:</em> {customer.email}</h6>

                                                    </CardBody>
                                                </Card>
                                            </Col>
                                        )
                                    }
                                </Row>
                            </Col>
                        </Row>
                        :
                        <Row className="mt-4">
                            <Col>
                                <div style={{ height: "500px" }}
className="d-flex align-content-center justify-content-center">
                                    <div class="spinner-border text-primary"
role="status">
                                        <span
class="sr-only">Loading...</span>
                                    </div>
                                </div>
                            </Col>
                        </Row>
                }
            </>
        )
    }
}
```

4.8.2 Waiter.js Component

```
import React, { Component } from 'react';
import { faExclamationCircle } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { Button, Card, CardBody, CardColumns, Col, Modal, ModalBody,
ModalFooter, ModalHeader, Row } from 'reactstrap';
import CreateWaiter from '../Management/Waiter/CreateWaiter';
import WaiterService from '../../services/WaiterService';


export default class Waiter extends Component {
    constructor(props) {
        super(props)
        this.state = {
            mode: 'index',
            waiters: [],
            waiter: {
                id: "",
                firstName: "",
                lastName: "",
                userName: "",
                phoneNumber: "",
                email:""
            },
            isModalOpen:false
        }

        this.getWaiters = this.getWaiters.bind(this);
        this.Delete = this.Delete.bind(this);
        this.ConfirmDelete = this.ConfirmDelete.bind(this);
    }

    componentDidMount() {
        this.getWaiters();
    }

    getWaiters() {
        let self = this;
        WaiterService.getWaiters().then(function (resp) {
            console.log(resp.data.user);
            self.setState({ waiters: resp.data.user });
        }).catch(function (error) {
```

```
            console.log(error.response);
        });
    }


    async ConfirmDelete(waiter) {
        await this.setState({ waiter: waiter });
        this.setState({ isModalOpen: true });
    }
    async Delete(waiter) {

        let self = this;
        this.setState({ isModalOpen: false });
        WaiterService.deleteWaiter(waiter.userName).then(function (resp) {
            self.getWaiters();
        }).catch(function (error) {
            console.log(error.response);
        });

    }

    render() {

        const ConfirmDeleteModal = () => {
            return (
                <Modal returnFocusAfterClose={true}
isOpen={this.state.isModalOpen} className="modal-dialog-centered" >
                    <ModalHeader className="text-danger">
                        <FontAwesomeIcon icon={faExclamationCircle}
size={"lg"} /> Are you sure to delete the waiter account?
                    </ModalHeader>
                    <ModalBody>
                        <Row>
                            <Col>
                                <h6><em
className="font-weight-bold">ID:</em> {this.state.waiter.id}</h6>
                                <h6><em
className="font-weight-bold">Name:</em> {this.state.waiter.firstName + " " +
this.state.waiter.lastName}</h6>
                                <h6><em
className="font-weight-bold">Username:</em>
{this.state.waiter.userName}</h6>
                                <h6><em
className="font-weight-bold">Phone:</em>
{this.state.waiter.phoneNumber}</h6>
                                <h6><em
className="font-weight-bold">Email:</em> {this.state.waiter.email}</h6>
                            </Col>
```

```
                        </Row>
                    </ModalBody>
                    <ModalFooter>
                        <Button onClick={() =>
this.Delete(this.state.waiter)} className="btn  btn-danger">Yes</Button>
                        <Button onClick={() => this.setState({ isModalOpen:
false })} className="btn  btn-primary" > No</Button>
                    </ModalFooter>
                </Modal>
            )
        }

        return (
            <>
                {
                    this.state.mode === 'index' &&
                    <div className="row mt-4">
                        <div className="col-12">

                            {this.state.waiters.length < 1 ?
                                <div className="row">
                                    <div className="col-12 h3">
                                        No waiter accounts added yet
                                    <Button className="btn btn-outline-success
float-right" onClick={() => this.setState({ mode: 'create' })}>Add New
Account</Button>
                                    </div>
                                </div> :
                                <>
                                    <div className="row">
                                        <div className="col-12 h3">
                                            Waiter accounts
                                    <Button className="btn btn-outline-success
float-right" onClick={() => this.setState({ mode: 'create' })}>Add New
Account</Button>

                                    </div>
                                </div>
                                <Row className="mt-4">

                                    {
                                        this.state.waiters.map(waiter =>


                                            <Col
key={Math.random().toString(36).substring(0)} xs={12} xs={12} sm={6} lg={4}
className=" mb-4">
                                                <Card
```

```
className="menuItemCard" >
                                                    <CardBody
className="text-purple">
                                                        <h6><em
className="font-weight-bold">Name:</em> {waiter.firstName + " " +
waiter.lastName}</h6>
                                                        <h6><em
className="font-weight-bold">Username:</em> {waiter.userName}</h6>
                                                        <h6><em
className="font-weight-bold">Phone:</em> {waiter.phoneNumber}</h6>
                                                        <h6><em
className="font-weight-bold">Email:</em> {waiter.email}</h6>
                                                        <button
onClick={() => this.ConfirmDelete(waiter)} className="btn btn-outline-danger
my-0">Delete</button>
                                                    </CardBody>
                                                </Card>
                                            </Col>

                                        )
                                    }

                                </Row>
                            </>
                        }

                    </div>
                </div>

            }
            {
                this.state.mode === 'create' && <div>
                    <CreateWaiter />
                </div>
            }
            <ConfirmDeleteModal />
        </>
    )

}

}
```

4.8.3 Category.js Component

```
import { faExclamationCircle } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import React, { Component } from 'react';
import { Button, Card, CardBody, Col, Modal, ModalBody, ModalFooter,
ModalHeader, Row } from 'reactstrap';
import CategoryServices from '../../../services/CategoryServices';
import CreateCategory from './CreateCategory';
import EditCategory from './EditCategory';

export default class Category extends Component {

    constructor(props) {
        super(props)
        this.state = {
            mode: 'index',
            categories: [],
            category: {
                name: "",
                categoryId:-1,
            },
            isModalOpen:false

        }
        this.getCategories = this.getCategories.bind(this);
        this.ConfirmDelete = this.ConfirmDelete.bind(this);
        this.Delete = this.Delete.bind(this);
        this.setEditMode = this.setEditMode.bind(this);
    }

    componentDidMount() {
        this.getCategories()
    }

    async getCategories() {
        let resp = await CategoryServices.getCategories().catch(function
(error) {
            console.log(error.response);
        });
        if (resp !== undefined) {
            if (resp.status === 200) {
                console.log(resp.data);
```

```
                this.setState({ categories: resp.data })
            }
        }
    }

    async ConfirmDelete(category) {
        await this.setState({ category: category });
        this.setState({ isModalOpen: true });
    }
    async Delete(category) {

        CategoryServices.DeleteCategory(category.categoryId).then(function
(resp) {
            console.log(resp);
            window.location.reload(false);
        }).catch(function (error) {
            console.log(error.response);
        });

    }

    async setEditMode(category) {
        await this.setState({ category: category });
        await this.setState({ mode: 'edit' });
    }

    render() {

        const ConfirmDelete = () => {
            return (
                <Modal returnFocusAfterClose={true}
isOpen={this.state.isModalOpen} className="modal-dialog-centered" >
                    <ModalHeader className= "text-danger">
                        <FontAwesomeIcon  icon={faExclamationCircle}
size={"lg"}/> Are you sure to delete this category?
                    </ModalHeader>
                    <ModalBody>
                        <Row>
                            <Col>
                                ID: {this.state.category.categoryId}
                            </Col>
                        </Row>
                        <Row>
                            <Col>
                                Name: {this.state.category.name}
                            </Col>
                        </Row>
                    </ModalBody>
```

```
                    <ModalFooter>
                        <Button onClick={() =>
this.Delete(this.state.category)} className="btn  btn-danger">Yes</Button>
                        <Button onClick={() => this.setState({ isModalOpen:
false })} className="btn  btn-primary" > No</Button>
                    </ModalFooter>
                </Modal>
                )
        }

        return (
            <>
                {
                    this.state.mode === 'index' &&
                    <div className="row mt-4">
                        <div className="col-12">

                            {this.state.categories.length < 1 ?
                                <div className="row">
                                    <div className="col-12 h3">
                                        No categories added yet
                                    <Button className="btn btn-outline-success
float-right" onClick={() => this.setState({ mode: 'create' })}>Create New
Category</Button>
                                    </div>
                                </div> :
                                <>
                                <div className="row">
                                    <div className="col-12 h3">
                                        Categories
                                    <Button className="btn btn-outline-success
float-right" onClick={() => this.setState({ mode: 'create' })}>Create New
Category</Button>

                                    </div>
                                    </div>

                                    {
                                        this.state.categories.map((category,
index) =>
                                            <Row
key={Math.random().toString(36).substring(0)} xs={12} className="mt-4">
                                                <Col xs={12}>
                                                    <Card
className="shadow-sm">

                                                        <CardBody>
                                                            <Row>
                                                                <Col
```

```
className="align-middle text-center" xs={12} sm={6} md={8}>

<h4>{category.name}</h4>
                                                                    </Col>
                                                                    <Col xs={6}
sm={3} md={2}>
                                                                        <Button
onClick={() => this.setEditMode(category)} className = "btn btn-block
btn-custom" > Edit</Button>
                                                                    </Col>
                                                                    <Col xs={6}
sm={3} md={2}>
                                                                        <Button
onClick={() => this.ConfirmDelete(category)} className="btn btn-block
btn-danger">Delete</Button>
                                                                    </Col>
                                                                </Row>
                                                            </CardBody>
                                                        </Card>
                                                    </Col>
                                                </Row>
                                            )
                                        }
                                    </>
                                }
                                </div>
                                <ConfirmDelete/>
                            </div>

                        }
                        {
                            this.state.mode === 'create' && <div>
                                <CreateCategory />
                            </div>
                        }
                        {
                            this.state.mode === 'edit' && <div>
                                <EditCategory category={this.state.category} />
                            </div>
                        }
                    </>

            )
        }
}
```

4.8.4 CreateCategory.js Component

```
import { faTag, faUser } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import React, { Component,useState } from 'react';
import { Alert, Card, CardBody, CardHeader, Col, Form, Row } from
'reactstrap';
import CategoryServices from '../../../services/CategoryServices';

function CreateCategory(props) {
    const [category, SetCategory] = useState("");
    const [alertOpen, showAlert] = useState(false);
    const [alertMsg, setAlert] = useState('');

    const handleSubmit = (e) => {
        e.preventDefault();

        if (category.trim().length > 0) {
            CategoryServices.insertCategory({ name: category
}).then(function (resp) {
                console.log(resp);
                window.location.reload(false);
            }).catch(function (error) {

                console.log(error.response);
                setAlert(JSON.stringify(error.response.data.title));
                showAlert(true);
            });
        }
    }

    return (

        <div>
            <Row className="text-purple mt-5">
                <Col xs="12" sm={{ size: 8, offset: 2 }} md={{ size: 6,
offset: 3 }}>
                    <Card className="shadow-custom">
                        <CardHeader>
                            <Alert aria-live="polite" className="text-center
" color="danger" isOpen={alertOpen} toggle={() => showAlert(false)}>
                                {alertMsg}
                            </Alert>
```

```
                        </CardHeader>
                        <CardBody>
                            <Form onSubmit={handleSubmit}>
                                <div className=" form-group">

                                    <div className="input-group">
                                        <span>

                                            <FontAwesomeIcon icon={faTag} />
                                            <input type="text"
                                                defaultValue={category}
                                                required
                                                className="form-control"
                                                placeholder={"Category
Name"}

                                                onChange={(e) =>
SetCategory(e.target.value)}

                                            />

                                        </span>
                                    </div>
                                </div>
                                <div className=" form-group">
                                    <div className="custom">
                                        <button type={"submit"}
disabled={!(category.trim().length>0)} className="btn btn-block
btn-outline-primary" >{"Save Category"}</button>
                                    </div>
                                </div>
                            </Form>
                        </CardBody>
                    </Card>


                </Col>
            </Row>
        </div>

        )
}

export default CreateCategory;
```

4.8.5 EditCategory.js Component

```
import { faTag, faUser } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import React, { Component, useState } from 'react';
import { Alert, Card, CardBody, CardHeader, Col, Form, Row } from
'reactstrap';
import CategoryServices from '../../../services/CategoryServices';

function EditCategory(props) {
    const [category, SetCategory] = useState(props.category.name);
    const [alertOpen, showAlert] = useState(false);
    const [alertMsg, setAlert] = useState('');

    const handleSubmit = (e) => {
        e.preventDefault();

        if (category.trim().length > 0) {
            props.category.name = category;
            CategoryServices.updateCategory(props.category).then(function
(resp) {
                console.log(resp);
                window.location.reload(false);
            }).catch(function (error) {

                console.log(error.response);
                setAlert(JSON.stringify(error.response.data.title));
                showAlert(true);
            });
        }
    }

    return (

        <div>
            <Row className="text-purple mt-5">
                <Col xs="12" sm={{ size: 8, offset: 2 }} md={{ size: 6,
offset: 3 }}>
                    <Card className="shadow-custom">
                        <CardHeader>
                            <Alert aria-live="polite" className="text-center
" color="danger" isOpen={alertOpen} toggle={() => showAlert(false)}>
                                {alertMsg}
                            </Alert>
                        </CardHeader>
                        <CardBody>
                            <Form onSubmit={handleSubmit}>
```

```
                                Category ID: {props.category.categoryId}
                                <div className=" form-group">

                                    <div className="input-group">
                                        <span>

                                            <FontAwesomeIcon icon={faTag} />
                                            <input type="text"
                                                defaultValue={category}
                                                required
                                                className="form-control"
                                                placeholder={"Category
Name"}

                                                onChange={(e) =>
SetCategory(e.target.value)}

                                            />

                                        </span>
                                    </div>
                                </div>
                                <div className=" form-group">
                                    <div className="custom">
                                        <button type={"submit"}
disabled={!(category.trim().length > 0)} className="btn btn-block
btn-outline-primary" >{"Save Category"}</button>
                                    </div>
                                </div>
                            </Form>
                        </CardBody>
                    </Card>


                </Col>
            </Row>
        </div>

    )
}
export default EditCategory;
```

4.8.6 CreateMenuItem.js Component

```
import { faAlignJustify, faAllergies, faBiohazard, faDrumstickBite,
faEuroSign, faImage, faSeedling, faTag, faUser, faUtensils } from
'@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import React, { Component, useEffect, useState } from 'react';
import { Alert, Card, CardBody, CardHeader, Col, Form, Row, Button,
ButtonGroup } from 'reactstrap';
import CategoryServices from '../../../services/CategoryServices';
import MenuItemService from '../../../services/MenuItemService';
import { FaUtensilSpoon, MdShortText } from 'react-icons/all';
import FileService from '../../../services/FileService';

function CreateMenuItem(props) {
    const [alertOpen, showAlert] = useState(false);
    const [alertMsg, setAlert] = useState('');
    const [categories, setCategories] = useState([]);
    const [name, setName] = useState('');
    const [description, setDesc] = useState('');
    const [price, setPrice] = useState("");
    const [imgPath, setImgPath] = useState('');
    const [categoryId, setCategoryId] = useState('');
    const [allergens, setAllergens] = useState('');
    const [imgFile, setImageFile] = useState(null);
    const [availability, setAvailability] = useState(true);
    const [isVegan, setVegan] = useState(false);

    useEffect(() => {
        if (categories.length <= 0) {
            CategoryServices.getCategories().then(function (resp) {
                console.log(resp);
                setCategories(resp.data);
            }).catch(function (error) {
                console.log(error.response);
            });
        }
    });

    const validate = () => {

    }

    const handleSubmit = (e) => {
        e.preventDefault();

        let menuItem = {
```

```
            "name": name,
            "description": description,
            "price": price,
            "imgPath": imgPath,
            "availability": availability,
            "categoryId": categoryId,
            "allergens": allergens,
            "isVegan": isVegan
        }

        MenuItemService.insertMenuItem(menuItem).then(function (resp) {
            console.log(resp)
            window.location.reload(false);
        }).catch(function (error) {
            console.log(error.response);
        });
    }

    const processFile = (e) => {
        setImageFile(e.target.files[0]);
        FileService.postImage(e.target.files[0]).then(function (resp) {
            console.log(resp)
            setImgPath(resp.data.url)
        }).catch(function (error) {
            console.log(error.response);
        });
    }

    return (

        <div>
            <Row className="text-purple mt-3">
                <Col xs="12" sm={{ size: 8, offset: 2 }} md={{ size: 6,
offset: 3 }}>
                    <Card className="shadow-custom">
                        <CardHeader>
                            <Alert aria-live="polite" className="text-center
" color="danger" isOpen={alertOpen} toggle={() => showAlert(false)}>
                                {alertMsg}
                            </Alert>
                        </CardHeader>
                        <CardBody>
                            <Form onSubmit={handleSubmit}>
                                <div className=" form-group">

                                    <div className="input-group">
                                        <span>
```

```
                                        <FontAwesomeIcon
icon={faDrumstickBite} />
                                        <input type="text"
                                            defaultValue={name}
                                            required
                                            className="form-control"
                                            placeholder={"Name"}

                                            onChange={(e) =>
setName(e.target.value)}
                                        />

                                    </span>
                                </div>
                            </div>
                            <div className=" form-group">

                                <div className="input-group">
                                    <span className="align-items-start">

                                        <FontAwesomeIcon
icon={faAlignJustify} className="mt-2" />
                                        <textarea type="text"
                                            defaultValue={description}
                                            required
                                            className="form-control"
                                            placeholder={"Description"}
                                            onChange={(e) =>
setDesc(e.target.value)}

                                        ></textarea>

                                    </span>
                                </div>
                            </div>
                            <div className=" form-group">

                                <div className="input-group">
                                    <span>

                                        <FontAwesomeIcon
icon={faEuroSign} />
                                        <input type="number"
                                            defaultValue={price}
                                            required
                                            min="0"
                                            step="0.01"
                                            className="form-control "
```

```
                                                placeholder={"Price"}
                                                onChange={(e) =>
setPrice(e.target.value)}
                                            />
                                        </span>
                                    </div>
                                </div>
                                <div className="from-group">
                                    <div>
                                        <div className=" w-100 text-center">
                                            <label htmlFor={"img3"}
className="w-100">
                                                <div id={"input-div"}

className={imgPath.length > 0 ? "d-none text-purple rounded square
align-middle" : " text-purple rounded square align-middle"}>
                                                    <FontAwesomeIcon
icon={faImage} size={"lg"} />
                                                    <p className=" small
circle-p text-dark">Upload Image</p>
                                                </div>

                                                <div
className={imgPath.length > 0 ? "d-flex justify-content-center
align-items-center" :"d-none  justify-content-center align-items-centerz" }
>
                                                    <img src={imgPath.length
> 0 ? imgPath : ""} className={imgPath.length > 0 ? " rounded d-block" :
"d-none rounded"}
                                                        style={{ height:
'200px',width: '100%', objectFit: "cover" }}
                                                    />
                                                </div>

                                            </label>
                                            <input type='file' required
onChange={processFile} className="d-none"
                                                id={"img3"}
accept={"image/*"} />

                                        </div>
                                    </div>
                                </div>
                                <div className=" form-group">

                                    <div className="input-group">
                                        <span>
                                            <FontAwesomeIcon
```

```
icon={faUtensils}/>

                                            <ButtonGroup required
className={"custom btn-block mr-2 ml-2 rounded"}>
                                                <Button role="radio"
className="btn btn-outline-primary border" onClick={() =>
setAvailability(false)} active={availability === false}>Not
Available</Button>
                                                <Button role="radio"
className="btn btn-outline-primary border" onClick={() =>
setAvailability(true)} active={availability === true}>Available</Button>

                                            </ButtonGroup>

                                        </span>
                                    </div>
                                </div>
                                <div className=" form-group">

                                    <div className="input-group">
                                        <span>
                                            <FontAwesomeIcon
icon={faSeedling} />

                                            <ButtonGroup required
className={"custom btn-block mr-2 ml-2 rounded"}>
                                                <Button role="radio"
className="btn btn-outline-primary border" onClick={() => setVegan(false)}
active={isVegan === false}>Non Veg</Button>
                                                <Button role="radio"
className="btn btn-outline-primary border" onClick={() => setVegan(true)}
active={isVegan === true}>Veg</Button>

                                            </ButtonGroup>

                                        </span>
                                    </div>
                                </div>
                                <div className=" form-group">

                                    <div className="input-group">
                                        <span>

                                            <FontAwesomeIcon
icon={faBiohazard} />
                                            <input type="text"
                                                defaultValue={allergens}
                                                required
```

```
                                                    className="form-control"
                                                    placeholder={"Allergens"}

                                                    onChange={(e) =>
setAllergens(e.target.value)}
                                                />

                                            </span>
                                        </div>
                                    </div>
                                    <div className=" form-group">

                                        <div className="input-group">
                                            <span>
                                                <FontAwesomeIcon icon={faTag} />

                                                <select onChange={(e) =>
setCategoryId(e.target.value)} defaultValue={categoryId} required
class="form-control text-purple">
                                                    <option value="">Select
Category</option>
                                                    {

categories.map((category) =>
                                                        <option
value={category.categoryId}>{category.name}</option>
                                                        )
                                                    }
                                                </select>

                                            </span>
                                        </div>
                                    </div>

                                    <div className=" form-group">
                                        <div className="custom">
                                            <button type={"submit"}
disabled={!(name.trim().length > 0
                                                && description.trim().length > 0
                                                && (price != 0 && price != "")
                                                && imgPath.trim().length > 0
                                                && categoryId != ""
                                                && allergens.trim().length > 0)}
className="btn btn-block btn-outline-primary" >{"Save Menu Item"}</button>
                                        </div>
                                    </div>
                                </Form>
                            </CardBody>
```

```
                    </Card>


                </Col>
            </Row>
        </div>
    )
}

export default CreateMenuItem;
```

4.8.7 EditMenuItem.js Component

```
import { faAlignJustify, faBiohazard, faDrumstickBite, faEuroSign, faImage,
faSeedling, faTag, faUser, faUtensils } from
'@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import React, { Component, useEffect, useState } from 'react';
import { Alert, Card, CardBody, CardHeader, Col, Form, Row, Button,
ButtonGroup } from 'reactstrap';
import CategoryServices from '../../../services/CategoryServices';
import MenuItemService from '../../../services/MenuItemService';
import { FaUtensilSpoon, MdShortText } from 'react-icons/all';
import FileService from '../../../services/FileService';

function EditMenuItem(props) {
    const [alertOpen, showAlert] = useState(false);
    const [alertMsg, setAlert] = useState('');
    const [categories, setCategories] = useState([]);
    const [name, setName] = useState(props.menu.name);
    const [description, setDesc] = useState(props.menu.description);
    const [price, setPrice] = useState(props.menu.price);
    const [imgPath, setImgPath] = useState(props.menu.imgPath);
    const [categoryId, setCategoryId] = useState(props.menu.categoryId);
    const [imgFile, setImgeFile] = useState(null);
    const [isVegan, setVegan] = useState(props.menu.isVegan);
    const [availability, setAvailability] =
useState(props.menu.availability);
    const [allergens, setAllergens] = useState(props.menu.allergens);

    useEffect(() => {
        if (categories.length <= 0) {
            CategoryServices.getCategories().then(function (resp) {
                console.log(resp);
                setCategories(resp.data);
```

```
            }).catch(function (error) {
                console.log(error.response);
            });
        }
    });

    const validate = () => {

    }

    const handleSubmit = (e) => {
        e.preventDefault();

        let menuItem = props.menu;
        menuItem.name = name;
        menuItem.description = description;
        menuItem.price = price;
        menuItem.imgPath = imgPath;
        menuItem.availability = availability;
        menuItem.categoryId = categoryId;
        menuItem.allergens = allergens;
        menuItem.menuCategory = null;
        menuItem.isVegan = isVegan;

        MenuItemService.updateMenuItem(menuItem).then(function (resp) {
            console.log(resp)
            window.location.reload(false);
        }).catch(function (error) {
            console.log(error.response);
        });
    }

    const processFile = (e) => {
        setImgeFile(e.target.files[0]);
        FileService.postImage(e.target.files[0]).then(function (resp) {
            console.log(resp)
            setImgPath(resp.data.url)
        }).catch(function (error) {
            console.log(error.response);
        });
    }

    return (

        <div>
            <Row className="text-purple mt-3">
                <Col xs="12" sm={{ size: 8, offset: 2 }} md={{ size: 6,
offset: 3 }}>
```

```
                    <Card className="shadow-custom">
                        <CardHeader>
                            <Alert aria-live="polite" className="text-center
" color="danger" isOpen={alertOpen} toggle={() => showAlert(false)}>
                                {alertMsg}
                            </Alert>
                        </CardHeader>
                        <CardBody>
                            <Form onSubmit={handleSubmit}>
                                <div className=" form-group">

                                    <div className="input-group">
                                        <span>

                                            <FontAwesomeIcon
icon={faDrumstickBite} />
                                            <input type="text"
                                                defaultValue={name}
                                                required
                                                className="form-control"
                                                placeholder={"Name"}

                                                onChange={(e) =>
setName(e.target.value)}
                                            />
                                        </span>
                                    </div>
                                </div>
                                <div className=" form-group">

                                    <div className="input-group">
                                        <span className="align-items-start">

                                            <FontAwesomeIcon
icon={faAlignJustify} className="mt-2" />
                                            <textarea type="text"
                                                defaultValue={description}
                                                required
                                                className="form-control"
                                                placeholder={"Description"}
                                                onChange={(e) =>
setDesc(e.target.value)}

                                            ></textarea>

                                        </span>
                                    </div>
                                </div>
```

```
                                <div className=" form-group">

                                    <div className="input-group">
                                        <span>

                                            <FontAwesomeIcon
icon={faEuroSign} />
                                            <input type="number"
                                                defaultValue={price}
                                                required
                                                min="0"
                                                step="0.01"
                                                className="form-control "
                                                placeholder={"Price"}
                                                onChange={(e) =>
setPrice(e.target.value)}
                                            />
                                        </span>
                                    </div>
                                </div>
                                <div className="from-group">
                                    <div>
                                        <div className=" w-100 text-center">
                                            <label htmlFor={"img3"}
className="w-100">
                                                <div id={"input-div"}

className={imgPath.length > 0 ? "d-none text-purple rounded square
align-middle" : " text-purple rounded square align-middle"}>
                                                    <FontAwesomeIcon
icon={faImage} size={"lg"} />
                                                    <p className=" small
circle-p text-dark">Upload Image</p>
                                                </div>

                                                <div
className={imgPath.length > 0 ? "d-flex justify-content-center
align-items-center" : "d-none  justify-content-center align-items-centerz"}
>
                                                    <img src={imgPath.length
> 0 ? imgPath : ""} className={imgPath.length > 0 ? " rounded d-block" :
"d-none rounded"}
                                                        style={{ height:
'200px', width: '100%', objectFit: "cover" }}
                                                    />
                                                </div>

                                            </label>
```

```
                                            <input type='file'
onChange={processFile} className="d-none"
                                                id={"img3"}
accept={"image/*"} />

                                        </div>
                                    </div>
                                </div>
                                <div className=" form-group">

                                    <div className="input-group">
                                        <span>
                                            <FontAwesomeIcon
icon={faUtensils} />

                                            <ButtonGroup required
className={"custom btn-block mr-2 ml-2 rounded"}>
                                                <Button role="radio"
className="btn btn-outline-primary border" onClick={() =>
setAvailability(false)} active={availability === false}>Not
Available</Button>
                                                <Button role="radio"
className="btn btn-outline-primary border" onClick={() =>
setAvailability(true)} active={availability === true}>Available</Button>

                                            </ButtonGroup>

                                        </span>
                                    </div>
                                </div>
                                <div className=" form-group">

                                    <div className="input-group">
                                        <span>
                                            <FontAwesomeIcon
icon={faSeedling} />

                                            <ButtonGroup required
className={"custom btn-block mr-2 ml-2 rounded"}>
                                                <Button role="radio"
className="btn btn-outline-primary border" onClick={() => setVegan(false)}
active={isVegan === false}>Non Veg</Button>
                                                <Button role="radio"
className="btn btn-outline-primary border" onClick={() => setVegan(true)}
active={isVegan === true}>Veg</Button>

                                            </ButtonGroup>
```

```
                                        </span>
                                    </div>
                                </div>
                                <div className=" form-group">

                                    <div className="input-group">
                                        <span>

                                            <FontAwesomeIcon
icon={faBiohazard} />
                                            <input type="text"
                                                defaultValue={allergens}
                                                required
                                                className="form-control"
                                                placeholder={"Allergens"}

                                                onChange={(e) =>
setAllergens(e.target.value)}
                                            />
                                        </span>
                                    </div>
                                </div>

                                <div className=" form-group">

                                    <div className="input-group">
                                        <span>
                                            <FontAwesomeIcon icon={faTag} />

                                            <select onChange={(e) =>
setCategoryId(e.target.value)} defaultValue={categoryId} required
class="form-control text-purple">
                                                <option value="">Select
Category</option>
                                                {

categories.map((category) =>
                                                        <option
selected={categoryId == category.categoryId}
value={category.categoryId}>{category.name}</option>
                                                    )
                                                }
                                            </select>

                                        </span>
                                    </div>
                                </div>
```

```
                                <div className=" form-group">
                                    <div className="custom">
                                        <button type={"submit"}
disabled={!(name.trim().length > 0
                                                && description.trim().length > 0
                                                && (price != 0 && price != "")
                                                && imgPath.trim().length > 0
                                                && categoryId != ""
                                                && allergens.trim().length > 0)}
className="btn btn-block btn-outline-primary" >{"Save Menu Item"}</button>
                                    </div>
                                </div>
                            </Form>
                        </CardBody>
                    </Card>


                </Col>
            </Row>
        </div>
    )
}

export default EditMenuItem;
```

4.8.8 MenuItem.js Component

```
import { faEllipsisH, faEuroSign, faExclamationCircle } from
'@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { get } from 'jquery';
import React, { Component } from 'react';
import { Button, Card, CardBody, CardImg, Col, UncontrolledDropdown,
DropdownItem, DropdownMenu, DropdownToggle, Modal, ModalBody, ModalFooter,
ModalHeader, Row } from 'reactstrap';
import MenuItemService from '../../../services/MenuItemService';
import CreateMenuItem from './CreateMenuItem';
import EditMenuItem from './EditMenuItem';

export default class MenuItem extends Component {

    constructor(props) {
        super(props)

        this.state = {
```

```
            mode: 'index',
            menus: [],
            menu: {},
            isModalOpen: false,
        }

        this.getMenuItems = this.getMenuItems.bind(this);
        this.Delete = this.Delete.bind(this);
        this.ConfirmDelete = this.ConfirmDelete.bind(this);
    }

    componentDidMount() {
        this.getMenuItems();
    }

    getMenuItems() {
        let self = this;
        MenuItemService.getMenuItem().then(async function (resp) {

            console.log(resp);
            await self.setState({ menus: resp.data });
        }).catch(function (error) {
            console.log(error.response);
        });
    }

    async ConfirmDelete(menu) {
        await this.setState({ menu: menu });
        this.setState({ isModalOpen: true });
    }

    async Delete(menu) {

        MenuItemService.deleteMenuItem(menu.menuItemId).then(function (resp)
{
            console.log(resp);
            window.location.reload(false);
        }).catch(function (error) {
            console.log(error.response);
        });

    }

    async setEditMode(menu) {
        await this.setState({ menu: menu });
        await this.setState({ mode: 'edit' });
    }
```

```
    render() {

        const ConfirmDelete = () => {
            return (
                <Modal centered  returnFocusAfterClose={true}
isOpen={this.state.isModalOpen} className="modal-dialog-centered" >
                    <ModalHeader className="text-danger">
                        <FontAwesomeIcon icon={faExclamationCircle}
size={"lg"} /> Are you sure to delete this menu item?
                    </ModalHeader>
                    <ModalBody>

                        <Card className="shadow-custom">
                            <CardImg top src={this.state.menu.imgPath} />
                            <CardBody>
                                <h6><em
className="font-weight-bold">ID:</em> {this.state.menu.menuItemId}</h6>
                                <h6><em
className="font-weight-bold">Name:</em> {this.state.menu.name}</h6>
                                <h6><em
className="font-weight-bold">Price:</em>
{this.state.menu.price}<FontAwesomeIcon icon={faEuroSign} /></h6>
                                <h6><em
className="font-weight-bold">Availability:</em>
{this.state.menu.availability ? "Available" : "Unavailable"}</h6>
                                <h6><em className="font-weight-bold">Special
Dietary:</em> {this.state.menu.isVegan ? "Vegetarian" : "Non
Vegetarian"}</h6>
                                <h6><em
className="font-weight-bold">Category:</em> {this.state.menu.menuCategory ==
undefined ? "" : this.state.menu.menuCategory.name}</h6>
                                <h6><em
className="font-weight-bold">Description:</em>
{this.state.menu.description}</h6>
                                <h6><em
className="font-weight-bold">Allergens:</em>
{this.state.menu.allergens}</h6>
                            </CardBody>
                        </Card>
                    </ModalBody>
                    <ModalFooter>
                        <Button onClick={() => this.Delete(this.state.menu)}
className="btn  btn-danger">Yes</Button>
                        <Button onClick={() => this.setState({ isModalOpen:
false })} className="btn  btn-primary" > No</Button>
                    </ModalFooter>
                </Modal>
            )
```

```
        }
        return (
            <>
                {
                    this.state.mode === 'index' &&
                    <div className="row mt-4">
                        <div className="col-12">
                            {this.state.menus.length < 1 ?
                                <div className="row">
                                    <div className="col-12 h3">
                                        No menu items added yet
                                <Button className="btn btn-outline-success
float-right" onClick={() => this.setState({ mode: 'create' })}>Add menu
item</Button>
                                    </div>
                                </div> :
                                <>
                                    <div className="row">
                                        <div className="col-12 h3">
                                            Menu Items
                                        <Button className="btn
btn-outline-success float-right" onClick={() => this.setState({ mode:
'create' })}>Add menu item</Button>
                                        </div>
                                    </div>
                                    <Row className="mt-4">
                                        {
                                            this.state.menus.map((menu) =>

                                                <Col
key={Math.random().toString(36).substring(0)} xs={12} xs={12} sm={6 }
lg={4} className=" mb-4" >
                                                    <Card
className="shadow-custom h-100">
                                                        <CardImg top
src={menu.imgPath} />
                                                        <CardBody>

<UncontrolledDropdown>

<DropdownToggle nav className="text-right p-0">

<FontAwesomeIcon icon={faEllipsisH} size={"lg"} />

</DropdownToggle>

<DropdownMenu right>
```

```
<DropdownItem><span style={{ cursor: "pointer" }} onClick={() =>
this.setEditMode(menu)} className="btn btn-block btn-custom" >
Edit</span></DropdownItem>

<DropdownItem><span style={{ cursor: "pointer" }} onClick={() =>
this.ConfirmDelete(menu)} className="btn btn-block
btn-danger">Delete</span></DropdownItem>

</DropdownMenu>

</UncontrolledDropdown>
                                                                    <h6><em
className="font-weight-bold">Name:</em> {menu.name}</h6>
                                                                    <h6><em
className="font-weight-bold">Price:</em> {menu.price}<FontAwesomeIcon
icon={faEuroSign} /></h6>
                                                                    <h6><em
className="font-weight-bold">Availability:</em> {menu.availability ?
"Available" : "Unavailable"}</h6>
                                                                    <h6><em
className="font-weight-bold">Special Dietary:</em> {menu.isVegan ?
"Vegetarian" : "Non Vegetarian"}</h6>
                                                                    <h6><em
className="font-weight-bold">Category:</em> {menu.menuCategory.name}</h6>
                                                                    <h6><em
className="font-weight-bold">Description:</em> {menu.description}</h6>
                                                                    <h6><em
className="font-weight-bold">Allergens:</em> {menu.allergens}</h6>
                                                                </CardBody>
                                                            </Card>
                                                        </Col>
                                                    )
                                                    }
                                                </Row>
                                            </>
                                        }
                                    </div>

                                </div>
                            }
                            {
                                this.state.mode === 'create' && <div>
                                    <CreateMenuItem/>
                                </div>
                            }
                            {
                                this.state.mode === 'edit' && <div>
                                    <EditMenuItem menu={this.state.menu} />
```

```
                    </div>
                }
                <ConfirmDelete/>
            </>
        )
    }
}
```

4.8.9 SelectedItem.js Component

```
import { faTrashAlt } from '@fortawesome/free-regular-svg-icons';
import { faAlignJustify, faEuroSign, faTrash } from
'@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { timers } from 'jquery';
import React, { Component } from 'react'
import { Card, CardBody, CardImg, CardImgOverlay, Col, Modal, ModalBody,
ModalHeader, Row } from 'reactstrap';
import BraintreeService from '../../../services/BraintreeService';
import OrderService from '../../../services/OrderService';
import CreatePayment from '../Payment/CreatePayment';


export default class SelectedItem extends Component {
    constructor(props) {
        super(props);
        this.state = {
            selectedItems: {},
            totalPrice: 0,
            instruction: "",
            table: "",
            order: {
                selectedItems: {},
                totalPrice: 0,
                instruction: "",
                table: '',
                token:""
            },
            isOpen: false,
            payType:"",

        }

        this.changeQuantity = this.changeQuantity.bind(this);
        this.proceedToCheckOut = this.proceedToCheckOut.bind(this);
```

```
        this.toggleModal = this.toggleModal.bind(this);
        this.setInstruction = this.setInstruction.bind(this);
        this.removeItem = this.removeItem.bind(this);
    }

    async componentDidMount() {
        if (localStorage.getItem('selectedItems') != undefined) {
            this.setState({ selectedItems:
JSON.parse(localStorage.getItem('selectedItems')) });
            let total = 0;

Object.values(JSON.parse(localStorage.getItem('selectedItems'))).forEach((it
em, index) => {
                console.log(item.itemAmount);
                console.log(item.price);
                total += parseInt(item.itemAmount) * parseFloat(item.price);
            });
            total = total.toFixed(2);
            this.setState({ totalPrice: total });
        }
    }

    async setInstruction(key, instruction) {
        this.state.selectedItems[key].instruction = instruction;
        await this.setState({ selectedItems: this.state.selectedItems });
        localStorage.setItem('selectedItems',
JSON.stringify(this.state.selectedItems));
    }

    async removeItem(key) {
        let item = this.state.selectedItems[key];
        delete this.state.selectedItems[key];
        await this.setState({ selectedItems: this.state.selectedItems });
        let diff = -1 * parseInt(item.itemAmount)
        let total = parseFloat(this.state.totalPrice);
        total += parseInt(diff) * parseFloat(item.price);

        await this.setState({ totalPrice: total.toFixed(2) });
        localStorage.setItem('dishItemCount',
parseInt(localStorage.getItem('dishItemCount')) + diff);
        localStorage.setItem('selectedItems',
JSON.stringify(this.state.selectedItems));
        window.location.reload(false);
    }

    async changeQuantity(key, qty) {
        let diff = qty - this.state.selectedItems[key].itemAmount
        this.state.selectedItems[key].itemAmount = qty;
```

```
        await this.setState({ selectedItems: this.state.selectedItems });

        let total = parseFloat(this.state.totalPrice);
        total += parseInt(diff) *
parseFloat(this.state.selectedItems[key].price);

        this.setState({ totalPrice: total.toFixed(2) });

        localStorage.setItem('dishItemCount',
parseInt(localStorage.getItem('dishItemCount')) + diff);
        localStorage.setItem('selectedItems',
JSON.stringify(this.state.selectedItems));

        this.props.updateThis();
    }

    async proceedToCheckOut(e) {
        e.preventDefault();
        if (localStorage.getItem('dishItemCount') == 0) {
            alert("The quantity of selected items should be greater than
zero");
            return;
        }

        console.log(localStorage.getItem('pendingPayment'));

        let order = {
            selectedItems: this.state.selectedItems,
            totalPrice: this.state.totalPrice,
            instruction: this.state.instruction,
            table: this.state.table,
            token:""
        }

        if (this.state.payType == 'now') {
            let self = this;

            await BraintreeService.getToken().then(function (resp) {
                order.token = resp.data.token;
            }).catch(function (error) {
                console.log(error);
                console.log(error.response);
            });

            await this.setState({ order: order });
            console.log(this.state.order);
            this.toggleModal();
        }
```

```
        else {
            await this.setState({ order: order });
            console.log(this.state.payType);

            if (localStorage.getItem('pendingPayment') == null ||
localStorage.getItem('pendingPayment') == undefined) {
                let selectedOrderItems =
Object.values(this.state.selectedItems);
                let orderItems = [];

                for (let i = 0; i < selectedOrderItems.length; i++) {
                    console.log(selectedOrderItems[i]);

                    let selecteditem = selectedOrderItems[i];
                    if (selecteditem.itemAmount != 0) {
                        let orderItem = {
                            "itemAmount": parseInt(selecteditem.itemAmount),
                            "price": selecteditem.price,
                            "menuItemId": selecteditem.menuItemId,
                            "orderItemStatus": "Unserved",
                            "instruction": selecteditem.instruction
                        }
                        orderItems.push(orderItem);
                    }
                }

                let order = {
                    "total": this.state.totalPrice,
                    "instruction": this.state.instruction,
                    "tableIdentity": this.state.table,
                    "status": "Unpaid",
                    "orderItems": orderItems
                }

                console.log(order);

                OrderService.insertOrder(order).then(function (resp) {
                    localStorage.setItem('pendingPayment',
resp.data.orderId);
                    console.log(resp.data);
                    localStorage.removeItem("selectedItems");
                    localStorage.removeItem('dishItemCount');
                    let orderIds = (localStorage.getItem('orderIds') != null
|| localStorage.getItem('orderIds') != undefined) ?
localStorage.getItem('orderIds') : "";
                    console.log(orderIds);
                    orderIds += ' ' + resp.data.orderId;
                    localStorage.setItem('orderIds', orderIds);
```

```
                    window.location.href = "/orderHistory";
                }).catch(function (error) {
                    console.log(error);
                    console.log(error.response);
                });
            }
            else {
                let self = this;
                let selectedOrderItems =
Object.values(this.state.selectedItems);
                let orderItems = [];

                for (let i = 0; i < selectedOrderItems.length; i++) {
                    console.log(selectedOrderItems[i]);

                    let selecteditem = selectedOrderItems[i];
                    if (selecteditem.itemAmount != 0) {
                        let orderItem = {
                            "itemAmount": parseInt(selecteditem.itemAmount),
                            "price": selecteditem.price,
                            "menuItemId": selecteditem.menuItemId,
                            "orderItemStatus": "Unserved",
                            "instruction": selecteditem.instruction
                        }
                        orderItems.push(orderItem);

                    }
                }
                let order = {
                    "orderId": localStorage.getItem('pendingPayment'),
                    "total": this.state.totalPrice,
                    "instruction": this.state.instruction,
                    "tableIdentity": this.state.table,
                    "status": "Unpaid",
                    "orderItems": orderItems
                }
                console.log(order);
                OrderService.updateOrderToAddItem(order).then(function
(resp) {
                    console.log(resp.data);
                    localStorage.removeItem("selectedItems");
                    localStorage.removeItem('dishItemCount');
                    window.location.href = "/orderHistory";
                }).catch(function (error) {
                    console.log(error.response);
                });
            }
        }
```

```
    }

    toggleModal() {
        this.setState({ isOpen: !this.state.isOpen });
    }

    render() {

        const PaymentModal = () => {
            return (
                <Modal className={" rounded text-purple"}
isOpen={this.state.isOpen} centered scrollable backdrop="static"  >

                    <ModalHeader toggle={this.toggleModal}>
                        <div className="h3">Checkout</div>
                    </ModalHeader>
                <ModalBody style={{ overflowX: "hidden", overflowY:
"auto" }}>
                        <CreatePayment order={this.state.order} />
                </ModalBody>
            </Modal>
            )
        }

        return (
            <>
            {Object.keys(this.state.selectedItems).length < 1 &&
                <div>
                    No Menu Items Selected
                </div>
            }
            {Object.keys(this.state.selectedItems).length > 0 &&

                < Row className="mt-4">
                    <Col xs={12} md={6} lg={8}>
                    <h4>Your Order</h4>
                    <Row style={{ height: '350px', overflowX: "hidden",
overflowY: "auto" }}>
                                {

Object.keys(this.state.selectedItems).map((key) =>

                                        <Col
key={Math.random().toString(36).substring(0)} xs={12} xs={12} lg={6}
className=" mt-2 mb-2" >
                                            <Card  className="shadow-custom"
style={{ height: '200px' }}>
                                                <CardImg top
```

```
className="rounded" src={this.state.selectedItems[key].imgPath} style={{
height: '100%', objectFit: "cover" }} />
                                                <CardImgOverlay style={{
backgroundColor: 'rgba(255,255,255,.8)' }} className="pt-2">
                                                    <div>
                                                        <h6><em
className="font-weight-bold">Name:</em>
{this.state.selectedItems[key].name}</h6>
                                                        <button style={{
float: 'right', marginTop: '-20px' }} onClick={() => this.removeItem(key)}
title='Remove Item' className="btn btn-custom btn-sm"><FontAwesomeIcon
icon={faTrash} /></button>
                                                    </div>
                                                    <h6><em
className="font-weight-bold">Price:</em>
{this.state.selectedItems[key].price}<FontAwesomeIcon icon={faEuroSign} />{'
* ' + this.state.selectedItems[key].itemAmount + ' = ' +
(this.state.selectedItems[key].itemAmount *
this.state.selectedItems[key].price).toFixed(2)}<FontAwesomeIcon
icon={faEuroSign} /></h6>
                                                    <h6><em
className="font-weight-bold">Category:</em>
{this.state.selectedItems[key].menuCategory.name}</h6>
                                                    <form>
                                                        <div className="">
                                                            <div
className="input-group">
                                                                <span>

Quantity:
                                                                    <input

defaultValue={this.state.selectedItems[key].itemAmount}

type="number"

min={0}

required

className="form-control bg-transparent qty"


placeholder={"Quantity"}

onMouseLeave={(e) => this.changeQuantity(key, e.target.value)}

onBlur={(e) => this.changeQuantity(key, e.target.value)}
```

```
                                                                    />
                                                                </span>
                                                            </div>
                                                        </div>
                                                        <div className=" ">
                                                            <div
className="input-group">
                                                                <span
className="align-items-start">

<textarea type="text"

defaultValue={this.state.selectedItems[key].instruction}

className="form-control bg-transparent"

style={{ maxHeight: "35px", marginLeft: "-0.9rem" }}

placeholder={"Instruction"}

onMouseLeave={(e) => this.setInstruction(key, e.target.value)}

onBlur={(e) => this.setInstruction(key, e.target.value)}

></textarea>
                                                                </span>
                                                            </div>
                                                        </div>
                                                    </form>
                                                </CardImgOverlay>
                                            </Card>
                                        </Col>
                                    )
                                }
                            </Row>
                        </Col>
                        <Col xs={12} md={6} lg={4}>
                            <form onSubmit={this.proceedToCheckOut}>

                                <Card className="mt-md-5 ">
                                    <CardBody>
                                    <table className="table text-purple">
                                            <tbody>
                                                <tr>
                                                    <td>
                                                        Total
                                                    </td>
                                                    <td
```

```
className="text-center">

{this.state.totalPrice} <FontAwesomeIcon icon={faEuroSign} />
                                                    </td>
                                                </tr>
                                                <tr>
                                                    <td colSpan="2">
                                                        <div
className="mb-2">
                                                            Special
Instructions
                                                        <textarea
className='form-control' onChange={(e) => this.setState({ instruction:
e.target.value })}></textarea>
                                                        </div>
                                                        <div>
                                                            Table
                                                        <input type="text"
className='form-control' required onChange={(e) => this.setState({ table:
e.target.value })} />
                                                        </div>
                                                    </td>
                                                </tr>
                                            </tbody>
                                        </table>
                                        <div className="text-center form-group
align-middle">
                                            <button onClick={() =>
this.setState({ payType: "now" })} role="submit" className="btn btn-custom"
>Proceed to checkout</button>
                                        </div>
                                        <div className="text-center
align-middle">
                                            <button onClick={() =>
this.setState({ payType:"later" })} role="submit" className="btn
btn-warning" >Place Order & Pay Later</button>
                                        </div>
                                        </CardBody>
                                    </Card>
                                </form>
                            </Col>
                        </Row>
                    }
                    <PaymentModal />
                </>
            )
        }
```

```
}
```

4.8.10 Order.js Component

```
import React, { Component } from 'react';
import { Badge, Card, CardBody, CardHeader, CardImg, CardImgOverlay,
CardText, CardTitle, Col, Row } from 'reactstrap';
import OrderService from '../../../services/OrderService';

class Orders extends Component {
    static displayName = Orders.name;
    constructor(props) {
        super(props);
        this.state = {
            served: [],
            current: []
        }
        this.getOrders = this.getOrders.bind(this);
        this.serveOrder = this.serveOrder.bind(this);
    }

    componentDidMount() {
        this.getOrders();
    }
    async serveOrder(order) {
        order.status = "Served";
        OrderService.updateOrder(order).then(function (resp) {
            console.log(resp.data);
        }).catch(function (error) {
            console.log(error.response);
        });
        this.forceUpdate();
    }

    async getOrders() {
        let self = this;
        this.state.current.length = 0;
        await OrderService.getOrders().then(function (resp) {
            console.log(resp.data);
            let orders = resp.data;
            for (let i = 0; i < orders.length; i++) {

                if (orders[i].status != "Served") {

                    self.state.current.push(orders[i]);
                }
```

```
            }
        }).catch(function (error) {
            console.log(error);
        });

        await this.setState({ current: this.state.current });
        //console.log(this.state.current.length);
        setTimeout(() => this.getOrders(), 5000);
    }

    render() {
        return (
            <div className="text-purple">
                <Row>
                    <Col>
                        <Card className="mt-3">
                            <CardHeader className="h3">
                                Current Orders
                            </CardHeader>
                            <CardBody>
                                {this.state.current.length == 0 ? "There is
no orders currently!!" :

                                    <Row>
                                        {
                                            this.state.current.map(order =>
                                                <Col
key={Math.random().toString(36).substring(0)} xs={12} className=" mb-4">
                                                    <Card>
                                                        <CardHeader>
                                                            <div
className="row">
                                                                <div
className="col-6">

<em>Table:</em> {order.tableIdentity}<br />

{(order.instruction != null && order.instruction.length > 0) &&

<><em>Instruction:</em> {order.instruction}<br /></>
                                                                }

<em>Status:</em> {order.status}
                                                                </div>
                                                                <div
className="col-6 align-middle custom d-flex justify-content-end
align-content-center">
                                                                    <button
```

```
onClick={() => this.serveOrder(order)} className="btn
btn-outline-primary">Mark as served</button>
                                                                    </div>
                                                                </div>
                                                            </CardHeader>
                                                            <CardBody>
                                                                <Row>
                                                                    {

order.orderItems.map(orderItem =>
                                                                                                    <Col
key={Math.random().toString(36).substring(0)} xs={12} sm={6} lg={4}
className=" mb-2 mt-2">

<Card className="h-100" style={{ maxHeight: "200px" }}>

<CardImg src={orderItem.menuItem.imgPath} className="h-100" style={{
objectFit: "cover" }} />

<CardImgOverlay style={{ backgroundColor: 'rgba(255,255,255,.6)' }}
className='p-1 text-center'>

<p className="p-0 m-0">{orderItem.menuItem.name}</p>

<p className="p-0 m-0">Quantity:  {orderItem.itemAmmount}</p>

{orderItem.instruction != null > 0 &&

<p className="p-0 m-0">Instruction:  {orderItem.instruction}</p>

}


</CardImgOverlay>

</Card>

</Col>
                                                                                                        )
                                                                    }
                                                                </Row>
                                                            </CardBody>
                                                        </Card>
                                                    </Col>
                                                )
                                            }
                                        </Row>
                                    }
```

```
                            </CardBody>
                        </Card>
                    </Col>
                </Row>
                {
                }
            </div>
        );
    }
}

export default Orders;
```

4.8.11 OrderHistory.js Component

```
import React, { Component } from "react";
import OrderService from "../../../services/OrderService";
import { Badge, Button, Card, CardBody, CardHeader, CardImg, CardImgOverlay,
CardText, CardTitle, Col, Modal, ModalBody, ModalFooter, ModalHeader, Row }
from 'reactstrap';
import DuePayment from "../Payment/DuePayment";
import { Link } from "react-router-dom";
import BraintreeService from "../../../services/BraintreeService";
import CreateReview from "../Review/CreateReview";

class OrderHistory extends Component {

    constructor(props) {
        super(props);
        this.state = {
            served: [],
            current: [],
            isOpen: false,
            isFeedbackOpen: false,
            order: {
                total: 0,
                orderId: 0,
                token: ""

            },
            orderItem: {
                orderItemId: 0,
                itemAmmount: 0,
```

```
                price: 0,
                orderItemStatus: "string",
                isReviewed: true,
                isLoginOrRegistrationModalOpen : false,
                menuItemId: 0,
                menuItem: {
                    menuItemId: 0,
                    name: "string",
                    description: "string",
                    price: 0,
                    avgRate: 0,
                    imgPath: "string",
                    availability: true,
                    allergens: "string",
                    categoryId: 0,
                    menuCategory: {
                        categoryId: 0,
                        name: "string"
                    }
                }
            }
        }
        this.getPreviousOrders = this.getPreviousOrders.bind(this);
        this.payNow = this.payNow.bind(this);
        this.toggleModal = this.toggleModal.bind(this);
        this.toggleFeedbackModal = this.toggleFeedbackModal.bind(this);
        this.reviewItem = this.reviewItem.bind(this);
        this.toggleOnSubmitFeedback =
this.toggleOnSubmitFeedback.bind(this);
    }

    componentDidMount() {
        this.getPreviousOrders();
    }

    async getPreviousOrders() {
        let self = this;

OrderService.getOrderByIds(localStorage.getItem("orderIds")).then(function
(resp) {
            console.log(resp.data);
            let orders = resp.data;
            for (let i = 0; i < orders.length; i++) {

                if (orders[i].status != "Served") {

                    self.state.current.push(orders[i]);
```

```
                }
                else if (orders[i].status == "Served") {
                    self.state.served.push(orders[i]);
                }
            }
            self.setState({ served: self.state.served });
            self.setState({ current: self.state.current });
        }).catch(function (error) {
            console.log(error.response);
        });
    }

    async payNow(order) {
        console.log(order);
        let self = this;

        await BraintreeService.getToken().then(function (resp) {
            order.token = resp.data.token;
        }).catch(function (error) {
            console.log(error);
            console.log(error.response);
        });
        await this.setState({
            order: order
        });
        this.toggleModal();
    }

    async reviewItem(orderItem) {
        await this.setState({ orderItem: orderItem });
        this.toggleFeedbackModal();

    }

    toggleModal() {
        this.setState({ isOpen: !this.state.isOpen });
    }
    toggleFeedbackModal() {
        this.setState({ isFeedbackOpen: !this.state.isFeedbackOpen });

    }

    toggleOnSubmitFeedback() {
        this.getPreviousOrders();
        this.setState({ isFeedbackOpen: !this.state.isFeedbackOpen });
        console.log(localStorage.getItem('loggedIn'));
        if (localStorage.getItem('loggedIn') != 'true') {
```

```
            this.setState({ isLoginOrRegistrationModalOpen:
!this.state.isLoginOrRegistrationModalOpen });
        }
    }

    render() {

        const PaymentModal = () => {
            return (
                <Modal className={" rounded text-purple"}
isOpen={this.state.isOpen} centered scrollable backdrop="static"  >


                    <ModalHeader toggle={this.toggleModal}>
                        <div className="h3">Pay due</div>
                    </ModalHeader>
                    <ModalBody style={{ overflowX: "hidden", overflowY:
"auto" }}>
                        <DuePayment order={this.state.order} />
                    </ModalBody>

                </Modal>
            )
        }

        const FeedbackModal = () => {
            return (
                <Modal className={" rounded text-purple"}
isOpen={this.state.isFeedbackOpen} centered scrollable backdrop="static"  >


                    <ModalHeader toggle={this.toggleFeedbackModal}>
                        <div className="h3">Your Feedback</div>
                    </ModalHeader>
                    <ModalBody style={{ overflowX: "hidden", overflowY:
"auto" }}>
                        <CreateReview orderItem={this.state.orderItem}
toggle={this.toggleOnSubmitFeedback} />
                    </ModalBody>

                </Modal>
            )
        }

        const LoginOrRegistrationModal = () => {
            return (
                <Modal className={" rounded text-purple"}
isOpen={this.state.isLoginOrRegistrationModalOpen} toggle={() =>
```

```
this.setState({ isLoginOrRegistrationModalOpen:
!this.state.isLoginOrRegistrationModalOpen })} centered scrollable >
                    <ModalHeader
toggle={this.isLoginOrRegistrationModalOpen}>
                        <h3> </h3>
                    </ModalHeader>
                    <ModalBody style={{ overflowX: "hidden", overflowY:
"auto" }}>
                        <a href="/signin" className="btn btn-custom
text-white btn-block">Sign In</a>
                        <div className="text-center textf-purple">
                            -- OR --
                        </div>
                        <a href="/signup" className="btn btn-custom
text-white btn-block">Create an account</a>
                    </ModalBody>
                    <ModalFooter className="text-left">
                        <button onClick={() => this.setState({
isLoginOrRegistrationModalOpen: !this.state.isLoginOrRegistrationModalOpen
})} className="btn btn-danger">Close</button>
                    </ModalFooter>

                </Modal>
            )
        }

        return (

            <div className="text-purple ">
                <Row>
                    <Col>
                        <Card className="mt-3">
                            <CardHeader className="h3">
                                Current Orders
                            </CardHeader>
                            <CardBody>
                                {this.state.current.length == 0 ? "There is
no orders currently!!" :
                                    <Row>
                                        {
                                            this.state.current.map(order =>
                                                <Col
key={Math.random().toString(36).substring(0)} xs={12} className=" mb-4">
                                                    <Card>
                                                        <CardHeader>
                                                            <div
className="row">
                                                                <div
```

```
className="col-6">

<em>Table:</em> {order.tableIdentity}<br />

{(order.instruction != null && order.instruction.length > 0) &&

<><em>Instruction:</em> {order.instruction}<br /></>
                                                                    }

<em>Status:</em> {order.status}

                                                                </div>

{order.status == "Unpaid" && <div className="col-6 align-middle custom
d-flex justify-content-end align-content-center">
                                                                    <Link
tag={Link} to="/" className="align-middle d-flex justify-content-end
align-content-center text-decoration-none pr-2"><button className="btn
btn-outline-primary">Add More Items</button> </Link>
                                                                    <button
onClick={() => this.payNow(order)} className="btn btn-outline-primary">Pay
Now</button>
                                                                </div>}

                                                            </div>
                                                        </CardHeader>
                                                        <CardBody>
                                                            <Row>
                                                                {

order.orderItems.map(orderItem =>
                                                                    <Col
key={Math.random().toString(36).substring(0)} xs={12} sm={6} lg={4}
className=" mb-2 mt-2">

<Card className="h-100" style={{ maxHeight: "200px" }}>

<CardImg src={orderItem.menuItem.imgPath} className="h-100" style={{
objectFit: "cover" }} />

<CardImgOverlay style={{ backgroundColor: 'rgba(255,255,255,.6)' }}
className='p-1 text-center'>

<p className="p-0 m-0">{orderItem.menuItem.name}</p>

<p className="p-0 m-0">Quantity:  {orderItem.itemAmmount}</p>

{orderItem.instruction != null &&
```

```
<p className="p-0 m-0">Instruction:  {orderItem.instruction}</p>

}

</CardImgOverlay>

</Card>

</Col>
                                                                    )
                                                                }
                                                            </Row>
                                                        </CardBody>
                                                    </Card>
                                                </Col>
                                            )
                                        }
                                    </Row>
                                }
                            </CardBody>
                        </Card>
                    </Col>
                </Row>

                <Row>
                    <Col>
                        <Card className="mt-2">
                            <CardHeader className="h3">
                                Served Orders
                                    </CardHeader>
                            <CardBody>
                                {this.state.served.length == 0 ? "No order
is served yet!!" :

                                    <Row>

                                        {this.state.served.map(order =>
                                            <Col
key={Math.random().toString(36).substring(0)} xs={12} className=" mb-4">
                                                <Card>
                                                    <CardHeader>
                                                        <div
className="row">
                                                            <div
className="col-6">

<em>Table:</em> {order.tableIdentity}<br />
```

```
{(order.instruction != null && order.instruction.length > 0) &&

<><em>Instruction:</em> {order.instruction}<br /></>
                                                                    }

<em>Status:</em> {order.status}

                                                                </div>
                                                                <div
className="col-6 align-middle custom d-flex justify-content-end
align-content-center">
                                                                    <div
className="display-4"><Badge color="success">Served</Badge></div>
                                                                </div>
                                                            </div>
                                                        </CardHeader>
                                                        <CardBody>
                                                            <Row>
                                                                {

order.orderItems.map(orderItem =>
                                                                    <Col
key={Math.random().toString(36).substring(0)} xs={12} sm={6} lg={4}
className=" mb-2 mt-2">

<Card className="h-100" style={{ maxHeight: "200px" }}>

<CardImg src={orderItem.menuItem.imgPath} className="h-100" style={{
objectFit: "cover" }} />

<CardImgOverlay className='text-center'>

<CardTitle className="bg-light mt-auto
mb-auto">{orderItem.menuItem.name}</CardTitle>

<CardText className="bg-light">Quantity:  {orderItem.itemAmmount}</CardText>

{orderItem.isReviewed != true &&

<Button onClick={() => this.reviewItem(orderItem)} className={'btn
btn-success'} >Give feedback</Button>

}

</CardImgOverlay>

</Card>
```

```
                                                                    </Col>
                                                                )
                                                            }
                                                        </Row>
                                                    </CardBody>
                                                </Card>
                                            </Col>
                                        )
                                        }
                                    </Row>
                                }
                            </CardBody>
                        </Card>
                    </Col>
                </Row>
                <PaymentModal />
                <FeedbackModal />
                <LoginOrRegistrationModal />
            </div>
        );
    }
}

export default OrderHistory;
```

4.8.12 CreatePayment.js Component

```
import { faCalendarAlt, faCreditCard, faEuroSign, faInfoCircle, faShieldAlt,
faUser, } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import React, { Component, useEffect, useState } from 'react';
import { Alert, Button, Col, Form, Modal, ModalBody, ModalFooter,
ModalHeader } from 'reactstrap';
import SelectedItem from '../MenuItem/selectedItem';
import { formatCreditCardNumber, formatCVC, formatExpirationDate,
formatFormData } from '../../../util/utils'
import PaymentService from '../../../services/PaymentService';
import BraintreeService from '../../../services/BraintreeService';
import $ from 'jquery';


const CreatePayment = (props) => {
    const [selectedItems, setSelectedItems] =
useState(props.order.selectedItems);
```

```
    const [total, setTotal] = useState(props.order.totalPrice);
    const [instruction, setInstruction] = useState(props.order.instruction);
    const [cardHolderName, setCardHolder] = useState("");
    const [cardNumber, setCardNumber] = useState("");
    const [cardCVC, setCardCVC] = useState("");
    const [cardExpireDate, setExpireDate] = useState("");
    const [table, setTable] = useState(props.order.table);
    const [isOpen, toggleModal] = useState(false);
    const [visible, setVisible] = useState(false);
    const [msg, setMsg] = useState("");
    const [token, setToken] = useState(props.order.token);
    const [dData, setDeviceData] = useState(null);

    useEffect(() => {

        renderForm();
    }, [])

    const renderForm = async () => {
        var authToken = token;
        var form = document.querySelector('#paymentForm');
        var submit = document.querySelector('input[type="submit"]');


        async function getDeviceData(err, dataCollectorInstance) {
            if (err) {
                // Handle error in creation of data collector
                return;
            }
            // At this point, I should access the
dataCollectorInstance.deviceData value and provide it
            // to the server, e.g. by injecting it into your form as a hidden
input.
            var deviceData = dataCollectorInstance.deviceData;


            window.braintree.client.create({
                authorization: authToken,
            }, function (clientErr, clientInstance) {
                if (clientErr) {
                    console.error(clientErr);
                    return;
                }

                // This example shows Hosted Fields, but you can also use
this
                // client instance to create additional components here, such
as
```

```
                // PayPal or Data Collector.

                window.braintree.hostedFields.create({
                    client: clientInstance,
                    styles: {
                        'input': {
                            'font-size': '14px'
                        },
                        'input.invalid': {
                            'color': 'red'
                        },
                        'input.valid': {
                            'color': 'green'
                        }
                    },
                    fields: {
                        cardholderName: {
                            selector: '#cc-name',
                            placeholder: 'Name as it appears on your card'
                        },
                        number: {
                            selector: '#card-number',
                            placeholder: '4111 1111 1111 1111'
                        },
                        cvv: {
                            selector: '#cvv',
                            placeholder: '123'
                        },
                        expirationDate: {
                            selector: '#expiration-date',
                            placeholder: '10/2022'
                        }
                    }
                },
                    function (hostedFieldsErr, hostedFieldsInstance) {
                        if (hostedFieldsErr) {
                            console.error(hostedFieldsErr);
                            return;
                        }

                        submit.removeAttribute('disabled');

                        form.addEventListener('submit', function (event) {
                            event.preventDefault();

                            var formIsInvalid = false;
                            var state = hostedFieldsInstance.getState();
```

```
                            Object.keys(state.fields).forEach(function
(field) {
                                if (!state.fields[field].isValid) {

$(state.fields[field].container).addClass('is-invalid');
                                    formIsInvalid = true;
                                }
                            });

                            if (formIsInvalid) {
                                // skip tokenization request if any fields
are invalid
                                alert("Card input is not valid");
                                return;
                            }
                            hostedFieldsInstance.tokenize({
                                cardholderName: $('#cc-name').val()
                            },

                                function (tokenizeErr, payload) {
                                    if (tokenizeErr) {
                                        console.error(tokenizeErr);
                                        return;
                                    }

                                    console.log(deviceData);
                                    console.log('Got a nonce: ' +
payload.nonce);

                                    BraintreeService.payment({
                                        "correlation_id":
JSON.parse(deviceData).correlation_id,
                                        "nonce": payload.nonce,
                                        "amount": total
                                    }).then(function (resp) {
                                        console.log(resp.data);
                                        var result = resp.data.result;
                                        makePayment(result);
                                    }).catch(function (error) {
                                        alert("Transaction failed");
                                        console.log(error.response);
                                    })

                                });
                        }, false);
                    });
            });
        }
```

```
        await window.braintree.client.create({
            authorization: authToken
        }, async function (err, clientInstance) {
            // Creation of any other components...
            await window.braintree.dataCollector.create({
                client: clientInstance,
                hostedFields: true
            }, (err, dataCollectorInstance) => getDeviceData(err,
dataCollectorInstance));
        });
    }
    const handleSubmit = (event) => {
        event.preventDefault();
        toggleModal(true);
    }

    const makePayment = (result) => {

        toggleModal(false);
        let orderItems = [];
        //console.log(selectedItems);
        //console.log(cardNumber.split(" ").join(""));
        //console.log(cardCVC);
        //console.log(cardExpireDate);

        let selectedOrderItems = Object.values(selectedItems);

        for (let i = 0; i < selectedOrderItems.length; i++) {
            console.log(selectedOrderItems[i]);

            let selecteditem = selectedOrderItems[i];
            if (selecteditem.itemAmount != 0) {
                let orderItem = {
                    "itemAmount": parseInt(selecteditem.itemAmount),
                    "price": selecteditem.price,
                    "menuItemId": selecteditem.menuItemId,
                    "instruction": selecteditem.instruction
                }
                orderItems.push(orderItem);
            }

            console.log(orderItems);
        }
        var card = result.target.creditCard;

        let payData = {
            "cardHolderName": card.cardholderName,
```

```
            "amount": parseFloat(total),
            "cardNumber": card.maskedNumber,
            "expireDate": card.expirationDate,
            "cvv": cardCVC,
            "order": {

                "total": parseFloat(total),
                "instruction": instruction,
                "tableIdentity": table,
                "status": "Paid",
                "orderItems": orderItems
            }
        }

        console.log(payData);
        PaymentService.insertPayment(payData).then(function (resp) {
            console.log(resp);
            setMsg(<><div>Order placed successfully!!</div>
                <div>Your order will be served in a while!!</div>
                <div><a href="/" className="text-decoration-none text-purple
font-weight-bold">Go back to menu items</a></div></>);
            setVisible(true);
            let orderIds = localStorage.getItem('orderIds') != undefined ?
localStorage.getItem('orderIds') : "";

            orderIds += ' ' + resp.data.order.orderId;
            localStorage.setItem('orderIds', orderIds)

            localStorage.removeItem("selectedItems");
            localStorage.removeItem("dishItemCount");

        }).catch(function (error) {
            console.log(error.response);
        });

    }
    const ConfirmModal = () => {
        return (

            <Modal className={" rounded text-purple"} isOpen={isOpen}
centered scrollable backdrop="static"  >
                <div className="shadow-custom">

                    <ModalHeader toggle={() => toggleModal(!isOpen)}>
                        <div className="h3">Confirm Payment</div>
                    </ModalHeader>
                    <ModalBody>
                        <em>Total: </em>{total}<FontAwesomeIcon
```

```
icon={faEuroSign} /><br />
                        <em>Number: </em>{cardNumber}<br />
                        <em>Name: </em>{cardHolderName}<br />
                        <em>Expiry: </em>{cardExpireDate}<br />
                        <em>CVC: </em>{cardCVC}
                    </ModalBody>
                    <ModalFooter className="text-right">
                        <button onClick={() => makePayment()} className="btn
btn-custom" >Pay</button>
                    </ModalFooter>
                </div>
            </Modal>
        )
    }

    const SuccessModal = () => {
        return (
            <Modal className={" rounded text-purple"} isOpen={visible}
centered scrollable backdrop="static"  >
                    <ModalBody>
                        <Alert color="info" className="text-center"
isOpen={visible} toggle={() => window.location.href = "/orderHistory"}>
                            {msg}
                        </Alert>
                    </ModalBody>
                    <ModalFooter className="text-right">
                        <button onClick={() => window.location.href =
"/orderHistory"} className="btn btn-custom" >OK</button>
                    </ModalFooter>
            </Modal>
        )
    }

    return (

        <>
            <div className='demo-frame'>
                <form id="paymentForm" method="post">
                    <label className="hosted-fields--label"
><FontAwesomeIcon icon={faInfoCircle} size={"sm"} /> Please enter your card
details.</label>
                    <label className="hosted-fields--label"
htmlFor="amount">Amount Total</label>
                    <div className="hosted-field" style={{ paddingTop: '5px'
}} id="ammount">
                        {total}<FontAwesomeIcon icon={faEuroSign} />
                    </div>
                    <label className="hosted-fields--label" style={{ cursor:
```

```
"pointer" }} htmlFor="cc-name">Card Holder Name</label>
                    <div className="hosted-field" id="cc-name"></div>
                    <label className="hosted-fields--label" style={{ cursor:
"pointer" }} htmlFor="card-number">Card Number</label>
                    <div className="hosted-field" id="card-number"></div>

                    <label className="hosted-fields--label" style={{ cursor:
"pointer" }} htmlFor="cvv">CVV</label>
                    <div className="hosted-field" id="cvv"></div>

                    <label className="hosted-fields--label" style={{ cursor:
"pointer" }} htmlFor="expiration-date">Expiration Date</label>
                    <div className="hosted-field"
id="expiration-date"></div>

                    <div className="text-center custom">
                        <input type="submit" value="Pay and Place order"
className="btn btn-outline-primary" />
                    </div>
                </form>
            </div>
            <ConfirmModal />
            <SuccessModal />
        </>
    )
}

export default CreatePayment;
```

4.8.13 DuePayment.js Component

```
import { faCalendarAlt, faCreditCard, faEuroSign, faInfoCircle, faShieldAlt,
faUser, } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import React, { Component, useEffect, useState } from 'react';
import { Alert, Button, Col, Form, Modal, ModalBody, ModalFooter,
ModalHeader } from 'reactstrap';
import SelectedItem from '../MenuItem/selectedItem';
import { formatCreditCardNumber, formatCVC, formatExpirationDate,
formatFormData } from '../../../util/utils'
import PaymentService from '../../../services/PaymentService';
import OrderService from '../../../services/OrderService';
import $ from 'jquery';
import BraintreeService from '../../../services/BraintreeService';
```

```
const DuePayment = (props) => {

    const [order, setOrder] = useState(props.order);
    const [cardHolderName, setCardHolder] = useState("");
    const [cardNumber, setCardNumber] = useState("");
    const [cardCVC, setCardCVC] = useState("");
    const [cardExpireDate, setExpireDate] = useState("");
    const [isOpen, toggleModal] = useState(false);
    const [visible, setVisible] = useState(false);
    const [msg, setMsg] = useState("");
    const [token, setToken] = useState(props.order.token);
    const [dData, setDeviceData] = useState(null);

    useEffect(() => {

        renderForm();
    }, [])

    const renderForm = async () => {
        var authToken = token;
        var form = document.querySelector('#paymentForm');
        var submit = document.querySelector('input[type="submit"]');


        async function getDeviceData(err, dataCollectorInstance) {
            if (err) {
                return;
            }
            var deviceData = dataCollectorInstance.deviceData;
            window.braintree.client.create({
                authorization: authToken,
            }, function (clientErr, clientInstance) {
                if (clientErr) {
                    console.error(clientErr);
                    return;
                }
                window.braintree.hostedFields.create({
                    client: clientInstance,
                    styles: {
                        'input': {
                            'font-size': '14px'
                        },
                        'input.invalid': {
                            'color': 'red'
                        },
                        'input.valid': {
                            'color': 'green'
                        }
```

```
                    },
                    fields: {
                        cardholderName: {
                            selector: '#cc-name',
                            placeholder: 'Name as it appears on your card'
                        },
                        number: {
                            selector: '#card-number',
                            placeholder: '4111 1111 1111 1111'
                        },
                        cvv: {
                            selector: '#cvv',
                            placeholder: '123'
                        },
                        expirationDate: {
                            selector: '#expiration-date',
                            placeholder: '10/2022'
                        }
                    }
                },

                    function (hostedFieldsErr, hostedFieldsInstance) {
                        if (hostedFieldsErr) {
                            console.error(hostedFieldsErr);
                            return;
                        }
                        submit.removeAttribute('disabled');

                        form.addEventListener('submit', function (event) {
                            event.preventDefault();

                            var formIsInvalid = false;
                            var state = hostedFieldsInstance.getState();
                            Object.keys(state.fields).forEach(function
(field) {
                                if (!state.fields[field].isValid) {

$(state.fields[field].container).addClass('is-invalid');
                                    formIsInvalid = true;
                                }
                            });

                            if (formIsInvalid) {
                                alert("Card input is not valid");
                                return;
                            }

                            hostedFieldsInstance.tokenize({
```

```
                                cardholderName: $('#cc-name').val()
                            },

                                function (tokenizeErr, payload) {
                                    if (tokenizeErr) {
                                        console.error(tokenizeErr);
                                        return;
                                    }
                                    console.log(deviceData);
                                    console.log('Got a nonce: ' +
payload.nonce);

                                    BraintreeService.payment({
                                        "correlation_id":
JSON.parse(deviceData).correlation_id,
                                        "nonce": payload.nonce,
                                        "amount": order.total
                                    }).then(function (resp) {
                                        console.log(resp.data);
                                        var result = resp.data.result;
                                        makePayment(result);


                                    }).catch(function (error) {
                                        alert("Transaction failed");
                                        console.log(error.response);
                                    })

                                });
                        }, false);
                    });
            });
        }

        await window.braintree.client.create({
            authorization: authToken
        }, async function (err, clientInstance) {

            await window.braintree.dataCollector.create({
                client: clientInstance,
                hostedFields: true
            }, (err, dataCollectorInstance) => getDeviceData(err,
dataCollectorInstance));
        });
    }

    const handleInputChange = (event) => {
```

```
        if (event.target.name === "number") {
            event.target.value = formatCreditCardNumber(event.target.value);
            setCardNumber(event.target.value);
        } else if (event.target.name === "expiry") {
            event.target.value = formatExpirationDate(event.target.value);
            setExpireDate(event.target.value)
        } else if (event.target.name === "cvc") {
            event.target.value = formatCVC(event.target.value);
            setCardCVC(event.target.value);
        }
        //this.setState({ [target.name]: target.value });
    };

    const handleSubmit = (event) => {
        event.preventDefault();
        toggleModal(true);
    }

    const makePayment = (result) => {

        toggleModal(false);

        //console.log(selectedItems);
        //console.log(cardNumber.split(" ").join(""));
        //console.log(cardCVC);
        //console.log(cardExpireDate);


        var card = result.target.creditCard;
        let payData = {
            "cardHolderName": card.cardholderName,
            "amount": parseFloat(order.total),
            "cardNumber": card.maskedNumber,
            "expireDate": card.expirationDate,
            "cvv": cardCVC,
            "orderId": order.orderId,
            "order": order
        }

        console.log(payData);
        PaymentService.insertDuePayment(payData).then(function (resp) {
            console.log(resp);
            setMsg(<><div>Payment Success </div>
                <div><a href="/" className="text-decoration-none text-purple
font-weight-bold">Go back to menu items</a></div></>);
            setVisible(true);
            localStorage.removeItem("pendingPayment");
```

```
        }).catch(function (error) {
            console.log(error.response);
        });
    }

    const ConfirmModal = () => {
        return (
            <Modal className={" rounded text-purple"} isOpen={isOpen}
centered scrollable backdrop="static"  >
                    <ModalHeader toggle={() => toggleModal(!isOpen)}>
                        <div className="h3">Confirm Payment</div>
                    </ModalHeader>
                    <ModalBody>
                        <em>Total: </em>{order.total}<FontAwesomeIcon
icon={faEuroSign} /><br />
                        <em>Number: </em>{cardNumber}<br />
                        <em>Name: </em>{cardHolderName}<br />
                        <em>Expiry: </em>{cardExpireDate}<br />
                        <em>CVC: </em>{cardCVC}
                    </ModalBody>
                    <ModalFooter className="text-right">
                        <button onClick={() => makePayment()} className="btn
btn-custom" >Pay</button>
                    </ModalFooter>
            </Modal>
        )
    }

    const SuccessModal = () => {
        return (
            <Modal className={" rounded text-purple"} isOpen={visible}
centered scrollable backdrop="static"  >
                <div className="shadow-custom">
                    <ModalBody>
                        <Alert color="info" className="text-center"
isOpen={visible} toggle={() => window.location.reload(false)}>
                            {msg}
                        </Alert>
                    </ModalBody>
                    <ModalFooter className="text-right">
                        <button onClick={() =>
window.location.reload(false)} className="btn btn-custom" >OK</button>
                    </ModalFooter>
                </div>
            </Modal>
        )
    }
```

```
    return (
        <>
            <div className='demo-frame'>
                <form id="paymentForm" method="post">
                    <label className="hosted-fields--label"
><FontAwesomeIcon icon={faInfoCircle} size={"sm"} /> Please enter your card
details.</label>
                    <label className="hosted-fields--label"
htmlFor="amount">Total Amount</label>
                    <div className="hosted-field " style={{ paddingTop:'5px'
}} id="ammount">
                        {order.total}<FontAwesomeIcon icon={faEuroSign} />
                    </div>
                    <label className="hosted-fields--label" style={{ cursor:
"pointer" }} htmlFor="cc-name">Card Holder Name</label>
                    <div className="hosted-field" id="cc-name"></div>
                    <label className="hosted-fields--label" style={{ cursor:
"pointer" }} htmlFor="card-number">Card Number</label>
                    <div className="hosted-field" id="card-number"></div>

                    <label className="hosted-fields--label" style={{ cursor:
"pointer" }} htmlFor="cvv">CVV</label>
                    <div className="hosted-field" id="cvv"></div>

                    <label className="hosted-fields--label" style={{ cursor:
"pointer" }} htmlFor="expiration-date">Expiration Date</label>
                    <div className="hosted-field"
id="expiration-date"></div>

                    <div className="text-center custom">
                        <input type="submit" value="Pay" className="btn
btn-outline-primary" />
                    </div>
                </form>
                <ConfirmModal />
                <SuccessModal />
            </div>
        </>
    )
}

export default DuePayment;
```

4.8.14 CreateReview.js Component

```
import { faComment, faUser } from '@fortawesome/free-regular-svg-icons';
import { faAlignJustify } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import React, { useState } from 'react';
import { Col, Row } from 'reactstrap';
import MenuItemService from '../../../services/MenuItemService';
import OrderItemService from '../../../services/OrderItemService';
import ReviewService from '../../../services/ReviewService';
import './StarRating.css';

const CreateReview = (props) => {

    const [orderItem, setOrderItem] = useState(props.orderItem);
    const [reviewer, setReviewer] = useState("");
    const [rate, setRate] = useState(1);
    const [comment, setComment] = useState("");

    const handleSubmit = async (event) => {
        event.preventDefault();

        orderItem.menuItem.menuCategory = null;
        orderItem.isReviewed = true;
        console.log(orderItem);
        let menuItem = orderItem.menuItem;
        menuItem.reviewCount = parseInt(menuItem.reviewCount) + 1;
        menuItem.avgRate = parseFloat((menuItem.reviewCount - 1) *
menuItem.avgRate) + rate / parseFloat(menuItem.reviewCount)

        let itemReview = {
            "rate": rate,
            "comment": comment,
            "reviewer": reviewer,
            "orderItemId": orderItem.orderItemId,
            "menuItemId": orderItem.menuItem.menuItemId
        }

        await ReviewService.postReview(itemReview).then(async function
(resp) {
            console.log(resp.data);
            await OrderItemService.putOrderItem(orderItem).then(function
(res) {
                console.log(res);
                props.toggle();
            }).catch(function (error) {
                console.log(error.reponse);
```

```
            });
            //await MenuItemService.updateMenuItem(menuItem).then(function
(res) {
            //    console.log(res);
            //    props.toggle();
            //}).catch(function (error) {
            //    console.log(error.reponse);
            //});

        }).catch(function (error) {
            console.log(error.response);
        })
    }

    return (
        <>

            <Row>
                <Col>
                    <form id="ratingForm" onSubmit={(e) => handleSubmit(e)}>
                        <div className="rating">
                            <h4>Rate it:</h4>
                            <input type="radio" id="star5" checked={rate ==
5} name="rating" onChange={() => setRate(5)} value="5" /><label
htmlFor="star5" title="Delicious!">5 stars</label>
                            <input type="radio" id="star4" checked={rate ==
4} name="rating" onChange={() => setRate(4)} value="4" /><label
htmlFor="star4" title="Yummy">4 stars</label>
                            <input type="radio" id="star3" checked={rate ==
3} name="rating" onChange={() => setRate(3)} value="3" /><label
htmlFor="star3" title="Meh">3 stars</label>
                            <input type="radio" id="star2" checked={rate ==
2} name="rating" onChange={() => setRate(2)} value="2" /><label
htmlFor="star2" title="Kinda bad">2 stars</label>
                            <input type="radio" id="star1" checked={rate ==
1} name="rating" onChange={() => setRate(1)} value="1" /><label
htmlFor="star1" title="Sucks big time">1 star</label>
                        </div>
                        <div className="clearfix"></div>
                        <div className=" form-group">
                            <div className="input-group">
                                <span>
                                    <FontAwesomeIcon icon={faUser} />
                                    <input type="text"
                                        defaultValue={reviewer}
                                        required
                                        className="form-control"
                                        placeholder={"Your Name"}
```

```
                                        onChange={(e) => setReviewer(
e.target.value )}
                                    />
                                </span>
                            </div>
                        </div>
                        <div className=" form-group">

                            <div className="input-group">
                                <span className="align-items-start">

                                    <FontAwesomeIcon icon={faComment}
className="mt-2" />
                                    <textarea type="text"
                                        defaultValue={comment}
                                        required
                                        className="form-control"
                                        placeholder={"Your feedback"}
                                        onChange={(e) =>
setComment(e.target.value)}
                                    ></textarea>
                                </span>
                            </div>
                        </div>
                        <div className="text-center custom">
                            <input type="submit" className="btn
btn-outline-primary my-2 my-sm-0" value ='Submit Feedback' />
                        </div>
                    </form>
                </Col>
            </Row>
        </>
        )
}

export default CreateReview;
```

4.8.15 CreateWaiter.js Component

```
import { faEnvelope, faInfoCircle, faLock, faPhone, faUser } from
'@fortawesome/free-solid-svg-icons';
import { faUser as faUserR } from '@fortawesome/free-regular-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import React, { useState } from 'react';
import { Card, CardBody, CardHeader, Col, Form, Row, Alert, Tooltip } from
'reactstrap';
import WaiterService from '../../../services/WaiterService';

const CreateWaiter = (props) => {
    const [username, setUsername] = useState("");
    const [firstname, setFN] = useState("");
    const [lastname, setLN] = useState("");
    const [password, setPass] = useState("");
    const [conPass, setConPass] = useState("");
    const [phone, setPhone] = useState("");
    const [email, setEmail] = useState("");
    const [alertMsg, setAlert] = useState('');
    const [alertOpen, showAlert] = useState(false);
    const [isTooltipOpen, showTooltip] = useState(false);

    const handleSubmit = (e) => {
        e.preventDefault();

        let user = {
            "username": username,
            "firstname": firstname,
            "lastname": lastname,
            "password": password,
            "phone": phone,
            "email": email
        };

        WaiterService.PostWaiter(user).then(function (resp) {
            window.location.reload(false);
        }).catch(function (error) {
            console.log(error.response);
        })

    }

    return (

        <div>
            <Row className="text-purple mt-5">
```

```
                <Col xs="12" sm={{ size: 8, offset: 2 }} md={{ size: 6,
offset: 3 }}>
                    <Card className="shadow-custom">
                        <CardHeader>
                            <Alert aria-live="polite" className="text-center
" color="danger" isOpen={alertOpen} toggle={() => showAlert(false)}>
                                {alertMsg}
                            </Alert>
                        </CardHeader>
                        <CardBody>
                            <Form onSubmit={handleSubmit}>
                                <div className=" form-group row">
                                    <div className="col-12 col-sm-6">
                                        <div className="input-group">
                                            <span>

                                                <FontAwesomeIcon
icon={faUser} />
                                                <input type="text"
                                                    defaultValue={username}
                                                    required
                                                    className="form-control"
                                                    placeholder={"Username"}
                                                    onChange={(e) =>
setUsername(e.target.value)}
                                                />
                                            </span>
                                        </div>
                                    </div>
                                    <div className="col-12 mt-sm-0 mt-3
col-sm-6">
                                        <div className="input-group">
                                            <span>
                                                <FontAwesomeIcon
icon={faPhone} />
                                                <input type="tel"
                                                    defaultValue={phone}
                                                    required

pattern="[+]{0,1}[0-9]{9,15}"
                                                    title="Input a valid
phone number"
                                                    className="form-control"
                                                    placeholder={"Phone
no."}
                                                    onChange={(e) =>
setPhone(e.target.value)}
                                                />
```

```
                                        </span>
                                    </div>
                                </div>
                            </div>
                            <div className=" form-group row">
                                <div className="col-12 col-sm-6">
                                    <div className="input-group">
                                        <span>

                                            <FontAwesomeIcon
icon={faUserR} />
                                            <input type="text"
                                                defaultValue={firstname}
                                                required
                                                className="form-control"
                                                placeholder={"First
Name"}
                                                onChange={(e) =>
setFN(e.target.value)}
                                            />
                                        </span>
                                    </div>
                                </div>
                                <div className="col-12 mt-sm-0 mt-3
col-sm-6">
                                    <div className="input-group">
                                        <span>

                                            <FontAwesomeIcon
icon={faUserR} />
                                            <input type="text"
                                                defaultValue={lastname}
                                                required
                                                className="form-control"
                                                placeholder={"Last
Name"}
                                                onChange={(e) =>
setLN(e.target.value)}
                                            />
                                        </span>
                                    </div>
                                </div>
                            </div>
                            <div className=" form-group row">
                                <div className="col-12 col-sm-6">
                                    <div className="input-group">
                                        <span>
```

```
                                                    <FontAwesomeIcon
icon={faLock} />
                                                    <input type="password"
                                                        defaultValue={password}
                                                        required
                                                        className="form-control
password"
pattern="^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$%^&*_=+-]).{8,}$"
                                                        placeholder={"Password"}
                                                        onChange={(e) =>
setPass(e.target.value)}
                                                        title="[1]At least 1
Uppercase
                                                               [2]At least 1
Lowercase
                                                               [3]At least 1
Number
                                                               [4]At least 1
Symbol, symbol allowed --> !@#$%^&*_=+-
                                                               [5]Minimum 8
characters"
                                                         />
                                                    <FontAwesomeIcon
icon={faInfoCircle} size={'lg'} onClick={() => showTooltip(!isTooltipOpen)}
href="#" id="passwordRequirement" />
                                                    <Tooltip autohide={false}
isOpen={isTooltipOpen} toggle={() => showTooltip(!isTooltipOpen)}
placement="right" target="passwordRequirement">

                                                        At least 1 Uppercase, At
least 1 Lowercase, At least 1 Number, At least 1 Symbol, symbol allowed
!@#$%^&*_=+- and minimum 8 characters

                                                    </Tooltip>
                                                    </span>
                                                </div>
                                            </div>
                                            <div className="col-12 mt-sm-0 mt-3
col-sm-6">
                                                <div className="input-group">
                                                    <span>
                                                        <FontAwesomeIcon
icon={faLock} />
                                                        <input type="password"
                                                            defaultValue={conPass}
                                                            required
                                                            className="form-control"
```

```
                                                placeholder={"Confirm
Password"}
                                                onChange={(e) =>
setConPass(e.target.value)}
                                                onBlur={(e) => {
                                                    if (password !==
conPass) {

setAlert("Password doesn't match");
                                                        showAlert(true);

                                                    }
                                                    else {
                                                        setAlert("");

showAlert(false);
                                                    }
                                                }}
                                            />

                                        </span>
                                    </div>
                                </div>
                            </div>
                            <div className=" form-group">
                                <div className="input-group">
                                    <span>
                                        <FontAwesomeIcon
icon={faEnvelope} />
                                        <input type="email"
                                            defaultValue={email}
                                            required
                                            className="form-control"
                                            placeholder={"Email
Address"}
                                            onChange={(e) =>
setEmail(e.target.value)}
                                        />
                                    </span>
                                </div>
                            </div>
                            <div className=" form-group">
                                <div className="custom">
                                    <button type={"submit"}
disabled={!(password === conPass)} className="btn btn-block
btn-outline-primary" >{"Register Waiter Account"}</button>
                                </div>
                            </div>
```

```
                            </Form>
                        </CardBody>
                    </Card>
                </Col>
            </Row>
        </div>
    )
}

export default CreateWaiter;
```

4.8.16 Analytics.js Component

```
//import { extend } from 'jquery'
import { faEuroSign, faPrint } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import React, { Component } from 'react'
import { Col, Row } from 'reactstrap';
import PaymentService from '../../services/PaymentService';
import html2canvas from 'html2canvas';
import jsPDF from 'jspdf';

class Analytics extends Component {
    constructor(props) {
        super(props);
        this.state = {
            sales: [],
            total: 0
        }
        this.getSales = this.getSales.bind(this);
        this.printAsPdf = this.printAsPdf.bind(this);
        this.dataURItoBlob = this.dataURItoBlob.bind(this);
    }
    componentDidMount() {
        this.getSales("1");
    }

    getSales(period) {
        //period = $('#salesPeriods').val();
        let self = this;
        PaymentService.getSalesByPeriod(period).then(async function (resp) {
            console.log(resp.data);
            await self.setState({
                sales: resp.data
            });
```

```
            let total = 0;
            for (let i = 0; i < resp.data.length; i++) {
                total += resp.data[i].amount;

            }
            await self.setState({ total: total });
        }).catch(function (error) {

        })
    }

    dataURItoBlob(dataURI) {
    // convert base64 to raw binary data held in a string
    // doesn't handle URLEncoded DataURIs - see SO answer #6850276 for code
that does this
    var byteString = atob(dataURI.split(',')[1]);

    // separate out the mime component
    var mimeString = dataURI.split(',')[0].split(':')[1].split(';')[0];

    // write the bytes of the string to an ArrayBuffer
    var ab = new ArrayBuffer(byteString.length);
    var ia = new Uint8Array(ab);
    for (var i = 0; i < byteString.length; i++) {
        ia[i] = byteString.charCodeAt(i);
    }
    return new Blob([ab], { type: mimeString });

    }

    printAsPdf() {
        html2canvas(document.querySelector("#printableReport")).then(canvas
=> {

            const imgData = canvas.toDataURL('image/png');
            const pdf = new jsPDF();
            const imgProps = pdf.getImageProperties(imgData);
            const pdfWidth = pdf.internal.pageSize.getWidth();
            const height = pdf.internal.pageSize.getHeight();
            const pdfHeight = (imgProps.height * pdfWidth) / imgProps.width;
            const pageNo = parseInt(Math.ceil(pdfHeight / height));
            pdf.addImage(imgData, 'PNG', 0, 0, pdfWidth / pageNo, (height <
pdfHeight ? height : pdfHeight));
            pdf.save("download.pdf");
        });
    }

    render() {
```

```
        return (

            <div>
                <Row className="pt-3 pb-2 border-bottom">
                    <Col className="text-right">
                        <button onClick={this.printAsPdf} className="btn
btn-danger btn-sm"><FontAwesomeIcon icon={faPrint} /> Print</button>
                    </Col>
                </Row>
                <Row className="pt-3 pb-2 border-bottom">
                    <Col xs={6}><h4>Sales</h4></Col>
                    <Col xs={6}>
                        <div>
                            <select  defaultValue="1" onChange={(e) =>
this.getSales(e.target.value)} className="form-control">
                                <option value="1">Today's sales</option>
                                <option value="7">Last 7 day's
sales</option>
                                <option value="30">Last 30 days's
sales</option>
                                <option value="year">This years's
sales</option>
                            </select>
                        </div>
                    </Col>
                </Row>
                <Row id="printableReport" className="pt-3 pb-2 border-bottom
overflow-auto">
                    <Col>
                        <table className="table  text-center text-purple
table-striped">
                            <thead >
                                <tr >
                                    <th
className="font-weight-bold">Item</th>
                                    <th
className="font-weight-bold">Category</th>
                                    <th
className="font-weight-bold">Total</th>
                                    <th
className="font-weight-bold">Status</th>
                                </tr>
                            </thead>
                            <tbody >
                                {
                                    this.state.sales.map(sale =>
                                        < >
                                            {
```

```
sale.order.orderItems.map(orderItem =>
                                                    <tr
key={orderItem.orderItemId}>

                                                        <td
className="font-weight-normal">{orderItem.menuItem.name}</td>
                                                        <td
className="font-weight-normal">{orderItem.menuItem.menuCategory.name}</td>
                                                        <td
className="font-weight-normal"><FontAwesomeIcon icon={faEuroSign} />
{orderItem.itemAmmount * orderItem.price}</td>
                                                        <td
className="font-weight-normal">{sale.order.status}</td>
                                                    </tr>
                                                )
                                            }
                                        </>
                                    )
                                }
                                <tr>
                                    <td colSpan="2"
className="font-weight-bold text-right">Total sales:</td>
                                    <td
className="font-weight-bold"><FontAwesomeIcon icon={faEuroSign} />
{this.state.total}</td>
                                    <td className="font-weight-normal"></td>
                                </tr>

                            </tbody>
                        </table>
                    </Col>
                </Row>

            </div>
        );
    }
}
export default Analytics;
```

4.8.17 ChangePassword.js Component

```
import { faInfoCircle, faLock } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import React, { useState } from 'react';
import { Card, CardBody, CardHeader, Alert, Col, Form, Row, Tooltip,
ListGroup, ListGroupItem } from 'reactstrap';
import Auth from '../services/Auth';

const ChangePassword = (props) => {

    const [oldPass, setOldPass] = useState("");
    const [password, setPass] = useState("");
    const [conPass, setConPass] = useState("");
    const [alertMsg, setAlert] = useState('');
    const [alertOpen, showAlert] = useState(false);
    const [isTooltipOpen, showTooltip] = useState(false);

    const handleSubmit = (e) => {
        e.preventDefault();

        let passwordData = {
            "oldPassword": oldPass,
            "newPassword": password,
            "confirmPassword": conPass
        };

        Auth.changePassword(passwordData).then(function (resp) {
            console.log(resp);
            alert(resp.data.title);
            localStorage.removeItem('username')
            localStorage.removeItem('type');
            localStorage.removeItem('loggedIn');
            localStorage.removeItem('userId')
            localStorage.removeItem('token')
            window.location.href = '/signin'
        }).catch(function (error) {
            console.log(error.response);
            setAlert(error.response.data.title);
            showAlert(true);
        })
    }

    return (

        <div>
            <Row className="text-purple mt-5">
```

```
                <Col xs="12" sm={{ size: 8, offset: 2 }} md={{ size: 6,
offset: 3 }}>
                    <Card className="shadow-custom">
                        <CardHeader>
                            <Alert aria-live="polite" className="text-center
" color="danger" isOpen={alertOpen} toggle={() => showAlert(false)}>
                                {alertMsg}
                            </Alert>
                        </CardHeader>
                        <CardBody>
                            <Form onSubmit={(e) => handleSubmit(e)}>
                                <div className=" form-group ">
                                    <div className="input-group">
                                        <span>

                                            <FontAwesomeIcon icon={faLock}
/>
                                            <input type="password"
                                                defaultValue={oldPass}
                                                required
                                                className="form-control"
                                                placeholder={"Current
Password"}
                                                onChange={(e) =>
setOldPass(e.target.value)}
                                            />
                                        </span>
                                    </div>
                                </div>
                                <div className=" form-group ">
                                    <div className="input-group">
                                        <span>
                                            <FontAwesomeIcon icon={faLock}
/>
                                            <input type="password"
                                                defaultValue={password}
                                                required
                                                className="form-control
password"

pattern="^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$%^&*_=+-]).{8,}$"
                                                placeholder={"New Password"}
                                                onChange={(e) =>
setPass(e.target.value)}
                                                title=" [1]At least 1
Uppercase
                                                        [2]At least 1
Lowercase
```

```
                                                    [3]At least 1 Number
                                                    [4]At least 1
Symbol, symbol allowed --> !@#$%^&*_=+-
                                                    [5]Minimum 8
characters"
                                                />
                                                <FontAwesomeIcon
icon={faInfoCircle} size={'lg'} onClick={() => showTooltip(!isTooltipOpen)}
href="#" id="passwordRequirement" />
                                                <Tooltip autohide={false}
isOpen={isTooltipOpen} toggle={() => showTooltip(!isTooltipOpen)}
placement="right" target="passwordRequirement">

                                                    At least 1 Uppercase, At
least 1 Lowercase, At least 1 Number, At least 1 Symbol, symbol allowed
!@#$%^&*_=+- and minimum 8 characters
                                                </Tooltip>
                                            </span>
                                        </div>
                                    </div>
                                    <div className="form-group ">
                                        <div className="input-group">
                                            <span>

                                                <FontAwesomeIcon icon={faLock}
/>
                                                <input type="password"
                                                    defaultValue={conPass}
                                                    required
                                                    className="form-control"
                                                    placeholder={"Confirm
Password"}
                                                    onChange={(e) =>
setConPass(e.target.value)}
                                                    onBlur={(e) => {
                                                        if (password !==

conPass) {
                                                            setAlert("Password
doesn't match");
                                                            showAlert(true);
                                                        }
                                                        else {
                                                            setAlert("");
                                                            showAlert(false);
                                                        }
                                                    }}
                                                />
                                            </span>
```

```
                                    </div>
                                </div>
                                <div className=" form-group">
                                    <div className="custom">
                                        <button type={"submit"}
disabled={!(password === conPass && oldPass.length >= 8)} className="btn
btn-block btn-outline-primary" >{"Change Password"}</button>
                                    </div>
                                </div>
                            </Form>
                        </CardBody>
                    </Card>
                </Col>
            </Row>
        </div>
    )
}
export default ChangePassword;
```

4.8.18 Home.js Component

```
import { faCaretDown, faCaretUp, faEuroSign, faInfoCircle, faPlusCircle,
faShoppingBasket, faStar } from '@fortawesome/free-solid-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import React, { Component } from 'react';
import { Link } from 'react-router-dom';
import { Button, Card, CardBody, CardImg, Col, UncontrolledDropdown,
DropdownItem, DropdownMenu, DropdownToggle, Modal, ModalBody, ModalFooter,
ModalHeader, Row, UncontrolledTooltip, CardImgOverlay, Badge,
UncontrolledCollapse, Alert } from 'reactstrap';
import MenuItemService from '../services/MenuItemService';
import ReviewService from '../services/ReviewService';
import SelectedItem from './Management/MenuItem/selectedItem';
import './Management/Review/StarRating.css'


export class Home extends Component {
    static displayName = Home.name; constructor(props) {
        super(props)

        this.state = {
            menus: [],
            selectedItems: {},
            reviews: [],
            menusByCategory: {},
```

```
            isOpen: false,
            menuItem: {},
            isInfoOpen: false,
            isSelectedModalOpen: false,
        }

        this.getMenuItems = this.getMenuItems.bind(this);
        this.selectItem = this.selectItem.bind(this);
        this.toggleModal = this.toggleModal.bind(this);
        this.getReviewes = this.getReviewes.bind(this);
        this.showMenuDetails = this.showMenuDetails.bind(this);
        this.toggleInfoModal = this.toggleInfoModal.bind(this);
        this.toggleSelectedModal = this.toggleSelectedModal.bind(this);
    }

    async componentDidMount() {
        this.getMenuItems();
        if (localStorage.getItem('selectedItems') != undefined) {
            await this.setState({ selectedItems:
JSON.parse(localStorage.getItem('selectedItems')) });
        }
        ///console.log(this.state.selectedItems);
    }

    async getMenuItems() {
        let self = this;
        await this.setState({ menusByCategory: {} });
        MenuItemService.getMenuItem().then(async function (resp) {

            //console.log(resp);
            let data = resp.data;
            //console.log(data);
            await self.setState({ menus: data });
            //console.log(self.state.menus);
            let len = data.length;
            //console.log("length: " + len);
            for (let i = 0; i < len; i++) {

                if (self.state.menusByCategory[data[i].menuCategory.name] ==
undefined) {
                    self.state.menusByCategory[data[i].menuCategory.name] =
[];
                }
                if (data[i].availability) {

self.state.menusByCategory[data[i].menuCategory.name].push(data[i]);
                }
            }
```

```
            await self.setState({ menusByCategory:
self.state.menusByCategory });
            //console.log(self.state.menusByCategory);

        }).catch(function (error) {
            //console.log(error.response);
        });
    }

    toggleSelectedModal() {
        this.setState({ isSelectedModalOpen: !this.state.isSelectedModalOpen
})
    }

    toggleInfoModal() {
        this.setState({ isInfoOpen: !this.state.isInfoOpen })
    }

    async showMenuDetails(menuItem) {

        await this.setState({ menuItem: menuItem });
        this.toggleInfoModal();
    }

    async selectItem(menuItem) {

        let keys = Object.keys(this.state.selectedItems);
        if (!(keys.includes(menuItem.menuItemId.toString()))) {
            this.state.selectedItems[menuItem.menuItemId] = menuItem;
            this.state.selectedItems[menuItem.menuItemId]["itemAmount"] = 1;
            this.state.selectedItems[menuItem.menuItemId]["instruction"] =
"";
            await this.setState({ selectedItems: this.state.selectedItems
});
        }
        else {
            this.state.selectedItems[menuItem.menuItemId]["itemAmount"] =
parseInt(this.state.selectedItems[menuItem.menuItemId]["itemAmount"]) + 1;
            await this.setState({ selectedItems: this.state.selectedItems
});
        }
        //console.log(this.state.selectedItems);
        localStorage.setItem('selectedItems',
JSON.stringify(this.state.selectedItems));
        this.props.selectMenuItem(menuItem);
    }

    toggleModal() {
```

```
        this.setState({ isOpen: !this.state.isOpen });
    }

    async getReviewes(menuItemId) {
        let self = this;
        await ReviewService.getReviewsByMenuItem(menuItemId).then(async
function (resp) {
            //console.log(resp);
            await self.setState({ reviews: resp.data });
            if (resp.data.length > 0) {
                self.toggleModal();
            }
        }).catch(function (error) {
            console.log(error.response);
        });
    }

    render() {

        const InfoModal = () => {
            return (
                <Modal className={" rounded text-purple"}
isOpen={this.state.isInfoOpen} toggle={this.toggleInfoModal} centered
scrollable  >
                    <ModalHeader toggle={this.toggleInfoModal}>
                        <div className="h3">Details</div>
                    </ModalHeader>
                    <ModalBody className="text-center">
                        {
                            Object.keys(this.state.menuItem).length === 0 ?
"" :
                                <>
                                    <h6><em
className="font-weight-bold">Special Dietary:</em>
{this.state.menuItem.isVegan ? "Vegetarian" : "Non Vegetarian"}</h6>
                                    <h4 ><u>Description</u></h4>
                                    <p>{this.state.menuItem.description}</p>
                                    <h4 ><u>Allergens</u></h4>
                                    <p>{this.state.menuItem.allergens}</p>
                                </>
                        }
                    </ModalBody>
                </Modal>
            )
        }
        const SelectItemModal = () => {
            return (
                <Modal className={" rounded text-purple"}
```

```
isOpen={this.state.isSelectedModalOpen} toggle={this.toggleSelectedModal}
centered scrollable  >
                    <ModalHeader toggle={this.toggleSelectedModal}>
                        <div className="h3">Menu item selected!!!</div>
                    </ModalHeader>
                    <ModalBody className="text-center">
                        <p>Menu item added to your order</p>
                        <Link className="text-decoration-none btn-custom btn
btn-sm" to="/dish">View your dish</Link>
                    </ModalBody>
                </Modal>
            )
        }

        const ReviewModal = () => {
            return (
                <Modal className={" rounded text-purple"}
toggle={this.toggleModal} isOpen={this.state.isOpen} centered scrollable   >
                    <ModalHeader className="shadow-custom"
toggle={this.toggleModal}>
                        <div className="h3">Reviews</div>
                    </ModalHeader>
                    <ModalBody style={{ overflowX: "hidden", overflowY:
"auto" }}>
                        {
                            this.state.reviews.map((review, index) =>
                                <Row
key={Math.random().toString(36).substring(0)}>
                                    <Col xs={12} className="d-flex
align-content-center justify-content-center">
                                        <div className="rating">
                                            <input type="radio" id="star5"
disabled checked={review.rate == 5} name={"rating" + index} value="5"
/><label htmlFor="star5" >5 stars</label>
                                            <input type="radio" id="star4"
disabled checked={review.rate == 4} name={"rating" + index} value="4"
/><label htmlFor="star4" >4 stars</label>
                                            <input type="radio" id="star3"
disabled checked={review.rate == 3} name={"rating" + index} value="3"
/><label htmlFor="star3" >3 stars</label>
                                            <input type="radio" id="star2"
disabled checked={review.rate == 2} name={"rating" + index} value="2"
/><label htmlFor="star2" >2 stars</label>
                                            <input type="radio" id="star1"
disabled checked={review.rate == 1} name={"rating" + index} value="1"
/><label htmlFor="star1" >1 star</label>
                                        </div>
                                    </Col>
```

```
                                    <Col xs={12}>
                                        <blockquote className="blockquote
text-center">
                                            <p className="mb-0
font-weight-normal">{review.comment}</p>
                                            <footer
className="blockquote-footer font-weight-light"><small>{review.reviewer + ",
"}<cite>{(new Date(review.createdAt)).toLocaleString('en-US', { month:
'short', day: 'numeric', year: "numeric" })}</cite></small></footer>
                                        </blockquote>
                                    </Col>
                                </Row>
                            )
                        }
                    </ModalBody>
                </Modal>
            )
        }

        return (
            <>
                <div className="row mt-2">
                    <div className="col-12">
                        {this.state.menus.length < 1 ?
                            <div className="row">
                                <div className="col-12 h3">
                                    <div style={{ height: "500px" }}
className="d-flex align-content-center justify-content-center">
                                        <div class="spinner-border
text-primary" role="status">
                                            <span
class="sr-only">Loading...</span>
                                        </div>
                                    </div>

                                </div>
                            </div> :
                            <>
                                <Row>
                                    <Col >
                                        <div className="home-menu-view">

                                        {

Object.keys(this.state.menusByCategory).map((category, index) =>
                                                <div className="border mt-2
rounded" key={Math.random().toString(36).substring(0)}>
                                                    <div className="row ml-1
```

```
mr-1 p-2">
                                                        <div style={{
cursor: "pointer" }} id={"category" + category.split(" ").join("")}
className="col-12 h3 mb-0 border text-purple rounded">
                                                            {category}
<FontAwesomeIcon icon={faCaretDown} size={"lg"} style={{ float: "right" }}
/><FontAwesomeIcon icon={faCaretUp} size={"lg"} style={{ float: "right" }}
/>

                                                        </div>
                                                    </div>
                                                    <UncontrolledCollapse
defaultOpen toggler={"#category" + category.split(" ").join("")}>
                                                        <Row className="mt-1
pl-1 pr-1 mr-1 ml-1 ">
                                                            {

this.state.menusByCategory[category].map((menu, index) => <Col
key={Math.random().toString(36).substring(0)} xs={6} sm={4} lg={3}
className="pl-1 pr-1 mb-2" >
                                                                <Card
className="menuItemCard h-100" style={{ maxHeight: "250px" }}>

<CardImg top src={menu.imgPath} style={{ objectFit: "cover", height: "60%"
}} />

<CardImgOverlay top className="p-1 d-flex flex-column align-content-center
bg-semi-transparent justify-content-center  text-center text-purple "
style={{ height: "60%" }}>


<p className="font-weight-bolder m-0"> {menu.name}</p>

<p className="font-weight-bolder m-0"> {menu.price}<FontAwesomeIcon
icon={faEuroSign} /></p>


</CardImgOverlay>

<CardBody className="pt-1 pb-1">

<div className="d-flex align-content-center justify-content-center h-100 ">

<div className="text-center pt-2 custom h-100 " >

<Badge onClick={() => this.getReviewes(menu.menuItemId)}
className="bg-purple btn " style={{ cursor: 'pointer' }}
>{parseFloat(menu.avgRate).toFixed(1)} <FontAwesomeIcon icon={faStar} />
```

```
{"(" + menu.reviewCount + ")"}</Badge>

{" "}

<button onClick={() => this.showMenuDetails(menu)} id={"infoTooltip" +
menu.menuItemId} className="btn btn-transparent ml-2 p-0" ><FontAwesomeIcon
className="text-purple" icon={faInfoCircle} size={"2x"} /></button>

<UncontrolledTooltip placement="right" target={"infoTooltip" +
menu.menuItemId}>

Details

</UncontrolledTooltip>

<div >

{!(localStorage.getItem('type') == 'admin' || localStorage.getItem('type')
== 'waiter') && menu.availability &&

<>

<button onClick={() => this.selectItem(menu)} id={"btnTooltip" +
menu.menuItemId} className="btn  btn-block btn-sm btn-select pt-0 mt-2 pb-0
btn-outline-primary font-weight-bold" >Select Item</button>

<UncontrolledTooltip placement="right" target={"btnTooltip" +
menu.menuItemId}>

Add item to order

</UncontrolledTooltip>

</>

}

</div>

</div>

</div>

</CardBody>
                                                                    </Card>
                                                                </Col>
                                                                )
                                                            }
```

```
                                                    </Row>
                                                </UncontrolledCollapse>
                                            </div>
                                        )
                                        }
                                    </div>
                                </Col>
                            </Row>
                            <Row>
                                <Col>
                                    {(localStorage.getItem('type') !=
"admin" && localStorage.getItem('type') != "waiter" &&
localStorage.getItem('dishItemCount') != undefined &&
localStorage.getItem('dishItemCount') != null &&
localStorage.getItem('dishItemCount') > 0) &&

                                        <Alert
isOpen={true}className="text-white text-center bg-purple">
                                            <Link to='/dish'
className="text-decoration-none text-white">
                                                <FontAwesomeIcon
icon={faShoppingBasket} size={'lg'} />
                                                <span style={{ top:
"-10px" }} className="badge position-relative text-purple bg-light
badge-pill border small mr-3 " >

{localStorage.getItem('dishItemCount') != undefined ?
localStorage.getItem('dishItemCount') : 0}
                                                </span>
                                    View your order
                                </Link>
                                        </Alert>
                                    }
                                </Col>
                            </Row>
                        </>
                    }

                </div>
            </div>
            <ReviewModal />
            <InfoModal />
            <SelectItemModal />
        </>
    )
  }
}
```

4.8.19 Layout.js Component

```
import { Alert } from 'reactstrap';
import React, { Component } from 'react';
import { Container } from 'reactstrap';
import { NavMenu } from './NavMenu';
import { Link } from 'react-router-dom';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faShoppingBasket } from '@fortawesome/free-solid-svg-icons';

export class Layout extends Component {
    static displayName = Layout.name;

  render () {
    return (
        <div>
            <NavMenu />
            <Container >
          {this.props.children}
        </Container>
      </div>
    );
  }
}
```

4.8.20 NavMenu.js Component

```
import React, { Component } from 'react';
import { Nav, Collapse, Container, Navbar, NavbarBrand, NavbarToggler,
NavItem, NavLink, DropdownToggle, DropdownMenu, Dropdown,
UncontrolledDropdown, DropdownItem, UncontrolledTooltip } from 'reactstrap';
import { Link } from 'react-router-dom';
import './NavMenu.css';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faHistory, faUtensils } from '@fortawesome/free-solid-svg-icons';
import SelectedItem from './Management/MenuItem/selectedItem';

export class NavMenu extends Component {
    static displayName = NavMenu.name;
```

```
    constructor(props) {
        super(props);

        this.toggleNavbar = this.toggleNavbar.bind(this);
        this.state = {
            collapsed: true,
            loggedIn: localStorage.getItem('loggedIn') === 'true',
            type: (localStorage.getItem('type') == 'user' ||
localStorage.getItem('type') == 'admin' || localStorage.getItem('type') ==
'waiter') ? localStorage.getItem('type').toString() : 'guest',
        };
    }
    toggleNavbar() {
        this.setState({
            collapsed: !this.state.collapsed
        });
    }

    render() {
        return (
            <>
                <Navbar id="nav" color="dark" className="shadow-custom"
sticky="top" expand="sm" dark>
                    <Container>
                        <NavbarBrand className="text-white" tag={Link}
to="/">Click&Eat</NavbarBrand>

                        <NavbarToggler onClick={this.toggleNavbar}
className="mr-0 ml-auto" />
                        <Collapse isOpen={!this.state.collapsed} navbar>
                            <Nav className="mr-0 ml-auto" navbar>
                                <NavItem>
                                    <Link tag={Link} className="nav-link
text-white" to="/">Home</Link>
                                </NavItem>
                                {(this.state.type == 'admin' ||
this.state.type == 'waiter') && (<>
                                    <NavItem>
                                        <Link tag={Link} className="nav-link
text-white" to="/Analytics">Analytics</Link>
                                    </NavItem>
                                    {
                                        this.state.type == 'admin' &&
                                        <UncontrolledDropdown nav>
                                            <DropdownToggle
className="text-white" nav caret>
                                                Accounts
                                                </DropdownToggle>
```

```
                                        <DropdownMenu>
                                            <DropdownItem>
                                                <Link tag={Link}
className="nav-link text-dark" to="/waiters">Waiters</Link>
                                            </DropdownItem>
                                            <DropdownItem>
                                                <Link tag={Link}
className="nav-link text-dark" to="/customers">Customers</Link>
                                            </DropdownItem>
                                        </DropdownMenu>
                                    </UncontrolledDropdown>
                                }
                                <div>
                                    {this.state.type == 'admin' &&

                                    <UncontrolledDropdown nav>
                                        <DropdownToggle
className="text-white" nav caret>
                                            Management
                                        </DropdownToggle>
                                        <DropdownMenu>
                                            <DropdownItem>
                                                <Link
className="text-dark nav-link" to="/category">Menu Category</Link>
                                            </DropdownItem>
                                            <DropdownItem>
                                                <Link
className="text-dark nav-link" to="/menuItems">Menu Item</Link>
                                            </DropdownItem>
                                            <DropdownItem>
                                                <Link className="
text-dark nav-link" to="/orders">Orders</Link>
                                            </DropdownItem>
                                        </DropdownMenu>
                                    </UncontrolledDropdown>
                                }
                                {
                                    this.state.type == 'waiter' &&
                                    <Link className=" nav-link
text-white" to="/orders">Orders</Link>
                                }
                            </div>
                        </>)
                        }
                        {!this.state.loggedIn && (
                            <>

                                < NavItem >
```

```
                                        <Link tag={Link}
className="nav-link text-white" to="/signup">Register</Link>
                                    </NavItem>
                                    < NavItem >
                                        <Link tag={Link}
className="nav-link text-white" to="/signin">Login</Link>
                                    </NavItem>

                                </>
                            )}
                            {this.state.loggedIn &&
                                <UncontrolledDropdown nav>
                                    <DropdownToggle
className="text-white" nav caret>
                                        My Account
                                            </DropdownToggle>
                                    <DropdownMenu>
                                        <DropdownItem >
                                            <Link tag={Link}
className="nav-link text-dark" to="/changePassword">Change Password</Link>
                                        </DropdownItem>
                                        <DropdownItem >
                                            <Link tag={Link}
className="nav-link text-dark" to="/signout">Logout</Link>
                                        </DropdownItem>
                                    </DropdownMenu>
                                </UncontrolledDropdown>
                            }
                        </Nav>
                    </Collapse>
                    {!(this.state.type == 'admin' || this.state.type ==
'waiter') && <span >
                        <UncontrolledDropdown  >
                            <DropdownToggle className="text-white pl-1 "
nav caret>
                                My Orders <span className="badge
text-purple bg-light badge-pill border small badge-light"
>{localStorage.getItem('dishItemCount') != undefined ?
localStorage.getItem('dishItemCount') : 0}
                                </span>
                            </DropdownToggle>
                            <DropdownMenu right>
                                <DropdownItem>
                                    <Link tag={Link} to='/dish'
className="text-purple nav-link font-weight-bold" >
                                        <FontAwesomeIcon
className="text-purple" /*icon={faUtensils} size={"lg"}*/ /> Selected Items
<span className="badge text-white bg-dark badge-pill border small
```

```
badge-light" >{localStorage.getItem('dishItemCount') != undefined ?
localStorage.getItem('dishItemCount') : 0}
                                            </span>
                                        </Link>
                                    </DropdownItem>
                                    <DropdownItem>
                                        <Link tag={Link} to='/orderHistory'
className="text-purple nav-link font-weight-bold" >
                                            <FontAwesomeIcon
className="text-purple" /*icon={faHistory} size={"lg"}*/ /> Previous Orders

{localStorage.getItem("pendingPayment") != null &&
                                                <>{" "}
                                                    <span className="badge
text-white badge-pill text-white bg-dark border small badge-light"
>{"Due"}</span>
                                                </>
                                            }
                                        </Link>
                                    </DropdownItem>
                                </DropdownMenu>
                            </UncontrolledDropdown>
                        </span>
                        }

                    </Container>
                </Navbar>
            </>
        );
    }
}
```

4.8.21 Signin.js Component

```
import React, { Component } from 'react';
import { Alert, Form, Row, Col, Card, CardHeader, CardBody } from
"reactstrap";
import "../Assets/Style.css";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import {faAddressBook, faLock, faUser} from
"@fortawesome/free-solid-svg-icons";
import Auth from '../services/Auth';
import { error } from 'jquery';
```

```
export class SignIn extends Component {
    static displayName = SignIn.name;

    constructor(props) {
        super(props);
        this.state = {
            alertMsg: "",
            alertOpen: false,
            username: "",
            password:"",
        }
        this.handleSubmit = this.handleSubmit.bind(this);
    }

    validate() {
        return (this.username.length > 0 &&
            this.props.length > 0);
    }

    async handleSubmit(e) {
        e.preventDefault();
        let user = {
            username: this.state.username,
            password: this.state.password
        }
        let self = this;
        let resp = await Auth.verifyUser(user).catch(function (error) {
            error = error.response.data;
            self.setState({ alertMsg: error.title });
            self.setState({ alertOpen: true });
        });
        if (resp !== undefined) {
            //console.log(resp);
            if (resp.status === 200) {
                let data = resp.data;
                localStorage.setItem('username', data.username)
                localStorage.setItem('type', data.type.toLowerCase());
                localStorage.setItem('loggedIn', 'true');
                localStorage.setItem('userId', data.userId)
                localStorage.setItem('token', data.token)

                //console.log(JSON.parse(atob(data.token.split('.')[1])));
                this.props.setType(data.type);
                this.props.setLoggedIn(true);
                //console.log(this);
                if (data.type.toLowerCase() === 'admin') {
                    window.location.href = '/Analytics';
```

```
                }
                else if (data.type.toLowerCase() === 'waiter') {
                    window.location.href = '/Analytics';
                }
                else {
                    window.location.href = '/'
                }
            }
        }
    }

    render() {
        return (
            <>
                <Row  className="text-purple mt-5">
                    <Col xs="12" sm={{ size: 8, offset: 2 }} md={{ size: 6,
offset: 3 }}>
                        <Card className="shadow-custom">
                            <CardHeader>
                                <Alert aria-live="polite"
className="text-center " color="danger" isOpen={this.state.alertOpen}
toggle={() => this.setState({ alertOpen: false })}>
                                    {this.state.alertMsg}
                                </Alert>
                            </CardHeader>
                            <CardBody>
                                <Form onSubmit={this.handleSubmit}>
                                    <div className=" form-group">
                                        <div className="input-group">
                                            <span>
                                                <FontAwesomeIcon
icon={faUser} />
                                                <input type="text"

defaultValue={this.state.username}
                                                    required
                                                    className="form-control"
                                                    placeholder={"Username"}
                                                    onChange={(e) =>
this.setState({ username: e.target.value })}
                                                />
                                            </span>
                                        </div>
                                    </div>
                                    <div className=" form-group">
                                        <div className="input-group">
                                            <span>
                                                <FontAwesomeIcon
```

```
icon={faLock} />
                                                    <input

defaultValue={this.state.password}
                                                        type="password"
                                                        required
                                                        className="form-control"
                                                        placeholder={"Password"}
                                                        onChange={(e) =>
this.setState({ password: e.target.value })}
                                                    />
                                                </span>
                                            </div>
                                        </div>
                                        <div className=" form-group">
                                            <div className="custom">
                                                <button type={"submit"}
disabled={!(this.state.username.length > 0 &&
                                                    this.state.password.length >
0)} className="btn btn-block btn-outline-primary" >{"Login"}</button>
                                            </div>
                                        </div>
                                    </Form>
                                </CardBody>
                            </Card>
                        </Col>
                    </Row>
                </>
            );
        }
}
```

4.8.22 SignUp.js Component

```
import { faEnvelope, faInfoCircle, faLock, faPhone, faUser } from
'@fortawesome/free-solid-svg-icons';
import { faUser as faUserR } from '@fortawesome/free-regular-svg-icons';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import React, { useState } from 'react';
import { Card, CardBody, CardHeader, Col, Form, Row, Alert, Tooltip,
UncontrolledTooltip, ListGroup, ListGroupItem } from 'reactstrap';
```

```
import UserService from '../services/UserService';


const SignUp = (props) => {
    const [username, setUsername] = useState("");
    const [firstname, setFN] = useState("");
    const [lastname, setLN] = useState("");
    const [password, setPass] = useState("");
    const [conPass, setConPass] = useState("");
    const [phone, setPhone] = useState("");
    const [email, setEmail] = useState("");
    const [alertMsg, setAlert] = useState('');
    const [alertOpen, showAlert] = useState(false);
    const [isTooltipOpen, showTooltip] = useState(false);

    const handleSubmit = (e) => {
        e.preventDefault();

        let user = {
            "username": username,
            "firstname": firstname,
            "lastname": lastname,
            "password": password,
            "phone": phone,
            "email": email
        };

        UserService.postUser(user).then(function (resp) {
            console.log(resp);
            window.location.href = '/signin'
        }).catch(function (error) {
            console.log(error.response);
        })

    }

    return (

        <div>
            <Row className="text-purple mt-5">
                <Col xs="12" sm={{ size: 8, offset: 2 }} md={{ size: 6,
offset: 3 }}>
                    <Card className="shadow-custom">
                        <CardHeader>
                            <Alert aria-live="polite" className="text-center
" color="danger" isOpen={alertOpen} toggle={() => showAlert(false)}>
                                {alertMsg}
                            </Alert>
```

```
                        </CardHeader>
                        <CardBody>
                            <Form onSubmit={handleSubmit}>
                                <div className=" form-group row">
                                    <div className="col-12 col-sm-6">
                                        <div className="input-group">
                                            <span>

                                                <FontAwesomeIcon
icon={faUser} />
                                                <input type="text"
                                                    defaultValue={username}
                                                    required
                                                    className="form-control"
                                                    placeholder={"Username"}
                                                    onChange={(e) =>
setUsername(e.target.value)}
                                                />
                                            </span>
                                        </div>
                                    </div>
                                    <div className="col-12 mt-sm-0 mt-3
col-sm-6">
                                        <div className="input-group">
                                            <span>

                                                <FontAwesomeIcon
icon={faPhone} />
                                                <input type="tel"
                                                    defaultValue={phone}
                                                    required

pattern="[+]{0,1}[0-9]{9,15}"
                                                    title="Input a valid
phone number"
                                                    className="form-control"
                                                    placeholder={"Phone
no."}
                                                    onChange={(e) =>
setPhone(e.target.value)}
                                                />
                                            </span>
                                        </div>
                                    </div>
                                </div>
                                <div className=" form-group row">
                                    <div className="col-12 col-sm-6">
                                        <div className="input-group">
```

```
                                        <span>
                                            <FontAwesomeIcon
icon={faUserR} />
                                            <input type="text"
                                                defaultValue={firstname}
                                                required
                                                className="form-control"
                                                placeholder={"First
Name"}
                                                onChange={(e) =>
setFN(e.target.value)}
                                            />
                                        </span>
                                    </div>
                                </div>
                                <div className="col-12 mt-sm-0 mt-3
col-sm-6">
                                    <div className="input-group">
                                        <span>

                                            <FontAwesomeIcon
icon={faUserR} />
                                            <input type="text"
                                                defaultValue={lastname}
                                                required
                                                className="form-control"
                                                placeholder={"Last
Name"}
                                                onChange={(e) =>
setLN(e.target.value)}
                                            />
                                        </span>
                                    </div>
                                </div>
                            </div>
                            <div className=" form-group row">
                                <div className="col-12 col-sm-6">
                                    <div className="input-group">
                                        <span>

                                            <FontAwesomeIcon
icon={faLock} />
                                            <input type="password"
                                                defaultValue={password}
                                                required
                                                className="form-control
password"
```

```
pattern="^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$%^&*_=+-]).{8,}$"
                                                    placeholder={"Password"}
                                                    onChange={(e) =>
setPass(e.target.value)}
                                                    title=" [1]At least 1
Uppercase
                                                            [2]At least 1
Lowercase
                                                            [3]At least 1
Number
                                                            [4]At least 1
Symbol, symbol allowed --> !@#$%^&*_=+-
                                                            [5]Minimum 8
characters"
                                                />
                                                <FontAwesomeIcon size={'lg'}
icon={faInfoCircle} onClick={() => showTooltip(!isTooltipOpen)} href="#"
id="passwordRequirement" />
                                                <Tooltip autohide={false}
isOpen={isTooltipOpen} toggle={() => showTooltip(!isTooltipOpen)}
placement="right" target="passwordRequirement">
                                                    At least 1 Uppercase, At
least 1 Lowercase, At least 1 Number, At least 1 Symbol, symbol allowed
!@#$%^&*_=+- and minimum 8 characters
                                                </Tooltip>
                                            </span>
                                        </div>
                                    </div>
                                    <div className="col-12 mt-sm-0 mt-3
col-sm-6">
                                        <div className="input-group">
                                            <span>
                                                <FontAwesomeIcon
icon={faLock} />
                                                <input type="password"
                                                    defaultValue={conPass}
                                                    required
                                                    className="form-control"
                                                    placeholder={"Confirm
Password"}
                                                    onChange={(e) =>
setConPass(e.target.value)}
                                                    onBlur={(e) => {
                                                        if (password !==
conPass) {
setAlert("Password doesn't match");
                                                            showAlert(true);
```

```
                                                    }
                                                    else {
                                                        setAlert("");
showAlert(false);
                                                    }
                                                }}
                                            />
                                        </span>
                                    </div>
                                </div>
                            </div>
                            <div className=" form-group">
                                <div className="input-group">
                                    <span>
                                        <FontAwesomeIcon
icon={faEnvelope} />
                                        <input type="email"
                                            defaultValue={email}
                                            required
                                            className="form-control"
                                            placeholder={"Email
Address"}
                                            onChange={(e) =>
setEmail(e.target.value)}
                                        />
                                    </span>
                                </div>
                            </div>
                            <div className=" form-group">
                                <div className="custom">
                                    <button type={"submit"}
disabled={!(password === conPass)} className="btn btn-block
btn-outline-primary" >{"Register"}</button>
                                </div>
                            </div>
                        </Form>
                    </CardBody>
                </Card>
            </Col>
        </Row>
    </div>
  )
}

export default SignUp;
```

4.9 Services

The Service folder contains the service files which return an Asynchronous object of the POST request to the API using Axios. Axios is a JavaScript library that helps the code connect to the web.

APIs are used to link resources, share data and gain access to services. Accessing resources on the web is not a quick operation, but this process is helped with the use of JavaScript's Promise API. When conducting asynchronous tasks like in this case, promises come in very useful.

4.9.1 Auth.js Service

```
import axios from "axios"

class Auth
{
    verifyUser(user)
    {
        return axios.post('/api/Authentication/login', user);
    }

    changePassword(passwordData)
    {
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        return axios.post('/api/Authentication/changePassword/' +
localStorage.getItem("username"), passwordData, config);
    }
}

export default new Auth();
```

4.9.2 BraintreeService.js Service

```
import React from 'react';
import axios from 'axios';

class BraintreeService
{
    getToken()
    {
        return axios.get('/api/Braintree');
    }

    payment(data)
    {
        return axios.post('/api/Braintree', data);
    }
}
export default new BraintreeService();
```

4.9.3 CategoryService.js Service

```
import { config } from '@fortawesome/fontawesome-svg-core';
import axios from 'axios';

class CategoryService
{
    getCategories()
    {
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        return axios.get('/api/MenuCategories', config);
    }

    insertCategory(category)
    {
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        return axios.post('/api/MenuCategories', category, config);
    }
```

```
    updateCategory(category)
    {
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        //console.log(category);
        return axios.put('/api/MenuCategories/' + category.categoryId,
category, config);
    }

    DeleteCategory(categoryId)
    {
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        return axios.delete('/api/MenuCategories/' + categoryId, config);
    }
}
export default new CategoryService();
```

4.9.4 FileService.js Service

```
import axios from 'axios';

class FileService
{
    postImage(file)
    {
        var data = new FormData();
        data.append('file', file);

        var config = {
            method: 'post',
            url: '/api/fileupload',
            data: data
        };
        return axios(config);
    }
}
export default new FileService();
```

4.9.5 MenuItemService.js Service

```
import axios from 'axios';

class MenuItemService
{
    getMenuItem()
    {
        return axios.get('/api/MenuItems');
    }

    insertMenuItem(menu)
    {
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        return axios.post('/api/MenuItems', menu, config);
    }

    updateMenuItem(menu)
    {
        //console.log(category);
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        return axios.put('/api/MenuItems/' + menu.menuItemId, menu, config);
    }

    deleteMenuItem(id)
    {
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        return axios.delete('/api/MenuItems/' + id, config);
    }
}
export default new MenuItemService();
```

4.9.6 OrderItemService.js Service

```
import React from 'react';
import axios from 'axios';

class OrderItemService
{
    putOrderItem(orderItem)
    {
        return axios.put('/api/OrderItems/' + orderItem.orderItemId,
orderItem);
    }
}
export default new OrderItemService();
```

4.9.7 OrderService.js Service

```
import { config } from '@fortawesome/fontawesome-svg-core';
import axios from 'axios';

class OrderService
{
    getOrders()
    {
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        return axios.get('/api/Orders', config);
    }

    getOrder(orderId)
    {
        return axios.get('/api/Orders/' + orderId);
    }

    getOrderByIds(ids)
    {
        var data = JSON.stringify(ids);

        var config = {
            method: 'post',
            url: '/api/Orders/userOrders/ids',
            headers: {
```

```
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            data: data
        };
        return axios(config);
    }

    insertOrder(order)
    {
        return axios.post('/api/Orders', order);
    }

    updateOrderToAddItem(order)
    {
        return axios.put('/api/orders/' + order.orderId +'/addItems',
order);
    }

    updateOrder(order)
    {
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        //console.log(category);
        return axios.put('/api/orders/' + order.orderId, order, config);
    }
 }
export default new OrderService();
```

4.9.8 PaymentService.js Service

```
import { config } from '@fortawesome/fontawesome-svg-core';
import axios from 'axios';

class PaymentService
{
    getPayments() {
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
```

```
        return axios.get('/api/Payments', config);
    }

    getSalesByPeriod(period)
    {
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        return axios.get('/api/Payments/period/' + period, config);
    }

    insertPayment(payment)
    {
        return axios.post('/api/Payments', payment);
    }

    insertDuePayment(payment)
    {
        return axios.post('/api/Payments/DuePayment', payment);
    }
}
export default new PaymentService();
```

4.9.9 ReviewService.js Service

```
import React from 'react';
import axios from 'axios';

class ReviewService
{
    getReviewsByMenuItem(menuItemId)
    {
        return axios.get('/api/ItemReviews/MenuItem/' + menuItemId);
    }

    postReview(review)
    {
        return axios.post('/api/ItemReviews', review);
    }
}
export default new ReviewService();
```

4.9.10 UserService.js Service

```
import React from 'react';
import axios from 'axios';

class UserService
{
    postUser(user)
    {
        return axios.post('/api/Authentication/register/user', user)
    }

    getCustomers()
    {
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        return axios.get('/api/Authentication/users/customers', config);
    }
}
export default new UserService();
```

4.9.11 WaiterService.js Service

```
import axios from 'axios';

class WaiterService
{
    PostWaiter(waiter)
    {
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        return axios.post('/api/Authentication/register/waiter', waiter,
config);
    }

    getWaiters()
    {
        var config = {
            headers: {
```

```
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        return axios.get('/api/Authentication/users/waiters', config);
    }

    deleteWaiter(username)
    {
        var config = {
            headers: {
                'Authorization': 'Bearer ' + localStorage.getItem('token')
            }
        }
        return axios.delete('/api/Authentication/user/' + username, config);
    }
}
export default new WaiterService();
```

4.10 App.js

This file is the main component, and it is the container for all the other components.

```
import React, { Component } from 'react';
import { Layout } from './components/Layout';
import { Home } from './components/Home';
import { BrowserRouter as Router, Route, Link, Switch } from
'react-router-dom';
import './custom.css'
import { SignIn } from './components/SiginIn';
import { AuthContext } from './context/auth';
import PrivateRoute from './_component/PrivateRoute';
import Dashboard from './components/Management/Analytics';
import Category from './components/Management/Category/Category';
import MenuItem from './components/Management/MenuItem/MenuItems';
import Waiter from './components/Admin/waiter';
import Orders from './components/Management/Order/Order';
import SelectedItem from './components/Management/MenuItem/selectedItem';
import '../src/Assets/fontawesome/css/all.css';
import OrderHistory from './components/Management/Order/OrderHistory';
import SignUp from './components/SignUp';
import Analytics from './components/Management/Analytics';
import Customers from './components/Admin/Customers';
import ChangePassword from './components/ChangePassword';

export default class App extends Component {
```

```
    static displayName = App.name;
    constructor(props) {
        super(props);
        this.state = {
            type: (localStorage.getItem('type') == 'user' ||
localStorage.getItem('type') == 'admin') ?
localStorage.getItem('type').toString() : 'guest',
            loggedin: localStorage.getItem('loggedIn') === 'true',
            count: 0,
        }
        this.setType = this.setType.bind(this);
        this.setLoggedIn = this.setLoggedIn.bind(this);
        this.selectMenuItem = this.selectMenuItem.bind(this);
        this.updateThis = this.updateThis.bind(this);
    }

    componentDidMount() {
        this.setType(localStorage.getItem('type'));
        this.setLoggedIn(localStorage.getItem('loggedIn'));
        let count = localStorage.getItem('dishItemCount') != undefined ?
localStorage.getItem('dishItemCount') : 0;
        this.setState({ count: parseInt(count) });
       // window.addEventListener("resize", () =>
window.location.reload(false));
    }
    componentWillUnmount() {
        window.removeEventListener('resize', () =>
window.location.reload(false));
    }

    async setType(type) {
        await this.setState({ type: type });
        //alert(this.state.type.toString());
    }
    async setLoggedIn(val) {
        await this.setState({ loggedin: val });
        //alert(this.state.loggedin.toString());
    }

    selectMenuItem(menu) {
        localStorage.setItem('dishItemCount', this.state.count + 1);
        this.setState({ count: this.state.count + 1 });
    }
    updateThis() {
        this.forceUpdate();
    }

    render() {
```

```
        const SignOut = () => {
            localStorage.removeItem('username')
            localStorage.removeItem('type');
            localStorage.removeItem('loggedIn');
            localStorage.removeItem('userId')
            localStorage.removeItem('token')

            return (
                <div>
                    We are signing you out
                    {
                        window.location.href = '/'
                    }
                </div>
                )
        }

        return (
            <Layout>
                <AuthContext.Provider value={(this.state.type == 'admin' &&
this.state.loggedin)}>
                    <Switch>
                        <Route exact path='/'><Home
selectMenuItem={this.selectMenuItem} /></Route>
                        <Route exact path='/signin' setType={this.setType}
><SignIn setType={this.setType} setLoggedIn={this.setLoggedIn} /></Route>
                        <Route exact path='/signup'  ><SignUp /></Route>
                        <Route exact path='/signout' component={SignOut} />
                        <Route exact path='/dish'  ><SelectedItem
updateThis={this.updateThis} /> </Route>
                        <Route exact path='/orderHistory'  ><OrderHistory />
</Route>
                        <PrivateRoute exact path='/category'
component={Category} />
                        <PrivateRoute exact path='/menuItems'
component={MenuItem} />
                        <PrivateRoute exact path='/waiters'
component={Waiter} />
                        <PrivateRoute exact path='/customers'
component={Customers} />
                    </Switch>
                </AuthContext.Provider>
                <AuthContext.Provider value={((this.state.type == 'admin' ||
this.state.type == 'waiter') && this.state.loggedin)}>
                    <Switch>
                        <PrivateRoute exact path='/orders'
component={Orders} />
```

```
                        <PrivateRoute exact path='/analytics'
component={Analytics} />
                    </Switch>
                </AuthContext.Provider>
                <AuthContext.Provider value={(this.state.loggedin)}>
                    <Switch>
                        <PrivateRoute exact path='/changePassword'
component={ChangePassword} />
                    </Switch>
                </AuthContext.Provider>
            </Layout>
        );
    }
}
```

4.11 index.js

```
import 'bootstrap/dist/css/bootstrap.css';
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter } from 'react-router-dom';
import App from './App';
import registerServiceWorker from './registerServiceWorker';

// index.js provides easy entry points for components

const baseUrl =
document.getElementsByTagName('base')[0].getAttribute('href');
const rootElement = document.getElementById('root');

ReactDOM.render(
  <BrowserRouter basename={baseUrl}>
    <App />
  </BrowserRouter>,
  rootElement);

registerServiceWorker();
```

4.12 index.html

This is the application's main Html file, which contains all the ReactJS code and serves as context for the React to render to. The file contains a div into which the React App will be shown, a process called mounting point for the React App.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <base href="%PUBLIC_URL%/" />
    <!--
      manifest.json provides metadata used when your web app is added to the
      homescreen on Android. See
https://developers.google.com/web/fundamentals/engage-and-retain/web-app-mani
fest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json">
    <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico">
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;
700;800;900&display=swap" rel="stylesheet">
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the
build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run
build`.
    -->
    <script
src="https://js.braintreegateway.com/web/3.72.0/js/client.min.js"></script>
    <script
src="https://js.braintreegateway.com/web/3.72.0/js/hosted-fields.min.js"></s
cript>
    <script
src="https://js.braintreegateway.com/web/3.72.0/js/data-collector.min.js"></
script>
    <title>ClickNEatReact</title>
</head>
  <body>
```

```
    <noscript>
      You need to enable JavaScript to run this app.
    </noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

4.13 CSS Files

4.13.1 Style.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins&display=swap');
.btn-login{
    border-radius: 12px !important;
    background-color: rgba(255,255,255,.1) !important;
    border: 1px solid rgba(255,255,255,.3) !important;
    color: #ffffff !important;
}
.btn-join{
    color: #ffffff !important;
    text-align: center !important;
    border-radius: 10px !important;
    background-color: rgba(255,255,255,.1) !important;
    border: 2px solid rgba(255,255,255,.3) !important;
}

.text-purple{
    color: #5D006B !important;
}
.input-group ::placeholder { /* Chrome, Firefox, Opera, Safari 10.1+ */
    color: #5D006B !important;
    opacity: 1; /* Firefox */
```

```
}

.input-group .form-control:focus {
    box-shadow: 0 0 0px rgba(81, 203, 238, 0);
    border: 0px !important;
    border-color: transparent !important;
}

.input-group .form-control{
    min-width:95%!important;
}

:-ms-input-placeholder { /* Internet Explorer 10-11 */
    color: #5D006B !important;
}

::-ms-input-placeholder { /* Microsoft Edge */
    color: #5D006B !important;
}
.input-group span{
    width: 100%;
    padding-left: 15px;
    padding-bottom: 5px;
    padding-top: 5px;
//position: absolute;
    display: flex;
    align-items: center;
    border: solid 1px;
    border-radius: 5px;
    overflow-x:hidden;

}

.input-group input {
    color: #5D006B;
    background-color: white;
    border: none !important;
    height: 30px;
}

.input-group select {
    color: #5D006B;
    background-color: white;
    border: none !important;
    height: 35px;
}

.input-group textarea {
```

```
    color: #5D006B;
    background-color: white;
    border: none !important;
}

.input-group button{
    color: #5D006B;
    background-color: white;
    border: none !important;

}

.btn-custom{
    background-color: #5D006B !important;
    color: white !important;
}
.btn-outline-custom{
    color: #5D006B;
    border-color: #5D006B;
}

.custom .btn-outline-primary {
    color: #5D006B !important;
    background-color: white !important;
    border-color: #5D006B !important;
}

.custom .btn-outline-primary:hover {
    background-color: #5D006B !important;
    color: white !important;
    border-color: #5D006B !important;
}

.custom .btn-select.btn-outline-primary {
    color: #5D006B !important;
    background-color: transparent !important;
    border-color: #5D006B !important;
    font-weight:700 !important;
}

.custom .btn-select.btn-outline-primary:hover {
    background-color: #5D006B !important;
    color: white !important;
    border-color: #5D006B !important;
    font-weight: 700 !important;
}

.custom .active{
```

```
    background-color: #5D006B !important;
    color: white !important;
    border-color: #5D006B !important;
}

.vh-35{
    height:35vh;
}

hr{
    height: 5px;
    border-width:0;
}
.dot {
    height: 150px;
    width: 150px;
    border-radius: 50%;
    border: black solid 1px;
    display: inline-block;
}

.circle
{
    width:170px;
    height:170px;
    border-radius:85px;
    color: black;
    line-height:85px;
    text-align:center;
    background:white;
    border: black solid 1px;
    cursor: pointer;
}

.circle-p{
    margin-bottom: 0px;
    line-height: normal;
}
.w-90{
    width: 85% !important;
}

.bg-size{
    height: 100vh;
}

@media (max-width:991px ) {
    .bg-size{
```

```
        height: 100%;
    }
}

.top-75px{
    top:75px !important;
}

.callout {
    /*padding: 20px;*/
    border: 1px solid #eee;
    border-left-width: 5px;
    border-radius: 0px;


}
.callout h4 {
    margin-top: 0;
    margin-bottom: 5px;
}
.callout p:last-child {
    margin-bottom: 0;
}
.callout code {
    border-radius: 0px;
}
.callout + .bs-callout {
    margin-top: -5px;
}

.callout-default {
    border-left-color: #777;
}
.callout-default h4 {
    color: #777;
}

.callout-primary {
    border-left-color: #428bca;
}
.callout-primary h4 {
    color: #428bca;
}

.callout-success {
    border-left-color: #5cb85c;
}
.callout-success h4 {
```

```
    color: #5cb85c;
}

.callout-danger {
    border-left-color: #d9534f;
}
.callout-danger h4 {
    color: #d9534f;
}

.callout-warning {
    border-left-color: #f0ad4e;
}
.callout-warning h4 {
    color: #f0ad4e;
}

.callout-info {
    border-left-color: #5bc0de;
}
.callout-info h4 {
    color: #5bc0de;
}

.callout-bdc {
    border-left-color: #29527a;
}
.callout-bdc h4 {
    color: #29527a;
}

.callout-purple{
    border-color: rgba(75,0,130 ,0.3 );
    border-left-color: #5D006B;
    background-color: rgba(75,0,130 ,0.3 )

}
.callout-purple h4{
    color: #5D006B;
}

.nav-pills .nav-link.active, .nav-pills .show>.nav-link {
    color: #5D006B !important;
    font-weight: bold;
    /*background-color: #007bff;*/
    border: 1px solid #eee;
    border-left-width: 5px;
    border-radius: 0px;
```

```
    border-color: rgba(75,0,130 ,0.3 );
    border-left-color: #5D006B;
    background-color: rgba(75,0,130 ,0.3 ) !important;
}
a{
    color: #000 !important;
}

@media (max-width: 767px) {
    .d-md-none-alt{
        display: none !important;
    }
}

.bg-grey{
    background-color: rgba(0,0,0,.1);
}
.line-height-normal{
    line-height: normal !important;
}
.progress-bar{
    background-color: #5D006B !important;
}

.border-dashed{
    border: 1px dashed #dee2e6!important;
}

.square
{
    width:100%;
    height:100px;
    color: black;
    line-height:65px;
    text-align:center;
    background:white;
    border: #5D006B solid 1px;
    cursor: pointer;
}

.custom.border {
    border: #5D006B solid 1px !important;
}

.input-custom input{
    color: #5D006B;
    background-color: white;
    border: none !important;
```

```
    height: 30px;
}
.custom-input span{

    background-color: white;
    border: none !important;
    cursor: pointer;

}

.input-custom{
    border-color: #5D006B !important;
}

.input-custom ::placeholder { /* Chrome, Firefox, Opera, Safari 10.1+ */
    color: #5D006B !important;
    opacity: 1; /* Firefox */
}

.shadow-custom{
    box-shadow: 0 .25rem .5rem rgba(75,0,130 ,.5 )!important;
}

body {
    font-family: "Poppins",sans-serif !important;
}

.cursor-pointer{
    cursor: pointer;
}

*{
    -ms-overflow-style: none;
}
::-webkit-scrollbar {
    width: 6px;
    /*margin-top:50px;*/
}

.form-control.qty{
    min-width:85% !important;
    width:60%
}

.form-control.password {
    min-width: 85% !important;
    width: 50%
}
```

```css
/* width */
/*::-webkit-scrollbar {*/
/*    width: 8px;*/
/*}*/

/* Track */
::-webkit-scrollbar-track {
    box-shadow: inset 0 0 4px grey;
    border-radius: 8px;
    margin-top: 15px;
}

/* Handle */
::-webkit-scrollbar-thumb {
    background: purple;
    border-radius: 8px;
}

::-webkit-scrollbar-thumb:hover {
    background: #5D006B;
}

.menuBorder{
    border: 2px solid #5D006B !important;
}

li.active{
    font-weight: bold;
}

.input-group .custom button {
    border:1px solid !important;
}
.input-group .custom button.active {
    border-color:#5D006B !important;
}

.form-control{
    min-width:100%;
}

.btn-outline-success {

    background-color: transparent;

}

.navbar{
```

```
    background-color:#5D006B !important
}

.pb-200 {
    padding-bottom: 100px;
}

@media (max-width: 575px) {
    .pb-200 {
        padding-bottom: 0px;
    }
}

.menuItemCard:hover {
    box-shadow: 0 .35rem .6rem rgba(75,0,130,.5 ) !important;
}

.bg-semi-transparent {
    background-color: rgba(255,255,255,.4) !important;
}

.badge-spacing{
    top: -15px !important;
}

.home-menu-view {
     height: calc(100vh - 115px);
     overflow-x: hidden;
     overflow-y: auto
}
@media (max-width: 767.98px) {
    .home-menu-view {
        height: calc(100vh - 160px);
    }
}
```

4.13.2 custom.css

```
a {
  color: #0366d6;
}

code {
  color: #E01A76;
}

.btn-primary {
  color: #fff;
  background-color: #1b6ec2;
  border-color: #1861ac;
}

div {
    font-weight: 700;
}

.nav-link {
    font-weight: 500;
    color: purple;
}

.navbar-brand {
    font-size: 20px;
    font-weight: 700;
    color: purple;
}

.form-control {
    font-weight: 500;
}

button, .btn {
    font-weight: 400 !important;
}

.hosted-field {
    height: 30px;
    box-sizing: border-box;
    width: 100%;
    padding-left: 5px;
    display: inline-block;
    box-shadow: none;
    font-weight: 600;
```

```
    font-size: 14px;
    border-radius: 6px;
    border: 1px solid #dddddd;
    line-height: 20px;
    background: #fcfcfc;
    margin-bottom: 6px;
    background: linear-gradient(to right, white 50%, #fcfcfc 50%);
    background-size: 200% 100%;
    background-position: right bottom;
    transition: all 300ms ease-in-out;
}

.hosted-fields--label {
    font-family: courier, monospace;
    text-transform: uppercase;
    font-size: 14px;
    display: block;
    margin-bottom: 6px;
}

.button-container {
    display: block;
    text-align: center;
}

.button {
    cursor: pointer;
    font-weight: 500;
    line-height: inherit;
    position: relative;
    text-decoration: none;
    text-align: center;
    border-style: solid;
    border-width: 1px;
    border-radius: 3px;
    -webkit-appearance: none;
    -moz-appearance: none;
    display: inline-block;
}

.button--small {
    padding: 10px 20px;
    font-size: 0.875rem;
}

.button--green {
    outline: none;
    background-color: #64d18a;
```

```
    border-color: #64d18a;
    color: white;
    transition: all 200ms ease;
}

    .button--green:hover {
        background-color: #8bdda8;
        color: white;
    }

.braintree-hosted-fields-focused {
    border: 1px solid #64d18a;
    border-radius: 1px;
    background-position: left bottom;
}

.braintree-hosted-fields-invalid {
    border: 1px solid #ed574a;
}

.braintree-hosted-fields-valid {
}

.bg-purple {
    background-color: #5D006B;
}

#paymentForm {
    max-width: 50.75em;
    margin: 0 auto;
    padding: 0em;
}

.btn:hover{
    color:yellow
}
```

4.13.3 NavMenu.css

```
a.navbar-brand {
  white-space: normal;
  text-align: center;
  word-break: break-all;
}
```

```
html {
  font-size: 14px;
}
@media (min-width: 768px) {
  html {
    font-size: 16px;
  }
}

.box-shadow {
  box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);
}
```

4.13.4 StarRating.css

```
.rating {
   float: left;
}

   .rating:not(:checked) > input {
      position: absolute;
      top: -9999px;
      clip: rect(0,0,0,0);
   }

   .rating:not(:checked) > label {
      float: right;
      width: 1em;
      padding: 0 .1em;
      overflow: hidden;
      white-space: nowrap;
      cursor: pointer;
      font-size: 200%;
      line-height: 1.2;
      color: #ddd;
      text-shadow: 1px 1px #bbb, 2px 2px #666, .1em .1em .2em rgba(0,0,0,.5);
   }

      .rating:not(:checked) > label:before {
         content: '★ ';
      }

   .rating > input:checked ~ label {
```

```
        color: #f70;
        text-shadow: 1px 1px #c60, 2px 2px #940, .1em .1em .2em rgba(0,0,0,.5);
    }

    .rating:not(:checked) > label:hover,
    .rating:not(:checked) > label:hover ~ label {
        color: gold;
        text-shadow: 1px 1px goldenrod, 2px 2px #B57340, .1em .1em .2em rgba(0,0,0,.5);
    }

    .rating > input:checked + label:hover,
    .rating > input:checked + label:hover ~ label,
    .rating > input:checked ~ label:hover,
    .rating > input:checked ~ label:hover ~ label,
    .rating > label:hover ~ input:checked ~ label {
        color: #ea0;
        text-shadow: 1px 1px goldenrod, 2px 2px #B57340, .1em .1em .2em rgba(0,0,0,.5);
    }

    .rating > label:active {
        position: relative;
        top: 2px;
        left: 2px;
    }

.clearfix:before,
.clearfix:after {
    content: " "; /* 1 */
    display: table; /* 2 */
}

.clearfix:after {
    clear: both;
}

.clearfix {
    *zoom: 1;
}
```

Code available on Github : [AnaGriga84/Click-Eat-Final-Year-Project (github.com)]

5. Deployment

The application is published on the Azure cloud and the following are the steps at the first deployment process.

- First, an account is required with a payable subscription.
- In Azure Portal, create a resource and choose the type of resource (Web App)

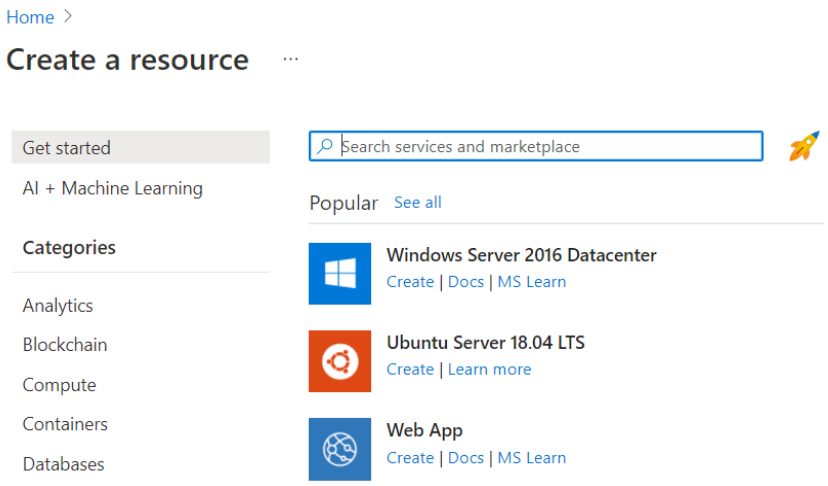

Figure 3 - Azure, Create Resource
Source: Ana Griga, 2021

- Choose a subscription, a name for the application, the operating system, and a region

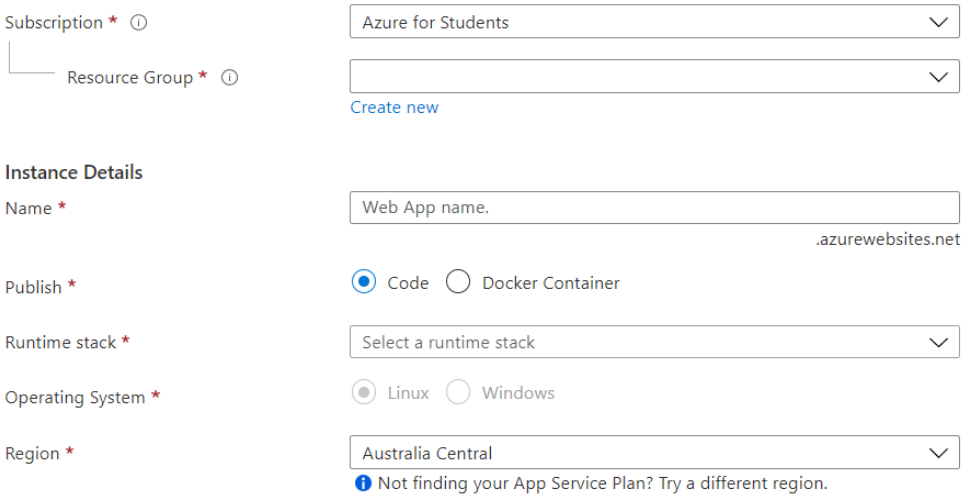

Figure 4 - Azure form
Source: Home - Microsoft Azure, 2021

- The database added the same way into the Azure Portal and the resources created

ClickEatDB (clickeat/ClickEatDB)	SQL database
ClickNEatReact20210318172719ResourceGroup	Resource group
ClickEat	App Service

- In Visual Studio, click Build, then Publish application and Start and login into the Azure account
- Select Azure and Next

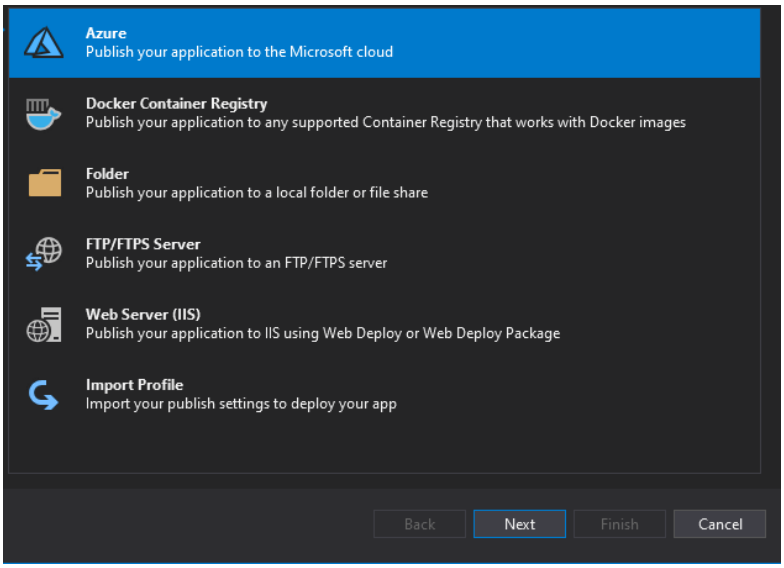

Figure 5 - VS Deployment
Source: Ana Griga, 2021

- Select Azure App Service(Windows) and Next

- Select the Resource and Next

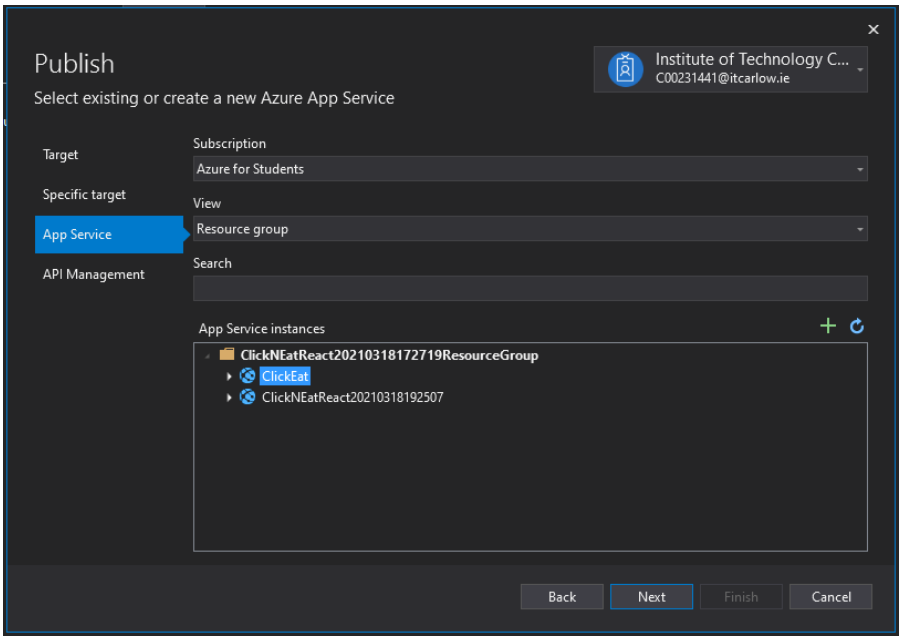

Figure 6 - VS Deployment
Source: Ana Griga, 2021

- Click Finish and Publish

Figure 7 - VS Publish
Source: Ana Griga, 2021

- Click Ctrl and F5 for the application to be refreshed from the last deployment
- On mobile phones, the application needs to be cleared and reset from the Site Settings