Technical Manual

Central Games Platform

Kyle Hennessy
C00227463

Table of Contents

Table of Figures

1.Abstract

The purpose of the Central Games Platform project is to develop a web based application intended for Windows 10 and Android that acts as a hybrid of both an e-commerce store to purchase and play video games, and an online casino, which is intended to be a one stop shop for gamers and casino players alike. Users will be able to purchase and play the latest and greatest blockbuster video game titles, or purchase casino passes to participate in casino games for a chance of winning real world currency. Video games can either be playable from the browser, or be installed to the supported device, depending on the game. Video games that are available on both platforms will be accessible to the user at no additional cost. The application will allow users to transfer winnings from casino games to a registered PayPal account. Impulsive gambling is a huge ethical issue, so a limit of 10 casino passes purchasable per day will be imposed on the user.

Central Games Platform is designed with a business to consumer/business to business approach. Users purchase games from the platform. Publishers of these games will receive the revenue the game has generated every month with a 15% cut. Casino games will be developed in-house. 100% of revenue from casino games goes straight to Central Games Platform.

2. Introduction

This technical manual will provide a detailed description of the functionality, technical aspects, and the design behind the final product of this project. This document will cover each screen, providing an overview of the main functionality of it, as well as the purpose of each component present. Additionally, explanations of external packages, libraries or APIs will be explained, and any complex functionality will be described in detail.

There are 9 different sections that will be broken down and explained in detail. Each section will contain any screens related to that particular section.

The project's two main aspects of development are the traditional games storefront and the online casino hub.

3.Functionality

3.1 Home

Home

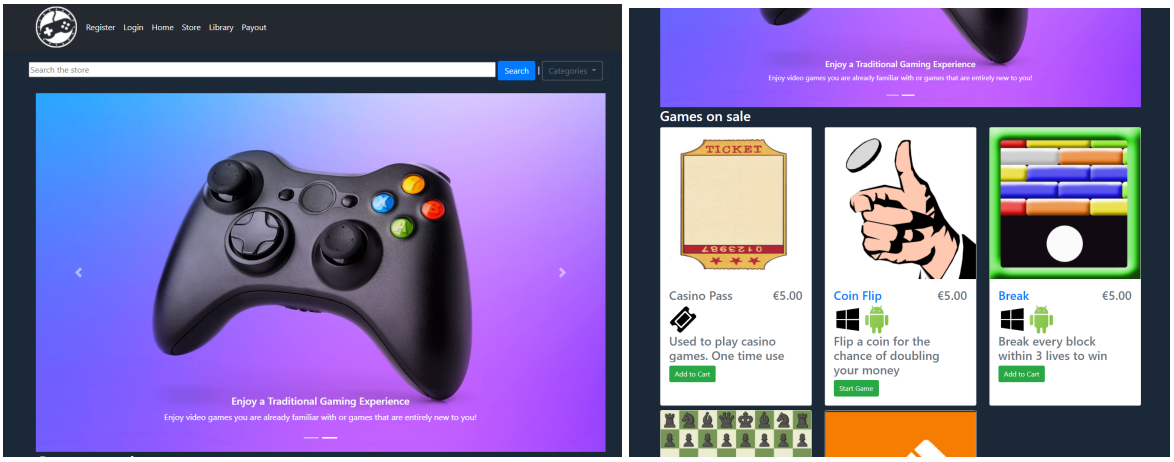

Figure 1 - Home page

This is the first page that every user will be greeted with upon visiting the website. Here, there will be a range of options available in the nav bar that will be featured throughout the entirety of this project. The home page will be used to feature new games or discounts. from here a user can view details of a game, add a game to their cart, register, login, browse the store, view their library, or payout. Most of these options are only available to authenticated users however.

A mobile first design approach has been taken to the design of this project. Elements on the page resize dynamically based on screen size, but the overall design will look very similar. Here is the mobile design for the home page, just to show the minor differences between the pc and mobile design. The main difference is that games are no longer displayed in rows of three, and the nav bar is now contained in a hamburger drop down button which suits the mobile platform very well.

These are the only core differences between the two designs, so the desktop design will be shown only from now on.

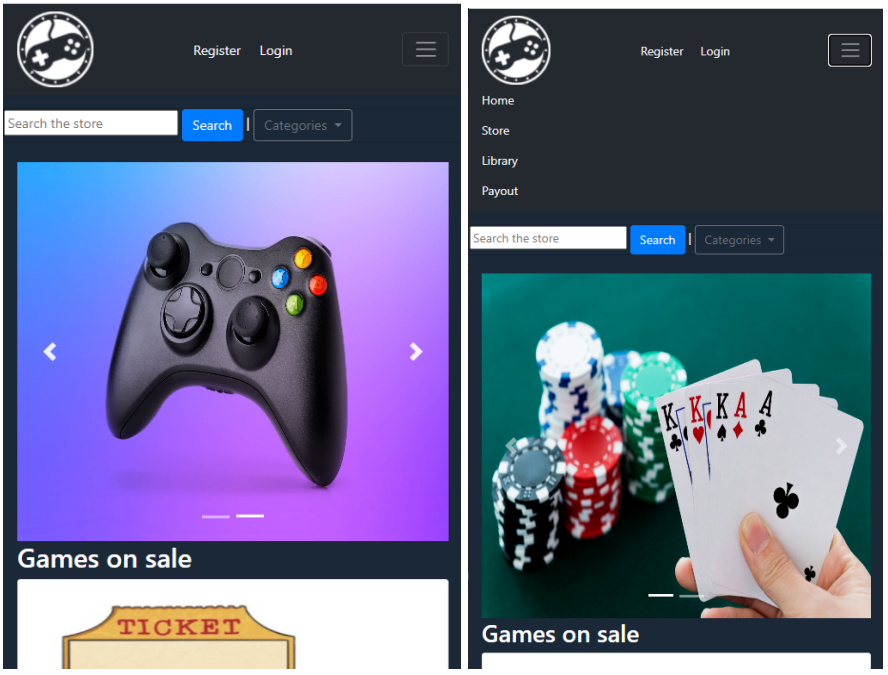

Figure 2 - Mobile design differences

3.2 Authentication

Authentication and authorization is made possible in this project through the use of the ASP.NET Core Identity API, or Identity for short. Identity provides a premade user interface and functionality for logging in, registering, and account management. It also provides many tools to handle authorization, including user roles which is useful for giving users admin privileges. Identity also securely stores passwords by using a key derivation function to generate a random salt and hash the password. The password is hashed using the SHA-1 hash function, which is called 1000 times to further increase the complexity of the password. This method will make it virtually impossible to brute force [1, 2].

Register

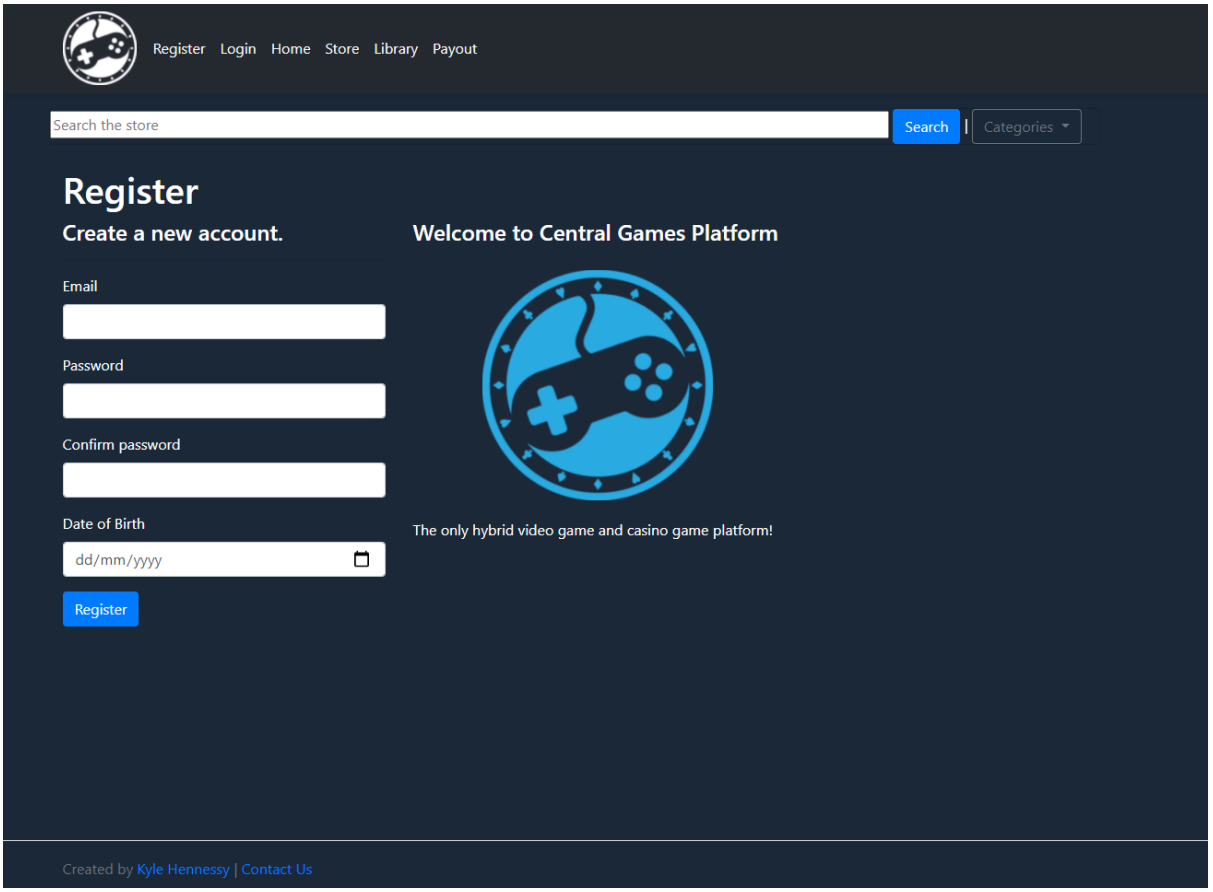

Figure 3 - Register page

To begin enjoying many of the features on this web app, a user must have a registered account. To register, a user enters their email address, password, confirmed password and their date of birth.

Every input field is required, making it impossible to progress unless they are filled in. Standard input validation is featured here, requiring the user to enter a valid email address, password that conforms to standard complexity rules, and email addresses must be unique to the system.

Login

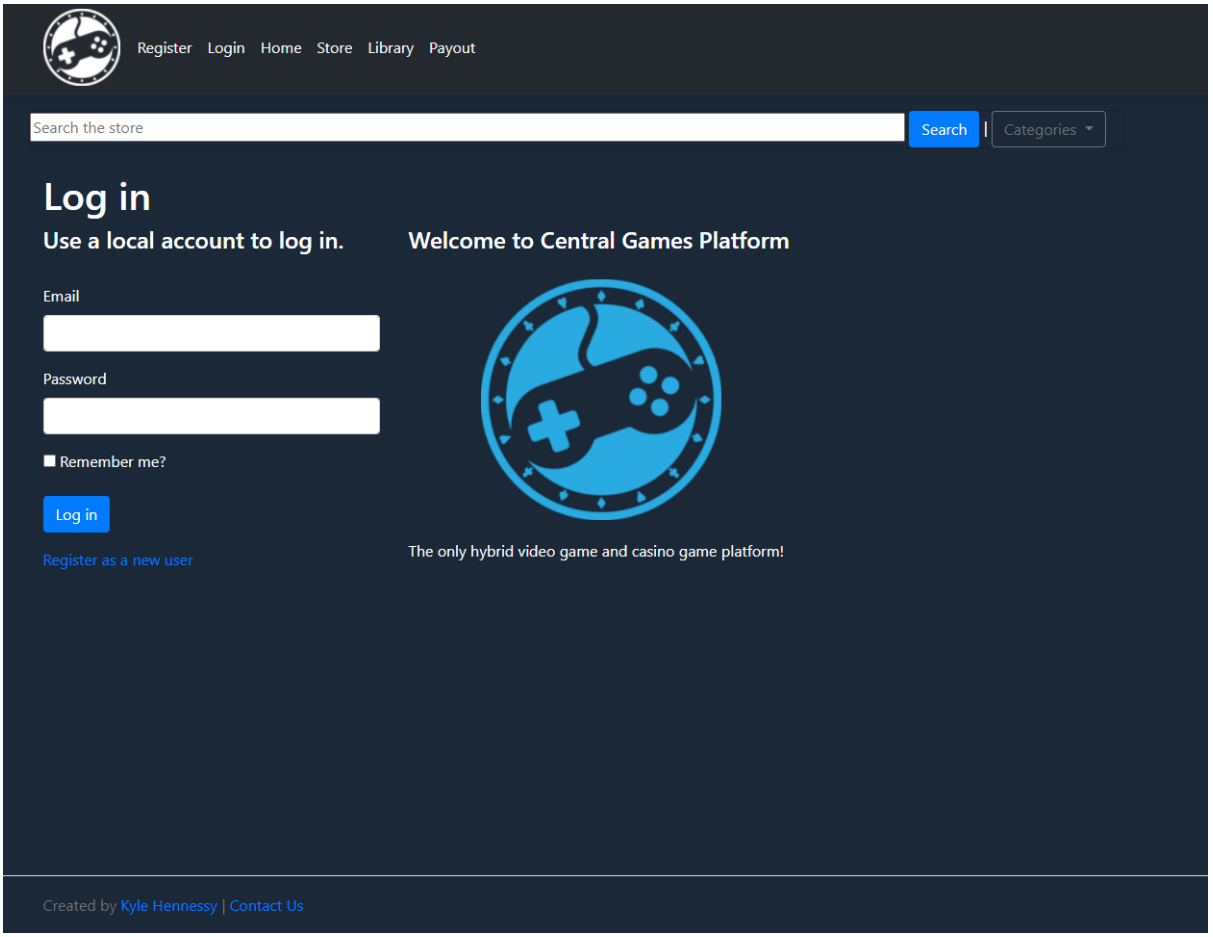

Figure 4 - Login page

This page will allow registered users to log in to their account. The user is required to enter their email and password in order to be authenticated. Users must have already registered to login to the system.

3.3 Store
Shop By Category

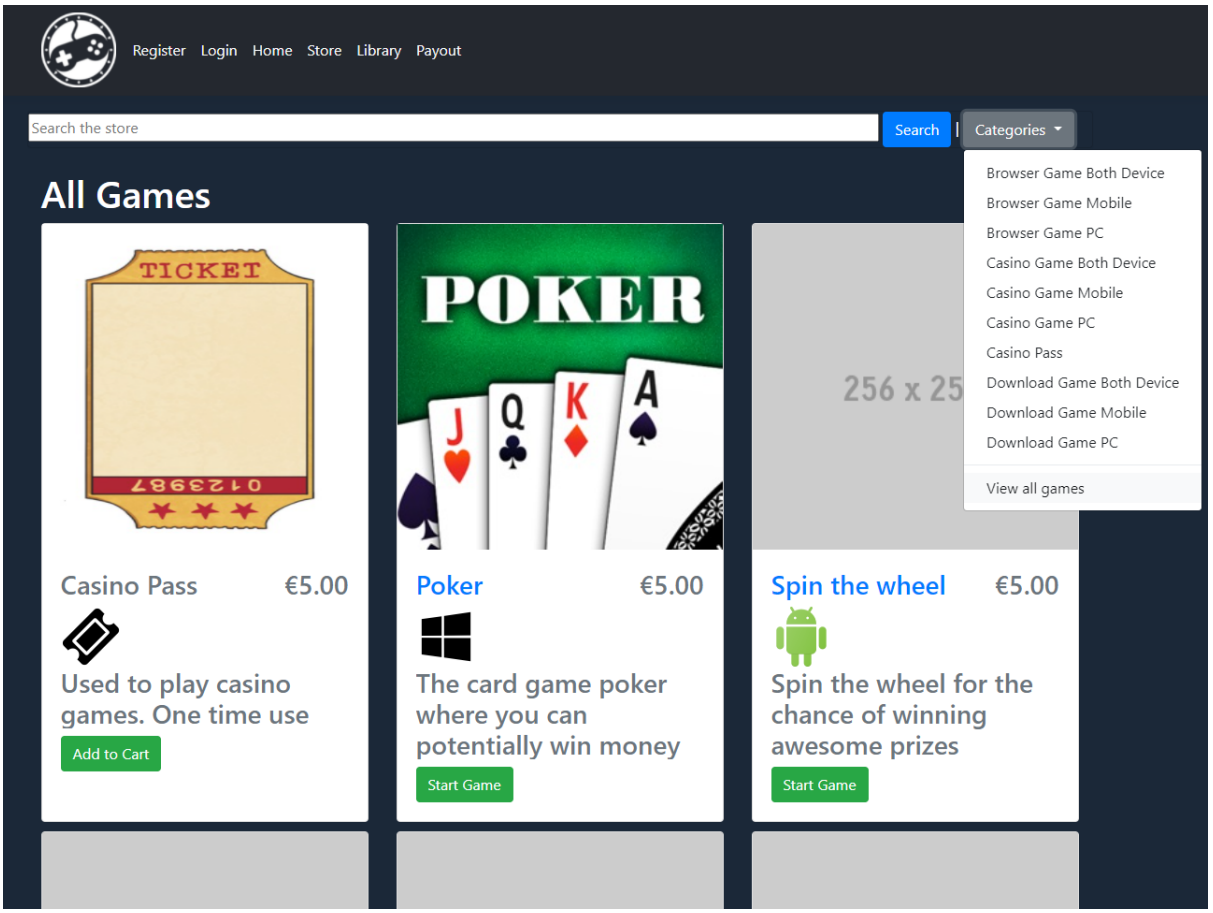

Figure 5 - Browse by category page

There are a lot of different games present on the platform's store page. A range of searching and browsing options are available to the user. Here they can browse by category and supported devices of games. Or if they wish, every game available can be displayed to the user. For every game, it is possible to view its store page, add the game to their cart if it's a video game or spend a casino pass to start playing it if it's a casino game, and if it's a game that the user already owns they will be able to play it from here too.

Search By Name

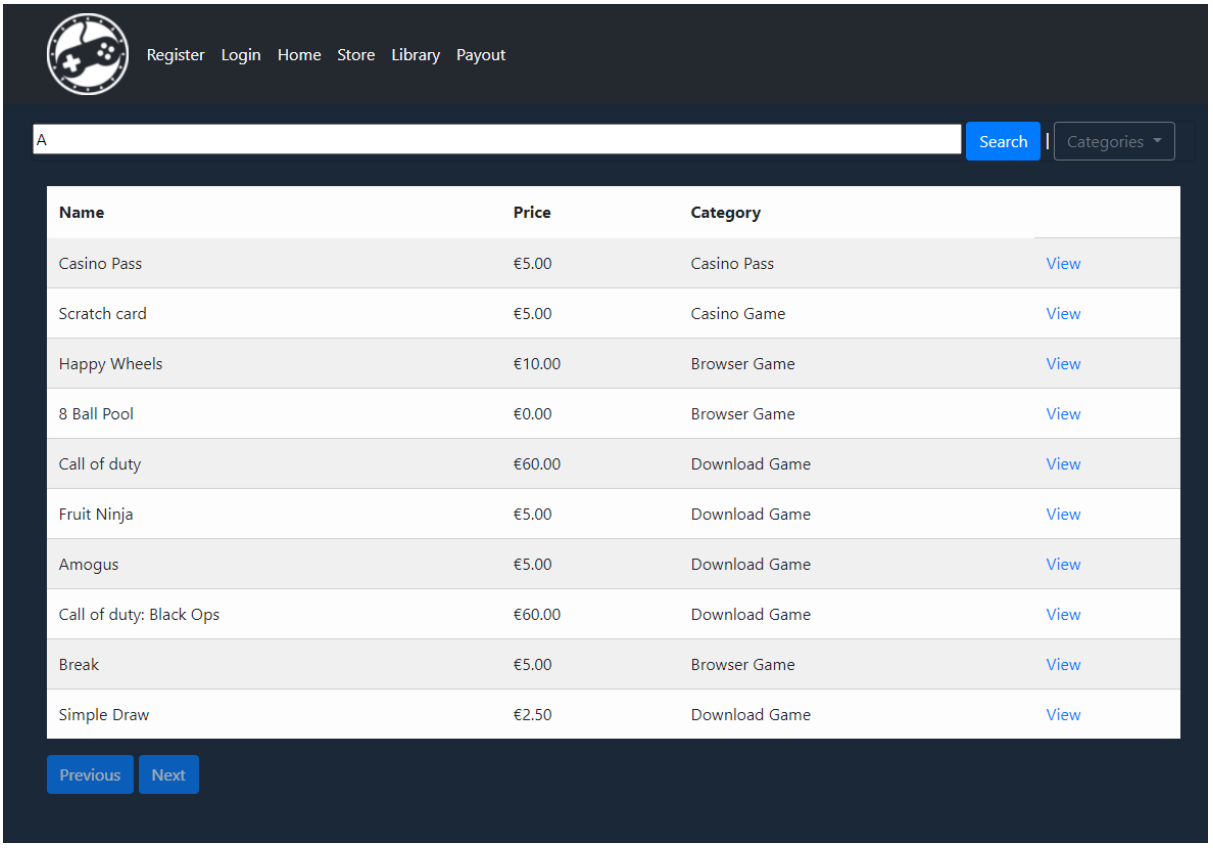

Figure 6 - Search the store page

There may be a certain game in particular that a user may be searching for. They can easily find the game that they wish by entering any search term into the search bar. Games containing the search term in their name will be displayed on the table. From here the user is able to view the game's store page or go to the next or previous page of the table, presuming there are more games to be displayed.

Game Store Page

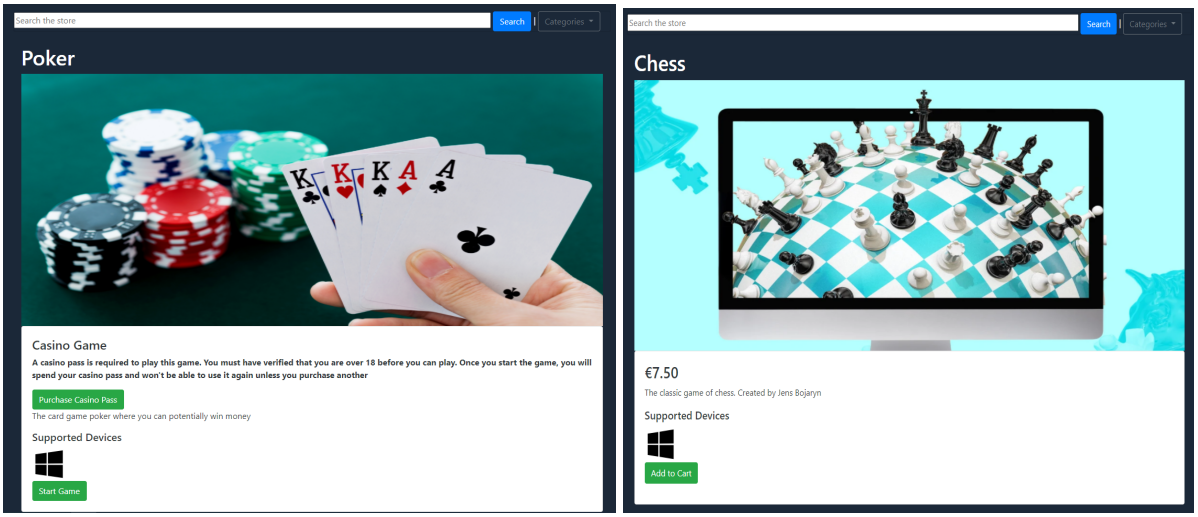

Figure 7 - Casino and video game store pages

This is the general layout for game store pages. A relevant image of the game along with a description of the game is showcased, giving the user an idea of what the game will be like, as well as the supported devices. In the context of casino games, a warning will be displayed clearly stating that this is a casino game and a casino pass is required to play the game. There is an option to purchase a casino pass or to start the game. For video games, the price is displayed, with an option to add the game to their cart if they wish.

Shopping Cart

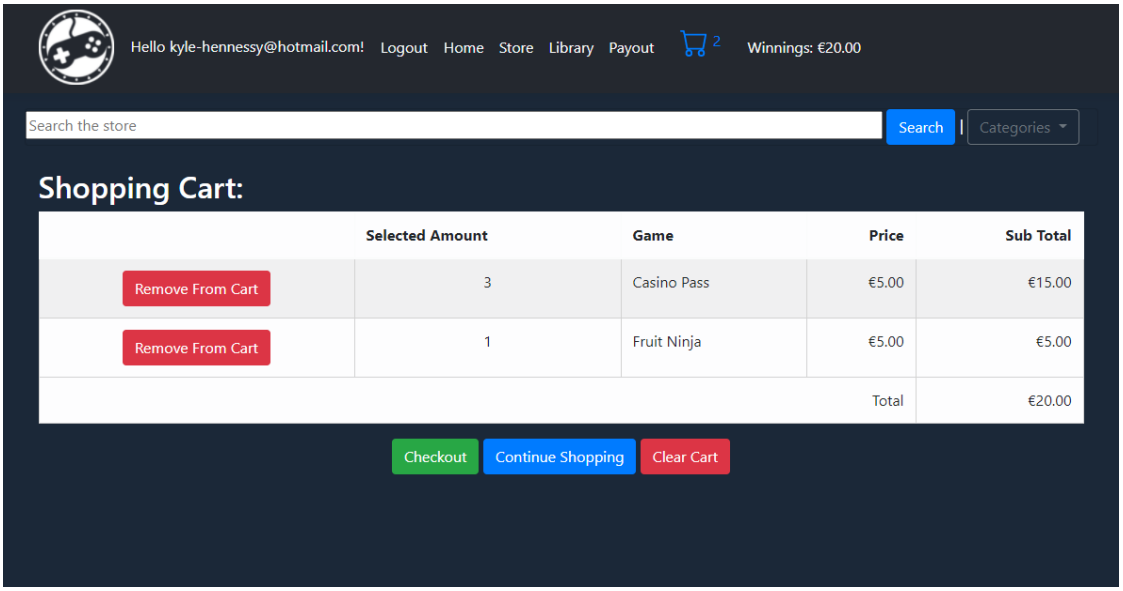

Figure 8 - Shopping cart page

When the user is shopping, they may wish to buy multiple products at once. This is what the shopping cart is for. Here the user can see every product that they have in their basket currently, with details on each of them as well as the overall total of the order. There are multiple options here for the user. They can select remove from cart to remove the product from their basket. In the context of casino passes, selecting remove from cart only removes one from the basket. They can select to continue shopping to be returned to the store, clear cart to remove every item from their basket, or checkout to proceed to the order process.

3.4 Order
Billing Details

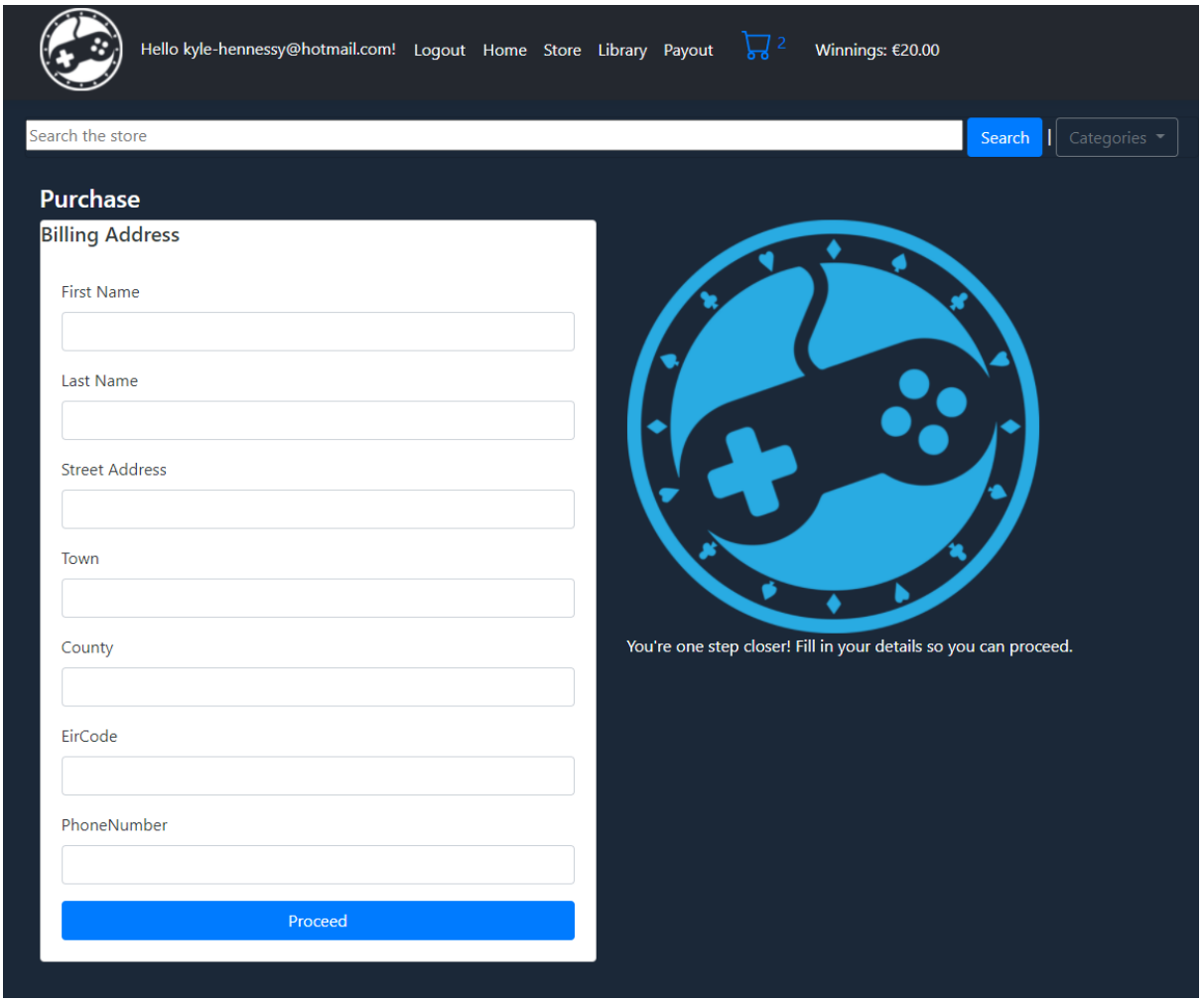

Figure 9 - Billing details page

Every order needs to contain billing details linking to the payment method used by the customer. This is both for record keeping and security reasons. Every field in this form is required. Once this form is submitted, an order will be created, but it will not be successful until the customer has paid for their items.

3.5 Payment

Handling payments is a time consuming and complex process, as you have to keep a record of the customers payment information which is of course very sensitive information and must be stored correctly by either encryption or hashing. Then, you have to establish a relationship with known payment providers such as Visa or MasterCard to allow you to even charge money from a supplied payment method. Then you have to verify that the payment method is real, belongs to the right user, among many other steps before a payment can finally be made which is its own separate project. Because of this, Stripe will be used to handle everything related to payments as they provide RESTful APIs that can be incorporated into any project, and they handle all of the record keeping, verification and every other aspect of payments imaginable. Stripe provides many APIs for different actions but the one that is being used here is the Checkout Session API, which allows you to create a session request with the order total, quantity, or payment method types among many other parameters. Once the payment session has been created, a session id will be created and the user will be redirected to the stripe checkout page for this particular checkout session. [3]

Order Summary

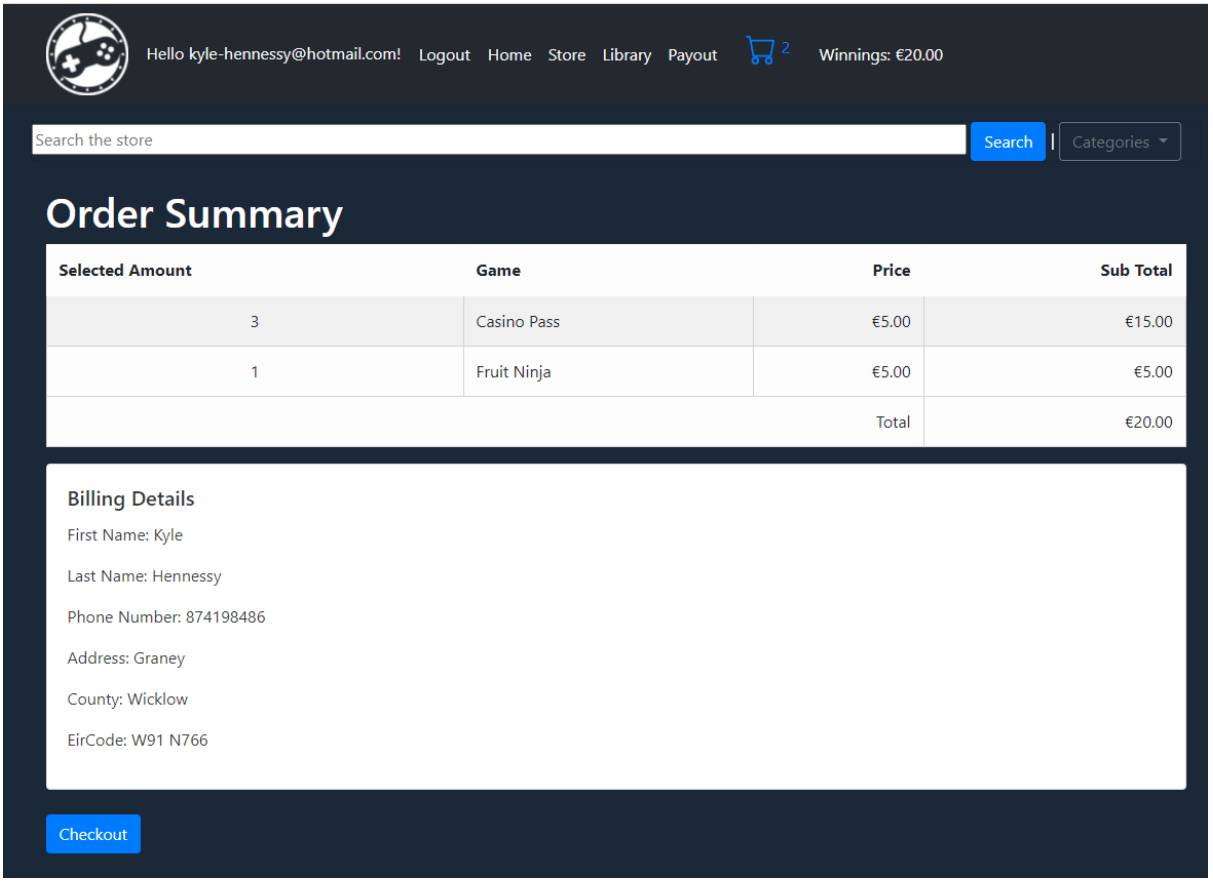

Figure 10 - Order summary page

Before payment is taken from the user, it would be useful to know that their order is in fact correct and their details are inputted correctly. That is what the order summary page displays. From here the user can select checkout, which will take them to the Stripe payment gateway.

Stripe Checkout

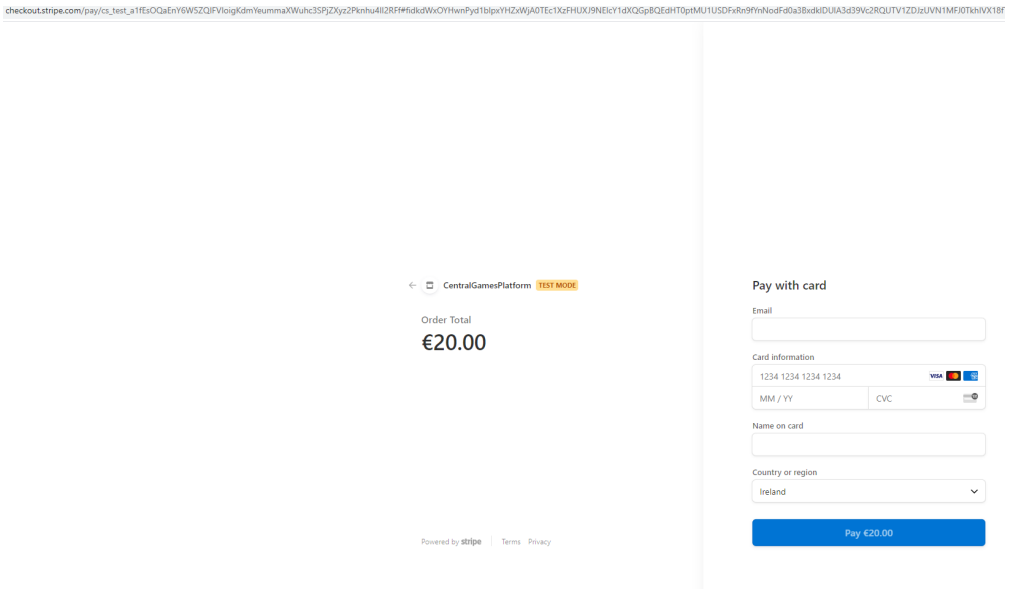

Figure 11 - Stripe checkout page

This is the Stripe checkout page that utilizes a session object to display information like the order total or the business name. In the url, the business id and session id for this transaction can be found. The user is required to enter their email address, card details, and country. Then and only then can they submit the form where money is then taken from their bank account.

The test card details are : 4242 4242 4242 4242, 01/22, 000.

Success/Failed

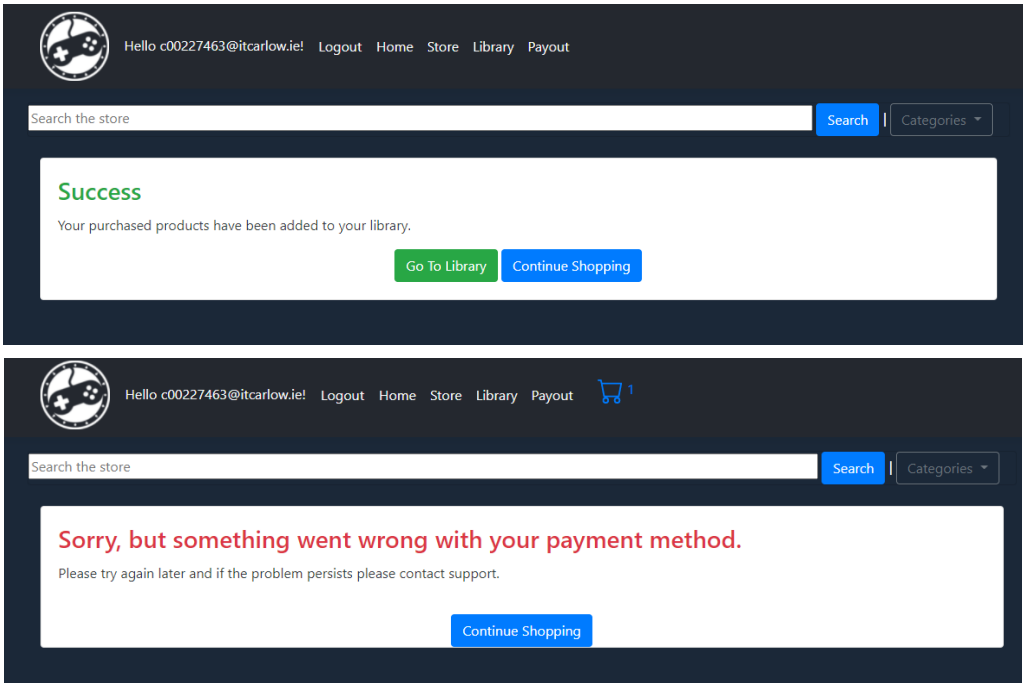

Figure 12 - Success and failure pages

After the payment details are verified by stripe, the user is then redirected to a success page or a failure page. The user is redirected to success if the payment details were verified correctly and money was taken from their account. A license for each game or casino pass purchased is generated and granted to the user, giving them ownership of them. From here, they can either select to Continue Shopping to go back to the store, or Go To Library where they will be redirected to their games library In the event of a failure, such as the payment method cannot be authorized or the user cancels the transaction, they will be redirected to the failure page.

3.6 Library
Library

Figure 13 - Library page

The user needs a way to manage their purchased games and casino passes, as well as view details on them or to play them. This is all managed by the game library page. Here the user can see the amount of casino game passes they own, the games they have active casino passes in, and the video games that they own. Here the user is able to select Play/Start Game to begin playing their game, or they can select the game's name to view its store page where they can see details about it.

3.6 Play Video Game

There are different types of video games offered on Central Games. The two main categories of video games are “Browser Games” and “Download Games”. There are different steps that are taken to play a game depending on both the category and sub category.

Download Game

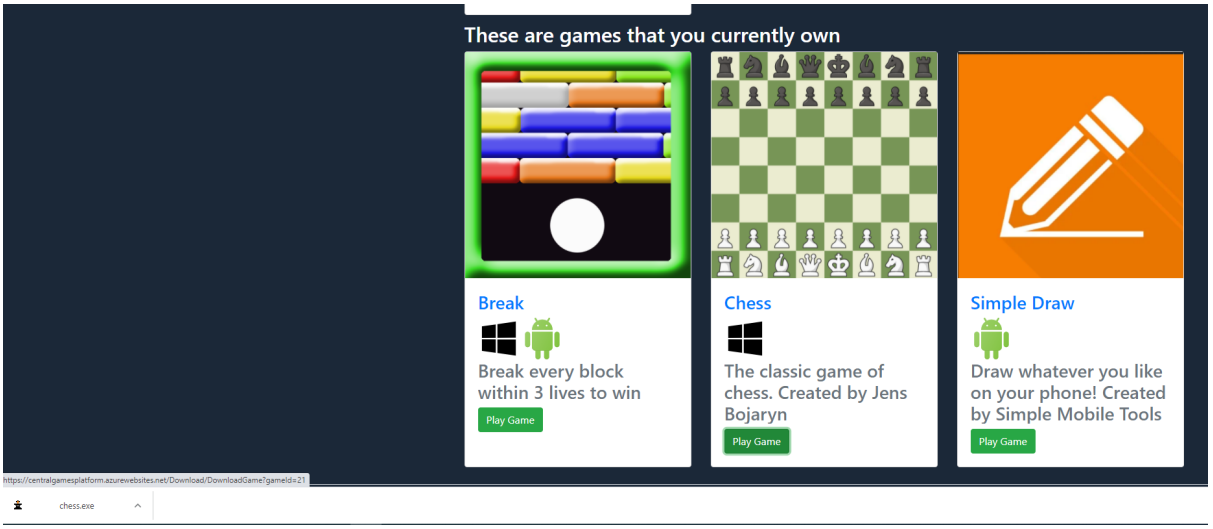

Figure 14 - Download game

Many games will be too complex to just be displayed in the browser, so it makes sense that instead the user will download and install it onto their device instead. Once the user selects Play Game for their download game of their choice, the game installer is downloaded from the database, to their device.

Browser Game

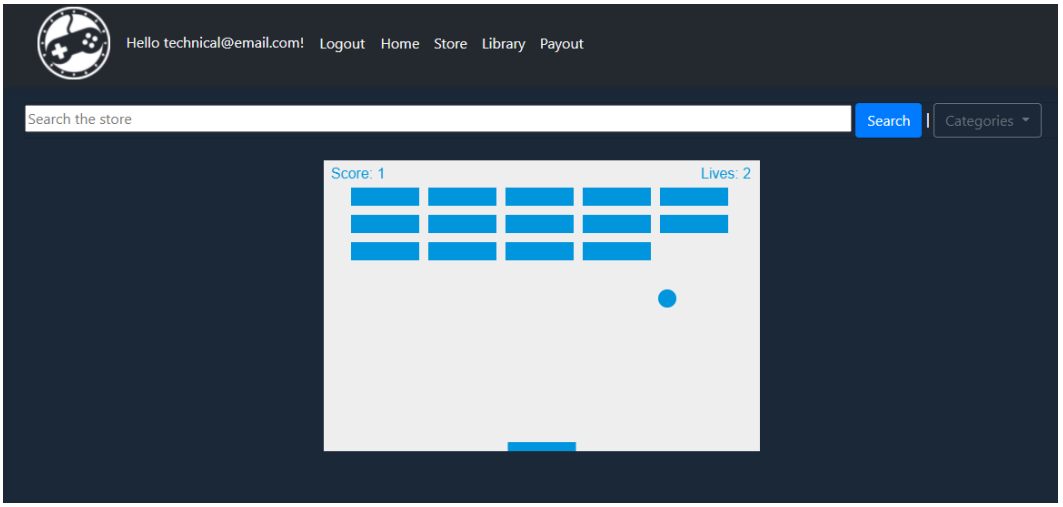

Figure 15 - Browser game page

Some games may be displayed in the browser, depending on the type of game of course. Once the user selects Play Game for a browser game, it is similar to a download game in that the game is retrieved from the database. However, it is not downloaded to the user’s machine, but instead it is embedded onto the web page for the user to play.

3.7 Start Casino Game
Age Verification

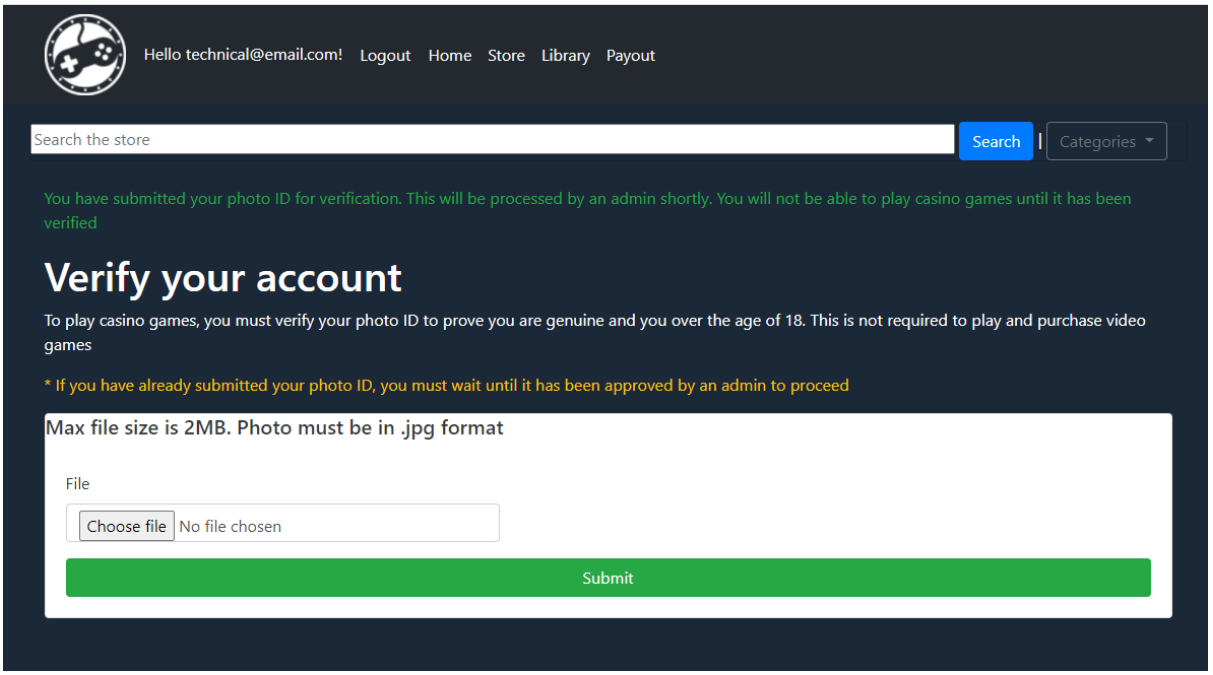

Figure 16 - Age verification page

Due to the nature of casino games, it would be both illegal and unethical to allow anyone to gamble without verifying that they are of age, so a user's age must be verified before they can play any casino games. If a user chooses to play a casino game for the first time, they will be brought to the verification page, where they are instructed that in order to play casino games, they must submit a photo ID that has to be approved by an admin. The user must include a photo in .jpg format before they can submit their request.

Casino Pass Check

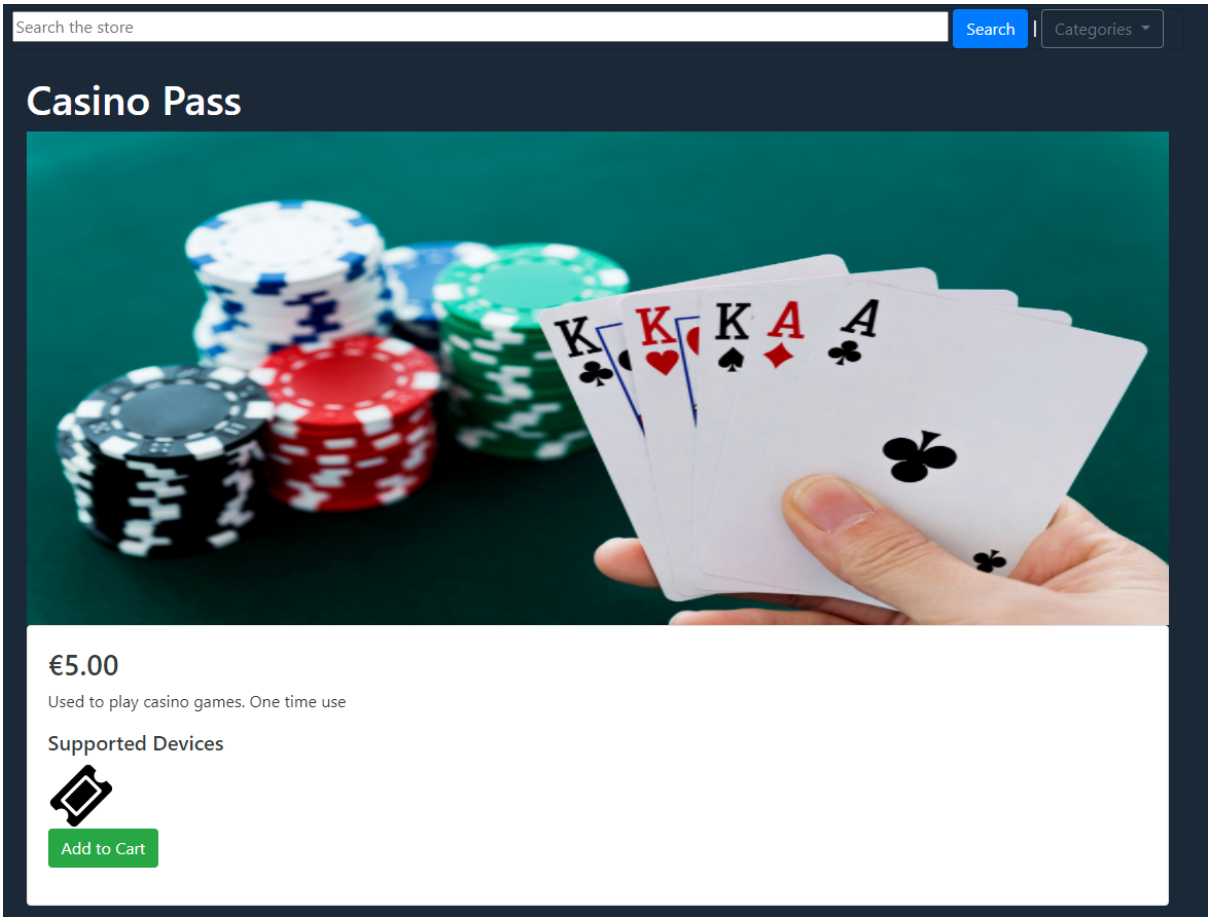

Figure 17 - Casino pass store page

If the user is verified by an admin, they are able to participate in casino games. However, a casino pass is required to play, which is consumed as soon as the game begins. If the user does not have a casino pass, they are redirected to the casino pass store page where they can purchase one like they would normally.

Start Game

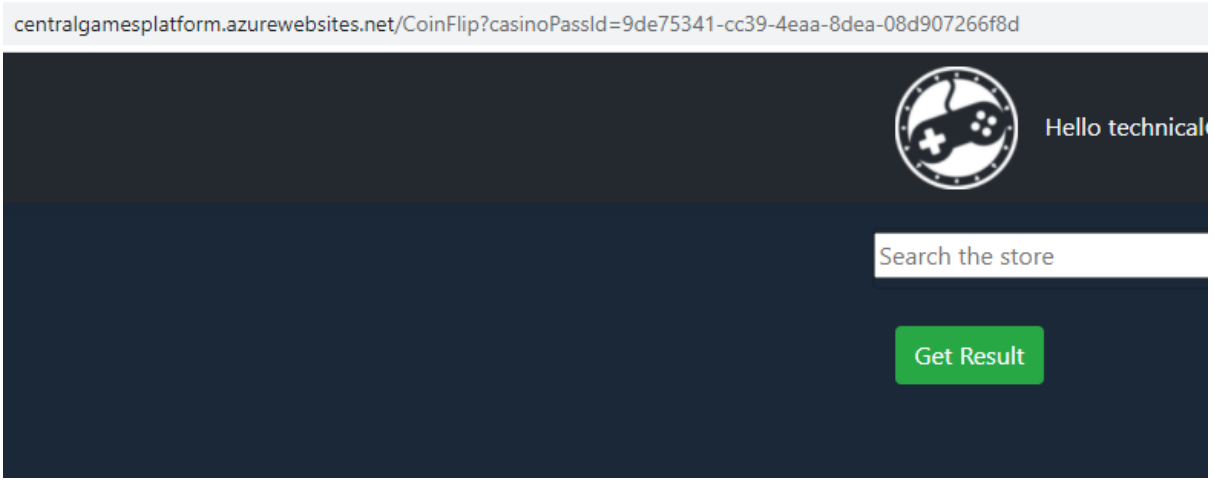

Figure 18 - Start Casino Game

Assuming the user is verified and they have a casino pass in their account, then they are able to start a casino game. For each game that is started, a casino pass is activated and assigned to this game and this game only. The active casino pass can be seen in the url.

Result

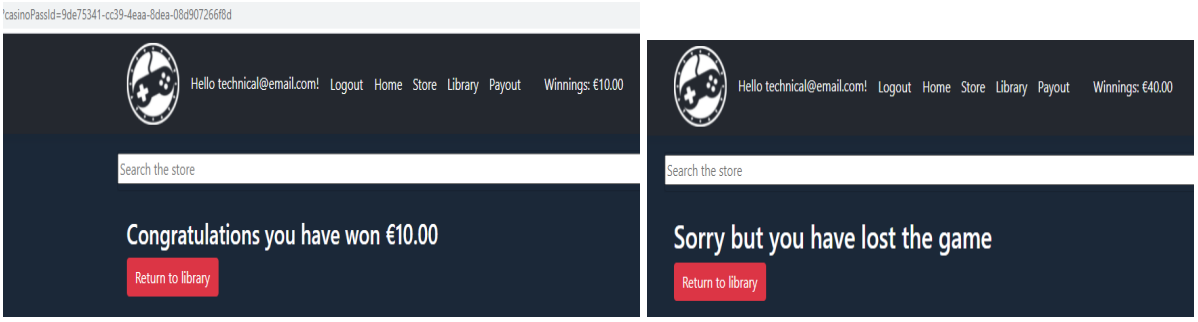

Figure 19 - Casino game result

After the casino game has concluded and the result has been determined, the user is shown the outcome. If they have won money, then they will be told as such and they will see that amount has been added to their total winnings which is viewable in the nav bar. After the result has been determined, the casino pass will expire so even if the user inputs the same casino pass id into the url, they will not be able to play again.

3.8 Payout

Similar to taking payments from customers, there is also a lot of complexity and security concerns when you wish to send money to them. This requires a large amount of work that would essentially become its own separate project. Because of this, instead PayPal is going to be used to handle payouts as they have the infrastructure set up already, and they provide a RESTful API that can be incorporated into any project. There is little documentation or information available on the payouts API for .NET, so it was no small task using it for this project and required days of work to incorporate. How it works is that a payout request is created containing the amount of money you wish to transfer and the email of the PayPal account you wish to send money to. If successful, a payout response is returned which can then be executed to transfer the funds over.

Payout

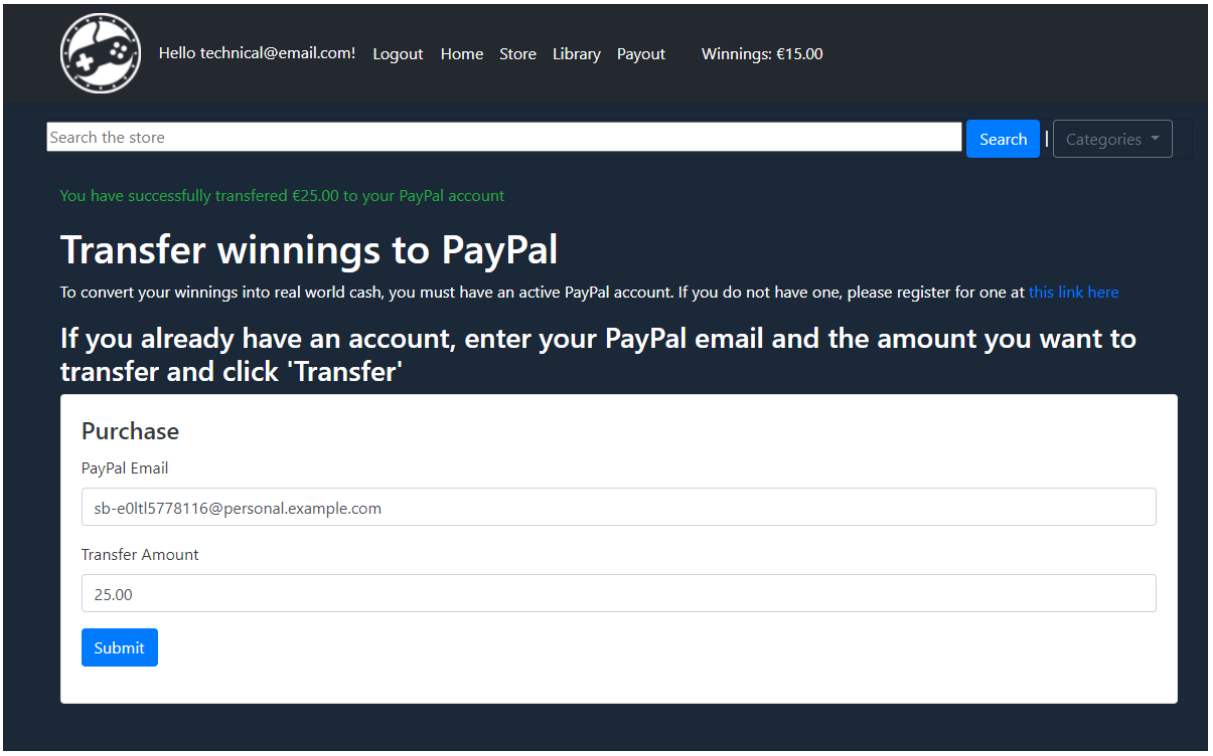

Figure 20 - Payouts page

When a user wins money from any casino game, they need a way to transfer their winnings into usable currency. To do this, the PayPal Payouts API is used to facilitate this request. A user must have a registered PayPal account to facilitate this request, so a link is provided that will redirect them to the PayPal registration page in case they do not already have an account. The user is prompted to enter the email address linked to their paypal account and the amount they wish to transfer. The minimum transfer amount is €5 as paypal will take a small cut for every transfer. Assuming that an account already exists with this email and the amount they have entered is valid and is less than or equal to the amount they have in their winnings, then the payout will process successfully and they will receive a message informing them as such.

PayPal test email for payouts: sb-e0ltl5778116@personal.example.com

3.9 Admin

An admin will be given the responsibilities of verifying users identifications and for adding, updating and maintaining games present on the store. This includes the product pages themselves, as well as any files associated with them like installers or source code.

Only accounts with admin privileges can access the admin portal.

Admin email: admin@centralgames.com
Admin password: Admin123!
Admin URL: centralgamesplatform.azurewebsites.net/admin

Access Denied

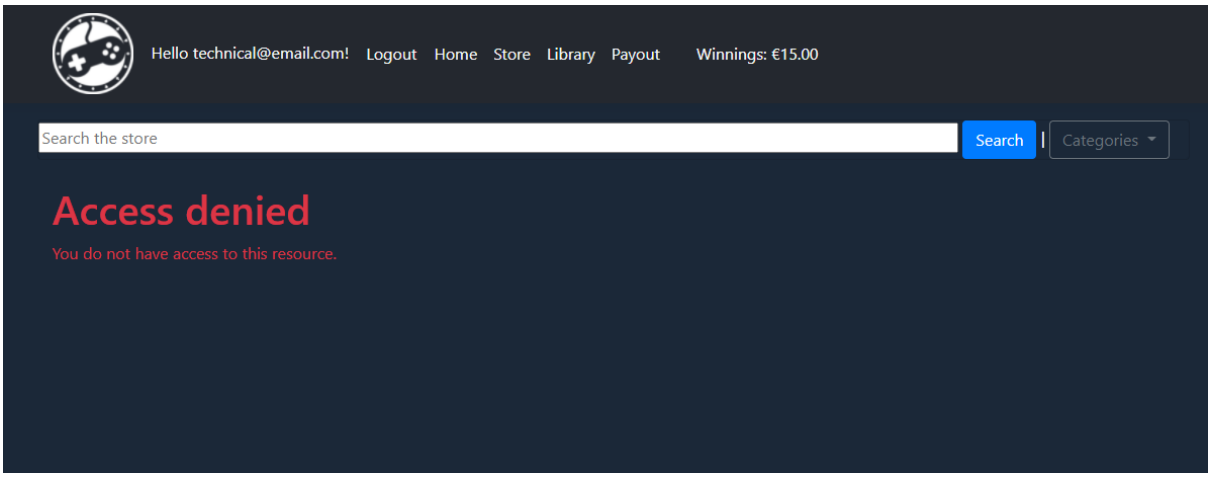

Figure 21 - Access denied page

Admin privileges contain very powerful commands and state modifying requests, and that is no different for this project. Admin privileges should be accessible only to those with the correct authentication. If any user who does not have access to admin privileges attempts to access the admin portal, they will be denied access.

Admin Home

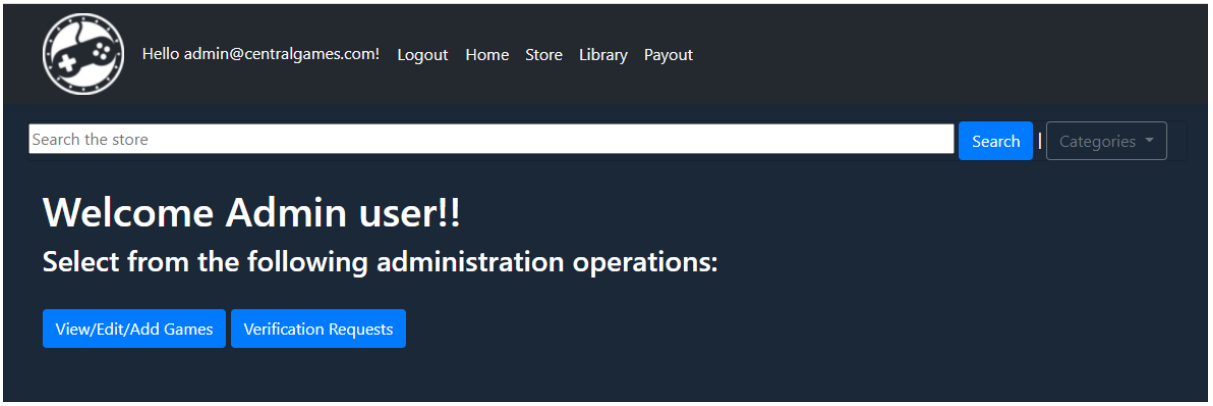

Figure 22 - Admin home page

This is the homepage for the admin. From here, they can access any of the admin commands available.

View/Edit/Add Games

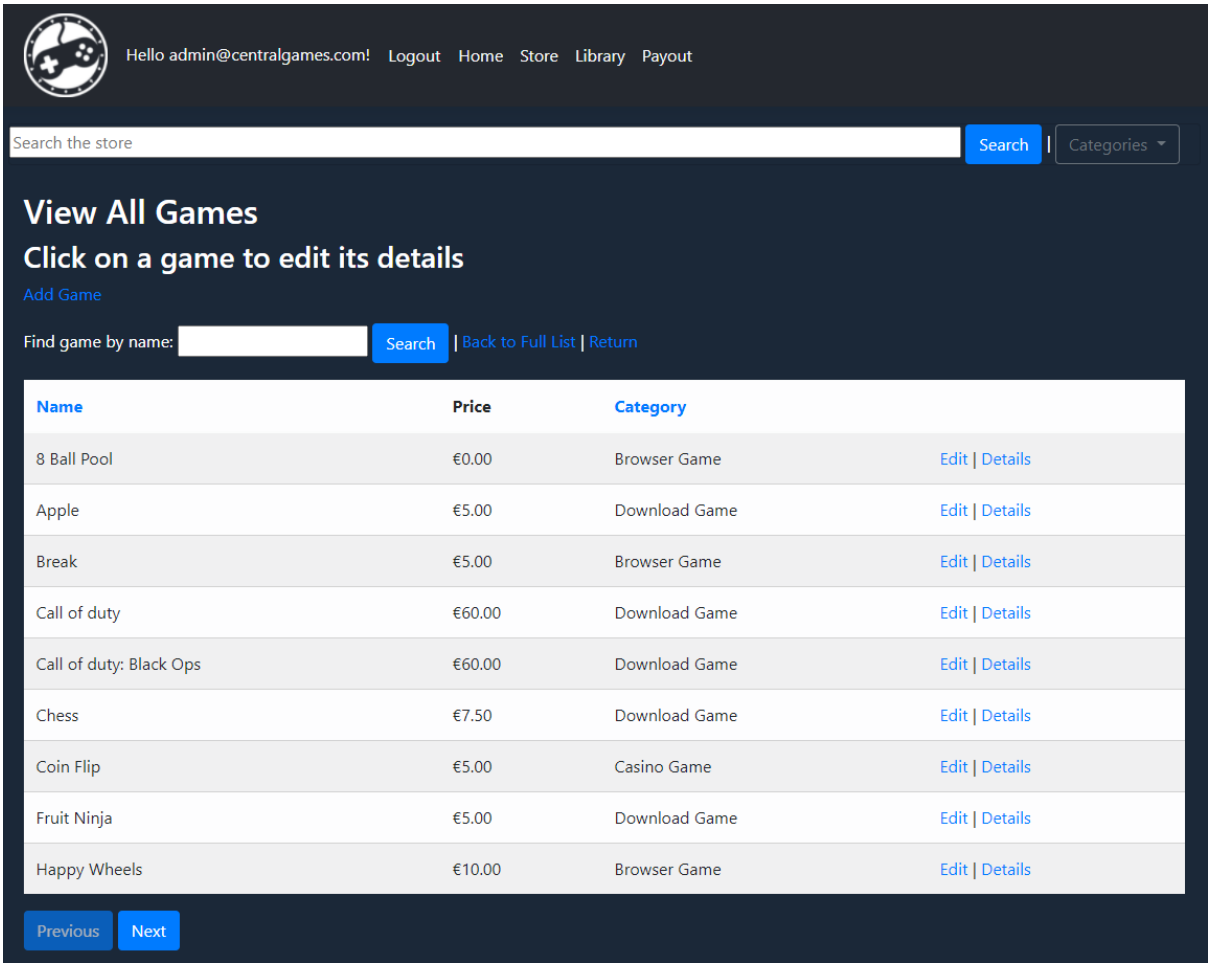

Figure 23 - View/Edit/Add games page

This is the page that the admin can use to manage and maintain all of the games on the website. Here they are given many options. They can select Add Game to add a new game to the store. If they wish to find a game in particular, they can enter a search term into the search bar and select search, or they can select Back to Full List to display the full list of games. If they wish, they can select Return to return to the admin home page. Games are displayed in pages so if they wish to search through more games, they can select Next or Previous to navigate the pages. It is possible to select Name or Category to sort by them in ascending or descending order. Finally, an admin can choose Details to view additional details on the game, or Edit to edit any details on the game or add/update game files.

Details

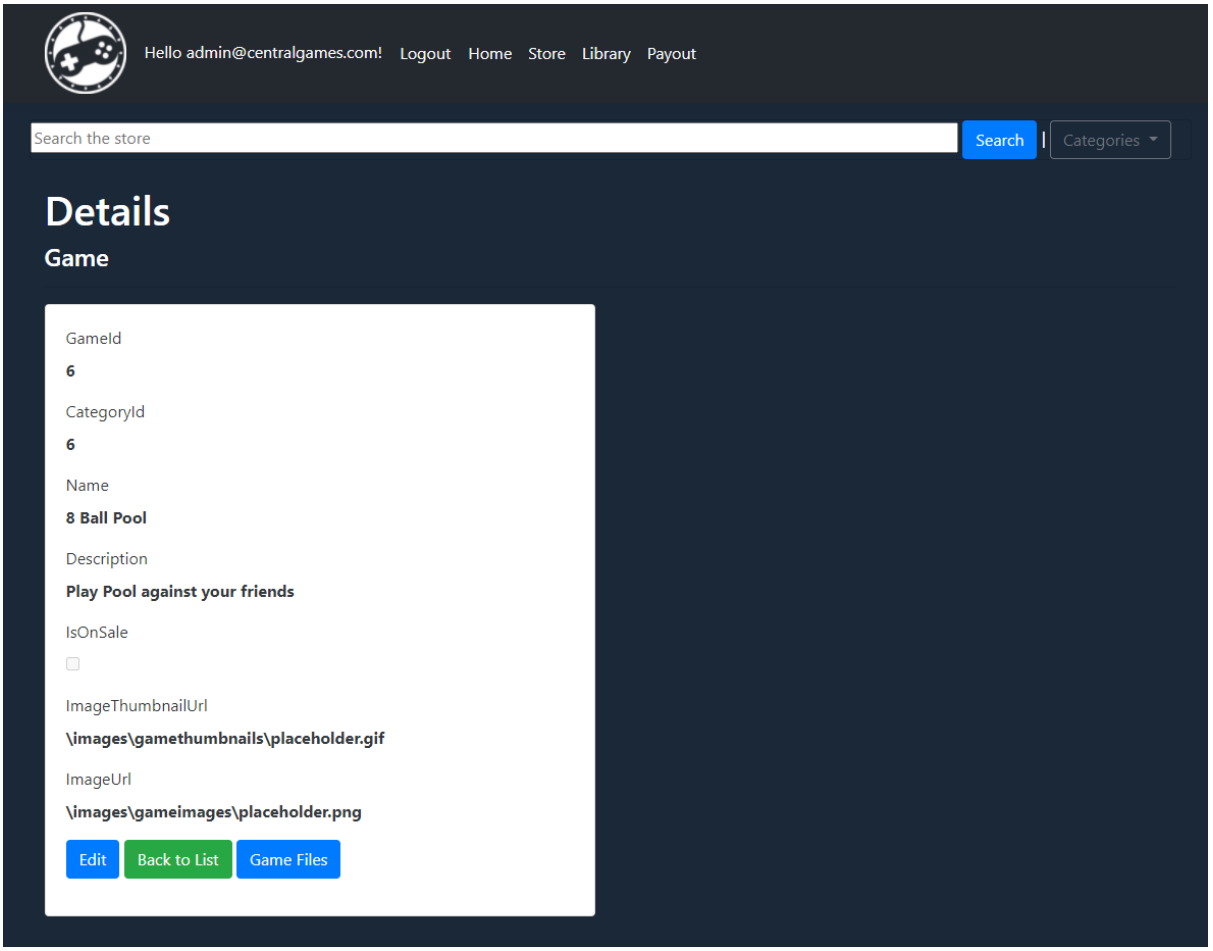

Figure 24 - Game details page

Sometimes the admin may just wish to view the details of a games store page without making any alterations to it. The details page allows for that, as this is a read only page, which makes it so that the details are displayed but there is no possible way to change any of them. From here there are 3 options. The admin can select to Edit the games store page, they can select Back to List to return to the list of games, or they can select Game Files to view/update game files for this particular game. The Game files option is only available for browser and download games, since casino games have to be worked into the web app itself.

Edit

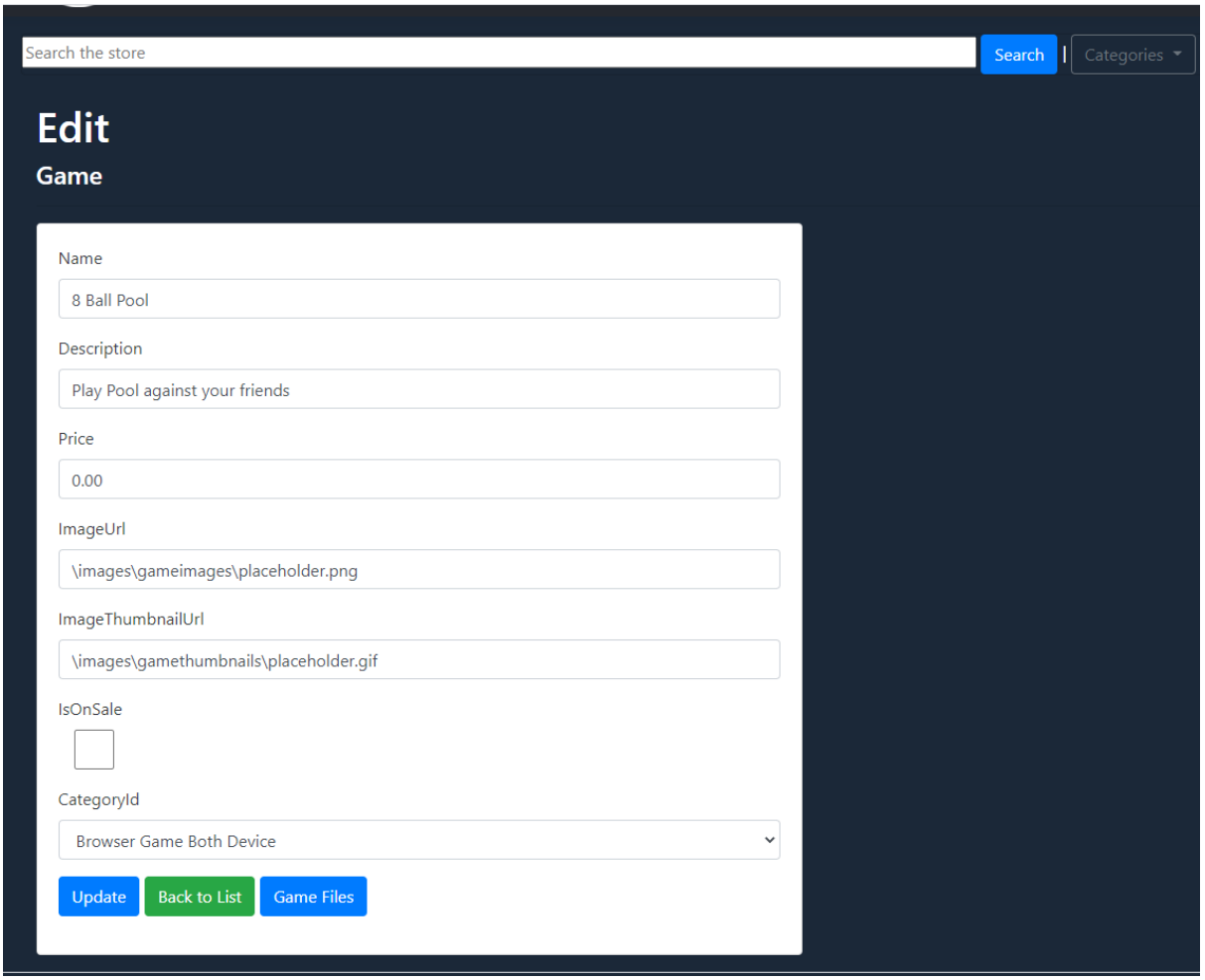

Figure 25 - Edit game page

There may be changes to a games store page that a publisher may require. This may be the description of a game, a screen shot or even the price. This can all be done by an Admin from the edit page. Here they are able to change any details related to the game, including the category, image thumbnail, and the option to put it on sale on the home page. Once they select Update, the details entered will be used to overwrite the pre-existing details. Alternatively, they can select Back To List to return to the list of games or Game Files if they wish to add/update game files associated with the game. Game Files is only available for browser or download games.

Add Game Files

If the admin selects Game Files from a game's edit or details page and there are no game files present, they are redirected to the Add Game Files page.

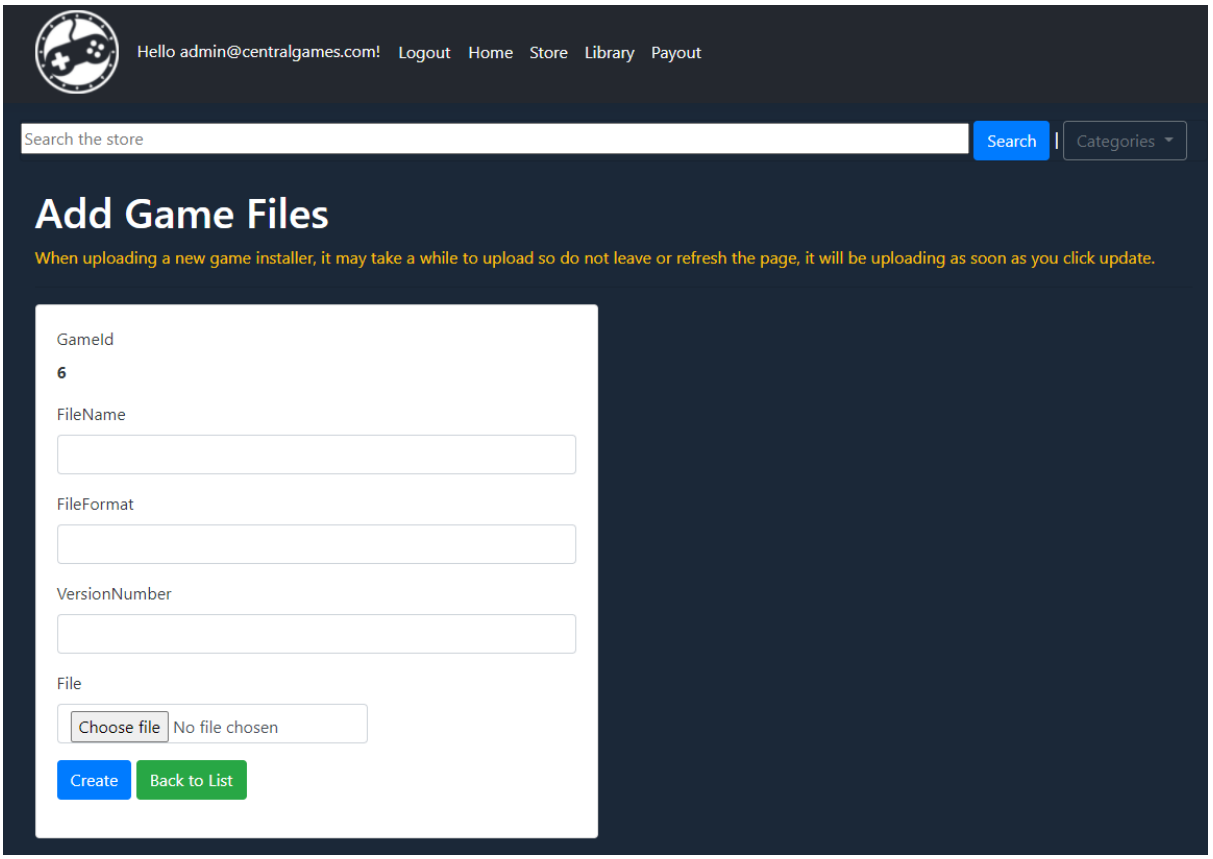

Figure 26 - Add game files page

Here the admin is able to see the current game Id for the game they are adding game files to. Every field is required and a file must be attached to the game. This can be an installer in the form of a .exe or .apk, or alternatively it can be a javascript or html file for browser games. From here the Admin has the option to select Create, which will add the game files to the database so then when customers choose to download/play the game, these are the game files that will be delivered to them.

Update Game Files

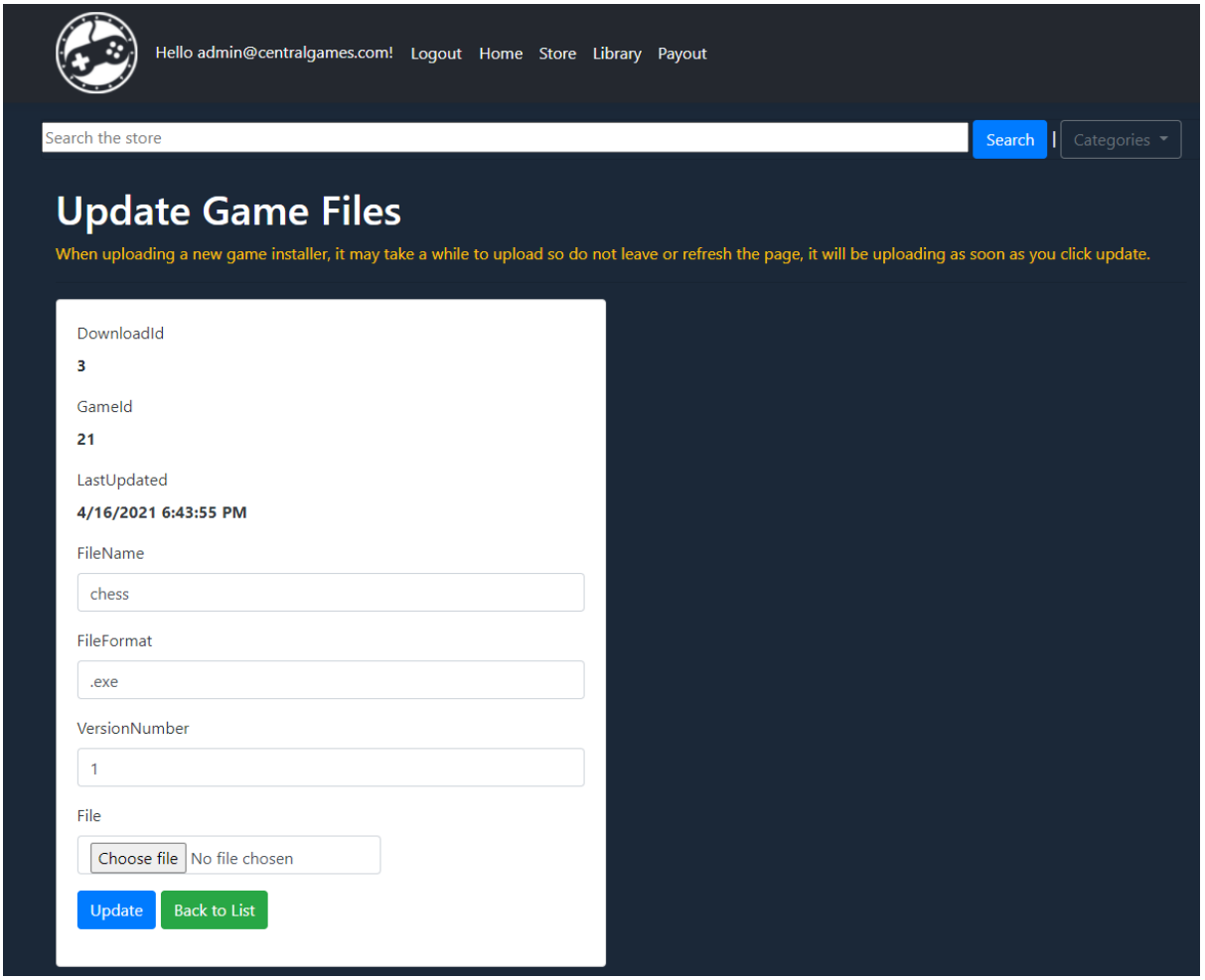

Figure 27 - Update Game Files page

The update game files page will allow an admin to update game files for games that already have game files associated with it. Here they are able to alter the file name, file format, file version or, if they wish, they can upload a new game file that will replace the pre-existing one

Details Game Files

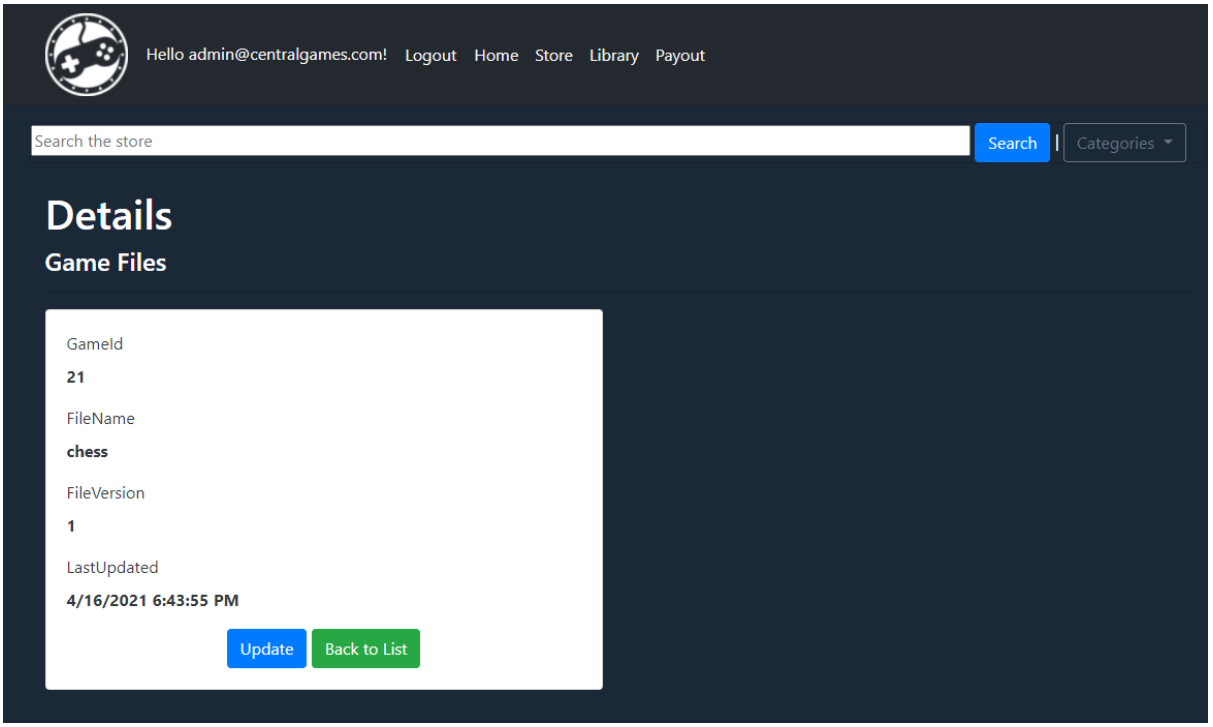

Figure 28 - Details Game Files Page

Here the admin is able to see the details on the game files for the current game. They can select Update to go to the update page for this particular game, or they can select Back to List to return to the list of games.

View Verification Requests

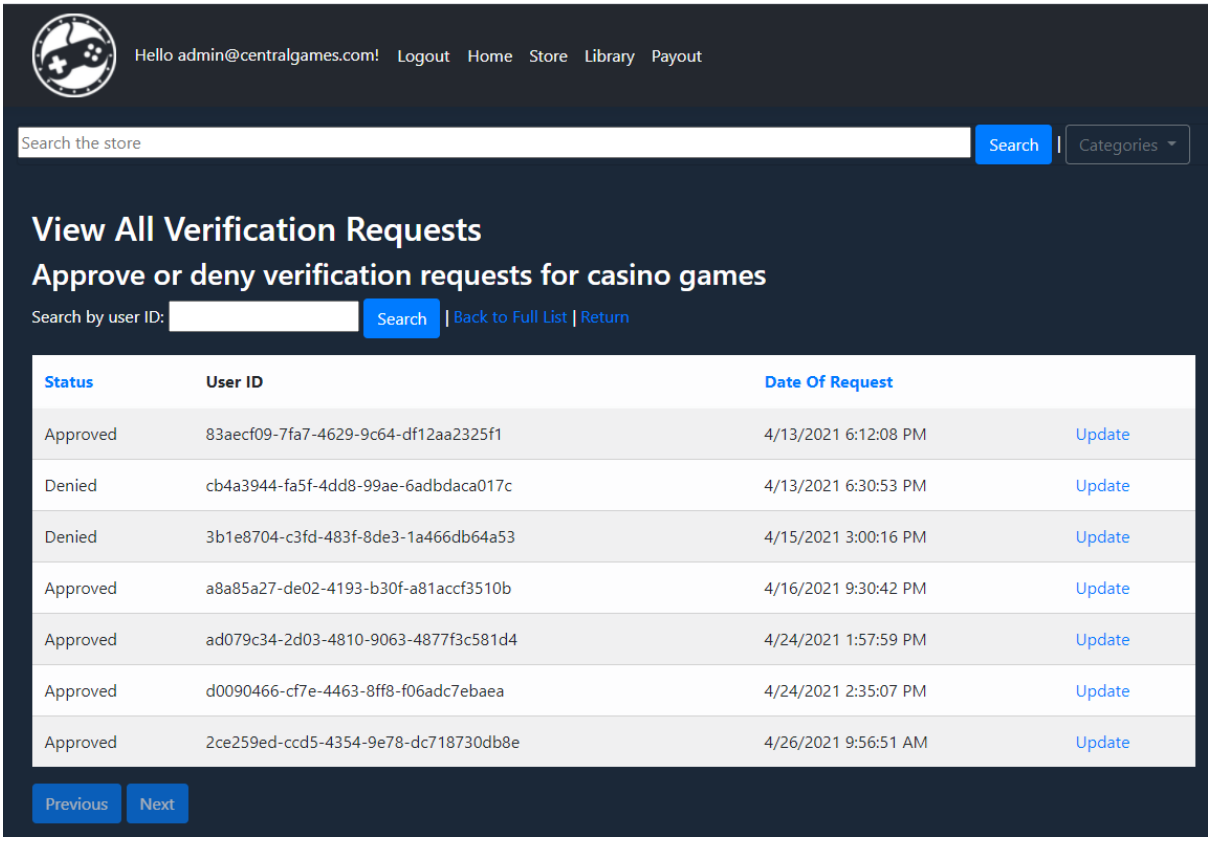

Figure 29 - View verification requests page

To play casino games, users must submit their photo identification to prove that they are of age. The admin can view all of the verification requests on this page. They have the option to search by user ID if they wish, or they can select Back to Full List to go back to the full list to view every verification request. Selecting Return will return the admin to the home page. The admin can select Status to order by status, or Date of Request to sort by date the request was created. Selecting Next or Previous will allow the admin to navigate between pages. Selecting Update on a request will redirect the admin to the update page where they can approve or deny a request.

Update Verification Request

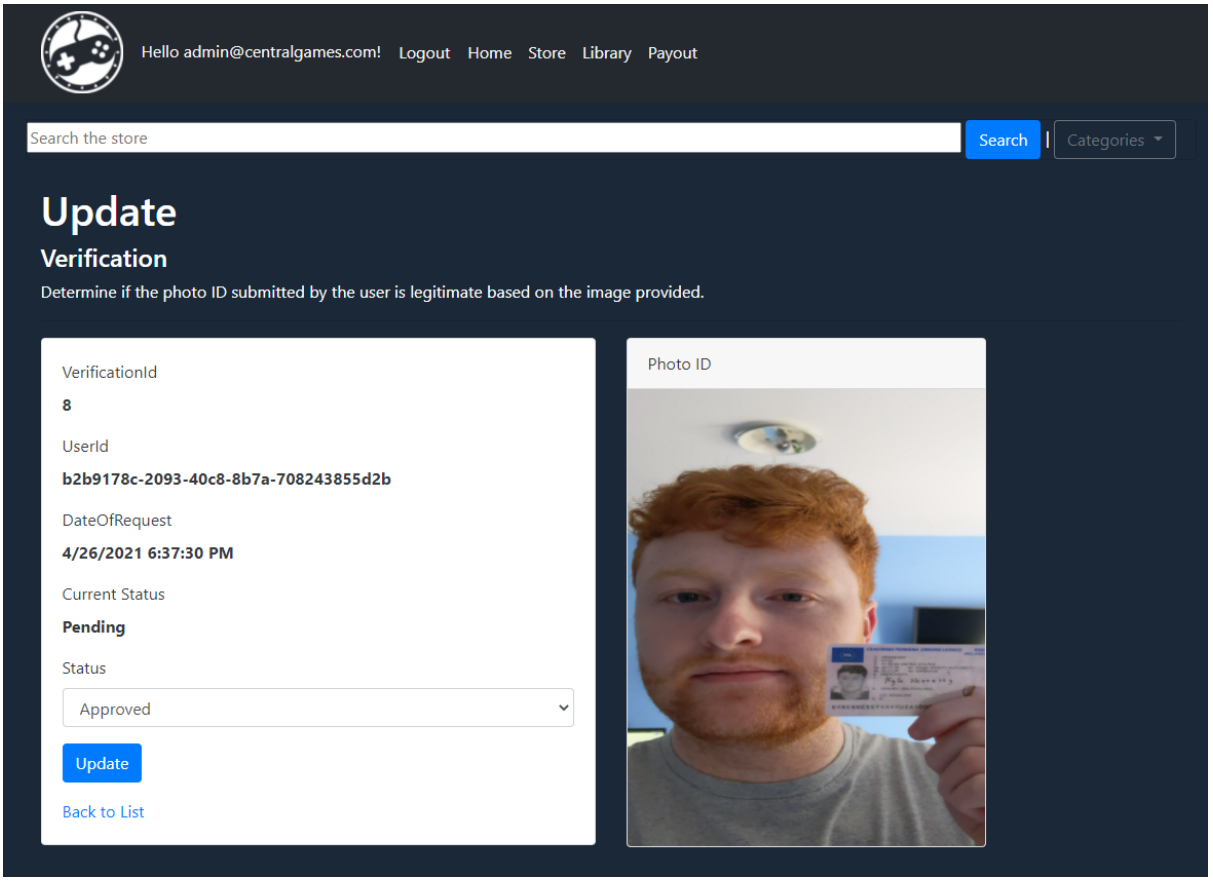

Figure 30 - Update verification request page

The admin is able to look at the image submitted by this particular user where they can decide if the photo ID submitted by the user is in fact authentic and the user is who they say they are. The admin is able to see the ID for this particular request, as well as the user id of this particular user. The date requested and the current status of the request is also shown. The admin can select either Approved or Denied from the dropdown where they can then select Update to update the verification request, or they can select Back To List to return to the list of verification requests.

Details Game Files

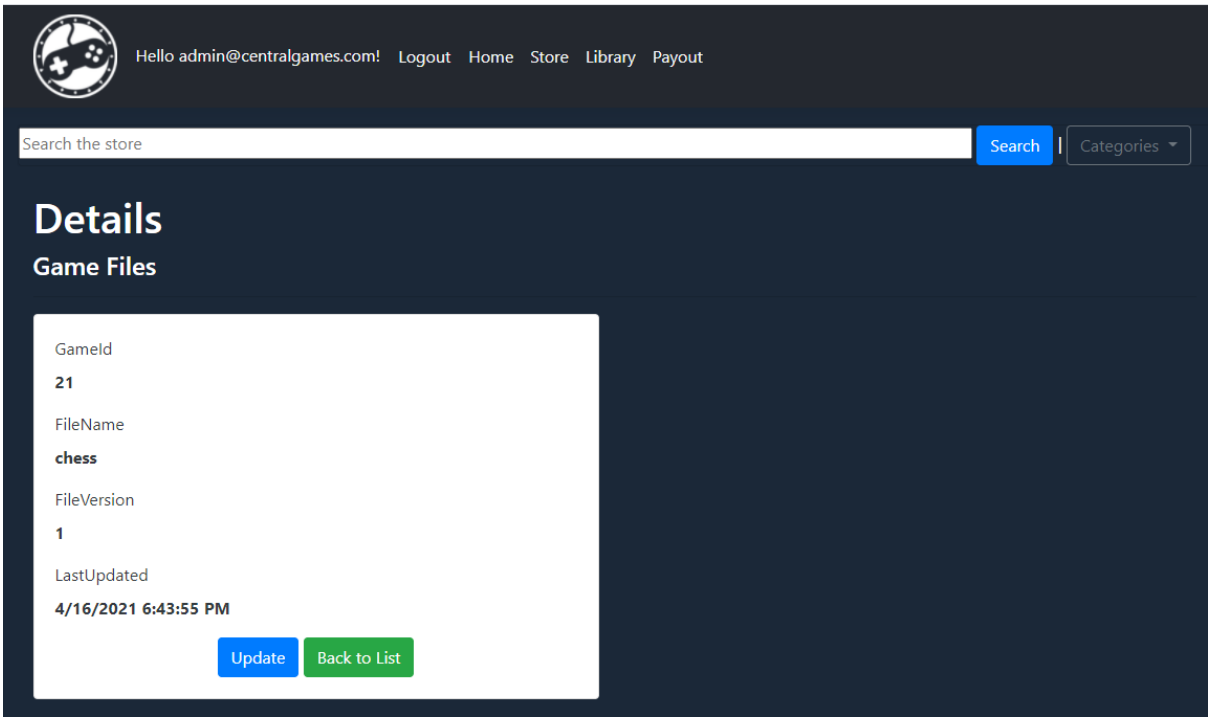

Figure 31 - Details game files page

Here the admin can view the details of a game's game files without updating any of it. Here they can view the name of the file, the game id that the game files are for, the file version, and when it was last updated. Here the admin can select Update to update the current game files, or Back to List to return to the list of games.

4.APIs, Packages and Frameworks

4.1 Stripe Checkout Session API and checkout page

As explained earlier, the Stripe Checkout Session API is used to generate a checkout session containing necessary details like the order total and accepted payment methods. To use Stripe, the secret key must be set in the constructor so that Stripe can transfer money to Central Games' Stripe account. **The secret key should not be viewable to anyone other than the business owner.** This is done in the Payment Controller like so:

PaymentController.cs

```
public PaymentController()
      {


          //Other code
           StripeConfiguration.ApiKey =
"sk_test_51IB0Z4BTwx1LYfRRop1pYWwRVKBAs0K7KZBRbKTubudFUXJPN5BlooRa
hipg8qIkpIQ49d6c4YZE9ErcziO23QtR00rzwq6cbk";
      }
```

```
[HttpPost("create-checkout-session")]
      public IActionResult CreateCheckoutSession()
      {
           decimal orderTotal;
           _shoppingCart.ShoppingCartItems =
_shoppingCart.GetShoppingCartItems();
           orderTotal = _shoppingCart.GetShoppingCartTotal() *
100;
           if(orderTotal == 0.00M)
          {
                return RedirectToAction("Success");
          }
           var options = new SessionCreateOptions
           {
                PaymentMethodTypes = new List<string>
                {
                     "card"
                },
                LineItems = new List<SessionLineItemOptions>
                {
                     new SessionLineItemOptions
                     {
```

```
                            PriceData = new
SessionLineItemPriceDataOptions
                            {
                                  UnitAmount = (long?)orderTotal,
                                  Currency = "eur",
                                  ProductData = new
SessionLineItemPriceDataProductDataOptions
                                  {
                                        Name = "Order Total"
                                  },

                            },
                            Quantity = 1
                      },
                 },
                 Mode = "payment",
                SuccessUrl =
"https://centralgamesplatform.azurewebsites.net/Payment/Success?se
ssion_id={CHECKOUT_SESSION_ID}",
                CancelUrl =
"https://centralgamesplatform.azurewebsites.net/Payment/Failed",
           };
            var service = new SessionService();
            Session session = service.Create(options);
            return Json(new { id = session.Id });
      }
```

What is happening here is that once the Create Checkout Session action is requested, the order total is calculated from every item currently present in the shopping cart. If the order total is 0, then the payment process is skipped. Otherwise, the session object is created for Stripe. Here, we set the following properties: Payment Method Types = “card”, UnitAmount = “orderTotal”, Name = “Order”, Quantity = 1, Mode = “payment”, SuccessUrl = "https://centralgamesplatform.azurewebsites.net/Payment/Success?session_id={CHECKOUT_SESSION_ID}", CancelUrl = "https://centralgamesplatform.azurewebsites.net/Payment/Failed". A session object is then created using all of these set properties, and the session Id is then sent to the view. With the Id present, javascript will then redirect to the Stripe Checkout Page for that current checkout object. When the project goes into full production, all that needs to be changed is the secret key.

The checkout page is a pre-built page provided by Stripe that is accessed by including the following javascript code in the view like so.

Payment/Index.cshtml

```
<button type="button" class="btn btn-primary" id="checkout-button"
asp-controller="Payment"
asp-action="CreateCheckoutSession">Checkout</button>

<script type="text/javascript">
      // Create an instance of the Stripe object with your
publishable API key
      var stripe =
Stripe('pk_test_51IB0Z4BTwx1LYfRREoDEBBJOh4HQM4tCzZgvmqRqnutsLFWU2
rzPvaUT0pa7vTVp5WqKMMPxyx1nFzeeLjW4o7Eo00umgSxWfI');
      var checkoutButton =
document.getElementById('checkout-button');

      checkoutButton.addEventListener('click', function () {
      // Create a new Checkout Session using the server-side
endpoint you
      // created in step 3.
      fetch('/create-checkout-session', {
            method: 'POST',
      })
            .then(function (response) {
                  return response.json();
            })
            .then(function (session) {
                  return stripe.redirectToCheckout({ sessionId:
session.id });
            })
            .then(function (result) {
                  // If `redirectToCheckout` fails due to a browser
or network
                  // error, you should display the localized error
message to your
                  // customer using `error.message`.
                  if (result.error) {
                  alert(result.error.message);
                  }
            })
            .catch(function (error) {
                  console.error('Error:', error);
            });
```

```
      });
</script>
```

This code will create a stripe session request with the public stripe key. This key can be accessible to anyone as it only is used to make requests and cannot actually authorize requests unless the secret key is supplied to the request in the controller. On a POST request, after the Checkout Session object has been created in the controller, the session Id returned is used to redirect the user to the Stripe Checkout page. When the project goes into full production, all that needs to be changed is the public key

4.2 PayPal Payouts API

The PayPal Payouts API is a RESTful API that is used to create payout requests so that money can be transferred from a business account to an individual. The accompanying .NET SDK that was used for this project has almost no documentation available and because of its niche use case, there are 0 guides or tutorials available online. Because of this, it was very difficult to incorporate into the project. The PayPal Payouts API requires both a Client ID and Client Secret ID to be set so that the PayPal Environment can be initialized to facilitate the sending and receiving of requests. This is done like so :

PayPalClient.cs

```
public class PayPalClient
      {
      public static PayPalEnvironment environment()
      {
            return new SandboxEnvironment(

System.Environment.GetEnvironmentVariable("PAYPAL_CLIENT_ID") !=
null ?

System.Environment.GetEnvironmentVariable("PAYPAL_CLIENT_ID"):
"AVESW8pAcvJQVcH6G11Z193gImhx8H8DQ9NIwV9sgP1XFZWaDztQym-JaolO1XX-g
GsL69VLChgzePxb",

System.Environment.GetEnvironmentVariable("PAYPAL_CLIENT_SECRET")
!= null ?

System.Environment.GetEnvironmentVariable("PAYPAL_CLIENT_SECRET"):
"ELslSeKzMAEFceAdQroyeo3gLrCU0f6G2b8zhQl54TpPPRfAXn1u5kfCT_MjyQHZH
lRHc4D1cQFGNodt");
      }
      public static HttpClient client()
      {
```

```
            return new PayPalHttpClient(environment());
      }

      public static HttpClient client(string refreshToken)
      {
            return new PayPalHttpClient(environment(),
refreshToken);
      }
```

Once any request is made, the client and secret keys are used in order to link it with the business account for Central Games Platform. Once the project is moved into full production, all that needs to be changed is the client and secret keys.

The Payouts API requires at the very least, an email address and an amount to be transferred. This is submitted by the user in a form located in /Payment/Index. This is then handled in the Payment Controller like so:

```
[HttpPost]
       public IActionResult Index(Payout submittedPayout)
      {
             string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
             decimal walletBalance =
_walletRepository.RetrieveBalance(userId);
             int walletId =
_walletRepository.RetrieveWalletId(userId);
             if (walletId == 0)
            {
                  ViewBag.ErrorMessage = "You have never won any
money in a casino game before. Come back here when you have";
                  return View();
            }
             submittedPayout.WalletId = walletId;
             if(submittedPayout.AmountTransfered < walletBalance &&
submittedPayout.AmountTransfered >= 5.00M)
            {
                  string amountTransfered =
submittedPayout.AmountTransfered.ToString();
                  string email = submittedPayout.PayPalEmail;
                  var response = CreatePayout(amountTransfered,
email);
                  HttpResponse createPayoutResponse =
```

```
response.Result;
                var payout =
createPayoutResponse.Result<CreatePayoutResponse>();

                submittedPayout.PayPalBatchId =
payout.BatchHeader.PayoutBatchId;
               _walletRepository.SubtractFromWallet(userId,
submittedPayout.AmountTransfered);
               _payoutRepository.CreatePayout(submittedPayout);
                ViewBag.SuccessMessage = "You have successfully
transfered €" + amountTransfered + " to your PayPal account";
            }
           else
           {
                ViewBag.ErrorMessage = "We were unable to
transfer funds into the PayPal account with the email provided. Do
you have enough in your wallet?";
           }
            return View();
     }

      private static CreatePayoutRequest buildRequestBody(string
amountTransfered, string email)
      {
            var request = new CreatePayoutRequest()
            {

                SenderBatchHeader = new SenderBatchHeader()
                {
                     EmailMessage = "You have received the
following amount from Central Games Platform: €" +
amountTransfered,
                     EmailSubject = "Central Games Platform
transfer"
                },
                Items = new List<PayoutItem>(){
                     new PayoutItem(){
                          RecipientType="EMAIL",

                          Amount=new Currency(){
                               CurrencyCode="EUR",
```

```
                                    Value=amountTransfered,
                               },
                              Receiver=email,

                         }
                    }
               };
               return request;

          }
          public async static Task<HttpResponse> CreatePayout(string
amountTransfered, string email)
          {
               try
               {
                    PayoutsPostRequest request = new
PayoutsPostRequest();

request.RequestBody(buildRequestBody(amountTransfered, email));
                    var response = await
PayPalClient.client().Execute(request);
                    var result =
response.Result<CreatePayoutResponse>();
                    return response;
               }
               catch
               {
                    return null;
               }
          }
```

Once the Index action is accessed with a POST request, the process of creating a payout begins. First a check is done to see if there is enough money in the users wallet and that it is over €5.00. If it is, then the CreatePayout method is called using the amount to be transferred and the email as parameters. The CreatePayout method creates a new PayoutsPostRequest object that is to be used to build the body of the response. A call is then made to the buildRequestBody method with the email and amount to be transferred being sent as parameters. In this method, a CreatePayoutRequest object is created and is populated with the following properties: EmailMessage = “You have received the following amount from Central Games Platform: €” + amountTransfered, EmailSubject = “Central Games Platform Platform Transfer”, RecipentType = “EMAIL”, CurrencyCode=”EUR”, Value=amountTransfered, Receiver=email.

The CreatePayoutRequest object is then returned back to the CreatePayout method. This object is then used to call the PayPalClient class mentioned earlier so that it can execute the CreatePayoutRequest. The result of this request is then returned back to the Index Action. Now that the request has been executed, the batch id is taken from the result and is recorded in the database, along with the details on the user and the amount transferred. A success message is then added to the ViewBag so a message can be displayed in the view to the user informing them that it was a success.

4.3 Entity Framework Core (Azure SQL Database)

Since this is an e-commerce web app containing a large and varied amount of data, a database is being used to store all of this. This means that there are many instances where SQL queries are used to query the database to read and write data. Additionally, instead of adding tables manually to the database, a code first approach to database development would have sped up development significantly. Entity Framework Core allows you to do just that. EFC is an object-database mapper for .NET Core. [4] It enables you to define your data in a model like for Categories:

Category.cs

```
public class Category
    {
       public int CategoryId { get; set; }
       public string CategoryName { get; set; }
       public string CategoryDescription { get; set; }
       public List<Game> Games { get; set; }
    }
```

The properties that can be directly mapped to the table for Categories is clearly identified as : int CategoryId, string CategoryName, string CategoryDescription. Then in the context for the database, a property can be created for Category that states that a table is to be created using the Category object. This context acts as a bridge between the application and the database. This can be seen in MyDatabaseContext like so:

```
 public class MyDatabaseContext :
IdentityDbContext<ApplicationUser>
     {
     public MyDatabaseContext (DbContextOptions<MyDatabaseContext>
options) :
           base(options)
     {
     }
     public DbSet<Category> Categories { get; set; }
}
```

The database context must be added as a service at startup so that it can be used throughout the entire project. This is done in the ConfigureServices method of Startup.cs like so:

```
public void ConfigureServices(IServiceCollection services)
      {
            services.AddDbContext<MyDatabaseContext>(options =>

options.UseSqlServer(Configuration.GetConnectionString("DefaultCon
nection")));
}
```

The connection string "DefaultConnection" can be found in the appsettings.json file. *The connection string for the database allows anyone to gain access to the database so it is confidential information.*

```
"DefaultConnection":
"Server=tcp:centralgamesplatformdbserver.database.windows.net,1433
;Initial Catalog=centralgamesplatformdb;Persist Security
Info=False;User
ID=superuser;Password=darak123!;MultipleActiveResultSets=False;Enc
rypt=True;TrustServerCertificate=False;Connection Timeout=0;"
```

To create the code-first migration for the categories table, use the command "Add-Migration <MigrationName>" in the package manager console in visual studio. This will create a migration that can be used to update the database. This can be seen in the migration below:

InitialMigration.cs

```
protected override void Up(MigrationBuilder migrationBuilder)
      {
            migrationBuilder.CreateTable(
                  name: "Categories",
                  columns: table => new
                  {
                  CategoryId = table.Column<int>(nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                  CategoryName = table.Column<string>(nullable:
true),
                  CategoryDescription =
table.Column<string>(nullable: true)
                  },
                  constraints: table =>
                  {
```

```
                table.PrimaryKey("PK_Categories", x =>
x.CategoryId);
                });
     }
 protected override void Down(MigrationBuilder migrationBuilder)
     {

          migrationBuilder.DropTable(
                name: "Categories");
     }
```

This generates 2 methods. Up, which is used to update the database, and Down, which is used to revert the changes in the database. To apply these changes to the database, simply use the command “Update-Database” in the package manager console which will use the Up method to generate the SQL code needed to update it. The code automatically generated for the Categories migration can be seen here:

```
CREATE TABLE [dbo].[Categories] (
     [CategoryId]          INT             IDENTITY (1, 1) NOT
NULL,
     [CategoryName]        NVARCHAR (MAX) NULL,
     [CategoryDescription] NVARCHAR (MAX) NULL,
     CONSTRAINT [PK_Categories] PRIMARY KEY CLUSTERED
([CategoryId] ASC)
);
```

4.4 LINQ (Language Integrated Query)

To query a database using entity framework core, LINQ statements are used to generate SQL queries. It is an extension of C# and is included with ASP.NET Core MVC automatically. LINQ statements can be written in either lambda expressions, or in a similar format to SQL statements, without having to worry about the syntax of SQL statements. A lamda expression used to return a Game using an ID can be found in GameRepository like so:

GameRepository.cs

```
public Game GetGameById(int gameId)
      {
           return _myDatabaseContext.Games.FirstOrDefault(g =>
g.GameId == gameId);
      }
```

Alternatively, a LINQ statement that uses a similar format to SQL statements to return a Wallet by ID can be found in WalletRepository like so:

WalletRepostiory.cs

```
public int RetrieveWalletId(string userId)
     {
          Wallet wallet = (from w in _myDatabaseContext.Wallets
                           where w.UserId == userId
                           select w).SingleOrDefault();
          if (wallet == null)
          {
               return 0;
          }
          int result = wallet.WalletId;
          return result;
     }
```

This approach to querying the database allows you to easily use C# variables in a query, as well as work with data retrieved from the database with ease, as can be seen in this example from the Wallet Repository. The result from the SQL query automatically creates a Wallet object so that it can be worked with in the code instantaneously without any sort of modifications.

4.5 Repository Pattern

Throughout the entire project, many common queries for reading and writing to the database will be used repeatedly. To prevent the need for rewriting code over and over, it can instead be abstracted into an interface so that any class that needs to use this operation can instead implement the interface through a process known as dependency injection . The repository pattern allows you to encapsulate the logic required for queries to the database. The result is a highly modular and flexible architecture where changes made to a repository will be reflected everywhere that the repository is used. [5] The interface IPaymentReposirory can be seen like so:

IPaymentRepository.cs

```
public interface IPaymentRepository
    {
      void CreatePayment(string stripeSession, Payment payment,
int orderid, decimal total);
    }
```

The interface for this repository defines one abstract method, CreatePayment, which takes in the parameters: string stripeSession, Payment payment, int orderid, decimal total. This behaviour is then implemented in PaymentRepository as can be seen below:

PaymentRepository.cs

```
public class PaymentRepository : IPaymentRepository
    {
       private readonly MyDatabaseContext _myDatabaseContext;

       public PaymentRepository(MyDatabaseContext
myDatabaseContext)
       {
            _myDatabaseContext = myDatabaseContext;
       }

       public void CreatePayment(string stripeSession, Payment
payment, int orderid, decimal total)
       {
            payment.OrderId = orderid;
            payment.StripeSession = stripeSession;
            payment.Total = total;
            payment.PaymentDateTime = DateTime.Now;

            _myDatabaseContext.Payments.Add(payment);
            _myDatabaseContext.SaveChanges();
       }
    }
```

The implementation of every repository must use dependency injection to create a reusable instance of the MyDatabaseContext class to communicate with the database. This is a standard that is required for Entity Framework Core to be usable in classes. The CreatePayment method is implemented here, and the necessary logic for creating a payment record in the database can be seen. The changes are saved by calling the _myDatabaseContext.Payments.Add(payment) function, with _myDatabaseContext.SaveChanges() being used to save the changes.

In the startup class, the newly created repository pattern has to be added to the service collection so that it can be used throughout the project. This can be seen in the ConfigureServices method in the Startup class like so:

Startup.cs

```
public void ConfigureServices(IServiceCollection services)
     {
          services.AddScoped<IPaymentRepository,
PaymentRepository>();
}
```

It is added as a scoped service so that instances of it last between HTTP requests, which is always going to be required for this project.

To instantiate a repository in a controller, dependency injection must be used to create a reusable implementation of the interface for the repository. Then any method defined in the interface can be used throughout that entire class, allowing for high reusability of data logic. This can be seen in the PaymentController like so:

PaymentController.cs

```
public class PaymentController : Controller
    {
      private readonly IPaymentRepository _paymentRepository;
           public PaymentControllerIPaymentRepository
paymentRepository)
      {
           _paymentRepository = paymentRepository;
      }
     public IActionResult Success()
      {
          //Code shortened for simplicity
          _paymentRepository.CreatePayment(sessionId, payment,
orderId, total);
     }
    }
```

4.6 ASP.NET Core Identity API

As mentioned at the beginning of this document, ASP.NET Core Identity API provides the tools and UI necessary to implement authentication and authorization into any project. This is built into visual studio and can be scaffolded into any ASP.NET Core project. Scaffolding will create a migration to add a number of tables related to Identity. The ones used in this project are AspNetUsers, AspNetRoles, and AspNetUserRoles. The migrations used for this can be seen below:

AddingIdentity.cs

```
protected override void Up(MigrationBuilder migrationBuilder)
      {
            migrationBuilder.CreateTable(
                  name: "AspNetRoles",
                  columns: table => new
                  {
                  Id = table.Column<string>(nullable: false),
                  Name = table.Column<string>(maxLength: 256,
nullable: true),
                  NormalizedName = table.Column<string>(maxLength:
256, nullable: true),
                  ConcurrencyStamp = table.Column<string>(nullable:
true)
                  },
                  constraints: table =>
                  {
                  table.PrimaryKey("PK_AspNetRoles", x => x.Id);
                  });

            migrationBuilder.CreateTable(
                  name: "AspNetUsers",
                  columns: table => new
                  {
                  Id = table.Column<string>(nullable: false),
                  UserName = table.Column<string>(maxLength: 256,
nullable: true),
                  NormalizedUserName =
table.Column<string>(maxLength: 256, nullable: true),
                  Email = table.Column<string>(maxLength: 256,
nullable: true),
                  NormalizedEmail = table.Column<string>(maxLength:
256, nullable: true),
                  EmailConfirmed = table.Column<bool>(nullable:
false),
                  PasswordHash = table.Column<string>(nullable:
true),
                  SecurityStamp = table.Column<string>(nullable:
true),
                  ConcurrencyStamp = table.Column<string>(nullable:
true),
```

```
                PhoneNumber = table.Column<string>(nullable:
true),
                PhoneNumberConfirmed =
table.Column<bool>(nullable: false),
                TwoFactorEnabled = table.Column<bool>(nullable:
false),
                LockoutEnd =
table.Column<DateTimeOffset>(nullable: true),
                LockoutEnabled = table.Column<bool>(nullable:
false),
                AccessFailedCount = table.Column<int>(nullable:
false)
                },
                constraints: table =>
                {
                table.PrimaryKey("PK_AspNetUsers", x => x.Id);
                });

 migrationBuilder.CreateTable(
                name: "AspNetUserRoles",
                columns: table => new
                {
                UserId = table.Column<string>(nullable: false),
                RoleId = table.Column<string>(nullable: false)
                },
                constraints: table =>
                {
                table.PrimaryKey("PK_AspNetUserRoles", x => new {
x.UserId, x.RoleId });
                table.ForeignKey(
                      name:
"FK_AspNetUserRoles_AspNetRoles_RoleId",
                      column: x => x.RoleId,
                      principalTable: "AspNetRoles",
                      principalColumn: "Id",
                      onDelete: ReferentialAction.Cascade);
                table.ForeignKey(
                      name:
"FK_AspNetUserRoles_AspNetUsers_UserId",
                      column: x => x.UserId,
                      principalTable: "AspNetUsers",
```

```
                    principalColumn: "Id",
                    onDelete: ReferentialAction.Cascade);
            });
}
```

To implement authorization throughout the project, it's very simple and requires just one line of code. This can be seen in the OrderController like so:

OrderController.cs

```
[Authorize]
    public class OrderController : Controller
    {
      //Code shortened for simplicity
    }
```

This will require that the entire controller can only be used by an authenticated user, but it can be applied to just specific methods too if required. For admin users, they are added to the "Admin" role. You can then specify if the user needs to be authenticated with the role of "Admin". This can be seen in the AdminController like so:

AdminController.cs

```
 [Authorize(Roles = "Admin")]
     public class AdminController : Controller
     {
     //Code shortened for simplicity
     }
```

4.7 ASP.NET Core MVC

ASP.NET Core MVC is the entire foundation of this project. It is an extension of the ASP.NET Core web development framework that utilizes the MVC design pattern. It follows the separation of concern design principle, which suggests that code should be separated based on the type of work it is doing. In this case, code is separated into models, views or controllers [6].

The core web app is built from the ground up using this framework and the separation of concern design principle. This can be seen from the directory structure of the project:

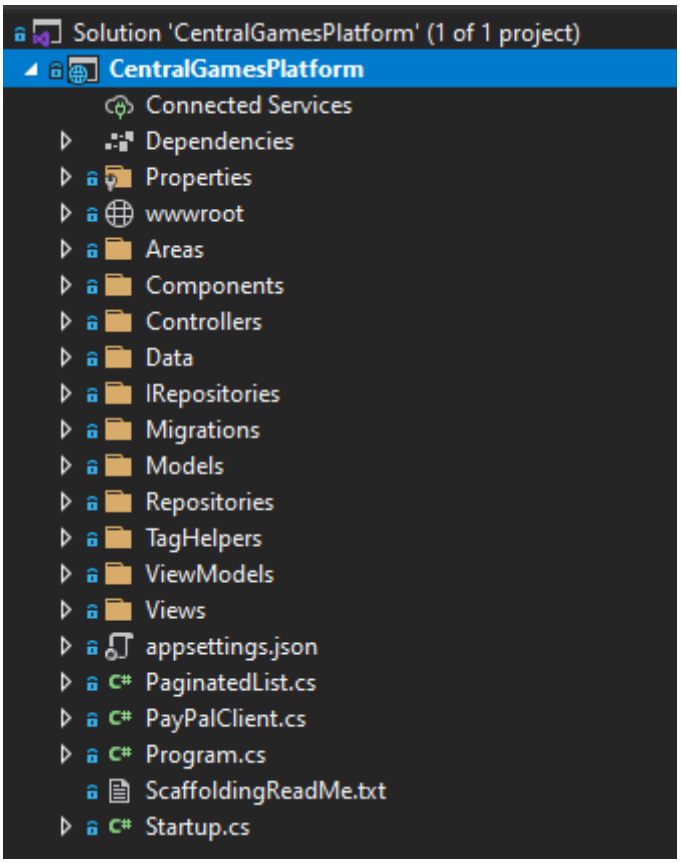

Figure 32 - Directory structure of project

On top of utilizing the MVC design pattern, ASP.NET Core MVC also provides a large amount of operations and functionality that is useful for any web development project. It allows you to use routing for url mapping, model binding to convert HTTP requests into C# objects, model validation to validate user input, and dependency injection to greatly reduce the amount of coupling in the project among many other useful functionalities.

5. Source Code

5.1 Controllers

AdminController.cs

```
using CentralGamesPlatform.IRepositories;
using CentralGamesPlatform.Models;
using CentralGamesPlatform.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Controllers
{
      [Authorize(Roles = "Admin")]
      public class AdminController : Controller
      {
      private readonly IGameRepository _gameRepository;
      private readonly IVerificationRepository
_verificationRepository;
      private readonly IDownloadRepository _downloadRepository;
      private readonly RoleManager<IdentityRole> _roleManager;
      private readonly MyDatabaseContext _myDatabaseContext;
      public AdminController(RoleManager<IdentityRole> roleManager,
IGameRepository gameRepository,
                             MyDatabaseContext myDatabaseContext,
IVerificationRepository verificationRepository,
                             IDownloadRepository downloadRepository)
      {
            _roleManager = roleManager;
            _gameRepository = gameRepository;
            _verificationRepository = verificationRepository;
            _downloadRepository = downloadRepository;
            _myDatabaseContext = myDatabaseContext;
      }
      public IActionResult Index()
```

```
      {
            return View();
      }

      //Manage Games
      public async Task <IActionResult> ViewGames(string sortOrder,
string currentFilter, string searchString, int? pageNumber)
      {
            ViewData["CurrentSort"] = sortOrder;
            ViewData["NameSortParam"] =
String.IsNullOrEmpty(sortOrder) ? "name_desc" : "";
            ViewData["CategoryIdSortParam"] = sortOrder ==
"CategoryId" ? "categoryid_desc" : "CategoryId";
            if(searchString != null)
            {
                  pageNumber = 1;
            }
            else
            {
                  searchString = currentFilter;
            }

            ViewData["CurrentFilter"] = searchString;
            var games = from g in _myDatabaseContext.Games select
g;
            if (!String.IsNullOrEmpty(searchString))
            {
                  games = games.Where(g =>
g.Name.Contains(searchString));
            }
            switch (sortOrder)
            {
                  case "name_desc":
                  games = games.OrderByDescending(g => g.Name);
                  break;
                  case "CategoryId":
                  games = games.OrderBy(g => g.CategoryId);
                  break;
                  case "categoryid_desc":
                  games = games.OrderByDescending(g =>
g.CategoryId);
```

```
                break;
                default:
                games = games.OrderBy(g => g.Name);
                break;
          }
          int pageSize = 10;
          return View(await
PaginatedList<Game>.CreateAsync(games.AsNoTracking(), pageNumber
?? 1, pageSize));
     }
     public IActionResult Details(int? gameId)
     {
          if (gameId == null)
          {
                Response.StatusCode = 404;
                return RedirectToAction("HandleError", "Error",
new { code = 404 });
          }
          var game = _gameRepository.GetGameById((int)gameId);
          if (game == null)
          {
                Response.StatusCode = 404;
                return RedirectToAction("HandleError", "Error",
new { code = 404 });
          }
          return View(game);
     }
     public IActionResult Create()
     {
          return View();
     }

     [HttpPost]
     [ValidateAntiForgeryToken]
     public IActionResult Create(Game game)
     {
          try
          {
                if (ModelState.IsValid)
                {
                _myDatabaseContext.Add(game);
```

```
                _myDatabaseContext.SaveChanges();
                return RedirectToAction("ViewGames");
                }
            }
            catch (DbUpdateException)
            {
                ModelState.AddModelError("", "Unable to save
changes to the database");
            }
            return View(game);
        }
        public IActionResult Edit(int? gameId)
        {
            if (gameId == null || gameId == -1)
            {
                Response.StatusCode = 404;
                return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            var game = _gameRepository.GetGameById((int)gameId);
            if (game == null)
            {
                Response.StatusCode = 404;
                return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            return View(game);
        }

        [HttpPost, ActionName("Edit")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> EditPost(int? gameId)
        {
            if (gameId == null)
            {
                Response.StatusCode = 404;
                return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            var gameToUpdate = await
_myDatabaseContext.Games.FirstOrDefaultAsync(g => g.GameId ==
```

```
gameId);
            if (await TryUpdateModelAsync<Game>(
                  gameToUpdate,
                  "",
                  g => g.Name, g => g.Description, g => g.Price, g
=> g.ImageUrl, g => g.ImageThumbnailUrl, g => g.IsOnSale, g =>
g.CategoryId))
            {
                  try
                  {
                  await _myDatabaseContext.SaveChangesAsync();
                  return RedirectToAction("ViewGames");
                  }
                  catch (DbUpdateException)
                  {
                  ModelState.AddModelError("", "Unable to save
changes to the database");
                  }
            }
            return View(gameToUpdate);
      }

      public async Task<IActionResult>
ViewVerificationRequests(string sortOrder, string currentFilter,
string searchString, int? pageNumber)
      {
            ViewData["CurrentSort"] = sortOrder;
            ViewData["StatusSortParam"] =
String.IsNullOrEmpty(sortOrder) ? "status_desc" : "";
            ViewData["DateOfRequestSortParam"] = sortOrder ==
"DateOfRequest" ? "dateofrequest_desc" : "DateOfRequest";
            if (searchString != null)
            {
                  pageNumber = 1;
            }
            else
            {
                  searchString = currentFilter;
            }

            ViewData["CurrentFilter"] = searchString;
```

```
            var verifications = from v in
_myDatabaseContext.Verifications select v;
            if (!String.IsNullOrEmpty(searchString))
            {
                  verifications = verifications.Where(v =>
v.UserId.Contains(searchString));
            }
            switch (sortOrder)
            {
                  case "status_desc":
                  verifications = verifications.OrderByDescending(v
=> v.Status);
                  break;
                  case "DateOfRequest":
                  verifications = verifications.OrderBy(v =>
v.DateOfRequest);
                  break;
                  case "dateofrequest_desc":
                  verifications = verifications.OrderByDescending(v
=> v.DateOfRequest);
                  break;
                  default:
                  verifications = verifications.OrderBy(v =>
v.DateOfRequest);
                  break;
            }
            int pageSize = 10;
            return View(await
PaginatedList<Verification>.CreateAsync(verifications.AsNoTracking
(), pageNumber ?? 1, pageSize));
      }
      public IActionResult UpdateVerificationRequest(int?
verificationId)
      {
            if (verificationId == null)
            {
                  Response.StatusCode = 404;
                  return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            var verificationRequest =
```

```
_verificationRepository.RetrieveVerificationById((int)verification
Id);
            if (verificationRequest == null)
            {
                  Response.StatusCode = 404;
                  return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            using (MemoryStream memoryStream = new
MemoryStream(verificationRequest.Content))
            {
                  ViewData["Image"] =
Convert.ToBase64String(verificationRequest.Content);
            }

            return View(verificationRequest);
      }


      [HttpPost, ActionName("UpdateVerificationRequest")]
      [ValidateAntiForgeryToken]
      public async Task<IActionResult>
UpdateVerificationRequestPost(int? verificationId)
      {
            if (verificationId == null)
            {
                  Response.StatusCode = 404;
                  return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            var verificationRequestToUpdate = await
_myDatabaseContext.Verifications.FirstOrDefaultAsync(g =>
g.VerificationId == verificationId);
            if (await TryUpdateModelAsync<Verification>(
                  verificationRequestToUpdate,
                  "",
                  v => v.Status))
            {
                  try
                  {
                  await _myDatabaseContext.SaveChangesAsync();
```

```
                return
RedirectToAction("ViewVerificationRequests");
                }
                catch (DbUpdateException)
                {
                ModelState.AddModelError("", "Unable to save
changes to the database");
                }
           }
           return View(verificationRequestToUpdate);
     }

     public IActionResult DownloadDetails(int? gameId)
     {
           if (gameId == null)
           {
                Response.StatusCode = 404;
                return RedirectToAction("HandleError", "Error",
new { code = 404 });
           }
           var game = _gameRepository.GetGameById((int)gameId);
           if (game == null)
           {
                Response.StatusCode = 404;
                return RedirectToAction("HandleError", "Error",
new { code = 404 });
           }
           string fileName =
_downloadRepository.GetDownloadFileName((int)gameId);
           if(fileName == null)
           {
                return RedirectToAction("DownloadCreate", new {
gameId = (int)gameId });
           }
           double fileVersion =
_downloadRepository.GetDownloadVersion((int)gameId);
           DateTime lastUpdated =
_downloadRepository.GetDownloadLastUpdated((int)gameId);
           var downloadViewModel = new AdminDownloadViewModel
           {
                FileName = fileName,
```

```
                FileVersion = fileVersion,
                LastUpdated = lastUpdated,
                GameId = (int)gameId
            };
            return View(downloadViewModel);
      }
      public IActionResult DownloadUpdate(int? gameId)
      {
            if (gameId == null)
            {
                Response.StatusCode = 404;
                return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            var download =
_downloadRepository.RetrieveDownloadByGameId((int)gameId);
            if (download == null)
            {
                Response.StatusCode = 404;
                return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            return View(download);
      }

      [HttpPost, ActionName("DownloadUpdate")]
      [ValidateAntiForgeryToken]
      public async Task<IActionResult> DownloadUpdatePost(Download
download, int? downloadId)
      {
            if (downloadId == null)
            {
                Response.StatusCode = 404;
                return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }

            var downloadToUpdate = await
_myDatabaseContext.Downloads.FirstOrDefaultAsync(d => d.DownloadId
== downloadId);
            downloadToUpdate.LastUpdated = DateTime.Now;
```

```
            downloadToUpdate.FileName = download.FileName;
            downloadToUpdate.FileFormat = download.FileFormat;
            downloadToUpdate.VersionNumber =
download.VersionNumber;
            if (download.fileObject.file.Length > 0)
            {
                  using (var memoryStream = new MemoryStream())
                  {
                  download.fileObject.file.CopyTo(memoryStream);
                  downloadToUpdate.Content = memoryStream.ToArray();
                  }
            }
            else
            {
                  Response.StatusCode = 404;
                  return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            try
            {
                  await _myDatabaseContext.SaveChangesAsync();
                  return RedirectToAction("DownloadDetails", new {
gameId = downloadToUpdate.GameId });
            }
            catch (DbUpdateException)
            {
                  ModelState.AddModelError("", "Unable to save
changes to the database");
            }

            return View(downloadToUpdate);
      }

      public IActionResult DownloadCreate()
      {
            return View();
      }

      [HttpPost]
      [ValidateAntiForgeryToken]
      public IActionResult DownloadCreate(Download download, int?
```

```
gameId)
      {
            try
            {
                  if (gameId != null)
                  {

                  var game =
_gameRepository.GetGameById((int)gameId);
                  if (game.CategoryId >= 4 && game.CategoryId <= 9)
                  {
                        var downloadNameExists =
_downloadRepository.GetDownloadFileName(download.GameId);
                        if (downloadNameExists == null)
                        {

                              if (download.fileObject.file.Length >
0)
                              {
                                    using (var memoryStream = new
MemoryStream())
                                    {

download.fileObject.file.CopyTo(memoryStream);
                                    download.Content =
memoryStream.ToArray();

_downloadRepository.CreateDownload(download);
                                    return
RedirectToAction("DownloadDetails", new { gameId = gameId });
                                    }
                              }
                        }
                  }
                  }
            }
            catch (DbUpdateException)
            {
                  ModelState.AddModelError("", "Unable to save
changes to the database");
            }
```

```
            return View(download);
        }

        }
}
```

ContactController.cs

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

namespace CentralGamesPlatform.Controllers
{
    public class ContactController : Controller
    {
       public IActionResult Index()
       {
            return View();
       }
    }
}
```

DownloadController.cs

```
using CentralGamesPlatform.IRepositories;
using CentralGamesPlatform.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Controllers
```

```
{
      [Authorize]
      public class DownloadController : Controller
      {
      private readonly IDownloadRepository _downloadRepository;
      private readonly MyDatabaseContext _myDatabaseContext;
      public DownloadController(IDownloadRepository
downloadRepository, MyDatabaseContext myDatabaseContext)
      {
            _downloadRepository = downloadRepository;
            _myDatabaseContext = myDatabaseContext;
      }
      public IActionResult DownloadGame(int? gameId)
      {
            if (gameId == null)
            {
                  Response.StatusCode = 404;
                  return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
            var query = (from l in _myDatabaseContext.Licences
                        where l.UserId == userId
                        join od in _myDatabaseContext.OrderDetails
on l.OrderDetailId equals od.OrderDetailId
                        join g in _myDatabaseContext.Games on
od.GameId equals g.GameId
                        where g.GameId == gameId
                        select g).SingleOrDefault();
            if (query == null)
            {
                  Response.StatusCode = 404;
                  return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            var download =
_downloadRepository.RetrieveDownloadByGameId((int)gameId);
            if (download == null)
            {
                  Response.StatusCode = 404;
```

```
                return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            return File(download.Content,
"application/force-download", download.FileName +
download.FileFormat);
      }
      public IActionResult PlayGame(int? gameId)
      {
            if (gameId == null)
            {
                  Response.StatusCode = 404;
                  return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
            var query = (from l in _myDatabaseContext.Licences
                        where l.UserId == userId
                        join od in _myDatabaseContext.OrderDetails
on l.OrderDetailId equals od.OrderDetailId
                        join g in _myDatabaseContext.Games on
od.GameId equals g.GameId
                        where g.GameId == gameId
                        select g).SingleOrDefault();
            if (query == null)
            {
                  Response.StatusCode = 404;
                  return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            var download =
_downloadRepository.RetrieveDownloadByGameId((int)gameId);
            if(download == null)
            {
                  Response.StatusCode = 404;
                  return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            using (MemoryStream memoryStream = new
MemoryStream(download.Content))
```

```
            {
                  ViewData["game"] =
Convert.ToBase64String(download.Content);
            }
            return View();
      }
      }
}
```

ErrorController.cs

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

namespace CentralGamesPlatform.Controllers
{
      public class ErrorController : Controller
      {

      public IActionResult HandleError(int? code)
      {
            if (code == null)
            {
                  return RedirectToAction("Index", "Home");
            }
            ViewData["ErrorMessage"] = $"Error code: {code}";
            return View();
      }
      }
}
```

Game Controller.cs

```
using CentralGamesPlatform.IRepositories;
using CentralGamesPlatform.Models;
using CentralGamesPlatform.ViewModels;
using Microsoft.AspNetCore.Mvc;
```

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

namespace CentralGamesPlatform.Controllers
{
    public class GameController : Controller
    {
       private readonly IGameRepository _gameRepository;
       private readonly ICategoryRepository _categoryRepository;
       private readonly MyDatabaseContext _myDatabaseContext;

       public GameController(IGameRepository gameRepository,
ICategoryRepository categoryRepository,
                                    MyDatabaseContext
myDatabaseContext)
       {
            _gameRepository = gameRepository;
            _categoryRepository = categoryRepository;
            _myDatabaseContext = myDatabaseContext;
       }

       public ViewResult List(string category)
       {
            List<int> query;
            if (User.Identity.IsAuthenticated)
            {
                 string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
                 query = (from l in _myDatabaseContext.Licences
                             where l.UserId == userId
                             join od in
_myDatabaseContext.OrderDetails on l.OrderDetailId equals
od.OrderDetailId
                             join g in _myDatabaseContext.Games on
od.GameId equals g.GameId
                             select g.GameId).ToList();
                 IEnumerable<Game> games;
```

```
                string currentCategory;
                //if category is null then current category is
all games
                if (string.IsNullOrEmpty(category))
                {
                      games =
_gameRepository.GetAllGames.OrderBy(g => g.GameId);
                      currentCategory = "All Games";
                }
                else
                {
                      //find games that matches category name
argument
                      games = _gameRepository.GetAllGames.Where(c
=> c.Category.CategoryName == category);
                      currentCategory =
_categoryRepository.GetAllCategories.FirstOrDefault(currentCategor
y => currentCategory.CategoryName == category)?.CategoryName;
                }

                return View(new GameListViewModel
                {
                      Games = games,
                      OwnedGameIds = query,
                      CurrentCategory = currentCategory
                });
          }
          else
          {
                IEnumerable<Game> games;
                string currentCategory;
                //if category is null then current category is
all games
                if (string.IsNullOrEmpty(category))
                {
                      games =
_gameRepository.GetAllGames.OrderBy(g => g.GameId);
                      currentCategory = "All Games";
                }
                else
                {
```

```
                        //find games that matches category name
argument
                        games = _gameRepository.GetAllGames.Where(c
=> c.Category.CategoryName == category);
                        currentCategory =
_categoryRepository.GetAllCategories.FirstOrDefault(currentCategor
y => currentCategory.CategoryName == category)?.CategoryName;
                  }

                  return View(new GameListViewModel
                  {
                        Games = games,
                        OwnedGameIds = null,
                        CurrentCategory = currentCategory
                  });
            }
      }

      public IActionResult Details(int id)
      {
            var game = _gameRepository.GetGameById(id);
            if (game == null)
                  return NotFound();

            return View(game);

      }
    }
}
```

HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using CentralGamesPlatform.Models;
using CentralGamesPlatform.ViewModels;
```

```
using Microsoft.EntityFrameworkCore;
using System.Security.Claims;
using CentralGamesPlatform.IRepositories;

namespace CentralGamesPlatform.Controllers
{
      public class HomeController : Controller
      {
      private readonly IGameRepository _gameRepository;
      private readonly MyDatabaseContext _myDatabaseContext;

       public HomeController(IGameRepository gameRepository,
MyDatabaseContext myDatabaseContext)
       {
            _gameRepository = gameRepository;
            _myDatabaseContext = myDatabaseContext;
       }
      public IActionResult Index()
      {
            if (User.Identity.IsAuthenticated)
            {
                  string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
                  var query = (from l in _myDatabaseContext.Licences
                               where l.UserId == userId
                               join od in
_myDatabaseContext.OrderDetails on l.OrderDetailId equals
od.OrderDetailId
                               join g in _myDatabaseContext.Games on
od.GameId equals g.GameId
                               select g.GameId).ToList();


                  var homeViewModel = new HomeViewModel
                  {
                  GamesOnSale = _gameRepository.GetGamesOnSale,
                  OwnedGameIds = query
                  };
                  return View(homeViewModel);
            }
            else
```

```
            {
                  var homeViewModel = new HomeViewModel
                  {
                  GamesOnSale = _gameRepository.GetGamesOnSale,
                  OwnedGameIds = null
                  };
                  return View(homeViewModel);
            }

      }
      public async Task <IActionResult> Search(string
currentFilter, string searchString, int? pageNumber)
      {
            if (searchString != null)
            {
                  pageNumber = 1;
            }
            else
            {
                  searchString = currentFilter;
            }
            ViewData["CurrentFilter"] = searchString;
            var games = from g in _myDatabaseContext.Games select
g;
            if (!String.IsNullOrEmpty(searchString))
            {
                  games = games.Where(g =>
g.Name.Contains(searchString));
            }
            else
            {
                  games = games.OrderBy(g => g.Name);
            }
            int pageSize = 10;
            return View(await
PaginatedList<Game>.CreateAsync(games.AsNoTracking(), pageNumber
?? 1, pageSize));
      }

      }
}
```

LibraryController.cs

```
using CentralGamesPlatform.Models;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Security.Claims;
using CentralGamesPlatform.ViewModels;
using Microsoft.AspNetCore.Authorization;
using CentralGamesPlatform.IRepositories;

namespace CentralGamesPlatform.Controllers
{
      [Authorize]
    public class LibraryController : Controller
    {
      private readonly IGameRepository _gameRepository;
      private readonly ILicenceRepository _licenceRepository;
      private readonly IOrderDetailRepository
_orderDetailRepository;
      private readonly ICasinoPassRepository _casinoPassRepository;
      private readonly MyDatabaseContext _myDatabaseContext;
      public LibraryController(IGameRepository gameRepository,
ILicenceRepository licenceRepository,
                               IOrderDetailRepository
orderDetailRepository, ICasinoPassRepository casinoPassRepository,
                               MyDatabaseContext
myDatabaseContext)
      {
            _gameRepository = gameRepository;
            _licenceRepository = licenceRepository;
            _orderDetailRepository = orderDetailRepository;
            _casinoPassRepository = casinoPassRepository;
            _myDatabaseContext = myDatabaseContext;
      }

      public IActionResult Index()
      {
            string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
```

```
            var licences = _licenceRepository.GetLicences(userId);
            var query = (from l in _myDatabaseContext.Licences
                         where l.UserId == userId
                         join od in _myDatabaseContext.OrderDetails
on l.OrderDetailId equals od.OrderDetailId
                         join g in _myDatabaseContext.Games on
od.GameId equals g.GameId
                         select g).ToList();
            query.RemoveAll(og => og.GameId == -1);
            var usersPasses =
_casinoPassRepository.GetCasinoPassesByUserId(userId);
            List<CasinoPass> ownedPasses = new List<CasinoPass>();
            List<Game> activeCasinoGames = new List<Game>();
            foreach(var pass in usersPasses)
            {
                  if(pass.Active == false && pass.Expired == false)
                  {
                  ownedPasses.Add(pass);
                  }
                  else if(pass.Active == true && pass.Expired ==
false)
                  {

activeCasinoGames.Add(_gameRepository.GetGameById(pass.GameId));
                  }
            }
            var libraryViewModel = new LibraryViewModel
            {
                  OwnedGames = query,
                  OwnedPasses = ownedPasses,
                  ActiveCasinoGames = activeCasinoGames
            };
            return View(libraryViewModel);
      }
      }
}
```

OrderController.cs

```
using CentralGamesPlatform.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Identity;
using System.Security.Principal;
using System.Security.Claims;
using CentralGamesPlatform.IRepositories;

namespace CentralGamesPlatform.Controllers
{
    [Authorize]
    public class OrderController : Controller
    {
       private readonly IOrderRepository _orderRepository;
       private readonly ICasinoPassRepository
_casinoPassRepository;
       private readonly UserManager<ApplicationUser> _userManager;
       private readonly ShoppingCart _shoppingCart;

       public OrderController(IOrderRepository orderRepository,
ICasinoPassRepository casinoPassRepository,
                                       ShoppingCart shoppingCart,
UserManager<ApplicationUser> userManager )
       {
            _orderRepository = orderRepository;
            _casinoPassRepository = casinoPassRepository;
            _shoppingCart = shoppingCart;
            _userManager = userManager;
       }

       public IActionResult Checkout()
       {
            _shoppingCart.ShoppingCartItems =
_shoppingCart.GetShoppingCartItems();
           if (_shoppingCart.ShoppingCartItems.Count == 0)
           {
```

```
                Response.StatusCode = 400;
                return RedirectToAction("Index", "Home");
            }
            return View();
        }

        [HttpPost]
        public IActionResult Checkout(Order order)
        {
            decimal total;
            _shoppingCart.ShoppingCartItems =
_shoppingCart.GetShoppingCartItems();
            total = _shoppingCart.GetShoppingCartTotal();

            if(_shoppingCart.ShoppingCartItems.Count == 0)
            {
                Response.StatusCode = 400;
                return RedirectToAction("HandleError", "Error",
new { code = 400 });
            }

            if (ModelState.IsValid)
            {
                string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
                order.UserId = userId;
                _orderRepository.CreateOrder(order);
                TempData["orderId"] = order.OrderId;
                return RedirectToAction("Index", "Payment", new {
id = order.OrderId });
            }

            return View(order);
        }
    }
}
```

PaymentController.cs

```
using CentralGamesPlatform.Models;
using Microsoft.AspNetCore.Mvc;
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Stripe;
using System.IO;
using Microsoft.Extensions.Logging;
using Stripe.Checkout;
using Newtonsoft.Json;
using Microsoft.AspNetCore.Authorization;
using System.Security.Claims;
using CentralGamesPlatform.ViewModels;
using CentralGamesPlatform.IRepositories;

namespace CentralGamesPlatform.Controllers
{
    [Authorize]
    public class PaymentController : Controller
    {
       private readonly ShoppingCart _shoppingCart;
       private readonly IOrderRepository _orderRepository;
       private readonly IPaymentRepository _paymentRepository;
       private readonly IOrderDetailRepository
_orderDetailRepository;
       private readonly ILicenceRepository _licenseRepository;
       private readonly ICasinoPassRepository
_casinoPassRepository;
       public PaymentController(IOrderRepository orderRepository,
ShoppingCart shoppingCart, IPaymentRepository paymentRepository,
                                          IOrderDetailRepository
orderDetailRepository, ILicenceRepository licenseRepository,
                                          ICasinoPassRepository
casinoPassRepository)
       {
             _orderRepository = orderRepository;
             _shoppingCart = shoppingCart;
             _paymentRepository = paymentRepository;
             _orderDetailRepository = orderDetailRepository;
             _licenseRepository = licenseRepository;
             _casinoPassRepository = casinoPassRepository;
```

```
            StripeConfiguration.ApiKey =
"sk_test_51IB0Z4BTwx1LYfRRop1pYWwRVKBAs0K7KZBRbKTubudFUXJPN5BlooRa
hipg8qIkpIQ49d6c4YZE9ErcziO23QtR00rzwq6cbk";
      }
      public IActionResult Index(int id)
      {

            _shoppingCart.ShoppingCartItems =
_shoppingCart.GetShoppingCartItems();
            var paymentSummaryViewModel = new
PaymentSummaryViewModel
            {
                  ShoppingCart = _shoppingCart,
                  ShoppingCartTotal =
_shoppingCart.GetShoppingCartTotal(),
                  Order = _orderRepository.GetOrder(id)
            };
            return View(paymentSummaryViewModel);
      }

      [HttpPost("create-checkout-session")]
      public IActionResult CreateCheckoutSession()
      {
            decimal orderTotal;
            _shoppingCart.ShoppingCartItems =
_shoppingCart.GetShoppingCartItems();
            orderTotal = _shoppingCart.GetShoppingCartTotal() *
100;
            if(orderTotal == 0.00M)
           {
                  return RedirectToAction("Success");
           }
            var options = new SessionCreateOptions
            {
                  PaymentMethodTypes = new List<string>
                  {
                        "card"
                  },
                  LineItems = new List<SessionLineItemOptions>
                  {
```

```
                    new SessionLineItemOptions
                    {
                          PriceData = new
SessionLineItemPriceDataOptions
                          {
                                UnitAmount = (long?)orderTotal,
                                Currency = "eur",
                                ProductData = new
SessionLineItemPriceDataProductDataOptions
                                {
                                      Name = "Order Total"
                                },

                          },
                          Quantity = 1
                    },
                },
                Mode = "payment",
                //SuccessUrl =
"https://localhost:44394/Payment/Success?session_id={CHECKOUT_SESS
ION_ID}",
                //CancelUrl =
"https://localhost:44394/Payment/Failed",
                SuccessUrl =
"https://centralgamesplatform.azurewebsites.net/Payment/Success?se
ssion_id={CHECKOUT_SESSION_ID}",
                CancelUrl =
"https://centralgamesplatform.azurewebsites.net/Payment/Failed",
           };
            var service = new SessionService();
            Session session = service.Create(options);
            return Json(new { id = session.Id });
      }


      public IActionResult Success()
      {
            if(TempData["orderId"] == null)
            {
                  return RedirectToAction("Failed");
            }
```

```
            try
            {
                  int orderId = (int)TempData["orderId"];
                  string sessionId =
HttpContext.Request.Query["session_id"];
                  TempData.Clear();
                  string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
                  //Create payment object in database
                  Payment payment = new Payment();
                  Models.Order order =
_orderRepository.GetOrder(orderId);
                  decimal total = order.OrderTotal;
                  _paymentRepository.CreatePayment(sessionId,
payment, orderId, total);
                  //Update order to be sucessful
                  _orderRepository.SuccessfulOrder(orderId);
                  //Generate License keys for each game/pass
purchased
                  var orderDetails =
_orderDetailRepository.GetAllOrderDetails(orderId);
                  _licenseRepository.CreateLicense(orderDetails,
userId);
                  //If pass was purchased then create pass in
database
                  foreach (var orderDetail in orderDetails)
                  {
                        if (orderDetail.GameId == -1)
                 {
                              CasinoPass casinoPass = new CasinoPass
                              {
                                    UserId = userId,
                                    Active = false,
                                    Expired = false
                              };

_casinoPassRepository.CreateCasinoPass(casinoPass);
                        }
                  }
                  _shoppingCart.ClearCart();
                  return View();
```

```
            }

            catch(Exception ex)
            {
                  return View(ex.Message);
            }
      }

      public IActionResult Failed()
      {
            return View();
      }

    }
}
```

PayoutController.cs

```
using CentralGamesPlatform.Models;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Stripe;
using System.IO;
using Microsoft.Extensions.Logging;
using Stripe.Checkout;
using Newtonsoft.Json;
using Microsoft.AspNetCore.Authorization;
using System.Security.Claims;
using CentralGamesPlatform.ViewModels;
using CentralGamesPlatform.IRepositories;

namespace CentralGamesPlatform.Controllers
{
    [Authorize]
    public class PaymentController : Controller
    {
      private readonly ShoppingCart _shoppingCart;
      private readonly IOrderRepository _orderRepository;
```

```
      private readonly IPaymentRepository _paymentRepository;
      private readonly IOrderDetailRepository
_orderDetailRepository;
      private readonly ILicenceRepository _licenseRepository;
      private readonly ICasinoPassRepository
_casinoPassRepository;
      public PaymentController(IOrderRepository orderRepository,
ShoppingCart shoppingCart, IPaymentRepository paymentRepository,
                                          IOrderDetailRepository
orderDetailRepository, ILicenceRepository licenseRepository,
                                          ICasinoPassRepository
casinoPassRepository)
      {
            _orderRepository = orderRepository;
            _shoppingCart = shoppingCart;
            _paymentRepository = paymentRepository;
            _orderDetailRepository = orderDetailRepository;
            _licenseRepository = licenseRepository;
            _casinoPassRepository = casinoPassRepository;


            StripeConfiguration.ApiKey =
"sk_test_51IB0Z4BTwx1LYfRRop1pYWwRVKBAs0K7KZBRbKTubudFUXJPN5BlooRa
hipg8qIkpIQ49d6c4YZE9ErcziO23QtR00rzwq6cbk";
      }
      public IActionResult Index(int id)
      {

            _shoppingCart.ShoppingCartItems =
_shoppingCart.GetShoppingCartItems();
            var paymentSummaryViewModel = new
PaymentSummaryViewModel
            {
                  ShoppingCart = _shoppingCart,
                  ShoppingCartTotal =
_shoppingCart.GetShoppingCartTotal(),
                  Order = _orderRepository.GetOrder(id)
            };
            return View(paymentSummaryViewModel);
      }
```

```
      [HttpPost("create-checkout-session")]
      public IActionResult CreateCheckoutSession()
      {
            decimal orderTotal;
            _shoppingCart.ShoppingCartItems =
_shoppingCart.GetShoppingCartItems();
            orderTotal = _shoppingCart.GetShoppingCartTotal() *
100;
            if(orderTotal == 0.00M)
           {
                  return RedirectToAction("Success");
           }
            var options = new SessionCreateOptions
            {
                  PaymentMethodTypes = new List<string>
                  {
                        "card"
                  },
                  LineItems = new List<SessionLineItemOptions>
                  {
                        new SessionLineItemOptions
                        {
                              PriceData = new
SessionLineItemPriceDataOptions
                              {
                                    UnitAmount = (long?)orderTotal,
                                    Currency = "eur",
                                    ProductData = new
SessionLineItemPriceDataProductDataOptions
                                    {
                                          Name = "Order Total"
                                    },

                              },
                              Quantity = 1
                        },
                  },
                  Mode = "payment",
                 //SuccessUrl =
"https://localhost:44394/Payment/Success?session_id={CHECKOUT_SESS
ION_ID}",
```

```
                //CancelUrl =
"https://localhost:44394/Payment/Failed",
                SuccessUrl =
"https://centralgamesplatform.azurewebsites.net/Payment/Success?se
ssion_id={CHECKOUT_SESSION_ID}",
                CancelUrl =
"https://centralgamesplatform.azurewebsites.net/Payment/Failed",
            };
            var service = new SessionService();
            Session session = service.Create(options);
            return Json(new { id = session.Id });
      }


      public IActionResult Success()
      {
            if(TempData["orderId"] == null)
            {
                  return RedirectToAction("Failed");
            }
            try
            {
                  int orderId = (int)TempData["orderId"];
                  string sessionId =
HttpContext.Request.Query["session_id"];
                  TempData.Clear();
                  string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
                  //Create payment object in database
                  Payment payment = new Payment();
                  Models.Order order =
_orderRepository.GetOrder(orderId);
                  decimal total = order.OrderTotal;
                  _paymentRepository.CreatePayment(sessionId,
payment, orderId, total);
                  //Update order to be sucessful
                  _orderRepository.SuccessfulOrder(orderId);
                  //Generate License keys for each game/pass
purchased
                  var orderDetails =
_orderDetailRepository.GetAllOrderDetails(orderId);
```

```
                _licenseRepository.CreateLicense(orderDetails,
userId);
                //If pass was purchased then create pass in
database
                foreach (var orderDetail in orderDetails)
                {
                      if (orderDetail.GameId == -1)
                {
                            CasinoPass casinoPass = new CasinoPass
                            {
                                  UserId = userId,
                                  Active = false,
                                  Expired = false
                            };

_casinoPassRepository.CreateCasinoPass(casinoPass);
                      }
                }
                _shoppingCart.ClearCart();
                return View();
            }

            catch(Exception ex)
            {
                  return View(ex.Message);
            }
      }

      public IActionResult Failed()
      {
            return View();
      }

    }
}
```

ShoppingCartController.cs

```
using CentralGamesPlatform.IRepositories;
using CentralGamesPlatform.Models;
using CentralGamesPlatform.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Controllers
{
    [Authorize]
    public class ShoppingCartController : Controller
     {
      private readonly IGameRepository _gameRepository;
      private readonly ShoppingCart _shoppingCart;
      private readonly MyDatabaseContext _myDatabaseContext;
      private readonly ICasinoPassRepository
_casinoPassRepository;

      public ShoppingCartController(IGameRepository
gameRepository, ShoppingCart shoppingCart,
                                                MyDatabaseContext
myDatabaseContext, ICasinoPassRepository casinoPassRepository)
      {
            _gameRepository = gameRepository;
            _shoppingCart = shoppingCart;
            _myDatabaseContext = myDatabaseContext;
            _casinoPassRepository = casinoPassRepository;
      }

      public IActionResult Index()
      {
            _shoppingCart.ShoppingCartItems =
_shoppingCart.GetShoppingCartItems();
           if(_shoppingCart.ShoppingCartItems.Count == 0)
          {
                return RedirectToAction("Index", "Home");
```

```
            }

            var shoppingCartViewModel = new ShoppingCartViewModel
            {
                  ShoppingCart = _shoppingCart,
                  ShoppingCartTotal =
_shoppingCart.GetShoppingCartTotal()
            };

            return View(shoppingCartViewModel);
      }

      public RedirectToActionResult AddToShoppingCart(int gameId)
      {
            string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
            var shoppingCartItems =
_shoppingCart.GetShoppingCartItems();
            if(shoppingCartItems == null)
            {
                  Response.StatusCode = 404;
                  return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            if(gameId == -1)
            {
                  int amountPurchasedToday =
_casinoPassRepository.AmountPurchasedToday(userId);
                  //var count = shoppingCartItems.Where(item =>
item.Game.GameId == -1).Count();
                  var count = shoppingCartItems.Where(g =>
g.Game.GameId == -1).Select(c => c.Amount).FirstOrDefault();
                  if (amountPurchasedToday >= 10 || (count >= 10 ||
count >= (10 - amountPurchasedToday)))
                  {
                        TempData["ErrorMessage"] = "You have
purchased too many casino passes today. This platform does not
enable or encourage impulsive gambling in any way. Consider
looking at some video games we have on offer for a safe and fun
outlet!";
                        _shoppingCart.ClearCart();
```

```
                        return RedirectToAction("Index",
"Library");
                }
            }
             var query = (from l in _myDatabaseContext.Licences
                                where l.UserId == userId
                                join od in
_myDatabaseContext.OrderDetails on l.OrderDetailId equals
od.OrderDetailId
                                join g in _myDatabaseContext.Games on
od.GameId equals g.GameId
                                select g.GameId).ToList();
             foreach(var item in shoppingCartItems)
            {
                  if (item.Game.GameId == gameId && gameId != -1)
                 {
                        return RedirectToAction("Index");
                  }
            }
             if(gameId == -1 || !query.Contains(gameId))
            {
                  var selectedGame =
_gameRepository.GetAllGames.FirstOrDefault(g => g.GameId ==
gameId);
                  if (selectedGame != null)
                  {
                        _shoppingCart.AddToCart(selectedGame, 1);
                  }

                  return RedirectToAction("Index");
             }
            else
            {
                  ViewBag.ErrorMessage = "You already own this
game";
                  return RedirectToAction("Index", "Library");
             }

      }

      public RedirectToActionResult RemoveFromShoppingCart(int
```

```
gameId)
      {
            var selectedGame =
_gameRepository.GetAllGames.FirstOrDefault(g => g.GameId ==
gameId);

            if (selectedGame != null)
            {
                  _shoppingCart.RemoveFromCart(selectedGame);
            }

            return RedirectToAction("Index");
      }

      public RedirectToActionResult ClearCart()
     {
            _shoppingCart.ClearCart();
            return RedirectToAction("Index");
     }
    }
}
```

StartController.cs

```
using CentralGamesPlatform.IRepositories;
using CentralGamesPlatform.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Claims;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Controllers
{
     [Authorize]
     public class StartController : Controller
     {
     private readonly ICasinoPassRepository _casinoPassRepository;
```

```
      private readonly IVerificationRepository
_verificationRepository;
      private readonly IGameRepository _gameRepository;

      public StartController(ICasinoPassRepository
casinoPassRepository, IVerificationRepository
verificationRepository,
                              IGameRepository gameRepository)
      {
            _casinoPassRepository = casinoPassRepository;
            _verificationRepository = verificationRepository;
            _gameRepository = gameRepository;
      }
      public RedirectToActionResult GiveAccessToCasinoGame(int
gameId)
      {
            //Due to coinflip being the only playable casino game,
I've hard coded it in so that
            //if the game being started is not coinflip, then
redirect the user to the error page.
            //This will not be the case in real life as every
casino game listed will have a game associated with it
            //but for the sake of demonstration there is multiple
games up that have no implementation
            if (gameId != 10)
            {
                  Response.StatusCode = 404;
                  return RedirectToAction("HandleError", "Error",
new { code = 404 });
            }
            string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
            var verification =
_verificationRepository.RetrieveVerificationByUserId(userId);
            if (verification == null ||verification.Status !=
"Approved" )
            {
                  return RedirectToAction("Index", "Verification",
new { code = 404 });
            }
            var ownedPasses =
```

```
_casinoPassRepository.GetCasinoPassesByUserId(userId);
            var gameDetails = _gameRepository.GetGameById(gameId);
            //remove space from game name
            string gameName = Regex.Replace(gameDetails.Name,
@"\s+", "");

            CasinoPass activePass = ownedPasses.FirstOrDefault(p =>
(p.GameId == gameId) && (p.Active == true) && (p.Expired ==
false));
            //if its active already
            if (activePass != null)
            {
                  return RedirectToAction("Index", "Play", new {
area = gameName, casinoPassId = activePass.CasinoPassId });
            }
            else
            {
                  CasinoPass firstValidPass = ownedPasses.Where(p =>
p.Active == false)
                                                .Where(p => p.Expired
== false)
                                                .Where(p => p.GameId ==
0)
                                                .FirstOrDefault();
                  //if no valid passes owned redirect to game pass
page
                  if (firstValidPass == null)
                  {
                  return RedirectToAction("Details", "Game", new {
id = -1 });
                  }
                  //activate pass and assign game to pass, redirect
to game library
                  else
                  {
                  Guid casinoPassId = firstValidPass.CasinoPassId;

_casinoPassRepository.ActivateCasinoPass(casinoPassId, gameId);
                  //return RedirectToAction("Index", "Play", new {
area = "CoinFlip", passId = casinoPassId });
                  return RedirectToAction("Index", "Library");
```

```
                }
            }
        }

        }
}
```

VerificationController.cs

```
using CentralGamesPlatform.IRepositories;
using CentralGamesPlatform.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Controllers
{
      [Authorize]
      public class VerificationController : Controller
      {
      private readonly IVerificationRepository
_verificationRepository;
      public VerificationController(IVerificationRepository
verificationRepository)
      {
            _verificationRepository = verificationRepository;
      }
      public IActionResult Index()
      {
            return View();
      }

      [HttpPost]
      [ValidateAntiForgeryToken]
      public IActionResult Index(FileUpload fileObject)
```

```
     {
          string[] permittedExtensions = { ".png", ".jpg" };
          var uploadedExtension =
Path.GetExtension(fileObject.file.FileName).ToLowerInvariant();
          string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
          var verificationExists =
_verificationRepository.RetrieveVerificationByUserId(userId);
          if (verificationExists == null)
          {
               if (fileObject.file.Length > 0 &&
permittedExtensions.Contains(uploadedExtension))
               {
               using (var memoryStream = new MemoryStream())
               {
                    fileObject.file.CopyTo(memoryStream);
                    if (memoryStream.Length < 2097152)
                    {
                         var verificationRequest = new
Verification()
                         {
                              Content = memoryStream.ToArray(),
                              UserId = userId,
                              Status = "Pending"

                         };

_verificationRepository.CreateVerification(verificationRequest);
                         ViewBag.SuccessMessage = "You have
submitted your photo ID for verification. This will be processed
by an admin shortly. You will not be able to play casino games
until it has been verified";
                    }
                    else
                    {
                         ModelState.AddModelError("File", "The
file is too large");
                    }
               }
               }
               else
```

```
                {
                ModelState.AddModelError("File", "The file is too
small or is in the wrong format");
                }
            }
            else if (verificationExists.Status == "Approved")
            {
                ViewBag.ErrorMessage = "Your account is already
verified";
            }
            else if (verificationExists.Status == "Denied")
            {
                ViewBag.ErrorMessage = "Your photo ID could not be
verified. Contact support if you think this is a mistake";
            }
            else if (verificationExists.Status == "Pending")
            {
                ViewBag.ErrorMessage = "Your verification request
is currently pending";
            }
            else if(verificationExists.Status == "")
            {
                ViewBag.ErrorMessage = "Something went wrong with
your verification. Please contact an admin";
            }

            return View();
        }
        }
}
```

WalletController.cs

```
using CentralGamesPlatform.IRepositories;
using CentralGamesPlatform.Models;
using CentralGamesPlatform.ViewModels;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Claims;
```

```
using System.Threading.Tasks;

namespace CentralGamesPlatform.Controllers
{
      public class WalletController : Controller
      {
      private readonly IWalletRepository _walletRepository;
      public WalletController(IWalletRepository walletRepository)
      {
            _walletRepository = walletRepository;
      }
      public ViewResult Index()
      {
            string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
            decimal balance =
_walletRepository.RetrieveBalance(userId);

            var walletViewModel = new WalletViewModel
            {
                  WalletBalance = balance
            };
            return View(walletViewModel);
      }
      }
}
```

Models

ApplicationUser.cs

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

namespace CentralGamesPlatform.Models
{
    public class ApplicationUser : IdentityUser
    {
```

```
        public DateTime DateOfBirth { get; set; }
    }
}
```

CasinoPass.cs

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

namespace CentralGamesPlatform.Models
{
      public class CasinoPass
      {
      [BindNever]
      public Guid CasinoPassId { get; set; }
      public string UserId { get; set; }
      public DateTime PassPlaced { get; set; }
      public bool Active { get; set; }
      public int GameId { get; set; }
      public bool Expired { get; set; }

      }
}
```

Category.cs

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

namespace CentralGamesPlatform.Models
{
    public class Category
    {
        public int CategoryId { get; set; }
```

```
        public string CategoryName { get; set; }
        public string CategoryDescription { get; set; }
        public List<Game> Games { get; set; }
    }
}
```

Download.cs

```
using Microsoft.AspNetCore.Mvc.ModelBinding;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Models
{
      public class Download
      {
      [BindNever]
      public int DownloadId { get; set; }
      public int GameId { get; set; }
      [Required]
      public byte[] Content { get; set; }
      [Required]
      public string FileName { get; set; }
      [Required]
      public string FileFormat { get; set; }
      public DateTime LastUpdated { get; set; }
      [Required]
      public double VersionNumber { get; set; }
      [NotMapped]
      public FileUpload fileObject { get; set; }
      }
}
```

FileUpload.cs

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

namespace CentralGamesPlatform.Models
{
      public class FileUpload
      {
      [Required]
      [Display(Name = "File")]
      public IFormFile file { get; set; }
      }
}
```

Game.cs

```
using Microsoft.AspNetCore.Mvc.ModelBinding;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Models
{
    public class Game
    {
       [BindNever]
       public int GameId { get; set; }
       [Required]
       public string Name { get; set; }
       [Required]
       public string Description { get; set; }
       [Required]
       [Range(0.00, 100.00)]
```

```
        public decimal Price { get; set; }
        [Required]
        public string ImageUrl { get; set; }
        [Required]
        public string ImageThumbnailUrl { get; set; }
        [Required]
        public bool IsOnSale { get; set; }
        [Required]
        public int CategoryId { get; set; }
        public Category Category { get; set; }
    }
}
```

License.cs

```
using Microsoft.AspNetCore.Mvc.ModelBinding;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Models
{
    public class Licence
    {
        [BindNever]
        public int LicenceId { get; set; }
        public Guid LicenseKey { get; set; }
        public int OrderDetailId { get; set; }
        public string UserId { get; set; }
    }
}
```

Order.cs

```
using Microsoft.AspNetCore.Mvc.ModelBinding;
using System;
```

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

namespace CentralGamesPlatform.Models
{
    public class Order
    {
       [BindNever]
       public int OrderId { get; set; }

       public string UserId { get; set; }

       [Required(ErrorMessage = "Required field")]
       [Display(Name = "First Name")]
       [StringLength(25)]
       public string FirstName { get; set; }

       [Required(ErrorMessage = "Required field")]
       [Display(Name = "Last Name")]
       [StringLength(50)]
       public string LastName { get; set; }

       [Required(ErrorMessage = "Required field")]
       [Display(Name = "Street Address")]
       [StringLength(100)]
       public string Address { get; set; }

       [Required(ErrorMessage = "Required field")]
       [StringLength(50)]
       public string Town { get; set; }

       [Required(ErrorMessage = "Required field")]
       [StringLength(50)]
       public string County { get; set; }

       [Required(ErrorMessage = "Required field")]
       [StringLength(10)]
       public string EirCode { get; set; }
```

```
        [Required(ErrorMessage = "Required field")]
        [DataType(DataType.PhoneNumber)]
        [Phone]
        public string PhoneNumber { get; set; }

        public List<OrderDetail> OrderDetails { get; set; }

        [BindNever]
        public decimal OrderTotal { get; set; }

        [BindNever]
        public DateTime OrderPlaced { get; set; }

        public bool IsSuccessful { get; set; }
    }
}
```

OrderDetail.cs

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

namespace CentralGamesPlatform.Models
{
    public class OrderDetail
    {
        public int OrderDetailId { get; set; }
        public int OrderId { get; set; }
        public int GameId { get; set; }
        public bool CasinoPass { get; set; }
        public Game Game { get; set; }
        public int Amount { get; set; }
        public decimal Price { get; set; }
        public Order Order { get; set; }

    }
}
```

Payment.cs

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

namespace CentralGamesPlatform.Models
{
    public class Payment
    {
      [BindNever]
      public int PaymentId { get; set; }
      public int OrderId { get; set; }
      public string StripeSession { get; set; }
      public decimal Total { get; set; }
      public DateTime PaymentDateTime { get; set; }
    }
}
```

Payout.cs

```
using Microsoft.AspNetCore.Mvc.ModelBinding;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Models
{
    public class Payout
    {
     [BindNever]
     public int PayoutId { get; set; }
     public int WalletId { get; set; }

     [Required(ErrorMessage = "Required field, must be greater be
between 5 and 100")]
     [Display(Name = "Transfer Amount")]
     [DataType(DataType.Currency)]
```

```
      [Range(5,100)]
      public decimal AmountTransfered { get; set; }

      [Required(ErrorMessage = "Required field")]
      [Display(Name = "PayPal Email")]
      [EmailAddress]
      public string PayPalEmail { get; set; }
      public string PayPalBatchId { get; set; }
      public DateTime PayoutDate { get; set; }

      }
}
```

Result.cs

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

namespace CentralGamesPlatform.Models
{
      public class Result
      {
      [BindNever]
      public int ResultId { get; set; }
      public Guid CasinoPassId { get; set; }
      public bool Win  { get; set; }
      public decimal AmountWon { get; set; }
      public DateTime DateResultPlaced { get; set; }
      }
}
```

ShoppingCart.cs

```
using Microsoft.AspNetCore.Http;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;
using System;
```

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

namespace CentralGamesPlatform.Models
{
    public class ShoppingCart
    {
       private readonly MyDatabaseContext _myDatabaseContext;
       public string ShoppingCartId { get; set; }
       public List<ShoppingCartItem> ShoppingCartItems { get; set;
}

       public ShoppingCart(MyDatabaseContext myDatabaseContext)
       {
            _myDatabaseContext = myDatabaseContext;
       }

       public static ShoppingCart GetCart(IServiceProvider
services)
       {
            ISession session =
services.GetRequiredService<IHttpContextAccessor>
                ()?.HttpContext.Session;

            var context =
services.GetService<MyDatabaseContext>();
            string cartId = session.GetString("CartId") ??
Guid.NewGuid().ToString();
            session.SetString("CartId", cartId);
            return new ShoppingCart(context) { ShoppingCartId =
cartId };
       }

       public void AddToCart(Game game, int amount)
       {
            var shoppingCartItem =
_myDatabaseContext.ShoppingCartItems.SingleOrDefault(
                s => s.Game.GameId == game.GameId &&
s.ShoppingCartId == ShoppingCartId);
```

```
            if(shoppingCartItem == null)
            {
                  shoppingCartItem = new ShoppingCartItem
                  {
                        ShoppingCartId = ShoppingCartId,
                        Game = game,
                        Amount = amount
                  };


_myDatabaseContext.ShoppingCartItems.Add(shoppingCartItem);
            }
            else
            {
                  shoppingCartItem.Amount++;
            }

            _myDatabaseContext.SaveChanges();
      }

      public int RemoveFromCart(Game game)
      {
            var shoppingCartItem =
_myDatabaseContext.ShoppingCartItems.SingleOrDefault(
                  s => s.Game.GameId == game.GameId &&
s.ShoppingCartId == ShoppingCartId);

            var localAmount = 0;
            if(shoppingCartItem != null)
            {
                  if(shoppingCartItem.Amount > 1)
                  {
                        shoppingCartItem.Amount--;
                        localAmount = shoppingCartItem.Amount;
                  }
                  else
                  {

_myDatabaseContext.ShoppingCartItems.Remove(shoppingCartItem);
                  }
            }
```

```
            _myDatabaseContext.SaveChanges();
            return localAmount;
      }

      public List<ShoppingCartItem> GetShoppingCartItems()
      {
            return ShoppingCartItems ?? (ShoppingCartItems =
_myDatabaseContext.ShoppingCartItems.Where(c
                  => c.ShoppingCartId == ShoppingCartId)
                  .Include(g => g.Game)
                  .ToList());
      }

      public void ClearCart()
      {
            var cartItems =
_myDatabaseContext.ShoppingCartItems.Where(c => c.ShoppingCartId
== ShoppingCartId);


_myDatabaseContext.ShoppingCartItems.RemoveRange(cartItems);
            _myDatabaseContext.SaveChanges();
      }

      public decimal GetShoppingCartTotal()
      {
            var total =
_myDatabaseContext.ShoppingCartItems.Where(c => c.ShoppingCartId
== ShoppingCartId)
                  .Select(g => g.Game.Price * g.Amount).Sum();

            return total;
      }
    }


}
```

ShoppingCartItem.cs

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

namespace CentralGamesPlatform.Models
{
    public class ShoppingCartItem
    {
       public int ShoppingCartItemId { get; set; }
       public string ShoppingCartId { get; set; }
       public Game Game { set; get; }
       public int Amount { get; set; }
    }
}
```

ShoppingCartItem.cs

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

namespace CentralGamesPlatform.Models
{
    public class ShoppingCartItem
    {
       public int ShoppingCartItemId { get; set; }
       public string ShoppingCartId { get; set; }
       public Game Game { set; get; }
       public int Amount { get; set; }
    }
}
```

Verification.cs

```
using Microsoft.AspNetCore.Mvc.ModelBinding;
using System;
using System.Collections.Generic;
```

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

namespace CentralGamesPlatform.Models
{
      public class Verification
      {
      [BindNever]
      public int VerificationId { get; set; }
      public string UserId { get; set; }
      public byte[] Content { get; set; }
      public string Status { get; set; }
      public DateTime DateOfRequest { get; set; }
      }
}
```

Wallet.cs

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

namespace CentralGamesPlatform.Models
{
      public class Wallet
      {
      public int WalletId { get; set; }
      public string UserId { get; set; }
      public decimal Balance { get; set; }
      }
}
```

Views

_ViewStart.cshtml

```
@{
      Layout = "_Layout";
}
```

_ViewImports.cshtml

```
@using CentralGamesPlatform
@using CentralGamesPlatform.Models
@using CentralGamesPlatform.ViewModels

@addTagHelper CentralGamesPlatform.TagHelpers.*,
CentralGamesPlatform
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

Admin

Create.cshtml

```
@model Game
@{
      ViewData["Title"] = "Create";
}

<h1>Create Game</h1>
<hr />
<div class="row">
      <div class="col-md-8">
      <form asp-action="Create" class="card">
            <div class="card-body">
                  <div asp-validation-summary="ModelOnly"
class="text-danger"></div>
                  <div class="form-group text-dark">
                  <label asp-for="Name"
class="control-label"></label>
                  <input asp-for="Name" class="form-control" />
                  <span asp-validation-for="Description"
class="text-danger"></span>
                  </div>
                  <div class="form-group text-dark">
                  <label asp-for="Description"
class="control-label"></label>
                  <input asp-for="Description" class="form-control"
/>
                  <span asp-validation-for="Description"
class="text-danger"></span>
                  </div>
                  <div class="form-group text-dark">
```

```
                <label asp-for="Price"
class="control-label"></label>
                <input asp-for="Price" class="form-control" />
                <span asp-validation-for="Price"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="ImageUrl"
class="control-label"></label>
                <input asp-for="ImageUrl" class="form-control" />
                <span asp-validation-for="ImageUrl"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="ImageThumbnailUrl"
class="control-label"></label>
                <input asp-for="ImageThumbnailUrl"
class="form-control" />
                <span asp-validation-for="ImageThumbnailUrl"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="IsOnSale"
class="control-label"></label>
                <input asp-for="IsOnSale" class="form-control"
style="width:10%;" />
                <span asp-validation-for="IsOnSale"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="CategoryId"
class="control-label"></label>
                <select asp-for="CategoryId" class="form-control">
                     <option value=1>Casino Game PC</option>
                     <option value=2>Casino Game Mobile</option>
                     <option value=3>Casino Game Both
Device</option>
                     <option value=4>Browser Game PC</option>
                     <option value=5>Browser Game Mobile</option>
                     <option value=6>Browser Game Both
Device</option>
```

```
                    <option value=7>Download Game PC</option>
                    <option value=8>Download Game
Mobile</option>
                    <option value=9>Download Game Both</option>
                </select>
                <span asp-validation-for="CategoryId"
class="text-danger"></span>
                </div>
                <div class="form-group">
                <input type="submit" value="Create" class="btn
btn-primary" />
                <a asp-action="Index" class="btn btn-success">Back
to List</a>
                </div>
            </div>
      </form>
      </div>
</div>

<div>

</div>
```

Details.cshtml

```
@model Game

@{
      ViewData["Title"] = "Details";
}

<h1>Details</h1>

<div>
      <h4>Game</h4>
      <hr />
      <div class="row">
      <div class="col-md-6">
            <div class="card">
                  <div class="card-body">
                  <div class="form-group text-dark">
```

```
                    <label asp-for="GameId"
class="control-label"></label><br />
                    <strong> @Html.DisplayFor(model =>
model.GameId)</strong>
                </div>

                <div class="form-group text-dark">
                    <label asp-for="CategoryId"
class="control-label"></label><br />
                    <strong> @Html.DisplayFor(model =>
model.CategoryId)</strong>
                </div>

                <div class="form-group text-dark">
                    <label asp-for="Name"
class="control-label"></label><br />
                    <strong> @Html.DisplayFor(model =>
model.Name)</strong>
                </div>
                <div class="form-group text-dark">
                    <label asp-for="Description"
class="control-label"></label><br />
                    <strong> @Html.DisplayFor(model =>
model.Description)</strong>
                </div>
                <div class="form-group text-dark">
                    <label asp-for="IsOnSale"
class="control-label"></label><br />
                    <strong> @Html.DisplayFor(model =>
model.IsOnSale)</strong>
                </div>
                <div class="form-group text-dark">
                    <label asp-for="ImageThumbnailUrl"
class="control-label"></label><br />
                    <strong> @Html.DisplayFor(model =>
model.ImageThumbnailUrl)</strong>
                </div>
                <div class="form-group text-dark">
                    <label asp-for="ImageUrl"
class="control-label"></label><br />
                    <strong> @Html.DisplayFor(model =>
```

```
model.ImageUrl)</strong>
                </div>
                <div class="form-group">
                     <a class="btn btn-primary" asp-action="Edit"
asp-route-gameId="@Model.GameId">Edit</a>
                     <a class="btn btn-success"
asp-action="ViewGames">Back to List</a>
                     @if (Model.CategoryId >= 4 &&
Model.CategoryId <= 9)
                     {
                          <a class="btn btn-primary"
asp-action="DownloadDetails" asp-route-gameId="@Model.GameId">Game
Files</a>
                     }
                </div>

                </div>
           </div>
      </div>
      </div>
</div>
<br/>
```

DownloadCreate.cshtml

```
@model Download
@{
      ViewData["Title"] = "Create Download";
}

<h1>Add Game Files</h1>
<p class="text-warning">When uploading a new game installer, it
may take a while to upload so do not leave or refresh the page, it
will be uploading as soon as you click update.</p>
<hr />
<div class="row">
      <div class="col-md-6">
      <form asp-action="DownloadCreate"
enctype="multipart/form-data" class="card">
           <div class="card-body">
                <div asp-validation-summary="ModelOnly"
```

```
class="text-danger"></div>

                <div class="form-group text-dark">
                <label asp-for="GameId"
class="control-label"></label><br />
                <strong>@Context.Request.Query["gameId"]</strong>
                </div>
                <div class="form-group text-dark">
                <label asp-for="FileName"
class="control-label"></label>
                <input asp-for="FileName" class="form-control" />
                <span asp-validation-for="FileName"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="FileFormat"
class="control-label"></label>
                <input asp-for="FileFormat" class="form-control"
/>
                <span asp-validation-for="FileFormat"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="VersionNumber"
class="control-label"></label>
                <input asp-for="VersionNumber"
class="form-control" />
                <span asp-validation-for="VersionNumber"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="fileObject.file"
class="control-label"></label>
                <input asp-for="fileObject.file"
class="form-control" type="file" style="width:60%;"/>
                <span asp-validation-for="fileObject.file"
class="text-danger"></span>
                </div>
                <div class="form-group">
                <input type="submit" value="Create" class="btn
btn-primary" asp-route-gameId="@Context.Request.Query["gameId"]"
```

```
/>
                <a class="btn btn-success"
asp-action="ViewGames">Back to List</a>
                </div>
           </div>
      </form>
      </div>
</div>
```

DownloadDetails.cshtml

```
@model AdminDownloadViewModel

@{
      ViewData["Title"] = "Download Details";
}

<h1>Details</h1>

<div>
      <h4>Game Files</h4>
      <hr />
      <div class="row">
      <div class="col-md-6">
           <div class="card">
                <div class="card-body">
                <div class="form-group text-dark">
                     <label asp-for="GameId"
class="control-label"></label><br />
                     <strong> @Html.DisplayFor(model =>
model.GameId)</strong>
                </div>

                <div class="form-group text-dark">
                     <label asp-for="FileName"
class="control-label"></label><br />
                     <strong> @Html.DisplayFor(model =>
model.FileName)</strong>
                </div>

                <div class="form-group text-dark">
```

```
                    <label asp-for="FileVersion"
class="control-label"></label><br />
                    <strong> @Html.DisplayFor(model =>
model.FileVersion)</strong>
                </div>
                <div class="form-group text-dark">
                    <label asp-for="LastUpdated"
class="control-label"></label><br />
                    <strong> @Html.DisplayFor(model =>
model.LastUpdated)</strong>
                </div>
                <div class="text-center">
                    <a class="btn btn-primary"
asp-action="DownloadUpdate"
asp-route-gameId="@Model.GameId">Update</a>
                    <a class="btn btn-success"
asp-action="ViewGames">Back to List</a>
                </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

DownloadUpdate.cshtml

```
@model Download
@{
    ViewData["Title"] = "Download Update";
}

<h1>Update Game Files</h1>
<p class="text-warning">When uploading a new game installer, it
may take a while to upload so do not leave or refresh the page, it
will be uploading as soon as you click update.</p>
<hr />
<div class="row">
    <div class="col-md-6">
    <form asp-action="DownloadUpdate"
enctype="multipart/form-data" class="card">
        <div class="card-body">
```

```
                <div asp-validation-summary="ModelOnly"
class="text-danger"></div>
                <div class="form-group text-dark">
                <label asp-for="DownloadId"
class="control-label"></label><br />
                <strong>@Html.DisplayFor(modelItem =>
Model.DownloadId)</strong>
                </div>
                <div class="form-group text-dark">
                <label asp-for="GameId"
class="control-label"></label><br />
                <strong>@Html.DisplayFor(modelItem =>
Model.GameId)</strong>
                </div>
                <div class="form-group text-dark">
                <label asp-for="LastUpdated"
class="control-label"></label><br />
                <strong>@Html.DisplayFor(modelItem =>
Model.LastUpdated)</strong>
                </div>
                <div class="form-group text-dark">
                <label asp-for="FileName"
class="control-label"></label>
                <input asp-for="FileName" class="form-control" />
                <span asp-validation-for="FileName"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="FileFormat"
class="control-label"></label>
                <input asp-for="FileFormat" class="form-control"
/>
                <span asp-validation-for="FileFormat"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="VersionNumber"
class="control-label"></label>
                <input asp-for="VersionNumber"
class="form-control" />
                <span asp-validation-for="VersionNumber"
```

```
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="fileObject.file"
class="control-label"></label>
                <input asp-for="fileObject.file"
class="form-control" type="file" style="width:60%;" />
                <span asp-validation-for="fileObject.file"
class="text-danger"></span>
                </div>
                <div class="form-group">
                <input type="submit" value="Update" class="btn
btn-primary" asp-route-downloadId="@Model.DownloadId" />
                <a class="btn btn-success"
asp-action="ViewGames">Back to List</a>
                </div>

           </div>
      </form>
      </div>
</div>
```

Edit.cshtml

```
@model Game

@{
      ViewData["Title"] = "Edit";
}

<h1>Edit</h1>

<h4>Game</h4>
<hr />
<div class="row">
      <div class="col-md-8">
      <form asp-action="Edit" class="card">
           <div class="card-body">
                <div asp-validation-summary="ModelOnly"
class="text-danger"></div>
                <div class="form-group text-dark">
```

```
                <label asp-for="Name"
class="control-label"></label>
                <input asp-for="Name" class="form-control" />
                <span asp-validation-for="Description"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="Description"
class="control-label"></label>
                <input asp-for="Description" class="form-control"
/>
                <span asp-validation-for="Description"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="Price"
class="control-label"></label>
                <input asp-for="Price" class="form-control" />
                <span asp-validation-for="Price"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="ImageUrl"
class="control-label"></label>
                <input asp-for="ImageUrl" class="form-control" />
                <span asp-validation-for="ImageUrl"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="ImageThumbnailUrl"
class="control-label"></label>
                <input asp-for="ImageThumbnailUrl"
class="form-control" />
                <span asp-validation-for="ImageThumbnailUrl"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="IsOnSale"
class="control-label"></label>
                <input asp-for="IsOnSale" class="form-control"
style="width:10%;" />
```

```
                <span asp-validation-for="IsOnSale"
class="text-danger" ></span>
                </div>
                <div class="form-group text-dark">
                <label asp-for="CategoryId"
class="control-label"></label>
                <select asp-for="CategoryId" class="form-control">
                     <option value=1>Casino Game PC</option>
                     <option value=2>Casino Game Mobile</option>
                     <option value=3>Casino Game Both
Device</option>
                     <option value=4>Browser Game PC</option>
                     <option value=5>Browser Game Mobile</option>
                     <option value=6>Browser Game Both
Device</option>
                     <option value=7>Download Game PC</option>
                     <option value=8>Download Game
Mobile</option>
                     <option value=9>Download Game Both</option>
                </select>
                <span asp-validation-for="CategoryId"
class="text-danger"></span>
                </div>
                <div class="form-group">
                <input type="submit" value="Update" class="btn
btn-primary" asp-route-gameId="@Model.GameId" />
                <a class="btn btn-success"
asp-action="ViewGames">Back to List</a>
                @if (Model.CategoryId >= 4 && Model.CategoryId <=
9)
                {
                     <a class="btn btn-primary"
asp-action="DownloadDetails" asp-route-gameId="@Model.GameId">Game
Files</a>
                }
                </div>
           </div>
      </form>
      </div>
</div>
```

Index.cshtml

```
@{
      ViewData["Title"] = "Admin";
}
<h1>Welcome Admin user!!</h1>
<h3>Select from the following administration operations:</h3>
<br />
<a asp-action="ViewGames">
      <input class="btn btn-primary" type="button"
value="View/Edit/Add Games">
</a>
<a asp-action="ViewVerificationRequests">
      <input class="btn btn-primary" type="button"
value="Verification Requests" />
</a>
```

UpdateVerificationRequest.cshtml

```
@model Verification

@{
      ViewData["Title"] = "Update Verification Request";
}

<h1>Update</h1>

<h4>Verification</h4>
<p>Determine if the photo ID submitted by the user is legitimate
based on the image provided.</p>
<hr />
<div class="row">
      <div class="col-md-6">
      <form asp-action="UpdateRequestPost" class="card">
            <div class="card-body">
                  <div asp-validation-summary="ModelOnly"
class="text-danger"></div>
                  <div class="form-group text-dark">
                  <label asp-for="VerificationId"
class="control-label"></label><br />
                  <strong>@Html.DisplayFor(modelItem =>
Model.VerificationId)</strong>
```

```
                </div>
                <div class="form-group text-dark">
                <label asp-for="UserId"
class="control-label"></label><br />
                <strong>@Html.DisplayFor(modelItem =>
Model.UserId)</strong>
                </div>
                <div class="form-group text-dark">
                <label asp-for="DateOfRequest"
class="control-label"></label><br />
                <strong>@Html.DisplayFor(modelItem =>
Model.DateOfRequest)</strong>
                </div>
                <div class="form-group text-dark ">
                <label asp-for="Status"
class="control-label">Current Status</label><br />
                <strong>@Html.DisplayFor(modelItem =>
Model.Status)</strong>
                </div>
                <div class="form-group text-dark">
                <label asp-for="Status"
class="control-label"></label>
                <select asp-for="Status" class="form-control">
                     <option value="Approved">Approved</option>
                     <option value="Denied">Denied</option>
                </select>
                <span asp-validation-for="Status"
class="text-danger"></span>
                </div>
                <div class="form-group text-dark">
                <input type="submit" value="Update" class="btn
btn-primary" asp-route-verificationId="@Model.VerificationId" />
                </div>
                <div>
                <a asp-action="ViewVerificationRequests">Back to
List</a>
                </div>
           </div>
      </form>
      </div>
```

```
      <div class="col-md-4">
      <div class="card">
            <div class="card-header text-dark">
                  Photo ID
            </div>
            <img src="@ViewData["Image"]"
height="445" width="300" class="card-img-bottom" />

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

ViewGames.cshtml

```
@model PaginatedList<Game>

@{
      ViewData["Title"] = "ViewGames";
}

<h2>View All Games</h2>
<h3>Click on a game to edit its details</h3>
<p>
      <a asp-action="Create">Add Game</a>
</p>

<form asp-action="ViewGames" method="get">
      <div class="form-actions no-color">
      <p>
            Find game by name: <input type="text"
name="SearchString" value="@ViewData["CurrentFilter"]" />
            <input type="submit" value="Search" class="btn
btn-primary" /> |
            <a asp-action="ViewGames">Back to Full List</a> |
            <a asp-action="Index">Return</a>
      </p>
      </div>
</form>

<table class="table table-light table-striped table-hover">
```

```
      <thead>
      <tr>
            <th>
                  <a asp-action="ViewGames"
asp-route-sortOrder="@ViewData["NameSortParam"]"
asp-route-currentFilter="@ViewData["CurrentFilter"]">Name</a>
            </th>
            <th>
                  Price
            </th>
            <th>
                  <a asp-action="ViewGames"
asp-route-sortOrder="@ViewData["CategoryIdSortParam"]"
asp-route-currentFilter="@ViewData["CurrentFilter"]">Category</a>
            </th>
            <th>

            </th>
      </tr>
      </thead>
      <tbody>
      @foreach (var item in Model)
      {
            if (item.GameId != -1)
            {
            <tr>
                  <td>
                  @Html.DisplayFor(modelItem => item.Name)
                  </td>
                  <td>
                  €@Html.DisplayFor(modelItem => item.Price)
                  </td>
                  <td>
                  @if (item.CategoryId >= 1 && item.CategoryId <= 3)
                  {
                        @:Casino Game
                  }
                  else if (item.CategoryId >= 4 && item.CategoryId
<= 6)
                  {
                        @:Browser Game
```

```
                }
                else if (item.CategoryId >= 7 && item.CategoryId
<= 9)
                {
                      @:Download Game
                }
                </td>
                <td>
                <a asp-action="Edit"
asp-route-gameId="@item.GameId">Edit</a> |
                <a asp-action="Details"
asp-route-gameId="@item.GameId">Details</a>
                </td>
            </tr>
            }
      }
      </tbody>
</table>

@{
      var prevDisabled = !Model.HasPreviousPage ? "disabled" : "";
      var nextDisabled = !Model.HasNextPage ? "disabled" : "";
}

<a asp-action="ViewGames"
asp-route-sortOrder="@ViewData["CurrentSort"]"
asp-route-pageNumber="@(Model.PageIndex - 1)"
asp-route-currentFilter="@ViewData["CurrentFilter"]" class="btn
btn-primary @prevDisabled">Previous</a>
<a asp-action="ViewGames"
asp-route-sortOrder="@ViewData["CurrentSort"]"
asp-route-pageNumber="@(Model.PageIndex + 1)"
asp-route-currentFilter="@ViewData["CurrentFilter"]" class="btn
btn-primary @nextDisabled">Next</a>
```

ViewVerificationRequests.cshtml

```
@model PaginatedList<Verification>

@{
      ViewData["Title"] = "View Verification Requests";
```

```
}
<p>
      <a asp-action="Index"></a>
</p>
<h2>View All Verification Requests</h2>
<h3>Approve or deny verification requests for casino games</h3>

<form asp-action="ViewVerificationRequests" method="get">
      <div class="form-actions no-color">
      <p>
            Search by user ID: <input type="text"
name="SearchString" value="@ViewData["CurrentFilter"]" />
            <input type="submit" value="Search" class="btn
btn-primary" /> |
            <a asp-action="ViewVerificationRequests">Back to Full
List</a> |
            <a asp-action="Index">Return</a>
      </p>
      </div>
</form>

<table class="table table-light table-striped table-hover">
      <thead>
      <tr>
            <th>
                  <a asp-action="ViewVerificationRequests"
asp-route-sortOrder="@ViewData["StatusSortParam"]"
asp-route-currentFilter="@ViewData["CurrentFilter"]">Status</a>
            </th>
            <th>
                  User ID
            </th>
            <th>
                  <a asp-action="ViewVerificationRequests"
asp-route-sortOrder="@ViewData["DateOfRequestSortParam"]"
asp-route-currentFilter="@ViewData["CurrentFilter"]">Date Of
Request</a>
            </th>
            <th>

            </th>
```

```
      </tr>
      </thead>
      <tbody>
      @foreach (var item in Model)
      {
            <tr>
                  <td>
                  @Html.DisplayFor(modelItem => item.Status)
                  </td>
                  <td>
                  @Html.DisplayFor(modelItem => item.UserId)
                  </td>
                  <td>
                  @Html.DisplayFor(modelItem => item.DateOfRequest)
                  </td>
                  <td>
                  <a asp-action="UpdateVerificationRequest"
asp-route-verificationId="@item.VerificationId">Update</a>
                  </td>
            </tr>
      }
      </tbody>
</table>

@{
      var prevDisabled = !Model.HasPreviousPage ? "disabled" : "";
      var nextDisabled = !Model.HasNextPage ? "disabled" : "";
}

<a asp-action="ViewVerificationRequests"
asp-route-sortOrder="@ViewData["CurrentSort"]"
asp-route-pageNumber="@(Model.PageIndex - 1)"
asp-route-currentFilter="@ViewData["CurrentFilter"]" class="btn
btn-primary @prevDisabled">Previous</a>
<a asp-action="ViewVerificationRequests"
asp-route-sortOrder="@ViewData["CurrentSort"]"
asp-route-pageNumber="@(Model.PageIndex + 1)"
asp-route-currentFilter="@ViewData["CurrentFilter"]" class="btn
btn-primary @nextDisabled">Next</a>
```

Contact

Index.cshtml

```
@{
      ViewData["Title"] = "Contact";
}
<h3>To get in contact with Central Games Platform, please use the
email link below</h3>

<email address="support@centralgamesplatform.com"
link-text="Contact Us"></email>
```

Download

PlayGame.cshtml

```
@model Download
@{
      ViewData["Title"] = "Play";
}
<canvas id="myCanvas" width="480" height="320"></canvas>
<script
src="data:text/javascript;base64,@ViewData["game"]"></script>
```

Error

HandleError.cshtml

```
@{
      ViewData["Title"] = "Error";
}
<h3 class="text-danger">An error has occurred</h3>
<p class="text-danger">@ViewBag.ErrorMessage</p>
```

Game

Details.cshtml

```
@model Game
@{
      ViewData["Title"] = "Game Details";
}
```

```
<h1>@Model.Name</h1>

<div class="figure">
     <img alt="@Model.Name" src="@Model.ImageUrl"
style="width:100%; height:480px;" />
</div>
<div class="card">
     <div class="card-body">
     @if (Model.CategoryId >= 1 && Model.CategoryId <= 3)
     {
          <h4 class="float-right text-dark">Casino Game</h4>
          <p class="text-dark">
               <strong>A casino pass is required to play this
game. You must have verified that you are over 18 before you can
play. Once you start the game, you will spend your casino pass and
won't be able to use it again unless you purchase another</strong>
          </p>
          <div class="purchaseCasinoPass">
               <a class="btn btn-success"
asp-controller="ShoppingCart" asp-action="AddToShoppingCart"
asp-route-gameId="-1">Purchase Casino Pass</a>
          </div>
     }
     else
     {
          <h3 class="float-right text-dark">€@Model.Price</h3>
     }

     <p class="text-dark">@Model.Description</p>
     <h5 class="text-dark">Supported Devices</h5>
     <div>
          @if (Model.CategoryId == 1 || Model.CategoryId == 4 ||
Model.CategoryId == 7)
          {
               <span>
               <img class="supportsWindows"
src="~/images/icons/windows.png" />
               </span>
          }
          else if (Model.CategoryId == 2 || Model.CategoryId == 5
```

```
|| Model.CategoryId == 8)
            {
                  <span>
                  <img class="supportsAndroid"
src="~/images/icons/android.png" />
                  </span>
            }
            else if (Model.CategoryId == -1)
            {
                  <span>
                  <img class="passIcon"
src="~/images/icons/pass.png" />
                  </span>
            }
            else
            {
                  <span>
                  <img class="supportsWindows"
src="~/images/icons/windows.png" />
                  </span>
                  <span>
                  <img class="supportsAndroid"
src="~/images/icons/android.png" />
                  </span>
            }
      </div>

      @if (Model.CategoryId >= 1 && Model.CategoryId <= 3)
      {
            <div class="startCasinoGame">
                  <a class="btn btn-success" asp-controller="Start"
asp-action="GiveAccessToCasinoGame"
asp-route-gameId="@Model.GameId">Start Game</a>
            </div>
      }
      else
      {
            <div class="addToCart">
                  <p class="button">
                  <a class="btn btn-success"
asp-controller="ShoppingCart" asp-action="AddToShoppingCart"
```

```
asp-route-gameId="@Model.GameId">Add to Cart</a>
                </p>
            </div>
      }
      </div>
</div>
```

List.cshtml

```
@model GameListViewModel
@{
      ViewData["Title"] = "Game List";
}

<h1>@Model.CurrentCategory</h1>
<div class="row">
      @foreach (var game in Model.Games)
      {
      if (Model.OwnedGameIds != null &&
Model.OwnedGameIds.Contains(game.GameId) && game.GameId != -1)
      {
            @await Html.PartialAsync("_GameCard", game, new
ViewDataDictionary(ViewData) { { "Owned", true } });
      }
      else
      {
            <partial name="_GameCard" model="game" />
      }

      }
</div>
```

Home

Index.cshtml

```
@model HomeViewModel
@{
      ViewData["Title"] = "Home Page";
}
```

```
<partial name="_Carousel" />

<h2>Games on sale</h2>

<div class="row">

      @foreach (var game in Model.GamesOnSale)
      {
      if (Model.OwnedGameIds != null &&
Model.OwnedGameIds.Contains(game.GameId) && game.GameId != -1)
      {
            @await Html.PartialAsync("_GameCard", game, new
ViewDataDictionary(ViewData) { { "Owned", true } });
      }
      else
      {
            <partial name="_GameCard" model="game" />
      }
      }

</div>
```

Search.cshtml

```
@model PaginatedList<Game>

@{
      ViewData["Title"] = "Search";
}

<table class="table table-light table-striped table-hover">
      <thead>
      <tr>
            <th>
                  Name
            </th>
            <th>
                  Price
            </th>
            <th>
                  Category
```

```
            </th>
      </tr>
      </thead>
      <tbody>
      @foreach (var item in Model)
      {
            <tr>
                  <td>
                  @Html.DisplayFor(modelItem => item.Name)
                  </td>
                  <td>
                  €@Html.DisplayFor(modelItem => item.Price)
                  </td>
                  <td>

                  @if (item.CategoryId >= 1 && item.CategoryId <= 3)
                  {
                  @:Casino Game
                  }
                  else if (item.CategoryId >= 4 && item.CategoryId
<= 6)
                  {
                  @:Browser Game
                  }
                  else if (item.CategoryId >= 7 && item.CategoryId
<= 9)
                  {
                  @:Download Game
                  }
                  else
                  {
                  @:Casino Pass
                  }
                  </td>
                  <td>
                  <a asp-action="Details" asp-controller="Game"
asp-route-id="@item.GameId">View</a>
                  </td>
            </tr>
      }
      </tbody>
```

```
</table>

@{
      var prevDisabled = !Model.HasPreviousPage ? "disabled" : "";
      var nextDisabled = !Model.HasNextPage ? "disabled" : "";
}

<a asp-action="Search" asp-route-pageNumber="@(Model.PageIndex -
1)" asp-route-currentFilter="@ViewData["CurrentFilter"]"
class="btn btn-primary text-light @prevDisabled">Previous</a>
<a asp-action="Search" asp-route-pageNumber="@(Model.PageIndex +
1)" asp-route-currentFilter="@ViewData["CurrentFilter"]"
class="btn btn-primary text-light @nextDisabled">Next</a>
```

Library

```
Index.cshtml
@model LibraryViewModel
@{
      ViewData["Title"] = "Library";
}
@if (TempData["ErrorMessage"] != null)
{
      <p class="text-warning">@TempData["ErrorMessage"]</p>
}
@if (Model.OwnedPasses.Any())
{
      <h2>You currently own @Model.OwnedPasses.Count() casino game
passes. These can be used only once on any casino game</h2>
}

@if (Model.ActiveCasinoGames.Any())
{
      <h2>You currently have active passes in the following
games:</h2>
      <div class="row">
      @foreach (var cg in Model.ActiveCasinoGames)
      {
            <partial name="_GameCard" model="cg" />
      }
      </div>
}
```

```
<h2>These are games that you currently own</h2>
<div class="row">
      @foreach (var game in Model.OwnedGames)
      {
      if (game.GameId != -1)
      {
            @await Html.PartialAsync("_GameCard", game, new
ViewDataDictionary(ViewData) { { "Owned", true } });
      }
      else
      {
            <partial name="_GameCard" model="game" />
      }
      }
</div>
```

Order

Checkout.cshtml

```
@model Order
@{
      ViewData["Title"] = "Checkout";
}
<h4>Purchase</h4>
<div class="row">
      <div class="col-md-6">
      <form asp-action="Checkout" method="post" class="card">
            <h5 class="card-title text-dark">Billing Address</h5>
            <div class="card-body">
                  <div class="form-group text-dark">
                  <label asp-for="FirstName"></label>
                  <input asp-for="FirstName" class="form-control" />
                  <span asp-validation-for="FirstName"
class="text-danger"></span>
                  </div>

                  <div class="form-group text-dark">
                  <label asp-for="LastName"></label>
                  <input asp-for="LastName" class="form-control" />
                  <span asp-validation-for="LastName"
```

```
class="text-danger"></span>
                </div>

                <div class="form-group text-dark">
                <label asp-for="Address"></label>
                <input asp-for="Address" class="form-control" />
                <span asp-validation-for="Address"
class="text-danger"></span>
                </div>

                <div class="form-group text-dark">
                <label asp-for="Town"></label>
                <input asp-for="Town" class="form-control" />
                <span asp-validation-for="Town"
class="text-danger"></span>
                </div>

                <div class="form-group text-dark">
                <label asp-for="County"></label>
                <input asp-for="County" class="form-control" />
                <span asp-validation-for="County"
class="text-danger"></span>
                </div>

                <div class="form-group text-dark">
                <label asp-for="EirCode"></label>
                <input asp-for="EirCode" class="form-control" />
                <span asp-validation-for="EirCode"
class="text-danger"></span>
                </div>

                <div class="form-group text-dark">
                <label asp-for="PhoneNumber"></label>
                <input asp-for="PhoneNumber" class="form-control"
/>
                <span asp-validation-for="PhoneNumber"
class="text-danger"></span>
                </div>

                <input type="submit" class="btn btn-primary"
value="Proceed" />
```

```
            </div>

      </form>
      </div>
      <div class="col-md-6">
      <img src="~/images/logo/logo-blue.png" height="400"
width="400" />
      <br />
      <p>
            You're one step closer! Fill in your details so you can
proceed.
      </p>
      </div>
</div>
```

Payment

Failed.cshtml

```
@{
      ViewData["Title"] = "Payment Failed";
}
<div class="card" style="width:100%;">
      <div class="card-body">
      <h3 class="text-danger card-title">Sorry, but something went
wrong with your payment method.</h3>
      <p class="text-dark">Please try again later and if the
problem persists please contact support.</p>
      </div>
      <div class="text-center">
      <a class="btn btn-primary" asp-controller="Home"
asp-action="Index">Continue Shopping</a>
      </div>
</div>
```

Index.cshtml

```
@model PaymentSummaryViewModel
@{
      ViewData["Title"] = "Payment";
}
<script src="https://js.stripe.com/v3/"></script>
```

```
<h1>Order Summary</h1>
<table class="table table-bordered table-striped table-light">
      <thead>
      <tr>
            <th>Selected Amount</th>
            <th>Game</th>
            <th class="text-right">Price</th>
            <th class="text-right">Sub Total</th>
      </tr>
      </thead>
      <tbody>
      @foreach (var item in Model.ShoppingCart.ShoppingCartItems)
      {
            <tr>
                  <td class="text-center">@item.Amount</td>
                  <td class="text-left">@item.Game.Name</td>
                  <td class="text-right">€@item.Game.Price</td>
                  <td class="text-right">
                  €@((item.Amount * item.Game.Price).ToString("F"))
                  </td>
            </tr>
      }
      </tbody>
      <tfoot>
      <tr>
            <td colspan="3" class="text-right">Total</td>
            <td
class="text-right">€@((Model.ShoppingCartTotal).ToString("F"))</td
>
      </tr>
      </tfoot>
</table>


<div class="card">
      <div class="card-body">
      <h5 class="card-title text-dark">Billing Details</h5>
      <p class="text-sm-left text-dark">
            First Name: @Model.Order.FirstName
      </p>
      <p class="text-sm-left text-dark">
```

```
            Last Name: @Model.Order.LastName
      </p>
      <p class="text-sm-left text-dark">
            Phone Number: @Model.Order.PhoneNumber
      </p>
      <p class="text-sm-left text-dark">
            Address: @Model.Order.Address
      </p>
      <p class="text-sm-left text-dark">
            County: @Model.Order.County
      </p>
      <p class="text-sm-left text-dark">
            EirCode: @Model.Order.EirCode
      </p>
      </div>
</div>
<br/>

<button type="button" class="btn btn-primary" id="checkout-button"
asp-controller="Payment"
asp-action="CreateCheckoutSession">Checkout</button>

<script type="text/javascript">
      // Create an instance of the Stripe object with your
publishable API key
      var stripe =
Stripe('pk_test_51IB0Z4BTwx1LYfRREoDEBBJOh4HQM4tCzZgvmqRqnutsLFWU2
rzPvaUT0pa7vTVp5WqKMMPxyx1nFzeeLjW4o7Eo00umgSxWfI');
      var checkoutButton =
document.getElementById('checkout-button');

      checkoutButton.addEventListener('click', function () {
      // Create a new Checkout Session using the server-side
endpoint you
      // created in step 3.
      fetch('/create-checkout-session', {
            method: 'POST',
      })
            .then(function (response) {
                  return response.json();
            })
```

```
            .then(function (session) {
                  return stripe.redirectToCheckout({ sessionId:
session.id });
            })
            .then(function (result) {
                  // If `redirectToCheckout` fails due to a browser
or network
                  // error, you should display the localized error
message to your
                  // customer using `error.message`.
                  if (result.error) {
                  alert(result.error.message);
                  }
            })
            .catch(function (error) {
                  console.error('Error:', error);
            });
      });
</script>
```

Success.cshtml

```
@{
      ViewData["Title"] = "Payment Success";
}
<div class="card">
      <div class="card-body" style="width:100%">
      <h3 class="card-title text-success">Success</h3>
      <p class="text-dark">
            Your purchased products have been added to your
library.
      </p>
      <div class="text-center">
            <a class="btn btn-success" asp-controller="Library"
asp-action="Index">Go To Library</a>
            <a class="btn btn-primary" asp-controller="Home"
asp-action="Index">Continue Shopping</a>
      </div>
      </div>
</div>
```

Payout

Index.cshtml

```
@model Payout
@{
      ViewData["Title"] = "Payout";
}
@if (ViewBag.SuccessMessage != null)
{
      <p class="text-success">@ViewBag.SuccessMessage</p>
}
@if (ViewBag.ErrorMessage != null)
{
      <p class="text-danger">@ViewBag.ErrorMessage</p>
}
<h1>Transfer winnings to PayPal</h1>
<p>To convert your winnings into real world cash, you must have an
active PayPal account. If you do not have one, please register for
one at <a href="www.paypal.com">this link here</a></p>

<h3>If you already have an account, enter your PayPal email and
the amount you want to transfer and click 'Transfer'</h3>
<form asp-action="Index" method="post" class="card">
      <div class="card-body">

      <h4 class="card-title text-dark">Purchase</h4>

      <div class="form-group text-dark">
            <label asp-for="PayPalEmail"></label>
            <input asp-for="PayPalEmail" class="form-control" />
            <span asp-validation-for="PayPalEmail"
class="text-danger"></span>
      </div>

      <div class="form-group text-dark">
            <label asp-for="AmountTransfered"></label>
            <input asp-for="AmountTransfered" class="form-control"
/>
            <span asp-validation-for="AmountTransfered"
class="text-danger"></span>
      </div>
      <div class="form-group">
```

```
            <input type="submit" class="btn btn-primary"
value="Submit" />
      </div>

      </div>
</form>
```

Shared

_Carousel.cshtml

```
<div id="genericCarousel" class="carousel slide"
data-ride="carousel">
      <ol class="carousel-indicators">
      <li data-target="#genericCarousel" data-slide="0"
class="active"></li>
      <li data-target="#genericCarousel" data-slide="1"></li>
      </ol>
      <div class="carousel-inner">
      <div class="carousel-item active">
            <img class="d-block w-100 carouselImage"
src="~/images/gameimages/pokerLarge.jpg" alt="Second slide" />
            <div class="carousel-caption d-none d-md-block">
                  <h5>Play Casino Games</h5>
                  <p>Purchase casino passes to play casino games for
a chance to win money</p>
            </div>
      </div>
      <div class="carousel-item">
            <img class="d-block w-100 carouselImage"
src="~/images/gameimages/gaming.jpg" alt="Third slide"  />
            <div class="carousel-caption d-none d-md-block">
                  <h5>Enjoy a Traditional Gaming Experience</h5>
                  <p>Enjoy video games you are already familiar with
or games that are entirely new to you! </p>
            </div>
      </div>
      </div>
      <a class="carousel-control-prev" href="#genericCarousel"
role="button" data-slide="prev">
      <span class="carousel-control-prev-icon"
```

```
aria-hidden="true"></span>
      <span class="sr-only">Previous</span>
      </a>
      <a class="carousel-control-next" href="#genericCarousel"
role="button" data-slide="next">
      <span class="carousel-control-next-icon"
aria-hidden="true"></span>
      <span class="sr-only">Next</span>
      </a>
</div>
```

_GameCard.cshtml

```
@model Game
<div class="col-sm-4 col-lg-4 col-md-4">
      <div class="card h-100 gameCard">
      <img src="@Model.ImageThumbnailUrl" alt="image thumbnail"
width="348" height="348" class="gameCardImage"/>
      <div class="card-body d-flex flex-column">
            <div class="figure-caption">
                  @if (!(ViewData["Owned"] != null &&
(bool)ViewData["Owned"] == true && Model.GameId != 1))
                  {
                  <h3 class="float-sm-right">€@Model.Price</h3>
                  }
                  @if (Model.CategoryId != -1)
                  {
                  <h3><a asp-controller="Game" asp-action="Details"
asp-route-id="@Model.GameId">@Model.Name</a></h3>
                  }
                  else
                  {
                  <h3>@Model.Name</h3>
                  }

                  <div>
                  @if (Model.CategoryId == 1 || Model.CategoryId ==
4 || Model.CategoryId == 7)
                  {
                        <span>
                              <img class="supportsWindows"
```

```
src="~/images/icons/windows.png" />
                    </span>
                }
                else if (Model.CategoryId == 2 || Model.CategoryId
== 5 || Model.CategoryId == 8)
                {
                    <span>
                        <img class="supportsAndroid"
src="~/images/icons/android.png" />
                    </span>
                }
                else if (Model.CategoryId == -1)
                {
                    <span>
                        <img class="passIcon"
src="~/images/icons/pass.png" />
                    </span>
                }
                else
                {
                    <span>
                        <img class="supportsWindows"
src="~/images/icons/windows.png" />
                    </span>
                    <span>
                        <img class="supportsAndroid"
src="~/images/icons/android.png" />
                    </span>
                }
                </div>
                <h3 class="game-description">@Model.Description
</h3>
            </div>
            @if (Model.CategoryId >= 1 && Model.CategoryId <= 3)
            {
                <div class="startCasinoGame">
                <a class="btn btn-success mt-auto start-game"
asp-controller="Start" asp-action="GiveAccessToCasinoGame"
asp-route-gameId="@Model.GameId">Start Game</a>
                </div>
            }
```

```
          else if (ViewData["Owned"] != null &&
(bool)ViewData["Owned"] == true && Model.GameId != 1)
          {
               if (Model.CategoryId >= 4 && Model.CategoryId <=
9)
               {
               <div class="playOwnedGame">
                    @if (Model.CategoryId >= 4 &&
Model.CategoryId <= 6)
                    {
                         <a class="btn btn-success mt-auto
start-game" asp-controller="Download" asp-action="PlayGame"
asp-route-gameId="@Model.GameId">Play Game</a>
                    }
                    else
                    {
                         <a class="btn btn-success mt-auto
start-game" asp-controller="Download" asp-action="DownloadGame"
asp-route-gameId="@Model.GameId">Play Game</a>

                    }
               </div>
               }
               else
               {
               <div class="playOwnedGame">
                    <a class="btn btn-success mt-auto
start-game" asp-controller="Start"
asp-action="GiveAccessToBrowserGame"
asp-route-gameId="@Model.GameId">Play Game</a>
               </div>
               }
          }
          else
          {
               <div class="addToCart">
               <a class="btn btn-success mt-auto start-game"
asp-controller="ShoppingCart" asp-action="AddToShoppingCart"
asp-route-gameId="@Model.GameId">Add to Cart</a>
               </div>
          }
```

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

_Layout.cshtml

```
@model Game
<div class="col-sm-4 col-lg-4 col-md-4">
      <div class="card h-100 gameCard">
      <img src="@Model.ImageThumbnailUrl" alt="image thumbnail"
width="348" height="348" class="gameCardImage"/>
      <div class="card-body d-flex flex-column">
            <div class="figure-caption">
                  @if (!(ViewData["Owned"] != null &&
(bool)ViewData["Owned"] == true && Model.GameId != 1))
                  {
                  <h3 class="float-sm-right">€@Model.Price</h3>
                  }
                  @if (Model.CategoryId != -1)
                  {
                  <h3><a asp-controller="Game" asp-action="Details"
asp-route-id="@Model.GameId">@Model.Name</a></h3>
                  }
                  else
                  {
                  <h3>@Model.Name</h3>
                  }

                  <div>
                  @if (Model.CategoryId == 1 || Model.CategoryId ==
4 || Model.CategoryId == 7)
                  {
                        <span>
                              <img class="supportsWindows"
src="~/images/icons/windows.png" />
                        </span>
                  }
                  else if (Model.CategoryId == 2 || Model.CategoryId
== 5 || Model.CategoryId == 8)
                  {
```

```
                    <span>
                        <img class="supportsAndroid"
src="~/images/icons/android.png" />
                    </span>
                }
                else if (Model.CategoryId == -1)
                {
                    <span>
                        <img class="passIcon"
src="~/images/icons/pass.png" />
                    </span>
                }
                else
                {
                    <span>
                        <img class="supportsWindows"
src="~/images/icons/windows.png" />
                    </span>
                    <span>
                        <img class="supportsAndroid"
src="~/images/icons/android.png" />
                    </span>
                }
                </div>
                <h3 class="game-description">@Model.Description
</h3>
            </div>
            @if (Model.CategoryId >= 1 && Model.CategoryId <= 3)
            {
                <div class="startCasinoGame">
                <a class="btn btn-success mt-auto start-game"
asp-controller="Start" asp-action="GiveAccessToCasinoGame"
asp-route-gameId="@Model.GameId">Start Game</a>
                </div>
            }
            else if (ViewData["Owned"] != null &&
(bool)ViewData["Owned"] == true && Model.GameId != 1)
            {
                if (Model.CategoryId >= 4 && Model.CategoryId <=
9)
                {
```

```
                <div class="playOwnedGame">
                    @if (Model.CategoryId >= 4 &&
Model.CategoryId <= 6)
                    {
                        <a class="btn btn-success mt-auto
start-game" asp-controller="Download" asp-action="PlayGame"
asp-route-gameId="@Model.GameId">Play Game</a>
                    }
                    else
                    {
                        <a class="btn btn-success mt-auto
start-game" asp-controller="Download" asp-action="DownloadGame"
asp-route-gameId="@Model.GameId">Play Game</a>

                    }
                </div>
                }
                else
                {
                <div class="playOwnedGame">
                    <a class="btn btn-success mt-auto
start-game" asp-controller="Start"
asp-action="GiveAccessToBrowserGame"
asp-route-gameId="@Model.GameId">Play Game</a>
                </div>
                }
            }
            else
            {
                <div class="addToCart">
                <a class="btn btn-success mt-auto start-game"
asp-controller="ShoppingCart" asp-action="AddToShoppingCart"
asp-route-gameId="@Model.GameId">Add to Cart</a>
                </div>
            }

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

_LoginPartial.cshtml

```
@using Microsoft.AspNetCore.Identity

@inject SignInManager<ApplicationUser> SignInManager
@inject UserManager<ApplicationUser> UserManager

<ul class="navbar-nav d-flex flex-row">
@if (SignInManager.IsSignedIn(User))
{
      <li class="nav-item">
      <a id="manage" class="nav-link text-light"
asp-area="Identity" asp-page="/Account/Manage/Index"
title="Manage">Hello @UserManager.GetUserName(User)!</a>
      </li>
      <li class="nav-item">
      <form id="logoutForm" class="form-inline" asp-area="Identity"
asp-page="/Account/Logout"
asp-route-returnUrl="@Url.Action("Index", "Home", new { area = ""
})">
            <button id="logout" type="submit" class="nav-link btn
btn-link text-light">Logout</button>
      </form>
      </li>
}
else
{
      <li class="nav-item">
      <a class="nav-link text-light" id="register"
asp-area="Identity" asp-page="/Account/Register">Register</a>
      </li>
      <li class="nav-item">
      <a class="nav-link text-light" id="login" asp-area="Identity"
asp-page="/Account/Login">Login</a>
      </li>
}
</ul>
```

_ValidationScriptsPartial.cshtml

```
<environment include="Development">
      <script
```

```
src="~/lib/jquery-validation/dist/jquery.validate.js"></script>
      <script
src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusi
ve.js"></script>
</environment>
<environment exclude="Development">
      <script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validate/1.17.0
/jquery.validate.min.js"

asp-fallback-src="~/lib/jquery-validation/dist/jquery.validate.min
.js"
            asp-fallback-test="window.jQuery &&
window.jQuery.validator"
            crossorigin="anonymous"

integrity="sha256-F6h55Qw6sweK+t7SiOJX+2bpSAa3b/fnlrVCJvmEj1A=">
      </script>
      <script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validation-unob
trusive/3.2.11/jquery.validate.unobtrusive.min.js"

asp-fallback-src="~/lib/jquery-validation-unobtrusive/jquery.valid
ate.unobtrusive.min.js"
            asp-fallback-test="window.jQuery &&
window.jQuery.validator && window.jQuery.validator.unobtrusive"
            crossorigin="anonymous"

integrity="sha256-9GycpJnliUjJDVDqP0UEu/bsm9U+3dnQUH8+3W10vkY=">
      </script>
</environment>
```

ShoppingCart

Index.cshtml

```
@model ShoppingCartViewModel
@{
      ViewData["Title"] = "Shopping Cart";
}
<h2>Shopping Cart:</h2>
```

```
<table class="table table-bordered table-striped table-light">
      <thead>
      <tr>
            <th> </th>
            <th>Selected Amount</th>
            <th>Game</th>
            <th class="text-right">Price</th>
            <th class="text-right">Sub Total</th>
      </tr>
      </thead>
      <tbody>
      @foreach (var item in Model.ShoppingCart.ShoppingCartItems)
      {
            <tr>
                  <td class="text-center"><a class="btn btn-danger"
asp-controller="ShoppingCart" asp-action="RemoveFromShoppingCart"
asp-route-gameId="@item.Game.GameId">Remove From Cart</a></td>
                  <td class="text-center">@item.Amount</td>
                  <td class="text-left">@item.Game.Name</td>
                  <td class="text-right">€@item.Game.Price</td>
                  <td class="text-right">
                  €@((item.Amount * item.Game.Price).ToString("F"))
                  </td>
            </tr>
      }
      </tbody>
      <tfoot>
      <tr>
            <td colspan="4" class="text-right">Total</td>
            <td
class="text-right">€@((Model.ShoppingCartTotal).ToString("F"))</td
>
      </tr>
      </tfoot>
</table>

<div class="text-center">
      <a class="btn btn-success" asp-controller="Order"
asp-action="Checkout">Checkout</a>
      <a class="btn btn-primary" asp-controller="Home"
asp-action="Index">Continue Shopping</a>
```

```
      <a class="btn btn-danger" asp-controller="ShoppingCart"
asp-action="ClearCart">Clear Cart</a>
</div>
```

Verification

Index.cshtml

```
@model FileUpload
@{
      ViewData["Title"] = "Verification";
}
@if (ViewBag.SuccessMessage != null)
{
      <p class="text-success">@ViewBag.SuccessMessage</p>
}
@if (ViewBag.ErrorMessage != null)
{
      <p class="text-danger">@ViewBag.ErrorMessage</p>
}
<h1>Verify your account</h1>
<p>To play casino games, you must verify your photo ID to prove
you are genuine and you are over the age of 18. This is not
required to play and purchase video games</p>
<p class="text-warning">* If you have already submitted your photo
ID, you must wait until it has been approved by an admin to
proceed</p>

<div class="card">
      <h5 class="card-title text-dark">Max file size is 2MB. Photo
must be in .jpg format</h5>
      <p class="text-secondary">In order to verify that you are
authentic, you must provide a picture of both your face and your
ID clearly in frame. If you cannot provide this, we have no choice
but to refuse your request.</p>
      <form asp-action="Index" enctype="multipart/form-data"
method="post" class="card-body">
      <div class="form-group text-dark">
            <label asp-for="file"></label>
            <input asp-for="file" class="form-control" type="file"
style="width:40%;" />
```

```
            <span asp-validation-for="file"
class="text-danger"></span>
      </div>
      <input type="submit" class="btn btn-success" value="Submit"
/>

      </form>
</div>
```

ViewModels

AdminDownloadViewModel.cs

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

namespace CentralGamesPlatform.ViewModels
{
      public class AdminDownloadViewModel
      {
      public string FileName { get; set; }
      public double FileVersion { get; set; }
      public DateTime LastUpdated { get; set; }
      public int GameId { get; set; }
      }
}
```

GameListViewModel.cs

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

namespace CentralGamesPlatform.ViewModels
{
    public class GameListViewModel
```

```
    {
        public IEnumerable<Game> Games { get; set; }
        public List<int> OwnedGameIds { get; set; }
        public string CurrentCategory { get; set; }
    }
}
```

HomeViewModel.cs

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

namespace CentralGamesPlatform.ViewModels
{
    public class HomeViewModel
    {
        public IEnumerable<Game> GamesOnSale { get; set; }
        public List<int> OwnedGameIds { get; set; }
    }
}
```

PaymentSummaryViewModel.cs

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

namespace CentralGamesPlatform.ViewModels
{
      public class LibraryViewModel
      {
      public IEnumerable<Game> OwnedGames { get; set; }
      public IEnumerable<CasinoPass> OwnedPasses { get; set; }
      public IEnumerable<Game> ActiveCasinoGames { get; set; }
      }
}
```

ShoppingCartViewModel.cs

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

namespace CentralGamesPlatform.ViewModels
{
    public class ShoppingCartViewModel
    {
       public ShoppingCart ShoppingCart { get; set; }
       public decimal ShoppingCartTotal { get; set; }
    }
}
```

WalletViewModel.cs

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

namespace CentralGamesPlatform.ViewModels
{
    public class ShoppingCartViewModel
    {
       public ShoppingCart ShoppingCart { get; set; }
       public decimal ShoppingCartTotal { get; set; }
    }
}
```

TagHelpers

EmailTagHelper.cs

```
using Microsoft.AspNetCore.Razor.TagHelpers;
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Threading.Tasks;

namespace CentralGamesPlatform.TagHelpers
{
    public class EmailTagHelper : TagHelper
    {
       public string Address { get; set; }
       public string LinkText { get; set; }

       public override void Process(TagHelperContext context,
TagHelperOutput output)
       {
            base.Process(context, output);

            output.TagName = "a";
            output.Attributes.SetAttribute("href", "mailto:" +
Address);
            output.Content.SetContent(LinkText);
       }
    }
}
```

Data

MyDatabaseContext.cs

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

namespace CentralGamesPlatform.Models
{
     public class MyDatabaseContext :
IdentityDbContext<ApplicationUser>
     {
     public MyDatabaseContext (DbContextOptions<MyDatabaseContext>
options) :
```

```
            base(options)
      {
            Database.SetCommandTimeout(10000);
      }
      public DbSet<Game> Games { get; set; }
      public DbSet<Category> Categories { get; set; }
      public DbSet<ShoppingCartItem> ShoppingCartItems { get; set;
}
       public DbSet<Order> Orders { get; set; }
       public DbSet<OrderDetail> OrderDetails { get; set; }
      public DbSet<Payment> Payments { get; set; }
      public DbSet<Licence> Licences { get; set; }
      public DbSet<CasinoPass> CasinoPasses { get; set; }
      public DbSet<Result> Results { get; set; }
      public DbSet<Wallet> Wallets { get; set; }
      public DbSet<Payout> Payouts { get; set; }
      public DbSet<Verification> Verifications { get; set; }
      public DbSet<Download> Downloads { get; set; }

       protected override void OnModelCreating(ModelBuilder
modelBuilder)
       {
            base.OnModelCreating(modelBuilder);
            modelBuilder.Entity<Category>().HasData(new Category {
CategoryId = -1, CategoryName = "Casino Pass"});
            modelBuilder.Entity<Category>().HasData(new Category {
CategoryId = 1, CategoryName = "Casino Game PC"});
            modelBuilder.Entity<Category>().HasData(new Category {
CategoryId = 2, CategoryName = "Casino Game Mobile" });
            modelBuilder.Entity<Category>().HasData(new Category {
CategoryId = 3, CategoryName = "Casino Game Both Device" });
            modelBuilder.Entity<Category>().HasData(new Category {
CategoryId = 4, CategoryName = "Browser Game PC" });
            modelBuilder.Entity<Category>().HasData(new Category {
CategoryId = 5, CategoryName = "Browser Game Mobile" });
            modelBuilder.Entity<Category>().HasData(new Category {
CategoryId = 6, CategoryName = "Browser Game Both Device" });
            modelBuilder.Entity<Category>().HasData(new Category {
CategoryId = 7, CategoryName = "Download Game PC" });
            modelBuilder.Entity<Category>().HasData(new Category {
CategoryId = 8, CategoryName = "Download Game Mobile" });
```

```
            modelBuilder.Entity<Category>().HasData(new Category {
CategoryId = 9, CategoryName = "Download Game Both Device" });

            modelBuilder.Entity<Game>().HasData(new Game
            {
                  GameId = -1,
                  Name = "Casino Pass",
                  Price = 5.00M,
                  Description = "Used to play casino games. One time
use",
                  CategoryId = -1,
                  ImageUrl = "\\images\\gameimages\\pokerLarge.jpg",
                  ImageThumbnailUrl =
"\\images\\gamethumbnails\\poker.jpg",
                  IsOnSale = false
            });

            modelBuilder.Entity<Game>().HasData(new Game
            {
                  GameId = 1,
                  Name = "Poker",
                  Price = 5.00M,
                  Description = "The card game poker where you can
potentially win money",
                  CategoryId = 1,
                  ImageUrl ="\\images\\gameimages\\pokerLarge.jpg",
                  ImageThumbnailUrl
="\\images\\gamethumbnails\\poker.jpg",
                  IsOnSale=true
            });

            modelBuilder.Entity<Game>().HasData(new Game
            {
                  GameId = 2,
                  Name = "Spin the wheel",
                  Price = 5.00M,
                  Description = "Spin the wheel for the chance of
winning awesome prizes",
                  CategoryId = 2,
                  ImageUrl =
"\\images\\gameimages\\placeholder.png",
```

```
                ImageThumbnailUrl =
"\\images\\gamethumbnails\\placeholder.gif",
                IsOnSale = true
            });

            modelBuilder.Entity<Game>().HasData(new Game
            {
                GameId = 3,
                Name = "Scratch card",
                Price = 5.00M,
                Description = "Scratch card where any prizes you
reveal are yours to keep",
                CategoryId = 3,
                ImageUrl =
"\\images\\gameimages\\placeholder.png",
                ImageThumbnailUrl =
"\\images\\gamethumbnails\\placeholder.gif",
                IsOnSale = true
            });

            modelBuilder.Entity<Game>().HasData(new Game
            {
                GameId = 4,
                Name = "Happy Wheels",
                Price = 10.00M,
                Description = "Ride a bike through obstacles to
get to the end",
                CategoryId = 4,
                ImageUrl =
"\\images\\gameimages\\placeholder.png",
                ImageThumbnailUrl =
"\\images\\gamethumbnails\\placeholder.gif",
                IsOnSale = true
            });

            modelBuilder.Entity<Game>().HasData(new Game
            {
                GameId = 5,
                Name = "Tetris",
                Price = 10.00M,
                Description = "Stack blocks neatly to increase
```

```
your score",
                CategoryId = 5,
                ImageUrl =
"\\images\\gameimages\\placeholder.png",
                ImageThumbnailUrl =
"\\images\\gamethumbnails\\placeholder.gif",
                IsOnSale = true
            });

            modelBuilder.Entity<Game>().HasData(new Game
            {
                GameId = 6,
                Name = "8 Ball Pool",
                Price = 0.00M,
                Description = "Play Pool against your friends",
                CategoryId = 6,
                ImageUrl =
"\\images\\gameimages\\placeholder.png",
                ImageThumbnailUrl =
"\\images\\gamethumbnails\\placeholder.gif",
                IsOnSale = true
            });

            modelBuilder.Entity<Game>().HasData(new Game
            {
                GameId = 7,
                Name = "Call of duty",
                Price = 60.00M,
                Description = "Use weapons to defeat your
enemies",
                CategoryId = 7,
                ImageUrl =
"\\images\\gameimages\\placeholder.png",
                ImageThumbnailUrl =
"\\images\\gamethumbnails\\placeholder.gif",
                IsOnSale = true
            });

            modelBuilder.Entity<Game>().HasData(new Game
            {
                GameId = 8,
```

```
                Name = "Fruit Ninja",
                Price = 5.00M,
                Description = "Slice the falling fruit to increase
your score",
                CategoryId = 8,
                ImageUrl =
"\\images\\gameimages\\placeholder.png",
                ImageThumbnailUrl =
"\\images\\gamethumbnails\\placeholder.gif",
                IsOnSale = false
           });

           modelBuilder.Entity<Game>().HasData(new Game
           {
                GameId = 9,
                Name = "Among us",
                Price = 5.00M,
                Description = "Defeat all enemies as the imposter
to win",
                CategoryId = 9,
                ImageUrl =
"\\images\\gameimages\\placeholder.png",
                ImageThumbnailUrl =
"\\images\\gamethumbnails\\placeholder.gif",
                IsOnSale = true
           });

           modelBuilder.Entity<Game>().HasData(new Game
           {
                GameId = 10,
                Name = "Coin flip",
                Price = 5.00M,
                Description = "Flip a coin for the chance of
doubling your money",
                CategoryId = 1,
                ImageUrl =
"\\images\\gameimages\\placeholder.png",
                ImageThumbnailUrl =
"\\images\\gamethumbnails\\placeholder.gif",
                IsOnSale = true
           });
```

```
      }
      }
}
```

Components

CategoryMenu.cs

```
using CentralGamesPlatform.IRepositories;
using CentralGamesPlatform.Models;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Components
{
    public class CategoryMenu : ViewComponent
    {
      private readonly ICategoryRepository _categoryRepository;
      public CategoryMenu(ICategoryRepository categoryRepository)
      {
            _categoryRepository = categoryRepository;
      }

      public IViewComponentResult Invoke()
      {
            var categories =
_categoryRepository.GetAllCategories.OrderBy(c => c.CategoryName);

            return View(categories);
      }
    }
}
```

ShoppingCartSummary.cs

```
using CentralGamesPlatform.Models;
using CentralGamesPlatform.ViewModels;
using Microsoft.AspNetCore.Mvc;
```

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

namespace CentralGamesPlatform.Components
{
    public class ShoppingCartSummary : ViewComponent
    {
       private readonly ShoppingCart _shoppingCart;
       public ShoppingCartSummary(ShoppingCart shoppingCart)
       {
            _shoppingCart = shoppingCart;
       }

       public IViewComponentResult Invoke()
       {
            _shoppingCart.ShoppingCartItems =
_shoppingCart.GetShoppingCartItems();

            var shoppingCartViewModel = new ShoppingCartViewModel
            {
                 ShoppingCart = _shoppingCart,
                 ShoppingCartTotal =
_shoppingCart.GetShoppingCartTotal()
            };

            return View(shoppingCartViewModel);
       }
    }
}
```

WalletBalance.cs

```
using CentralGamesPlatform.IRepositories;
using CentralGamesPlatform.Models;
using CentralGamesPlatform.ViewModels;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Components
{
      public class WalletBalance : ViewComponent
      {
      private readonly IWalletRepository _walletRepository;
      private readonly IHttpContextAccessor _httpContextAccessor;
      public WalletBalance(IWalletRepository walletRepository,
IHttpContextAccessor httpContextAccessor)
      {
            _walletRepository = walletRepository;
            _httpContextAccessor = httpContextAccessor;
      }

      public IViewComponentResult Invoke()
      {
            string userId =
_httpContextAccessor.HttpContext.User.FindFirstValue(ClaimTypes.Na
meIdentifier);
            decimal balance =
_walletRepository.RetrieveBalance(userId);
            var walletViewModel = new WalletViewModel
            {
                  WalletBalance = balance
            };
            return View(walletViewModel);
      }
      }
}
```

IRepositories

ICasinoPassRepository.cs

```
using CentralGamesPlatform.Models;
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Threading.Tasks;

namespace CentralGamesPlatform.IRepositories
{
      public interface ICasinoPassRepository
      {
      void CreateCasinoPass(CasinoPass casinoPass);
      IEnumerable<CasinoPass> GetCasinoPassesByUserId(string
userId);
      CasinoPass GetCasinoPass(Guid casinoPassId);
      void ActivateCasinoPass(Guid casinoPassId, int gameId);
      void ExpireCasinoPass(Guid casinoPassId);
      int AmountPurchasedToday(string userId);
      }
}
```

ICategoryRepository.cs

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

namespace CentralGamesPlatform.IRepositories
{
    public interface ICategoryRepository
    {
       IEnumerable<Category> GetAllCategories { get; }
    }
}
```

IDownloadRepository.cs

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

namespace CentralGamesPlatform.IRepositories
```

```
{
      public interface IDownloadRepository
      {
      void CreateDownload(Download download);
      void UpdateDownload(Download download);
      Download RetrieveDownloadById(int gameId);
      Download RetrieveDownloadByGameId(int gameId);
      string GetDownloadFileName(int gameId);
      double GetDownloadVersion(int gameId);
      DateTime GetDownloadLastUpdated(int gameId);
      }
}
```

IGameRepository.cs

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

namespace CentralGamesPlatform.IRepositories
{
    public interface IGameRepository
    {
       IEnumerable<Game> GetAllGames { get; }
       IEnumerable<Game> GetGamesOnSale { get; }
       Game GetGameById(int gameId);
    }
}
```

ILicenseRepository.cs

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

namespace CentralGamesPlatform.IRepositories
{
```

```
    public interface IGameRepository
    {
        IEnumerable<Game> GetAllGames { get; }
        IEnumerable<Game> GetGamesOnSale { get; }
        Game GetGameById(int gameId);
    }
}
```

IOrderRepository.cs

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

namespace CentralGamesPlatform.IRepositories
{
    public interface IOrderDetailRepository
    {
        List<OrderDetail> GetAllOrderDetails(int orderId);
        OrderDetail GetOrderDetailById(int orderDetailId);
    }
}
```

IPaymentRepository.cs

```
using Stripe.Checkout;
using CentralGamesPlatform.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CentralGamesPlatform.IRepositories
{
    public interface IPaymentRepository
    {
        void CreatePayment(string stripeSession, Payment payment,
int orderid, decimal total);
```

```
    }
}
```

IPayoutRepository.cs

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

namespace CentralGamesPlatform.IRepositories
{
      public interface IPayoutRepository
      {
      void CreatePayout(Payout payout);
      }
}
```

IResultRepository.cs

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

namespace CentralGamesPlatform.IRepositories
{
      public interface IResultRepository
      {
      void CreateResult(Result result);
      }
}
```

IVerificationRepository.cs

```
using CentralGamesPlatform.Models;
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Threading.Tasks;

namespace CentralGamesPlatform.IRepositories
{
      public interface IVerificationRepository
      {
      void CreateVerification(Verification verification);
      void UpdateVerification(int verificationId, string status);
      IEnumerable<Verification> RetrieveAllVerifications();
      Verification RetrieveVerificationById(int verificationId);
      Verification RetrieveVerificationByUserId(string userId);

      }
}
```

IWalletRepository.cs

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

namespace CentralGamesPlatform.IRepositories
{
      public interface IWalletRepository
      {
      void AddToWallet(string userId, decimal amountToAdd);
      void SubtractFromWallet(string userId, decimal
amountToSubtract);
      decimal RetrieveBalance(string userId);
      int RetrieveWalletId(string userId);
      }
}
```

CasinoPassRepository.cs

```
using CentralGamesPlatform.IRepositories;
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Threading.Tasks;

namespace CentralGamesPlatform.Models
{
      public class CasinoPassRepository : ICasinoPassRepository
      {
      private readonly MyDatabaseContext _myDatabaseContext;

      public CasinoPassRepository(MyDatabaseContext
myDatabaseContext)
      {
            _myDatabaseContext = myDatabaseContext;
      }
      public void CreateCasinoPass(CasinoPass casinoPass)
      {
            casinoPass.PassPlaced = DateTime.Now;
            _myDatabaseContext.CasinoPasses.Add(casinoPass);
      }

      public void ActivateCasinoPass(Guid casinoPassId, int gameId)
      {
            CasinoPass pass = (from p in
_myDatabaseContext.CasinoPasses
                              where p.CasinoPassId == casinoPassId
select p).Single();
            pass.Active = true;
            pass.GameId = gameId;
            _myDatabaseContext.SaveChanges();
      }

      public void ExpireCasinoPass(Guid casinoPassId)
      {
            CasinoPass pass = (from p in
_myDatabaseContext.CasinoPasses
                              where p.CasinoPassId == casinoPassId
                              select p).Single();
            pass.Active = false;
            pass.Expired = true;
            _myDatabaseContext.SaveChanges();
      }
```

```
      public CasinoPass GetCasinoPass(Guid casinoPassId)
      {
            CasinoPass pass = (from p in
_myDatabaseContext.CasinoPasses
                               where p.CasinoPassId == casinoPassId
                               select p).SingleOrDefault();
            return pass;
      }

      public IEnumerable<CasinoPass> GetCasinoPassesByUserId(string
userId)
      {
            var passes = (from p in _myDatabaseContext.CasinoPasses
                               where p.UserId == userId
                               select p).ToList();
            return passes;
      }

      public int AmountPurchasedToday(string userId)
      {
            DateTime todaysDateTime = DateTime.Today; //Todays date
and time at 00:00:00
            DateTime yesterdaysDateTime =
DateTime.Today.AddDays(1).AddTicks(-1); //Todays date and time at
23:59:59
            var count = (from p in
_myDatabaseContext.CasinoPasses.OrderByDescending(p =>
p.PassPlaced)
                         where (p.PassPlaced > todaysDateTime &&
p.PassPlaced <= yesterdaysDateTime && p.UserId == userId)
                         select p).Count();
            return count;
      }
      }
}
```

CategoryRepository.cs

```
using CentralGamesPlatform.IRepositories;
using System;
using System.Collections.Generic;
```

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

namespace CentralGamesPlatform.Models
{
    public class CategoryRepository : ICategoryRepository
    {
       private readonly MyDatabaseContext _myDatabaseContext;
       public CategoryRepository(MyDatabaseContext
myDatabaseContext)
       {
            _myDatabaseContext = myDatabaseContext;
       }
       public IEnumerable<Category> GetAllCategories =>
_myDatabaseContext.Categories;
    }
}
```

DownloadRepository.cs

```
using CentralGamesPlatform.IRepositories;
using CentralGamesPlatform.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Repositories
{
     public class DownloadRepository : IDownloadRepository
     {
     private readonly MyDatabaseContext _myDatabaseContext;
     public DownloadRepository(MyDatabaseContext
myDatabaseContext)
     {
          _myDatabaseContext = myDatabaseContext;
     }
     public void CreateDownload(Download download)
     {
          download.LastUpdated = DateTime.Now;
          _myDatabaseContext.Downloads.Add(download);
```

```
            _myDatabaseContext.SaveChanges();
      }

      public string GetDownloadFileName(int gameId)
      {
            var result = (from d in _myDatabaseContext.Downloads
                          where d.GameId == gameId
                          select d.FileName).SingleOrDefault();
            return result;
      }

      public DateTime GetDownloadLastUpdated(int gameId)
      {
            var result = (from d in _myDatabaseContext.Downloads
                          where d.GameId == gameId
                          select
d.LastUpdated).SingleOrDefault();
            return result;
      }

      public double GetDownloadVersion(int gameId)
      {
            var result = (from d in _myDatabaseContext.Downloads
                          where d.GameId == gameId
                          select
d.VersionNumber).SingleOrDefault();
            return result;
      }

      public Download RetrieveDownloadByGameId(int gameId)
      {
            var result = (from d in _myDatabaseContext.Downloads
                          where d.GameId == gameId
                          select d).SingleOrDefault();
            return result;
      }

      public Download RetrieveDownloadById(int downloadId)
      {
            var result = (from d in _myDatabaseContext.Downloads
                          where d.DownloadId == downloadId
```

```
                              select d).SingleOrDefault();
            return result;
      }

      public void UpdateDownload(Download download)
      {
            var result = (from d in _myDatabaseContext.Downloads
                              where d.DownloadId ==
download.DownloadId
                              select d).SingleOrDefault();
            result.Content = download.Content;
            result.VersionNumber = download.VersionNumber;
            result.LastUpdated = DateTime.Now;

            _myDatabaseContext.SaveChanges();
      }
      }
}
```

GameRepository.cs

```
using CentralGamesPlatform.IRepositories;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Models
{
    public class GameRepository : IGameRepository
    {
       private readonly MyDatabaseContext _myDatabaseContext;
       public GameRepository (MyDatabaseContext myDatabaseContext)
       {
             _myDatabaseContext = myDatabaseContext;
       }
       public IEnumerable<Game> GetAllGames
       {
             get
             {
```

```
                return _myDatabaseContext.Games.Include(g =>
g.Category);
            }
      }

      public IEnumerable<Game> GetGamesOnSale
      {
            get
            {
                return _myDatabaseContext.Games.Include(g =>
g.Category).Where(s => s.IsOnSale);
            }
      }

      public Game GetGameById(int gameId)
      {
            return _myDatabaseContext.Games.FirstOrDefault(g =>
g.GameId == gameId);
      }
    }
}
```

LicenseRepository.cs

```
using CentralGamesPlatform.IRepositories;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Models
{
    public class GameRepository : IGameRepository
    {
      private readonly MyDatabaseContext _myDatabaseContext;
      public GameRepository (MyDatabaseContext myDatabaseContext)
      {
            _myDatabaseContext = myDatabaseContext;
      }
      public IEnumerable<Game> GetAllGames
```

```
        {
            get
            {
                return _myDatabaseContext.Games.Include(g =>
g.Category);
            }
        }

        public IEnumerable<Game> GetGamesOnSale
        {
            get
            {
                return _myDatabaseContext.Games.Include(g =>
g.Category).Where(s => s.IsOnSale);
            }
        }

        public Game GetGameById(int gameId)
        {
            return _myDatabaseContext.Games.FirstOrDefault(g =>
g.GameId == gameId);
        }
    }
}
```

OrderDetailRepository.cs

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

namespace CentralGamesPlatform.Models
{
    public class OrderDetailRepository : IOrderDetailRepository
    {
        private readonly MyDatabaseContext _myDatabaseContext;
        public OrderDetailRepository(MyDatabaseContext
myDatabaseContext)
        {
```

```
            _myDatabaseContext = myDatabaseContext;
      }
      public List<OrderDetail> GetAllOrderDetails(int orderId)
      {
            var orderDetails = (from od in
_myDatabaseContext.OrderDetails
                                    where od.OrderId == orderId
                                    select od).ToList();
            return orderDetails;
      }

     public OrderDetail GetOrderDetailById(int orderDetailId)
     {
            return
_myDatabaseContext.OrderDetails.FirstOrDefault(od =>
od.OrderDetailId == orderDetailId);
     }
     }
}
```

OrderRepository.cs

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

namespace CentralGamesPlatform.Models
{
    public class OrderRepository : IOrderRepository
    {
      private readonly MyDatabaseContext _myDatabaseContext;
      private readonly ShoppingCart _shoppingCart;

      public OrderRepository(MyDatabaseContext myDatabaseContext,
ShoppingCart shoppingCart)
      {
            _myDatabaseContext = myDatabaseContext;
            _shoppingCart = shoppingCart;
      }
```

```
      public void CreateOrder(Order order)
      {
            order.OrderPlaced = DateTime.Now;
            order.OrderTotal =
_shoppingCart.GetShoppingCartTotal();
            _myDatabaseContext.Orders.Add(order);
            _myDatabaseContext.SaveChanges();

            var shoppingCartItems =
_shoppingCart.GetShoppingCartItems();

            foreach(var cartItem in shoppingCartItems)
            {
                  if (cartItem.Game.GameId == -1)
                  {
                        for (int i = 0; i < cartItem.Amount; i++)
                        {
                              var orderDetail = new OrderDetail
                              {
                                    Amount = cartItem.Amount,
                                    Price = cartItem.Game.Price,
                                    GameId = cartItem.Game.GameId,
                                    OrderId = order.OrderId,
                                    CasinoPass = true
                              };

_myDatabaseContext.OrderDetails.Add(orderDetail);
                        }
                  }
                  else
                  {
                        var orderDetail = new OrderDetail
                        {
                              Amount = cartItem.Amount,
                              Price = cartItem.Game.Price,
                              GameId = cartItem.Game.GameId,
                              OrderId = order.OrderId
                        };

_myDatabaseContext.OrderDetails.Add(orderDetail);
                  }
```

```
            }

            _myDatabaseContext.SaveChanges();
      }
      public void SuccessfulOrder(int orderId)
      {
            Order result = (from o in _myDatabaseContext.Orders
                            where o.OrderId == orderId
select o).SingleOrDefault();

            result.IsSuccessful = true;

            _myDatabaseContext.SaveChanges();
      }
      public Order GetOrder(int orderId)
      {
            Order result = (from o in _myDatabaseContext.Orders
                            where o.OrderId == orderId
                            select o).SingleOrDefault();
            return result;
      }

     public List<Order> GetUsersOrders(string userid)
     {
            var orders = (from o in _myDatabaseContext.Orders
                                                  where o.UserId
== userid
                                                  select
o).ToList();
            return orders;
     }
     }
}
```

PaymentRepository.cs

```
using CentralGamesPlatform.IRepositories;
using Newtonsoft.Json.Linq;
using Stripe.Checkout;
```

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

namespace CentralGamesPlatform.Models
{
    public class PaymentRepository : IPaymentRepository
    {
       private readonly MyDatabaseContext _myDatabaseContext;

       public PaymentRepository(MyDatabaseContext
myDatabaseContext)
       {
            _myDatabaseContext = myDatabaseContext;
       }

       public void CreatePayment(string stripeSession, Payment
payment, int orderid, decimal total)
       {
            payment.OrderId = orderid;
            payment.StripeSession = stripeSession;
            payment.Total = total;
            payment.PaymentDateTime = DateTime.Now;

            _myDatabaseContext.Payments.Add(payment);
            _myDatabaseContext.SaveChanges();
       }
    }
}
```

PayoutRepository.cs

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

namespace CentralGamesPlatform.Models
{
```

```
      public class PayoutRepository : IPayoutRepository
      {
      private readonly MyDatabaseContext _myDatabaseContext;
      public PayoutRepository(MyDatabaseContext myDatabaseContext)
      {
            _myDatabaseContext = myDatabaseContext;
      }
      public void CreatePayout(Payout payout)
      {
            payout.PayoutDate = DateTime.Now;

            _myDatabaseContext.Payouts.Add(payout);
            _myDatabaseContext.SaveChanges();
      }
      }
}
```

ResultRepository.cs

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

namespace CentralGamesPlatform.Models
{
      public class ResultRepository : IResultRepository
      {
      private readonly MyDatabaseContext _myDatabaseContext;
      public ResultRepository(MyDatabaseContext myDatabaseContext)
      {
            _myDatabaseContext = myDatabaseContext;
      }
      public void CreateResult(Result result)
      {
            result.DateResultPlaced = DateTime.Now;
            _myDatabaseContext.Results.Add(result);
            _myDatabaseContext.SaveChanges();
      }
      }
```

```
}
```

VerificationRepository.cs

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

namespace CentralGamesPlatform.Models
{
      public class VerificationRepository : IVerificationRepository
      {
      private readonly MyDatabaseContext _myDatabaseContext;
      public VerificationRepository(MyDatabaseContext
myDatabaseContext)
      {
            _myDatabaseContext = myDatabaseContext;
      }
      public void CreateVerification(Verification verification)
      {
            verification.DateOfRequest = DateTime.Now;
            _myDatabaseContext.Verifications.Add(verification);
            _myDatabaseContext.SaveChanges();
      }

      public IEnumerable<Verification> RetrieveAllVerifications()
      {
            var result = (from v in
_myDatabaseContext.Verifications
                                  select v).ToList();
            return result;
      }

      public Verification RetrieveVerificationById(int
verificationId)
      {
            var result = (from v in
_myDatabaseContext.Verifications
                                  where v.VerificationId ==
```

```
verificationId
                                    select v).SingleOrDefault();
            return result;
      }

      public Verification RetrieveVerificationByUserId(string
userId)
      {
            var result = (from v in
_myDatabaseContext.Verifications
                                    where v.UserId == userId
                                    select v).SingleOrDefault();
            return result;
      }

      public void UpdateVerification(int verificationId, string
status)
      {
            var result = (from v in
_myDatabaseContext.Verifications
                                    where v.VerificationId ==
verificationId
                                    select v).SingleOrDefault();

            result.Status = status;

            _myDatabaseContext.SaveChanges();
      }
      }
}
```

WalletRepository.cs

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

namespace CentralGamesPlatform.Models
{
```

```
public class WalletRepository : IWalletRepository
{
private readonly MyDatabaseContext _myDatabaseContext;
public WalletRepository(MyDatabaseContext myDatabaseContext)
{
      _myDatabaseContext = myDatabaseContext;
}
public void AddToWallet(string userId, decimal amountToAdd)
{
      Wallet wallet = (from w in _myDatabaseContext.Wallets
                       where w.UserId == userId
                       select w).SingleOrDefault();
      if (wallet == null)
      {
            Wallet newWallet = new Wallet();
            newWallet.UserId = userId;
            newWallet.Balance = 0.00M;
            _myDatabaseContext.Wallets.Add(newWallet);
            decimal result = newWallet.Balance + amountToAdd;
            newWallet.Balance = result;
            _myDatabaseContext.SaveChanges();
      }
      else
      {
            decimal result = wallet.Balance + amountToAdd;
            wallet.Balance = result;
            _myDatabaseContext.SaveChanges();
      }

}

public decimal RetrieveBalance(string userId)
{
      Wallet wallet = (from w in _myDatabaseContext.Wallets
                       where w.UserId == userId
                       select w).SingleOrDefault();
      if (wallet == null)
      {
            return 0.00M;
      }
      decimal result = wallet.Balance;
```

```
            return result;
      }

      public int RetrieveWalletId(string userId)
      {
            Wallet wallet = (from w in _myDatabaseContext.Wallets
                             where w.UserId == userId
                             select w).SingleOrDefault();
            if (wallet == null)
            {
                  return 0;
            }
            int result = wallet.WalletId;
            return result;
      }

      public void SubtractFromWallet(string userId, decimal
amountToSubtract)
      {
            Wallet wallet = (from w in _myDatabaseContext.Wallets
                             where w.UserId == userId
                             select w).SingleOrDefault();

            decimal result = wallet.Balance - amountToSubtract;
            wallet.Balance = result;
            _myDatabaseContext.SaveChanges();
      }
      }
}
```

Areas/Coinflip

Controllers

PlayController.cs

```
using CentralGamesPlatform.IRepositories;
using CentralGamesPlatform.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;

namespace CentralGamesPlatform.Areas.CoinFlip.Controllers
{
      [Authorize]
      [Area("CoinFlip")]
      public class PlayController : Controller
      {
      private readonly ICasinoPassRepository _casinoPassRepository;
      private readonly IResultRepository _resultRepostiory;
      private readonly IWalletRepository _walletRepository;
      public PlayController(ICasinoPassRepository
casinoPassRepository, IResultRepository resultRepository,
                              IWalletRepository walletRepository)
      {
            _casinoPassRepository = casinoPassRepository;
            _resultRepostiory = resultRepository;
            _walletRepository = walletRepository;
      }
      public IActionResult Index(Guid casinoPassId)
      {
            var pass =
_casinoPassRepository.GetCasinoPass(casinoPassId);
            //If pass is active and game id is the same as this
game
            if(pass == null)
            {
                  return RedirectToAction("Index", "Home", new {
area = "" });
            }
            if (pass.Active == true && pass.GameId == 10)
            {
                  return View(pass);
            }
            else
            {
                  return RedirectToAction("Index", "Library", new {
area = "" });
            }
```

```
      }

      public IActionResult Decide(Guid casinoPassId)
      {
            var casinoPass =
_casinoPassRepository.GetCasinoPass(casinoPassId);
            if(casinoPass == null)
            {
                  return RedirectToAction("Index", "Home", new {
area = "" });
            }
            if(casinoPass.Expired == true && casinoPass.Active ==
false)
            {
                  return RedirectToAction("Index", "Home", new {
area = "" });
            }
            Random rand = new Random();
            if (rand.Next(0, 2) == 0)
            {

                  Result result = new Result();
                  result.Win = false;
                  result.CasinoPassId = casinoPass.CasinoPassId;
                  result.AmountWon = 0.00M;
                  if (casinoPass.Expired == false &&
casinoPass.Active == true && casinoPass.GameId == 10)
                  {
                  _resultRepostiory.CreateResult(result);

_casinoPassRepository.ExpireCasinoPass(casinoPass.CasinoPassId);

                  }
                  else
                  {
                  RedirectToAction("Index", "Home", new { area = ""
});
                  }
                  return View("Loss");
            }
            else
```

```
            {
                  Result result = new Result();
                  result.Win = true;
                  result.CasinoPassId = casinoPass.CasinoPassId;
                  result.AmountWon = 10.00M;
                  string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;


                  if (casinoPass.Expired == false &&
casinoPass.Active == true && casinoPass.GameId == 10)
                  {
                  _resultRepostiory.CreateResult(result);
                  _walletRepository.AddToWallet(userId,
result.AmountWon);

_casinoPassRepository.ExpireCasinoPass(casinoPass.CasinoPassId);

                  }
                  else
                  {
                  RedirectToAction("Index", "Home", new { area = ""
});
                  }
                  return View("Win");
            }
      }
      }
}
```

Views

Index.cshtml

```
@model CasinoPass
<p>
      <a class="btn btn-success" asp-controller="Play"
asp-action="Decide"
asp-route-casinoPassId="@Model.CasinoPassId">Get Result</a>
</p>
```

Loss.cshtml

```
<h2>Sorry but you have lost the game</h2>
<a class ="btn btn-danger" asp-area="" asp-controller="Library"
asp-action="Index">Return to library</a>
```

Win.cshtml

```
<h3>Congratulations you have won €10.00</h3>
<a class="btn btn-danger" asp-area="" asp-controller="Library"
asp-action="Index">Return to library</a>
```

CentralGamesPlatform/ (ProjectRoot)

appsetting.json

```
{
  "Logging": {
     "LogLevel": {
     "Default": "Information"
     }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
     //[LOCAL]"MyDbConnection":
"Server=(localdb)\\mssqllocaldb;Database=MyDatabaseContext-2085246
d-aff5-4310-a0b8-93780358cdae;Trusted_Connection=True;MultipleActi
veResultSets=true"
     "DefaultConnection":
"Server=tcp:centralgamesplatformdbserver.database.windows.net,1433
;Initial Catalog=centralgamesplatformdb;Persist Security
Info=False;User
ID=superuser;Password=darak123!;MultipleActiveResultSets=False;Enc
rypt=True;TrustServerCertificate=False;Connection Timeout=0;"
  },
  "Stripe": {
     "TestSecretKey":
"sk_test_51IB0Z4BTwx1LYfRRop1pYWwRVKBAs0K7KZBRbKTubudFUXJPN5BlooRa
hipg8qIkpIQ49d6c4YZE9ErcziO23QtR00rzwq6cbk",
     "TestPublishableKey":
"pk_test_51IB0Z4BTwx1LYfRREoDEBBJOh4HQM4tCzZgvmqRqnutsLFWU2rzPvaUT
0pa7vTVp5WqKMMPxyx1nFzeeLjW4o7Eo00umgSxWfI"
```

```
  },
  "PayPal": {
     "clientId":
"AVESW8pAcvJQVcH6G11Z193gImhx8H8DQ9NIwV9sgP1XFZWaDztQym-JaolO1XX-g
GsL69VLChgzePxb",
     "secret":
"ELslSeKzMAEFceAdQroyeo3gLrCU0f6G2b8zhQl54TpPPRfAXn1u5kfCT_MjyQHZH
lRHc4D1cQFGNodt",

  }
}
```

PaginatedList.cs

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

namespace CentralGamesPlatform
{
     public class PaginatedList<T> : List<T>
     {
     public int PageIndex { get; private set; }
     public int TotalPages { get; private set; }

     public PaginatedList(List<T> items, int count, int pageIndex,
int pageSize)
     {
          PageIndex = pageIndex;
          TotalPages = (int)Math.Ceiling(count /
(double)pageSize);

          this.AddRange(items);
     }

     public bool HasPreviousPage
     {
          get
          {
```

```
                return (PageIndex > 1);
            }
        }

        public bool HasNextPage
        {
            get
            {
                return (PageIndex < TotalPages);
            }
        }

        public static async Task<PaginatedList<T>>
CreateAsync(IQueryable<T> source, int pageIndex, int pageSize)
        {
            var count = await source.CountAsync();
            var items = await source.Skip((pageIndex - 1) *
pageSize).Take(pageSize).ToListAsync();
            return new PaginatedList<T>(items, count, pageIndex,
pageSize);
        }

        internal static Task<object> CreateAsync(object p, int v, int
pageSize)
        {
            throw new NotImplementedException();
        }
        }
}
```

PayPalClient.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.Serialization.Json;
using System.Text;
using System.Threading.Tasks;
using PayoutsSdk.Core;
using PayPalHttp;
```

```
namespace CentralGamesPlatform
{
      public class PayPalClient
      {
      public static PayPalEnvironment environment()
      {
            return new SandboxEnvironment(

System.Environment.GetEnvironmentVariable("PAYPAL_CLIENT_ID") !=
null ?

System.Environment.GetEnvironmentVariable("PAYPAL_CLIENT_ID"):
"AVESW8pAcvJQVcH6G11Z193gImhx8H8DQ9NIwV9sgP1XFZWaDztQym-JaolO1XX-g
GsL69VLChgzePxb",

System.Environment.GetEnvironmentVariable("PAYPAL_CLIENT_SECRET")
!= null ?

System.Environment.GetEnvironmentVariable("PAYPAL_CLIENT_SECRET"):
"ELslSeKzMAEFceAdQroyeo3gLrCU0f6G2b8zhQl54TpPPRfAXn1u5kfCT_MjyQHZH
lRHc4D1cQFGNodt");
      }
      public static HttpClient client()
      {
            return new PayPalHttpClient(environment());
      }

      public static HttpClient client(string refreshToken)
      {
            return new PayPalHttpClient(environment(),
refreshToken);
      }

      /**
            This method can be used to Serialize Object to JSON
string.
      */
      public static String ObjectToJSONString(Object
serializableObject)
      {
```

```
            MemoryStream memoryStream = new MemoryStream();
            var writer = JsonReaderWriterFactory.CreateJsonWriter(
                        memoryStream, Encoding.UTF8, true, true, "
");
            DataContractJsonSerializer ser = new
DataContractJsonSerializer(serializableObject.GetType(), new
DataContractJsonSerializerSettings { UseSimpleDictionaryFormat =
true });
            ser.WriteObject(writer, serializableObject);
            memoryStream.Position = 0;
            StreamReader sr = new StreamReader(memoryStream);
            return sr.ReadToEnd();
      }
      }
}
```

Program.cs

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

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

      public static IHostBuilder CreateHostBuilder(string[] args)
=>
            Host.CreateDefaultBuilder(args)
```

```
                .ConfigureLogging(logging =>
                {
                logging.AddAzureWebAppDiagnostics();
                })
                .ConfigureWebHostDefaults(webBuilder =>
                {
                webBuilder.UseStartup<Startup>();
                });
      }
}
```

Startup.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.EntityFrameworkCore;
using CentralGamesPlatform.Models;
using Stripe;
using Microsoft.AspNetCore.Identity;
using CentralGamesPlatform.Repositories;
using CentralGamesPlatform.IRepositories;

namespace CentralGamesPlatform
{
      public class Startup
      {
      public IConfiguration Configuration { get; }
      public Startup(IConfiguration configuration)
      {
            Configuration = configuration;
```

```
        }

        // This method gets called by the runtime. Use this method to
add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {

services.AddDefaultIdentity<ApplicationUser>().AddRoles<IdentityRo
le>().AddEntityFrameworkStores <MyDatabaseContext>();
            services.AddControllersWithViews();
            services.AddScoped<ICategoryRepository,
CategoryRepository>();
            services.AddScoped<IGameRepository, GameRepository>();
            services.AddScoped<IOrderRepository,
OrderRepository>();
            services.AddScoped<IPaymentRepository,
PaymentRepository>();
            services.AddScoped<IOrderDetailRepository,
OrderDetailRepository>();
            services.AddScoped<ILicenceRepository,
LicenceRepository>();
            services.AddScoped<ICasinoPassRepository,
CasinoPassRepository>();
            services.AddScoped<IResultRepository,
ResultRepository>();
            services.AddScoped<IWalletRepository,
WalletRepository>();
            services.AddScoped<IPayoutRepository,
PayoutRepository>();
            services.AddScoped<IVerificationRepository,
VerificationRepository>();
            services.AddScoped<IDownloadRepository,
DownloadRepository>();
            services.AddScoped<ShoppingCart>(sc =>
ShoppingCart.GetCart(sc));
            services.AddHttpContextAccessor();
            services.AddSession();
            services.AddRazorPages();
            services.AddDbContext<MyDatabaseContext>(options =>
                  // options.UseSqlite("Data
Source=localdatabase.db"));
```

```
options.UseSqlServer(Configuration.GetConnectionString("DefaultCon
nection")));
      }

      // This method gets called by the runtime. Use this method to
configure the HTTP request pipeline.
      public void Configure(IApplicationBuilder app,
IWebHostEnvironment env)
      {
            StripeConfiguration.ApiKey =
(Configuration.GetSection("Stripe")["TestSecretKey"]);
            if (env.IsDevelopment())
            {
                  app.UseDeveloperExceptionPage();
            }
            else
            {
                  app.UseExceptionHandler("/Home/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.UseSession();

//StripeConfiguration.SetApiKey(Configuration.GetConnectionString(
"StripeTestSecretKey"));
            app.UseRouting();
            app.UseAuthentication();
            app.UseAuthorization();
            //app.UseStatusCodePages();

//app.UseStatusCodePagesWithReExecute("/Error/HandleError/{0}");
            app.Use(async (context, next) =>
            {
                  await next();
                  if (context.Response.StatusCode == 404)
                  {
```

```
                context.Request.Path = "/Error/HandleError";
                await next();
                }
            });


            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                name:"areas",

pattern:"{area:exists}/{controller=Play}/{action=Index}/{id?}"
                );
                endpoints.MapControllerRoute(
                name: "default",
                pattern: "{controller=Home}/{action=Index}/{id?}"
                );
                endpoints.MapRazorPages();
            });
        }
        }
}
```

10.Bibliography

[1] Microsoft. 2020. *Introduction to Identity on ASP.NET Core*. [online] Available at: https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-5.0&tabs=visual-studio [Accessed 23 April 2021].

[2] Microsoft. n.d. *Rfc2898DeriveBytes(String, Int32, Int32)*. [online] Available at: https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.rfc2898derivebytes.-ctor?redirectedfrom=MSDN&view=net-5.0#System_Security_Cryptography_Rfc2898DeriveBytes__ctor_System_String_System_Int32_System_Int32_ [Accessed 23 April 2021]

[3] Stripe. n.d. *Stripe API reference – Create a Session – .NET*. [online] Available at: https://stripe.com/docs/api/checkout/sessions/create?lang=dotnet [Accessed 24 April 2021].

[4] Microsoft. n.d. *Overview of Entity Framework Core - EF Core*. [online] Available at: https://docs.microsoft.com/en-us/ef/core/ [Accessed 26 April 2021].

[5] Petrakovich, J., n.d. *The WHY Series: Why should you use the repository pattern?*. [online] Making Loops. Available at: https://makingloops.com/why-should-you-use-the-repository-pattern/ [Accessed 26 April 2021].

[6] Microsoft. 2020. *Overview of ASP.NET Core MVC*. [online] Available at: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0 [Accessed 26 April 2021].

DECLARATION

*I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.

*I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.

*I have provided a complete bibliography of all works and sources used in the preparation of this submission.

*I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offence.

Student Name: (Printed)KYLE HENNESSY_____

Student Number(s): C00227463_____

Signature(s): Kyle Hennessy_____

Date: 30/04/2021_____

Please note:
The Institute regulations on plagiarism are set out in Section 10 of Examination and Assessment Regulations published each year in the Student Handbook.