Institiúid Teicneolaíochta Cheatharlach

INSTITUTE of TECHNOLOGY CARLOW

At the Heart of South Leinster

LifeReceitas

4th Year Project Technical Manual

Student Name: Fillipe Soares

Student ID: C00203202

Supervisor: Joseph Kehoe

Table of contents

Introduction

The goal of this manual is to demonstrate my code for the gastronomy application LifeRecipes. The code for this application will be provided in a structured manner, displaying all of the components required to build this application. A screenshot of each file inside each component will be included in the Application section. Following that, screenshots of the database structure will be presented.

All the code can be found on my Git Hub:

https://github.com/fillipeoi/LifeReceitas

Application Code

Components

Cookbook

Cookbook JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */


import React, {useEffect, useState}
from "react";
//MUI icons
import {Done, ExpandMore, MoreVert}
from "@mui/icons-material";
import ExpandMoreIcon
from "@mui/icons-material/ExpandMore";
//MUI materials
import {Box, CardActions, Collapse, Divider, IconButton, Menu,
MenuItem, Tooltip, Typography} from "@mui/material";
//Firebase
import {deleteDoc, doc, getDoc, updateDoc}
from "firebase/firestore";
import {database}
from "../../FirebaseConfig";
//React boostrap
import {Col, Row}
from "react-bootstrap";
//Components
import Recipe
from "../Recipe/Recipe";
//Styles
import {BootstrapInput} from "./CookbookStyle";

const Cookbook = ({id, name, recipes, setCurrentCookbook,
currentCookbook}) => {

    const user = JSON.parse(sessionStorage.getItem('user'));

    const [expandedCookbook, setExpandedCookbook] = useState(false);
    const [menu, setMenu] = useState(null);
    const [update, setUpdate] = useState(false);
    const [nameState, setNameState] = useState(name);
    const [recipesState, setRecipesState] = useState(recipes);

    //Menu settings
    const settings = ["Modify", "Remove"];

    //Display a cookbook
    const handleExpandCookbook = () => {
        if(update) return;
        setCurrentCookbook(id);
        currentCookbook = id;
        setExpandedCookbook(!expandedCookbook);
    }

    //Open options menu
    const handleOpenMenu = (event) => {
        setMenu(event.currentTarget);
        setCurrentCookbook(id);
    };

    //Close options menu
    const handleCloseMenu = () => {
        setMenu(null);
    };

    //Remove a cookbook in the database
    const handleRemoveCookbook = () => {
        //Remove the cookbook from the Firebase database
        deleteDoc(doc(database, "users/" + user?.uid + "/cookbooks/" +
id))
            .then(() => {
                window.location.reload();
            })
            .catch(() => {
                console.log("Error removing recipe");
            });
    }

    //Update the cookbook name in the database
    const handleUpdateName = () => {
        // Update the cookbook name in Firebase database
        if(nameState === "") return;
        if(!nameState) return;
        const cookbook = {
            name: nameState,
            recipes: recipes,
            owner: user?.uid
        };
        updateDoc(doc(database, "users/" + user?.uid + "/cookbooks/" +
id), cookbook)
        .catch(() => {
            console.log("Error updating recipe");
        });

        setUpdate(false);
    }

    //Change to update mode
    const controlUpdateMenu = () => {
        setUpdate(true);
        handleCloseMenu();
    }

    //Change the state of the cookbook name
    const updateName = (event) => {
        setNameState(event.target.value);
    }


    //Refresh recipes of the cookbook from the firebase database
    const refreshRecipes = async () => {
        recipes.map(async (recipe) => {
            if (recipe.id) {
                const docRef = doc(database, "recipes", recipe.id);
                const docSnap = await getDoc(docRef);

                if (docSnap.exists()) {
                    //if the recipe exists in the recipes array, update
the recipe
                    const recipe = docSnap.data();
                    recipe.id = docSnap.id;
                    recipe.ingredients =
recipe.ingredients.map(ingredient => ingredient.text);
                    setRecipesState(recipesState.map((r) => r.id ===
recipe.id ? recipe : r));
                } else {
                    // doc.data() will be undefined in this case
                    console.log("No such document!");
                }
            }
        });
    }

    useEffect(async () => {
        await refreshRecipes();
        if(currentCookbook === id) {
            setExpandedCookbook(true);
        } else {
            setExpandedCookbook(false);
        }
    }, [currentCookbook, id]);

    return (
        <Row>
            <Divider sx={{marginTop: "10px"}}/>
                <CardActions disableSpacing sx={{width:"100%",
cursor:"pointer"}}>
                    <Row className={"justify-content-between
align-items-center w-100"} onClick={handleExpandCookbook}
sx={{cursor:"pointer"}}>
                        <Col>
                            {update
                                // If the update mode is on, you can
modify the name of the cookbook.
                                ?
                                <Row className={"align-items-center"}>
                                    <Col lg={10} md={10} xs={10}>
                                        <BootstrapInput multiline
className="input-100 recipe-input"

onChange={updateName} required={true} value={nameState}/>
                                    </Col>
                                    <Col lg={2} md={2} xs={2}>
                                        <IconButton
onClick={handleUpdateName} sx={{p: '10px', color:"#FA4616"}}
aria-label="search">
                                            <Done/>
                                        </IconButton>
                                    </Col>
                                </Row>
                                // Else display the name of the
cookbook
                                :
                                <Col lg={10} md={10} xs={10}>
                                    <Typography sx={{fontSize: "24px",
cursor: "pointer", width:"100%"}}>{nameState}</Typography>
                                </Col>
                            }
                        </Col>
                        {/*If the cookbook is not updating, display the
expand button*/}
                            {!update &&
                                <Col lg={2} md={2} xs={2} sx={{cursor:
"pointer"}} >
                                    <ExpandMore
expand={expandedCookbook.toString()} onClick={handleExpandCookbook}
aria-expanded={expandedCookbook} aria-label="show more">
                                        <ExpandMoreIcon/>
                                    </ExpandMore>
                                </Col>
                            }
                    </Row>
                    <Box sx={{ flexGrow: 0 }}>
                        {/* Options button*/}
                        <Tooltip title="Open settings">
                            <IconButton aria-label="settings"
onClick={handleOpenMenu}>
                                <MoreVert />
                            </IconButton>
                        </Tooltip>
                        {/*Options menu*/}
                        <Menu
                            sx={{ mt: '45px' }}
                            id="menu-appbar"
                            anchorEl={menu}
                            anchorOrigin={{
                                vertical: 'top',
                                horizontal: 'right',
                            }}
                            transformOrigin={{
                                vertical: 'top',
                                horizontal: 'right',
                            }}
                            open={Boolean(menu)}
                            onClose={handleCloseMenu}
                            onClick={handleCloseMenu}
                        >
                            {/*List of options*/}
                            {settings.map((setting) => (
                                <MenuItem key={setting}
onClick={setting === "Remove" ? handleRemoveCookbook : () =>
controlUpdateMenu()}>
                                    <Typography
textAlign="center">{setting}</Typography>
                                </MenuItem>
                            ))}
                        </Menu>
                    </Box>
                </CardActions>
                {/*//If the cookbook is expanded, display the
content*/}
                <Collapse in={expandedCookbook} timeout="auto"
unmountOnExit>
                    <Row className={"justify-content-center"}>
                        <Col>
                          <div className="recipes">
                              {/*List of recipes in the cookbook*/}
                            {recipesState.map((recipe, index) => (
                              <Recipe
                                  key={index}
                                  title={recipe.title}
                                  calories={recipe.calories}
                                  image={recipe.image}
                                  method={recipe.method}
                                  ingredients={recipe.ingredients}
                                  rating={recipe.rating}
                                  reviews={recipe.reviews}
                                  uid={recipe.uid}
                                  id={recipe.id}
                                  photoURL={recipe.photoURL}
                                  displayName={recipe.displayName}
                                  disableAddToCookbook={true}
                                  cookBookMode={true}
                                  cookbookId={id}
                              />
                            ))}
                          </div>
                        </Col>
                    </Row>
                </Collapse>
        </Row>
      );
};

export default Cookbook;
```

Cookbook Style File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */



import {InputBase}      from "@mui/material";
import {alpha, styled} from "@mui/material/styles";

export const BootstrapInput = styled(InputBase)(({ theme }) => ({
    'label + &': {
        marginTop: theme.spacing(3),
    },
    '& .MuiInputBase-input': {
        borderRadius: 4,
        multiline: "true",
        position: 'relative',
        backgroundColor: theme.palette.mode === 'light' ? '#fcfcfb' :
'#2b2b2b',
        border: '1px solid #ced4da',
        fontSize: 16,
        width: '100%',
        padding: '10px 10px 10px 10px',
        margin: '5px 0px 5px 0px',
        transition: theme.transitions.create([
            'border-color',
            'background-color',
            'box-shadow',
        ]),
        // Use the system font instead of the default Roboto font.
        fontFamily: [
            '-apple-system',
            'BlinkMacSystemFont',
            '"Segoe UI"',
            'Roboto',
            '"Helvetica Neue"',
            'Arial',
            'sans-serif',
            '"Apple Color Emoji"',
            '"Segoe UI Emoji"',
            '"Segoe UI Symbol"',
        ].join(','),
        '&:focus': {
            //transparent for #FA4616
            boxShadow: `${alpha('#FA4616', 0.25)} 0 0 0 0.2rem`,
            borderColor: '#FA4616',
        },
    },
}));
```

Discover

Discover JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */

import {useCallback, useEffect, useState} from "react";
//MUI Icons
import SearchIcon                           from
'@mui/icons-material/Search';
//MUI material
import {IconButton, InputBase, Paper}       from "@mui/material";
//Firebase
import {database}                           from "../../FirebaseConfig";
import {collection, getDocs}                from "firebase/firestore";
//Lodash
import {debounce}                           from "lodash";
//React bootstrap
import {Col, Container, Row}                from "react-bootstrap";
import {useNavigate}                        from "react-router-dom";
//Components
import NavigationBar                        from
"../NavigationBar/NavigationBar";
```

```
import Recipe                                         from "../Recipe/Recipe";
//CSS
import "./Discover.css";


//Discover Component
 const Discover = () => {
    //Recipe API setup
    const APP_ID = "691a37a0";
    const API_KEY = "255600a0f7f4b613fe5e470a702545cb";

    const [recipes, setRecipes ] = useState ([]);
    const [search, setSearch] = useState("");
    const [stringNoMatch, setStringNoMatch] = useState(null);
    const [error, setError] = useState(false);
    const [cookbooks, setCookbooks] = useState([]);

    const user = JSON.parse(sessionStorage.getItem("user"));
    const navigate = useNavigate();

     //Get user cookbooks from firebase database
     const getUserCookbooks = async () => {
         const querySnapshot = await getDocs(collection(database,
"users/" + user?.uid + "/cookbooks"));
         querySnapshot.forEach((doc) => {
             let newCookbook = doc.data();
             newCookbook.id = doc.id;
             if(!cookbooks.some(cookbook => cookbook.id ===
newCookbook.id)){
                 setCookbooks(cookbooks => [...cookbooks,
newCookbook]);
             } else {
                 const index = cookbooks.findIndex(cookbook =>
cookbook.id === newCookbook.id);
                 cookbooks[index] = newCookbook;
                 setCookbooks(cookbooks);
             }
         });
     }

     //Depending on the search input, recipes are retrieved from the
API with 1 second delay
     //The debounce function is used to avoid the API request to be
made too often
```

```
    const delayedQuery = useCallback(
        debounce((q) => sendQuery(q), 1000),
        []
    );

    //Send query to the API
    const sendQuery = useCallback(
        async (search) => {
            if (error) return;
            var url;
            var oneRandomLetter = (Math.random() +
1).toString(36).substring(7)[0];
            //If the search input is empty, a random recipe is
retrieved
            if (search === "" || search === undefined) {
                url =
`https://api.edamam.com/search?q=${oneRandomLetter}&app_id=${APP_ID}&ap
p_key=${API_KEY}`;
            } else {
                //If the search input is not empty, recipes which
contain the search input are retrieved
                url =
`https://api.edamam.com/search?q=${search}&app_id=${APP_ID}&app_key=${A
PI_KEY}`;
            }
            //If there is no match with the search input, the app
doesn't fetch the API
            if (search?.includes(stringNoMatch)) {
                return;
            }

            fetch(url)
                .then(response => response.json())
                .then(data => {
                    if (data.hits.length === 0) {
                        setStringNoMatch(search);
                        setRecipes([]);
                    } else {
                        let recipesData = data?.hits.map(recipe => {
                            return {
                                uri: recipe.recipe.uri,
                                title: recipe.recipe.label,
                                image: recipe.recipe.image,
                                calories: recipe.recipe.calories,
                                ingredients:
recipe.recipe.ingredients.map((ingredient) => { return
ingredient.text}),
                            }
                        });
                        setRecipes(recipesData);
                    }
                })
                .catch(err => {
                    console.log(err);
                    setError(true);
                });
        },
        []
    );

    useEffect(async() => {
        //If the user is not logged in, redirect to the login page
        //To handle the case where the user is not logged in and tries
to access the page
        if(!user){
            navigate("/");
        }
        await getUserCookbooks();
        delayedQuery(search);

    }, [search, delayedQuery]);

    //Update the search input
   const updateSearch = e => {
       setSearch(e.target.value);
   }

    return(
       <>
           <Container className="mt-5">
               <Row>
                   <NavigationBar/>
               </Row>
           </Container>

           <Container className="mt-4">
               <Row className={"justify-content-center"}>
                   <Col className="mt-4" lg={6} mf={6} sm={6}>
```

```
                              {/*recipe search form*/}
                             <Paper className="mb-4"  sx={{ p: '0px 4px',
display: 'flex', alignItems: 'center', maxWidth: '100%' }}   >

                                 <InputBase sx={{ ml: 1, flex: 1 }}
placeholder="Search recipe or ingredient" value={search ?? ""}
onChange={updateSearch} />

                                 <IconButton  type="submit" sx={{ p: '10px'
}} aria-label="search">
                                     <SearchIcon className="searchIcon"/>
                                 </IconButton>
                             </Paper>
                         </Col>
                     </Row>
                 </Container>
                 <Row className={"justify-content-center"}>
                     <Col lg={5} md={7} sm={7}>
                         {/*List of recipes*/}
                         {recipes.map((recipe, index) =>(
                             <Container className="mt-2" key={index}>
                                 <Row>
                                     <Col>
                                         <Recipe
                                             key={recipe.uri}
                                             title={recipe.title}
                                             calories={recipe.calories}
                                             image={recipe.image}

ingredients={recipe.ingredients}
                                             userCookbooks={cookbooks}
                                         />
                                     </Col>
                                 </Row>
                              </Container>
                         ))}
                     </Col>
                 </Row>
             </>
     );
};

export default Discover;
```

Discover CSS File

```css
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */


.navBg{
    background: #FA4616;
    width: 100%;
}
.menuOptions{
    color: white;
}
 .logoutBtn{

    color: white;
 }
 .searchIcon{
   color: #FA4616;
 }
 .homeSearchBtn:hover{
    padding: auto;
    font-size: 15px;
    background-color: #FA4616;
    border: 2px solid #FA4616;
    border-radius: 4px;
    margin-bottom: 20px;
    color: #ffffff;
}

 .recipeImg{
     height: 300px;
     width: 300px;
     border-radius: 6px;
     object-fit: cover;
}

.filterSelect{
    padding: auto;
    font-size: 15px;
    background-color: #ffffff;
    border: 1px solid #FA4616;
    border-radius: 4px;
    margin-bottom: 8px;
    color: #FA4616;
}
.filterSelect:toggle{
   padding: auto;
   font-size: 15px;
   background-color: #FA4616;
   border: 2px solid #FA4616;
   border-radius: 4px;
   margin-bottom: 3px;
   color: #ffffff;
}
```

Forgot Password

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */

 import React, { useState } from "react";
 import { Container, Col, Row, Button, Form, Image, Alert } from
"react-bootstrap";
 import { Link, useNavigate  } from "react-router-dom";
 import '../Login/Login.css';
 import "bootstrap/dist/css/bootstrap.css";
 import "../Custom.scss";
 import { useUserAuth } from "../../context/UserAuthContext";



  const ForgotPass = () => {
      const [email, setEmail] = useState("");
      const [error, setError] = useState("");
      const [message, setMessage] = useState("");

      const { passwordReset } = useUserAuth();
      const navigate = useNavigate();



       //error handling
      const handleSubmit = async (e) => {
          e.preventDefault();
          setError("");
          setMessage("");

          try{
              await passwordReset(email);
              setMessage("Check your email for further instructions");
              navigate("/");

          }catch(err){
            setError("Failed to reset password");
          }
        };


      return (
          <>
             <Container className="mt-5">
                 <Row>
                     <Col lg={12} md={12} sm={12}>
                         <img className="mx-auto d-block img-fluid"
src="loginLogo.png"></img>
                     </Col>
                 </Row>
                 <Row>
                     <Col lg={3} mf={3} sm={0}></Col>
                     <Col lg={6} mf={6} sm={12}>

                         {error && <Alert variant="danger">{ error
}</Alert>}

                         <Form onSubmit={handleSubmit}>
                             <Form.Group className="mb-3"
controlId="formBasicEmail">
                                 <Form.Label>Email address</Form.Label>
                                 <Form.Control type="email"
placeholder="Enter email" onChange={(e) => setEmail(e.target.value)} />
                                 <Form.Text className="text-muted">
We'll never share your email with anyone else.</Form.Text>
                             </Form.Group>

                             <div className="d-grid">
```

```
                                   <Button className="text-light"
variant="primary" type="submit">Reset Password</Button>
                             </div>
                         </Form>

                         <div className="mt-2">
                            <small >Already have an account? <Link
to="/" className="forgotPass">Login</Link></small>
                         </div>

                         <div className="mt-1">
                                <small >Need an account? <Link
to="/signup" className="forgotPass">Sign up</Link></small>
                        </div>
                    </Col>


                </Row>

            </Container>
         </>
      )
   }
 export default ForgotPass;
```

Login

Login JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */

import React, {useEffect, useState}                  from "react";
//React Bootstrap
import { Container, Col, Row, Button, Form, Alert } from
"react-bootstrap";
import "bootstrap/dist/css/bootstrap.css";
//React Router Dom
import { Link, useNavigate }                         from
"react-router-dom";
//User auth
import { useUserAuth }                               from
"../../context/UserAuthContext";
```

```
//Google
import GoogleButton                                  from
"react-google-button";

import "./Login.css";
import "../Custom.scss";

 const Login = () => {
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const [error, setError] = useState("");
    const [rememberMe, setRememberMe] =
useState(sessionStorage.getItem("rememberMe"));

    const { logIn, googleSignIn, user } = useUserAuth();
    const navigate = useNavigate();

    //Handle remember me checkbox
     const handleRememberMe = (e) => {
         setRememberMe(e.target.checked);
         sessionStorage.setItem("rememberMe", e.target.checked);
    }

     //error handling
     const handleSubmit = async (e) => {
        e.preventDefault();
        setError("");

        try{
            const user = await logIn(email, password)
            if(user && user.emailVerified){
                navigate("/recipes");
            }
        }catch(error){
            //Handle error code message
            switch(error.code){
                case "auth/user-not-found":
                    setError("User not found");
                    break;
                case "auth/wrong-password":
                    setError("Wrong password");
                    break;
                case "auth/invalid-email":
                    setError("Invalid email");
                    break;
                case "auth/internal-erroror":
                    setError("Wrong password");
                    break;
                case "auth/too-many-requests":
                    setError("Access to this account has been
temporarily disabled due to many failed login attempts. You can
immediately restore it by resetting your password or you can try again
later.");
                    break;
                default:
                    setError(error.message);
                    break;
            }
        }
      };

     const handleGoogleSignIn = async (e) => {
        e.preventDefault();
        try{
            await googleSignIn();
            navigate("/recipes");
        } catch(error){
            //Handle erroror message with Google sign in
            switch(error.code) {
                case "auth/popup-closed-by-user":
                    setError("Google sign in was cancelled");
                    break;
                default:
                    setError(error.message);
                    break;
            }
        }
     };

     const handleUserChange = () => {
         //If user is logged in and verified, redirect to recipes page
        if(user && user.emailVerified){
            navigate("/recipes");
        }
     };


     useEffect(() => {
        handleUserChange();
     }, [user]);

     return (
         <>
                <Container className="mt-5">
                    <Row>
                        <Col lg={12} md={12} sm={12}>
                            <img className="mx-auto d-block img-fluid"
src="loginLogo.png"></img>
                        </Col>
                    </Row>
                    <Row>
                        <Col lg={3} mf={3} sm={0}></Col>
                        <Col lg={6} mf={6} sm={12}>

                            {error && <Alert variant="danger">{ error
}</Alert>}

                            <Form onSubmit={handleSubmit}>
                                <Form.Group className="mb-3"
controlId="formBasicEmail">
                                    <Form.Label>Email
address</Form.Label>
                                    <Form.Control type="email"
placeholder="Enter email" onChange={(e) => setEmail(e.target.value)} />
                                    <Form.Text className="text-muted">
We'll never share your email with anyone else.</Form.Text>
                                </Form.Group>

                                <Form.Group className="mb-3"
controlId="formBasicPassword">
                                    <Form.Label>Password</Form.Label>
                                    <Form.Control type="password"
placeholder="Password" onChange= {(e) => setPassword(e.target.value)}
/>
                                </Form.Group>

                                <Form.Group className="mb-4"
controlId="formBasicCheckbox">
                                    <Form.Check type="checkbox"
label="Remember me" checked={Boolean(rememberMe)} onChange={(e) =>
handleRememberMe(e)}/>
                                </Form.Group>
                                <div className="d-grid , mb-2" >
                                      <Button className="text-light"
variant="primary" type="submit">Login</Button>
                                </div>

                                <div className="mb-3">
                                    <Link to="/forgotpass"
className="forgotPass"><small >Forgot Password?</small></Link>
                                </div>

                                <div className="mb-2">
                                    <GoogleButton className="g-btn
w-100" type="dark" onClick={handleGoogleSignIn}/>
                                </div>

                                <div className="mt-2">
                                    <small >Need an account? <Link
to="/signup" className="forgotPass">Sign up</Link></small>
                                </div>
                            </Form>
                        </Col>
                    </Row>
                </Container>
         </>
     )
 }

 export default Login;
```

Login CSS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */
.btn-primary {
    background-color: #FA4616;
}

.forgotPass{
    color: black;
    text-decoration: none;
}
```

Messages

Messages JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */


import {ArrowBackIosNew}             from "@mui/icons-material";
import {ListItemAvatar}                      from "@mui/material";
import React, {useEffect, useRef, useState} from 'react';
//MUI material
import SendIcon                      from '@mui/icons-material/Send';
import Avatar                        from '@mui/material/Avatar';
import Container                     from "@mui/material/Container";
import ListItemButton                from
"@mui/material/ListItemButton";
import Divider                       from '@mui/material/Divider';
import Fab                           from '@mui/material/Fab';
import Grid                          from '@mui/material/Grid';
import List                          from '@mui/material/List';
import ListItem       from '@mui/material/ListItem';
import ListItemIcon   from '@mui/material/ListItemIcon';
import ListItemText   from '@mui/material/ListItemText';
import Paper          from '@mui/material/Paper';
import TextField      from '@mui/material/TextField';
import Typography     from '@mui/material/Typography';
import Badge from '@mui/material/Badge';
//Firebase
import {
    addDoc,
    increment,
    collection,
    doc,
    getDoc,
    getDocs,
    orderBy,
    Timestamp,
    query,
    onSnapshot,
    where,
    updateDoc,
    limit
} from "firebase/firestore";
import {database}
from "../../FirebaseConfig";
//Bootstrap
import {Col, Row}                                             from
"react-bootstrap";
//React router
import {useNavigate}                                          from
"react-router-dom";
//Components
import NavigationBar                                          from
"../NavigationBar/NavigationBar";

import "./Messages.css";

const Messages = () => {

    const user = JSON.parse(sessionStorage.getItem("user"));
    const navigate = useNavigate();

    const [users, setUsers] = useState([]);
    const [search, setSearch] = useState(null);
    const [formValue, setFormValue] = useState('');
    const [interlocutorUID, setInterlocutorUID] = useState('');
    const [messages, setMessages] = useState([]);
    const [currentRoomId, setCurrentRoomId] = useState('');
    const [displayUserList, setDisplayUserList] = useState(true);
    const [interlocutorName, setInterlocutorName] = useState('');

    const roomsRef = collection(database, "rooms");

    //Basic use effect
    useEffect(async () => {
        //If the user is not logged in, redirect to the login page
        //To handle the case where the user is not logged in and tries
to access the page
        if(!user){
            navigate("/");
        }
        //If a room is open by "send message" button
        if (sessionStorage.getItem("interlocutorUID")) {

setInterlocutorUID(sessionStorage.getItem("interlocutorUID"));
```

```
setInterlocutorName(sessionStorage.getItem("interlocutorName"));
            setDisplayUserList(false);
            await
createRoom(sessionStorage.getItem("interlocutorUID"));
        }
        await getUsers();
    }, []);

    //Update unread messages
    useEffect(async () => {
        if (user) {
            const userRef = doc(database, "users/" + user.uid);
            await updateDoc(userRef, {
                unreadMessages: false
            });
        }
    }, [user]);

    //Search for a user use effect
    useEffect(async () => {
        if (search && search !== "") {
            await searchFromAllUsers();
        } else {
            setUsers([]);
            await getUsers();
        }
    }, [search]);

    //Get messages depending on the room use effect
    useEffect(async () => {
        if (!user) return;
        if (currentRoomId) {
            const roomRef = doc(database, "rooms/" + currentRoomId);
            await updateDoc(roomRef, {
                [user.uid]: 0
            });
            setMessages([]);
            const q = query(collection(database, "rooms/" +
currentRoomId + "/messages"), orderBy("createdAt", "desc"));
            const unsubscribe = onSnapshot(q, (querySnapshot) => {
                querySnapshot.forEach(async () => {
                    const updatedMessages =
querySnapshot.docs.map((doc) => doc.data());
```

```
                    setMessages(updatedMessages);
                });
            });
            return unsubscribe;
        }
    }, [currentRoomId]);

    //Get the users from the database only if the user has started a
conversation
    const getUsers = async () => {
        //Get users existing in rooms current user
        const roomsRef = collection(database, "rooms");
        let usersTemp = [];
        getDocs(roomsRef, where("members." + user.uid, "==",
true)).then((querySnapshot) => {
            querySnapshot.forEach( async (oneDoc) => {
                //Get one of the two members of the room
                let newUserUid = "";
                if (Object.keys(oneDoc.data().members)[0] !== user.uid)
{
                    newUserUid = Object.keys(oneDoc.data().members)[0];
                } else {
                    newUserUid = Object.keys(oneDoc.data().members)[1];
                }
                //Check if messages collection of the room is not empty
                const room = oneDoc.data();
                const queryRoom = query(roomsRef,
where(`members.${user.uid}`, "==", true),
where(`members.${newUserUid}`, "==", true));
                const querySnapshotRoom = await getDocs(queryRoom);
                if (!querySnapshotRoom.empty) {
                    //Get room id
                    const roomId = querySnapshotRoom.docs[0].id;
                    const q = query(collection(database, "rooms/" +
roomId + "/messages"), orderBy("createdAt", "asc"));
                    const querySnapshotMessages =  await getDocs(q);
                    if (!querySnapshotMessages.empty) {
                        const docRef = doc(database, "users",
newUserUid);
                        const docSnap =  await getDoc(docRef);

                        if (docSnap.exists() && docSnap.data().uid !==
user.uid) {
```

```
                            //Add user in the list if he is not already
in the list
                            const userAlreadyInList =
usersTemp.find((oneUser) => oneUser.uid === newUserUid);
                            if (!userAlreadyInList) {
                                const newUser = docSnap.data();
                                newUser.uid = newUserUid;
                                newUser.roomId = roomId;
                                newUser.unreadMessages =
room[user.uid];

                                usersTemp.push(newUser);
                                setUsers(usersTemp);
                            }
                        }
                    }
                }
            });
        });
    };

    //Search from all users in database
    const searchFromAllUsers = async () => {
        const querySnapshot = await getDocs(collection(database,
"users"));
        querySnapshot.forEach((doc) => {
            let newUser = doc.data();
            let userData = {};
            userData.uid = newUser.uid;
            userData.displayName = newUser.displayName;
            userData.email = newUser.email;
            userData.photoURL = newUser.photoURL;

            if
((newUser?.displayName?.toLowerCase().includes(search.toLowerCase()) ||
newUser.email?.toLowerCase().includes(search?.toLowerCase())) &&
userData.uid !== user.uid) {
                if (!users.find(oneUser => oneUser.uid ===
userData.uid)) {
                    const newUsers = users;
                    newUsers.push(userData);
                    setUsers(newUsers);
                }
            }
            if (search !== "") {
                filterUsers();
            }

        });
    };

    //Filter users by display name and email
    const filterUsers = () => {
        if(search){
            if (users.length > 0) {
                const filteredUsers = users?.filter(oneUser =>
oneUser.displayName?.toLowerCase().includes(search.toLowerCase()) ||
oneUser.email?.toLowerCase().includes(search.toLowerCase()));
                setUsers(filteredUsers);
            }
        }
    };


    //Change the interlocutor
    const changeInterlocutor = async (uid) => {
        if (sessionStorage.getItem("interlocutorUID") === uid &&
!displayUserList) return;
        setCurrentRoomId(null);
        setInterlocutorUID(uid);
        //Set unread messages number to 0 for the current interlocutor
in the users array
        users.map((oneUser) => {
            if (oneUser.uid === uid) {
                oneUser.unreadMessages = 0;
            }
            return oneUser;
        });
        sessionStorage.setItem("interlocutorUID", uid);
        setInterlocutorName(users.find(oneUser => oneUser.uid ===
uid).displayName);
        setDisplayUserList(false);
        setMessages([]);
        await createRoom(uid);
    };

    //Convert the date to a string
    const timestampToString = (timestamp) => {
        const a = new Date(timestamp * 1000);
        let month = a.getMonth();
        month++;
        month = month < 10 ? '0' + month: month;
        const date = a.getDate();
        const hour = a.getHours();
        const min = a.getMinutes() < 10 ? '0' + a.getMinutes() :
a.getMinutes();
        const time = date + '/' + month + ' ' + hour + ':' + min;
        return time;
    };

    //Send the message
    const sendMessage = async (e) => {
        e.preventDefault();
        const message = formValue;
        if (message && message.trim() !== "") {
            const newMessage = {
                message: message,
                sender: user.uid,
                displayName: user.displayName,
                createdAt: Timestamp.fromDate(new Date()),
            };
            if (user.photoURL) {
                newMessage.photoURL = user.photoURL;
            }
            setFormValue("");
            addDoc(collection(database, "rooms/" + currentRoomId +
"/messages"), newMessage)
                .then(async () => {
                    //We increment the number of unread messages by 1
                    const roomRef = doc(database, "rooms/" +
currentRoomId);
                    await updateDoc(roomRef, {
                        [interlocutorUID]: increment(1)
                    });
                    const interlocutorRef = doc(database, "users/" +
interlocutorUID);
                    await updateDoc(interlocutorRef, {
                        unreadMessages: true
                    });
                }).catch(function (error) {
                    console.error("Error writing document: ", error);
            });
        }
    };

    //Create a room with interlocutor if it doesn't exist
    const createRoom = async (interlocutorUID) => {
        const uidString = user.uid;
        const q = query(roomsRef, where(`members.${uidString}`, "==",
true), where(`members.${interlocutorUID}`, "==", true));
        const querySnapshot = await getDocs(q);

        const room = {
            members: {
                [uidString]: true,
                [interlocutorUID]: true
            }
        };

        if (querySnapshot.empty) {
            addDoc(roomsRef, room).then((roomRef) => {
                setCurrentRoomId(roomRef.id);
            }).catch(function (error) {
                console.error("Error writing document: ", error);
            });
        } else {
            setCurrentRoomId(querySnapshot.docs[0].id);
        }
    };

    //Change search value
    const handleChangeSearch = (e) => {
        setSearch(e.target.value);
    };

    //Handle click on arrow back
    const handleClickArrowBack = () => {
        setDisplayUserList(true)
    };

    return (
        <div>
            <Container className="mt-5">
                <Row>
                    <NavigationBar/>
                </Row>
            </Container>
            <Grid container item lg={12} xs={12} md={12}
sx={{margin:"50px 0 0 0"}}>
                <Grid>
                    <Typography variant="h5"
className="header-message">Chat</Typography>
                </Grid>
            </Grid>
            <Grid container component={Paper} className={"chatSection"}
sx={{ boxShadow: "1px -1px rgb(0, 0, 0, 0.2)" }}>
                <Grid item lg={3} md={3} xs={12}
className={displayUserList ? "borderRight500" : "d-none d-md-block
borderRight500"}>
                    <List>
                        <ListItem key="user" >
                            <Row className={"justify-content-center
flex-wrap align-items-center"}>
                                <Col lg={3} xs={3}>
                                    <ListItemIcon>
                                        <Avatar alt="avatar"
src={user?.photoURL} />
                                    </ListItemIcon>
                                </Col>
                                <Col lg={9} xs={9}>
                                    <ListItemText
primary={user?.displayName }></ListItemText>
                                </Col>
                            </Row>
                        </ListItem>
                    </List>
                    <Divider />
                    <Grid item xs={12} style={{padding: '10px'}}
className={displayUserList ? "" : "d-none d-md-block"}>
                        <TextField id="outlined-basic-email"
label="Search" value={search ?? ""} onChange={handleChangeSearch}
variant="outlined" fullWidth />
                    </Grid>
                    <Divider />
                    <List className={displayUserList ? "d-block" :
"d-none d-md-block"}>
                        {/*List of users*/}
                        {users?.map((oneUser, index) => (
                            <div key={index}>
                                {oneUser?.uid !== user?.uid &&
                                    <ListItemButton onClick={() =>
changeInterlocutor(oneUser?.uid)}

selected={oneUser?.uid === interlocutorUID}>
                                        <Row
className={"justify-content-center flex-wrap align-items-center"}>
                                            <Col lg={3} xs={3}>
                                                <ListItemIcon>
                                                    <Avatar
alt="avatar" src={oneUser?.photoURL}/>
                                                </ListItemIcon>
                                            </Col>
                                            <Col lg={9} xs={9}>
                                                <ListItemText
primary={oneUser?.displayName ? oneUser?.displayName :
oneUser?.email}/>
                                            </Col>
                                        </Row>
                                        <Col>
                                            {oneUser?.unreadMessages >
0 &&
                                                <Badge
color="secondary" badgeContent={oneUser?.unreadMessages.toString()} />
                                            }
                                        </Col>
                                    </ListItemButton>
                                }
                            </div>
                        ))}
                    </List>
                </Grid>
                <Grid item xs={12} lg={9} md={9}
className={displayUserList ? "d-none d-md-block": ""} sx={{position:
"relative", width: "100%"}}>
                    {/*If no interlocutor is selected, then show this
message*/}
                    {!interlocutorUID &&
                        <Row className={"d-flex justify-content-center
align-items-center h-100"}>
                            <Col>
                                <Typography textAlign={"center"}
sx={{fontSize: 36}}>Select or search a user to start a
conversation</Typography>
                            </Col>
                        </Row>
                    }
                    <Grid item xs={12} style={{padding: '10px',
position: "absolute", top: 0, width:"100%"}}>
                        <Row className={"w-100"}>
                            <Col xs={3} className={displayUserList ?
"d-none" : "d-block d-md-none"}>
                                <ArrowBackIosNew sx={{cursor:
"pointer"}} onClick={handleClickArrowBack}/>
                            </Col>
                            <Col xs={9}>
                                <Typography
variant={"h6"}>{interlocutorName}</Typography>
                            </Col>
                        </Row>
                        <Divider/>
                    </Grid>

                    <List sx={{ width: '100%', height:"80%",
paddingBottom: "96px",position: "absolute", top: 50, overflow:"scroll",
overflowX: "hidden", display:"flex", flexDirection:"column-reverse"}}
className={"messageArea"}>
                        {messages?.map((message, index) => (
                            <ListItem alignItems="flex-start"
key={index} sx={{display: "block"}}>
                                <ListItemAvatar>
                                    <Avatar alt="Avatar"
src={message?.photoURL} />
                                </ListItemAvatar>
                                <ListItemText
                                    primary={
                                        <p className="p-0
m-0">{message?.displayName} <span className="text-secondary
fontSmall">{timestampToString(message?.createdAt)}</span></p>
                                    }
                                    secondary={
                                        <React.Fragment>
                                            <Typography
                                                sx={{display: 'inline',
overflowWrap: "break-word"}}
                                                component="span"
                                                variant="body2"
                                                color="text.primary"
                                            >
```

```
                                                {message?.message}
                                            </Typography>
                                        </React.Fragment>
                                    }
                                />
                            </ListItem>
                        ))}
                    </List>

                    <Grid container style={{padding: '20px'}}
sx={{position: "absolute", bottom: 0, backgroundColor: "white"}}>
                        <Grid item xs={11}>
                            {/*Input to type a message*/}
                            <TextField
                                id="outlined-basic-message"
                                label="Type a message..."
                                fullWidth
                                multiline
                                value={formValue}
                                onChange={(e) =>
setFormValue(e.target.value)}
                                onKeyPress={(e) => e.key === "Enter" ?
sendMessage(e) : null}
                            />
                        </Grid>
                        <Grid  item xs={1} align="right">
                            <Fab color="primary" sx={{backgroundColor:
"#FA4616"}} aria-label="add" onClick={sendMessage}><SendIcon /></Fab>
                        </Grid>
                    </Grid>
                </Grid>
            </Grid>
        </div>
    )
}

export default Messages;
```

Messages CSS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */
```

```css
.table {
    min-width: 650px;
}
.chatSection {
    width: 100%;
    height: 80vh;
    border-bottom: 0;
}
.headBG {
    background-color: #e0e0e0;
}
.borderRight500 {
    border-right: 1px solid #e0e0e0;
}
.messageArea {
    height: 70vh;
}

.fontSmall {
    font-size: 0.8em;
}
```

My Cookbooks

My Cookbooks JS File

```javascript
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */


import {Add}                                 from "@mui/icons-material";
import {addDoc, collection, doc, getDocs} from "firebase/firestore";
import {useNavigate}                         from "react-router-dom";
import NavigationBar                         from
"../NavigationBar/NavigationBar";
import React, {useEffect, useState}          from "react";
import {Col, Container, Row}                 from "react-bootstrap";
import Cookbook                              from "../CookBook/Cookbook";
import {database} from "../../FirebaseConfig";
import {BootstrapButton, BootstrapInput} from "./MyCookbooksStyle";

const MyCookbooks = () => {
    const user = JSON.parse(sessionStorage.getItem('user'));
    const navigate = useNavigate();

    const [cookbooks, setCookbooks] = useState([]);
    const [cookbook, setCookbook] = useState("");
    const [buttonDisabled, setButtonDisabled] = useState(true);
    const [currentIndexCookbook, setCurrentIndexCookbook] =
useState(null);

    useEffect( async () => {
        //If the user is not logged in, redirect to the login page
        //To handle the case where the user is not logged in and tries
to access the page
        if(!user){
            navigate("/");
        }
        await getUserCookbooks();
    }, []);

    //add Cookbook to firebase database
    const addCookbook = async () => {
        //if this cookbook name is already in cookbooks, don't add it
        if (cookbooks.includes(cookbook)) return;
        if (cookbook === "") return;
        const newCookbook = {
            name: cookbook,
            owner: user.uid,
            recipes: []
        };

        addDoc(collection(database, "users/" + user.uid +
"/cookbooks"), newCookbook)
            .then(() => {
                getUserCookbooks();
                window.location.reload();
            }).then(() => {
                setCookbook("");
            }).catch(function (error) {
                console.error("Error writing document: ", error);
            });
    }
    //Get user cookbooks from firebase database
    const getUserCookbooks = async () => {
        const querySnapshot = await getDocs(collection(database,
"users/" + user?.uid + "/cookbooks"));
        querySnapshot.forEach((doc) => {
            let newCookbook = doc.data();
            newCookbook.id = doc.id;

            if(!cookbooks.some(cookbook => cookbook.id ===
newCookbook.id)){
                setCookbooks(cookbooks => [...cookbooks, newCookbook]);
            } else {
                const index = cookbooks.findIndex(cookbook =>
cookbook.id === newCookbook.id);
                cookbooks[index] = newCookbook;
                setCookbooks(cookbooks);
            }
        });
    }

    //Change cookbook name
    const updateCookbook = (event) => {
        if(event.target.value === ""){
            setButtonDisabled(true);
        } else {
            setButtonDisabled(false);
        }
        setCookbook(event.target.value);
    }

    //Change current cookbook
    const handleChangeCurrentIndexCookbook = (index) => {
        setCurrentIndexCookbook(index);
        //close all others cookbooks
        cookbooks.forEach((cookbook, i) => {
            if(i !== index){
                cookbook.open = false;
            }
        });
    }

    return (
        <>
```

```
            <Container className="mt-5">
                <Row>
                    <NavigationBar/>
                </Row>
            </Container>
            <Container className="mt-4">
                <Row className={"justify-content-center"}>
                    <Col lg={5} mf={7} sm={7}>
                        <BootstrapInput multiline className="input-100
recipe-input" placeholder="Cookbook name" onChange={updateCookbook}
required={true} value={cookbook}/>
                        <BootstrapButton
                            variant="contained"
                            disableElevation
                            disabled={buttonDisabled}
                            sx={{width: '100%', display: 'flex',
justifyContent: 'center', alignItems: 'center'}}
                            onClick={addCookbook}
                            endIcon={<Add />}
                        >
                            Add a new Cookbook
                        </BootstrapButton>
                        {/*List of cookbooks*/}
                        {
                            cookbooks.length > 0 &&
cookbooks.map(cookbook => (
                                <Cookbook
                                    key={cookbook.id}
                                    name={cookbook.name}
                                    recipes={cookbook.recipes}
                                    id={cookbook.id}
                                    setCurrentCookbook={() =>
handleChangeCurrentIndexCookbook(cookbook.id)}

currentCookbook={currentIndexCookbook}
                                />
                            ))
                        }
                    </Col>
                </Row>
            </Container>
        </>
    );
};
```

```
export default MyCookbooks;
```

My Cookbooks Style File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */


import {Button, InputBase} from "@mui/material";
import {alpha, styled}      from "@mui/material/styles";

export const BootstrapButton = styled(Button)({
    boxShadow: 'none',
    textTransform: 'none',
    fontSize: 16,
    padding: '6px 12px',
    border: '1px solid',
    lineHeight: 1.5,
    backgroundColor: '#FA4616',
    borderColor: `${alpha('#FA4616', 0.25)} 0 0 0 0.2rem`,
    justifyContent: 'space-around',
    fontFamily: [
        '-apple-system',
        'BlinkMacSystemFont',
        '"Segoe UI"',
        'Roboto',
        '"Helvetica Neue"',
        'Arial',
        'sans-serif',
        '"Apple Color Emoji"',
        '"Segoe UI Emoji"',
        '"Segoe UI Symbol"',
    ].join(','),
    '&:hover': {
        backgroundColor: '#FA4616',
        borderColor: `${alpha('#FA4616', 0.25)} 0 0 0 0.2rem`,
        boxShadow: 'none',
    },
    '&:active': {
        boxShadow: 'none',
        backgroundColor: '#FA4616',
        borderColor: `${alpha('#FA4616', 0.25)} 0 0 0 0.2rem`,
    },
    '&:focus': {
        boxShadow: `${alpha('#FA4616', 0.25)} 0 0 0 0.2rem`,
    },
});
export const BootstrapInput = styled(InputBase)(({ theme }) => ({
    'label + &': {
        marginTop: theme.spacing(3),
    },
    '& .MuiInputBase-input': {
        borderRadius: 4,
        position: 'relative',
        backgroundColor: theme.palette.mode === 'light' ? '#fcfcfb' :
'#2b2b2b',
        border: '1px solid #ced4da',
        fontSize: 16,
        width: '100%',
        padding: '10px 10px 10px 10px',
        margin: '5px 0px 5px 0px',
        transition: theme.transitions.create([
            'border-color',
            'background-color',
            'box-shadow',
        ]),
        // Use the system font instead of the default Roboto font.
        fontFamily: [
            '-apple-system',
            'BlinkMacSystemFont',
            '"Segoe UI"',
            'Roboto',
            '"Helvetica Neue"',
            'Arial',
            'sans-serif',
            '"Apple Color Emoji"',
            '"Segoe UI Emoji"',
            '"Segoe UI Symbol"',
        ].join(','),
        '&:focus': {
            //transparent for #FA4616
            boxShadow: `${alpha('#FA4616', 0.25)} 0 0 0 0.2rem`,
            borderColor: '#FA4616',
        },
    },
}));
```

My Recipes

My Recipes JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */

//MUI Icons
import SearchIcon
from '@mui/icons-material/Search';
//MUI material
import {Alert, AppBar, IconButton, InputBase, Paper, Snackbar}
from "@mui/material";
//React bootstrap
import {Col, Container, Row}
from "react-bootstrap";
//Components
import NavigationBar                                          from
"../NavigationBar/NavigationBar";
import Recipe                                                 from "../Recipe/Recipe";
import MyRecipesLogic                                          from
"./MyRecipesLogic";


//MyRecipes Component
 const  MyRecipes = () => {
     //Import of variables and functions
     const {
         search,
         recipes,
         updateSearch,
         updateRecipe,
         removeRecipe,
         handleCloseSuccessSnackbar,
         handleCloseModifySnackbar,
         handleCloseRemoveSnackbar,
         openSuccessSnackbar,
         openModifySnackbar,
         openRemoveSnackbar,
     } = MyRecipesLogic();

     return(
        <>
            <Container className="mt-5">
                <Row>
                    <NavigationBar/>
                </Row>
            </Container>

            <Container className="mt-4">
                <Row className={"justify-content-center"}>
                     <Col lg={5} mf={7} sm={7}>
                         {/*recipe search form*/}
                        <Paper className="mb-4"  sx={{ p: '0px 4px',
display: 'flex', alignItems: 'center', maxWidth: '100%' }}   >

                            <InputBase sx={{ ml: 1, flex: 1 }}
placeholder="Search recipe or ingredient" value={search ?? ""}
onChange={updateSearch} />

                            <IconButton  sx={{ p: '10px' }}
aria-label="search">
                                <SearchIcon className="searchIcon"/>
                            </IconButton>
                        </Paper>
                        {/*List of recipes*/}
                        {recipes.map((recipe, index) =>(
                            <Container className="mt-2" key={index}>
                                <Row>
                                    <Col>
                                            <Recipe
                                                    key={index}

title={recipe.title}

calories={recipe.calories}

image={recipe.image}

method={recipe.method}

ingredients={recipe.ingredients.map(ingredient => ingredient.text)}
                                                    uid={recipe.uid}
photoURL={recipe.photoURL}

reviews={recipe.reviews}

displayName={recipe.displayName}

disableReview={true}

updateRecipe={updateRecipe}
                                                                     removeRecipe={(e)
=> removeRecipe(e, recipe.id)}
                                                                     id={recipe.id}
                                                          />
                                                 </Col>
                                             </Row>
                                         </Container>
                                   ))}
                             </Col>
                        </Row>
                    </Container>

                    {/*Snackbars*/}
                    <Snackbar open={openSuccessSnackbar}
autoHideDuration={6000} onClose={handleCloseSuccessSnackbar}>
                        <Alert onClose={handleCloseSuccessSnackbar}
severity="success" sx={{ width: '100%' }}>
                            Recipe successfuly added !
                        </Alert>
                    </Snackbar>
                    <Snackbar open={openModifySnackbar} autoHideDuration={6000}
onClose={handleCloseModifySnackbar}>
                        <Alert onClose={handleCloseModifySnackbar}
severity="success" sx={{ width: '100%' }}>
                            Recipe successfuly modified !
                        </Alert>
                    </Snackbar>
                    <Snackbar open={openRemoveSnackbar} autoHideDuration={6000}
onClose={handleCloseRemoveSnackbar}>
                        <Alert onClose={handleCloseRemoveSnackbar}
severity="success" sx={{ width: '100%' }}>
                            Recipe successfuly removed !
                        </Alert>
                    </Snackbar>
```

```
        </>
    );
};

export default MyRecipes;
```

My Recipes Logic Js File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */

import {useEffect, useState}
from "react";
//React Router
import {useNavigate}
from "react-router-dom";
//Firebase
import {collection, deleteDoc, doc, getDoc, getDocs, query, updateDoc,
where} from "firebase/firestore";
import {database}
from "../../FirebaseConfig";

const MyRecipesLogic = () => {

    const user = JSON.parse(sessionStorage.getItem('user'));
    const navigate = useNavigate();

    const [recipes, setRecipes ] = useState ([]);
    const [search, setSearch] = useState(null);
    const [openSuccessSnackbar, setOpenSuccessSnackbar] =
useState(false);
    const [openModifySnackbar, setOpenModifySnackbar] =
useState(false);
    const [openRemoveSnackbar, setOpenRemoveSnackbar] =
useState(false);

    //Fetching recipes from Firebase database
    const getUserRecipes = async () => {
        const q = query(collection(database, "recipes"), where("uid",
"==", user?.uid));
        const querySnapshot = await getDocs(q);

        querySnapshot.forEach((doc) => {
            let newRecipe = doc.data();
            //if the id of a recipe is not already in the array, add
the recipe
            newRecipe.id = doc.id;
            //Update user photo of the recipe
            newRecipe.userPhoto = user?.photoURL;
            if(!recipes.some(recipe => recipe.id === newRecipe.id)){
                setRecipes(recipes => [...recipes, newRecipe]);
            } else {
                //if the id of a recipe is already in the array, update
the recipe
                const index = recipes.findIndex(recipe => recipe.id ===
newRecipe.id);
                recipes[index] = newRecipe;
                setRecipes(recipes);
            }
        });

        if (search) {
            filterRecipes();
        }
    }

    //Filter recipes by title and ingredients
    const filterRecipes = () => {
        const filteredRecipesByTitle = recipes.filter(recipe =>
recipe?.title.toLowerCase().includes(search.toLowerCase()));
        const filteredRecipesByIngredients = recipes.filter(recipe =>
recipe?.ingredients.some(ingredient =>
ingredient.text.toLowerCase().includes(search.toLowerCase())));
        //merge the two arrays without same id
        const filteredRecipes = [...filteredRecipesByTitle,
...filteredRecipesByIngredients].filter((recipe, index, self) =>
                index === self.findIndex((t) => (
                    t.id === recipe.id
                ))
        );
        setRecipes(filteredRecipes);
    }
    //update recipe in Firebase database
    const updateRecipe = async (e, recipe, ingredients) => {
        e.preventDefault();
        recipe.ingredients = [];
        ingredients.map(ingredient => {
            recipe.ingredients.push({text: ingredient});
        });

        updateDoc(doc(database, "recipes", recipe.id), recipe).then(()
=> {
            getUserRecipes();
            setOpenModifySnackbar(true);
        }).catch(() => {
            console.log("Error updating recipe");
        });
    }

    //remove recipes by ID from Firebase database
    const removeRecipe = async (e, id) => {
        e.preventDefault();
        deleteDoc(doc(database, "recipes", id))
            .then(() => {
                //remove recipe from recipes
                setRecipes(recipes => recipes.filter(recipe =>
recipe.id !== id));
                setOpenRemoveSnackbar(true);
                sessionStorage.setItem("openDeleteSnackbar", "true");
                window.location.reload();
            }).catch(() => {
                console.log("Error removing recipe");
        });
    }

    //Refresh recipes of the cookbook from the firebase database
    const refreshRecipes = async () => {
        recipes.map(async (recipe) => {
            if (recipe.id) {
                const docRef = doc(database, "recipes", recipe.id);
                const docSnap = await getDoc(docRef);

                if (docSnap.exists()) {
                    //if the recipe exists in the recipes array, update
the recipe
```

```
                    const recipe = docSnap.data();
                    recipe.id = docSnap.id;
                    recipe.ingredients =
recipe.ingredients.map(ingredient => ingredient.text);
                    setRecipes(recipes.map((r) => r.id === recipe.id ?
recipe : r));
                } else {
                    // doc.data() will be undefined in this case
                    console.log("No such document!");
                }
            }
        });
    }


    useEffect(async () => {
        //If the user is not logged in, redirect to the login page
        //To handle the case where the user is not logged in and tries
to access the page
        if(!user){
            navigate("/");
        }
        await getUserRecipes(search);
        if (sessionStorage.getItem("openDeleteSnackbar")) {
            setOpenRemoveSnackbar(true);
            sessionStorage.removeItem("openDeleteSnackbar");
        }
        if (!search) {
            await refreshRecipes();
        }
    }, [search]);

    const updateSearch = e => {
        setSearch(e.target.value);
    }

    const handleCloseSuccessSnackbar = () => {
        setOpenSuccessSnackbar(false);
    };

    const handleCloseModifySnackbar = () => {
        setOpenModifySnackbar(false);
    };
    const handleCloseRemoveSnackbar = () => {
        setOpenRemoveSnackbar(false);
    };

    return {
        search,
        recipes,
        updateSearch,
        updateRecipe,
        removeRecipe,
        handleCloseSuccessSnackbar,
        handleCloseModifySnackbar,
        handleCloseRemoveSnackbar,
        openSuccessSnackbar,
        openModifySnackbar,
        openRemoveSnackbar
    }
}

export default MyRecipesLogic;
```

Navigation Bar

Navigation Bar File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */



import {doc, onSnapshot} from "firebase/firestore";
import {database}
from "../../FirebaseConfig";
import React, {useEffect, useState} from "react";
//MUI icons
import {
    AccountBoxRounded,
    ChatRounded,
    CloseRounded,
    FoodBankRounded,
    Menu,
    MenuBookRounded,
    Notifications, NotificationsActive,
    PublicRounded
} from "@mui/icons-material";
//MUI Material
import {
    Box,
    Divider,
    Drawer,
    IconButton,
    List,
    ListItem,
    ListItemIcon,
    ListItemText,
    Toolbar, Tooltip,
    Typography
} from "@mui/material";
import MuiAppBar from "@mui/material/AppBar";
//MUI styles
import {useTheme}            from "@mui/material/styles";
//React Bootstrap
import {Button, Col}                    from "react-bootstrap";
//Router DOM
import {useNavigate} from "react-router-dom";
//User auth context
import {useUserAuth} from "../../context/UserAuthContext";
//Styles
import {DrawerHeader, AppBar} from "./NavigationBarStyles";

//NavigationBar component
const NavigationBar = () => {
    const { logOut } = useUserAuth();
    const user = JSON.parse(sessionStorage.getItem("user"));
    const theme = useTheme();
    const navigate = useNavigate();

    const handleLogOut = async () => {
        try{
            sessionStorage.removeItem('user');
            navigate('/');
            await logOut();
        } catch(err) {
            console.log(err.message);
        }
    };
    //Burger Menu Start
    const drawerWidth = 260;
    const [open, setOpen] = useState(false);
    const [currentPage, setCurrentPage] = useState("");
    const [unreadMessages, setUnreadMessages] = useState(false);

    //Basic use effect
    useEffect(async () => {
        //Initialize the current page name, first letter is a major
letter
        let currentPage =
window.location.pathname.slice(1,2).toUpperCase() +
window.location.pathname.slice(2);
        if (currentPage === "Mycookbooks") currentPage = "My
Cookbooks";
        if (currentPage === "Myrecipes") currentPage = "My Recipes";
        setCurrentPage(currentPage);
    }, []);

    //Use effect when user changes
    useEffect(() => {
        if (sessionStorage.getItem("user")) {
            const unsubscribe = onSnapshot(doc(database, "users/" +
user.uid), (doc) => {
                setUnreadMessages(doc.data()?.unreadMessages);
            });
            return unsubscribe;
        }
    }, [user]);

    //Burger Menu Close
    const handleDrawerClose = () => {
        setOpen(false);
    };

    //Burger Menu Open
    const handleDrawerOpen = () => {
        setOpen(true);
    };

    //Navigate to a page
    const navigateTo = (path) =>{
        if (!user) {
            navigate("/login");
        } else {
            setOpen(false);
            //delete spaces in the path
            path = path.replace(/\s/g, '');
            //navigate to the path
            navigate('/' + path.toLowerCase());
        }
    }

    return(
        <Col lg={12} mf={12} sm={12}>
            <Box>
                <AppBar position="fixed">
                    <Toolbar className="navBg">
                        {/*Burger Menu Icon*/}
                        <IconButton size="large" edge="start"
color="inherit" onClick={handleDrawerOpen} aria-label="menu">
                            <Menu  color="inherit" aria-label="open
drawer"  edge="start" sx={{...(open && { display: 'none' }) }} />
                        </IconButton>

                        <Typography variant="h6" component="div" sx={{
flexGrow: 1, marginLeft: 2}}> {currentPage}</Typography>
                            <Tooltip title={unreadMessages ? "You have
unread messages" : "You don't have unread messages"}>
                                <IconButton onClick={() =>
navigate("/messages")}>
                                    {
                                        unreadMessages
                                            ?<NotificationsActive
sx={{color: "white"}}/>
                                            :<Notifications sx={{color:
"white"}}/>
                                    }
                                </IconButton>
                            </Tooltip>
                        <Button className="logoutBtn"
onClick={handleLogOut}>Logout</Button>
                    </Toolbar>
                </AppBar>
                {/*{Burger Menu}*/}
                <Drawer sx={{width: drawerWidth, flexShrink: 0,'&
.MuiDrawer-paper':{width: drawerWidth, boxSizing: 'border-box',
backgroundColor: '#FA4616'},}} variant="persistent" anchor="left"
open={open}>
                    <DrawerHeader className="burgerMenuBg">
                        <IconButton className="menuOptions"
onClick={handleDrawerClose}>
                            {theme.direction === 'ltr' ? <CloseRounded
className="menuOptions"/> : <CloseRounded /> }
                        </IconButton>
                    </DrawerHeader>
                    <Divider />
                    <List className="menuOptions">
                        {/*List of navigation pages*/}
                        {['Profile', 'Recipes', 'Discover', 'My
Recipes', 'My Cookbooks', 'Messages'].map((text, index) => (
                            <ListItem button key={index} onClick={ ()
=> navigateTo(text)}>
                                <ListItemIcon>
                                    {index === 0 && <AccountBoxRounded
className="menuOptions"/>}
                                    {index === 1 && <MenuBookRounded
className="menuOptions"/>}
                                    {index === 2 && <PublicRounded
className="menuOptions"/>}
                                    {index === 3 && <MenuBookRounded
className="menuOptions"/>}
                                    {index === 4 && <FoodBankRounded
fontSize="medium" className="menuOptions"/>}
                                    {index === 5 && <ChatRounded
className="menuOptions"/>}
                                </ListItemIcon>
                                <ListItemText primary={text} />
                            </ListItem>
                        ))}
                    </List>
                </Drawer>
            </Box>
        </Col>
    )
}

export default NavigationBar;
```

Navigation Bar Style File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */

import MuiAppBar from "@mui/material/AppBar";
import {styled}  from "@mui/material/styles";

const drawerWidth = 260;

export const DrawerHeader = styled('div')(({ theme }) => ({
    display: 'flex',
    alignItems: 'center',
    padding: theme.spacing(0, 1),
    // necessary for content to be below app bar
    ...theme.mixins.toolbar,
    justifyContent: 'flex-end',
}));

export const AppBar = styled(MuiAppBar, {
    shouldForwardProp: (prop) => prop !== 'open',
})(({ theme, open }) => ({
    transition: theme.transitions.create(['margin', 'width'], {
        easing: theme.transitions.easing.sharp,
        duration: theme.transitions.duration.leavingScreen,
    }),
    ...(open && {
        width: `calc(100% - ${drawerWidth}px)`,
        marginLeft: `${drawerWidth}px`,
        transition: theme.transitions.create(['margin', 'width'], {
            easing: theme.transitions.easing.easeOut,
            duration: theme.transitions.duration.enteringScreen,}),
    }),
}));
```

New Recipe Card

New Recipe Card JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */
```

```
import React                                                 from "react";
//MUI icons
import {Add, Done, FileUpload, Remove}                          from
"@mui/icons-material";
import ExpandMoreIcon                                           from
"@mui/icons-material/ExpandMore";
//MUI material
import {CardActionArea, CardActions, CardMedia, Collapse} from
"@mui/material";
//React bootstrap
import {Col, Row}                                               from
"react-bootstrap";
//Logic
import NewRecipeCardLogic from "./NewRecipeCardLogic";
//Styles
import {ExpandMore, BootstrapInput, BootstrapButton, CardContentStyled,
CardStyles} from "./NewRecipeCardStyles";

const NewRecipeCard = ({addRecipe}) => {

    const {
        expanded,
        expandedMethod,
        title,
        image,
        ingredients,
        calories,
        ingredient,
        method,
        handleExpandClick,
        handleExpandMethod,
        updateTitle,
        updateImageFile,
        updateMethod,
        addIngredient,
        updateCalories,
        updateIngredient,
        checkRecipe,
        deleteIngredient,
        emptyFields

    } = NewRecipeCardLogic();
    return (
        {/*card around add recipe */},
            <Col>
                <CardContentStyled>
                    <BootstrapInput multiline className="input-100
recipe-input" placeholder="Recipe" onChange={updateTitle}
required={true} value={title}/>
                    <BootstrapInput multiline className="input-100"
placeholder="Calories" onChange={updateCalories} required={true}
value={calories}/>
                </CardContentStyled>
                <CardActionArea >
                    <Row sx={{justifyContent:"center"}}>
                        <label htmlFor="image-input">
                            <CardMedia component="img" height="300"
width="100%" image={image} alt="Recipe image"
style={CardStyles.media}/>
                        </label>
                        <label htmlFor="image-input">
                            <FileUpload className="upload-icon"
style={CardStyles.overlay} fontSize="large" color="primary"/>
                        </label>
                        <input id="image-input" type="file"
onChange={updateImageFile} hidden={true} required={true}/>
                    </Row>
                </CardActionArea>
                <CardActions disableSpacing>
                    <div>Ingredients</div>
                    <ExpandMore expand={expanded}
onClick={handleExpandClick} aria-expanded={expanded} aria-label="show
more">
                        <ExpandMoreIcon />
                    </ExpandMore>
                </CardActions>
                <Collapse in={expanded} timeout="auto" unmountOnExit>
                    <CardContentStyled>
                        {/*List of ingredients*/}
                        {ingredients && ingredients.map((ingredient,
index) =>(
                                <BootstrapButton
                                    key={index}
                                    variant="contained"
                                    disableElevation
                                    aria-expanded={expanded}
                                    aria-label="show more"
                                    onClick={() =>
deleteIngredient(index)}
                                    endIcon={<Remove color={"error"}/>}
                                >
                                    {ingredient}
                                </BootstrapButton>
                        ))}
                        {/*Add ingredient input*/}
                            <BootstrapInput multiline
className="input-100 recipe-input" placeholder="Ingredient"
value={ingredient} onChange={updateIngredient} />
                            <BootstrapButton
                                variant="contained"
                                disableElevation
                                onClick={addIngredient}
                                endIcon={<Add color={"success"}/>}
                            >
                                Add an ingredient
                            </BootstrapButton>
                    </CardContentStyled>
                </Collapse>
                {/*Add a method*/}
                <CardActions disableSpacing>
                    <div>Method</div>
                    <ExpandMore expand={expandedMethod}
onClick={handleExpandMethod} aria-expanded={expandedMethod}
aria-label="show more">
                        <ExpandMoreIcon />
                    </ExpandMore>
                </CardActions>
                <Collapse in={expandedMethod} timeout="auto"
unmountOnExit>
                    <CardContentStyled>
                        <BootstrapInput multiline className="input-100
recipe-input" placeholder="Method" onChange={updateMethod}
required={true} value={method}/>
                    </CardContentStyled>
                </Collapse>
                <CardContentStyled sx={{paddingBottom: '0'}}>
                    {/*Add recipe button*/}
                    <BootstrapButton
                        color="secondary"
                        variant="contained"
                        disableElevation
                        disabled={!checkRecipe()}
                        onClick={(event) => addRecipe(event, {title,
image, calories, method}, ingredients).then(emptyFields)}
                        type="submit"
                        endIcon={<Done color={checkRecipe() ? "success"
: "disabled"}/>}
                    >
                        Validate
                    </BootstrapButton>
                </CardContentStyled>
            </Col>
    );
}

export default NewRecipeCard;
```

New Recipe Card Logic JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */


import * as firebaseServices from "../../services/FirebaseServices";
import {useState}            from "react";

const NewRecipeCardLogic = () => {
    const [expanded, setExpanded] = useState(true);
    const [expandedMethod, setExpandedMethod] = useState(true);
    const [title, setTitle] = useState("");
    const [image, setImage] =
useState("https://caer.univ-amu.fr/wp-content/uploads/default-placehold
er.png");
    const [ingredients, setIngredients] = useState([]);
    const [calories, setCalories] = useState("");
    const [ingredient, setIngredient] = useState("");
    const [imageFile, setImageFile] = useState(null);
    const [method, setMethod] = useState("");
    //empty the recipe fields
    const emptyFields = () => {
        setTitle("");

setImage("https://caer.univ-amu.fr/wp-content/uploads/default-placehold
er.png");
        setIngredients([]);
        setCalories("");
        setIngredient("");
        setMethod("");
    }

    //add an ingredient to the ingredients list with the plus button
    const addIngredient = () => {
        if (ingredient !== "") {
            setIngredients(ingredients => [...ingredients,
ingredient]);
            setIngredient("");
        }
    }

    //control accordion
    const handleExpandClick = () => {
        setExpanded(!expanded);
    };

    //control accordion
    const handleExpandMethod = () => {
        setExpandedMethod(!expandedMethod);
    };

    //delete an ingredient from the ingredients list
    const deleteIngredient = (index) => {
        const newIngredients = [...ingredients];
        newIngredients.splice(index, 1);
        setIngredients(newIngredients);
    }

    //check if the recipe is valid
    const checkRecipe = () => {
        if (title === "" || image ===
"https://caer.univ-amu.fr/wp-content/uploads/default-placeholder.png"
|| ingredients.length === 0 || calories === "") {
            return false;
        }
        return true;
    }

    //upload an image to the Firebase storage
    const uploadImage = (file) => {
        firebaseServices.uploadFile(file).then(url => {
            setImage(url);
        }).catch(error => {
            console.log(error);
        });
    }

    //update calories state if it's a number
    const updateCalories = e => {
        if(isNaN(e.target.value)) {
            return;
        }
        setCalories(e.target.value);
    }

    //update title state
    const updateTitle = e => {
        setTitle(e.target.value);
    }

    //update image state
    const updateImageFile = e => {
        if(e.target.files[0]) {
            setImageFile(e.target.files[0]);
            uploadImage(e.target.files[0])
        }
    }

    //update ingredient state
    const updateIngredient = e => {
        setIngredient(e.target.value);
    }

    //update method state
    const updateMethod = e => {
        setMethod(e.target.value);
    }
    return {
        expanded,
        expandedMethod,
        title,
        image,
        ingredients,
        calories,
        ingredient,
        method,
        handleExpandClick,
        handleExpandMethod,
        updateTitle,
        updateImageFile,
        updateMethod,
        addIngredient,
        updateCalories,
        updateIngredient,
        checkRecipe,
        deleteIngredient,
        emptyFields
    }
}

export default NewRecipeCardLogic;
```

New Recipe Card Style JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */



//recipe Card styling
import {Button, CardContent, IconButton, InputBase} from
"@mui/material";
import {alpha, styled}                               from
"@mui/material/styles";
import React                                         from "react";

export const ExpandMore = styled((props) => {
    const { expand, ...other } = props;
    return <IconButton {...other} />;
```

```
})
(({ theme, expand }) => ({
    transform: !expand ? 'rotate(0deg)' : 'rotate(180deg)',
    marginLeft: 'auto',
    transition: theme.transitions.create('transform', {
        duration: theme.transitions.duration.shortest,
    }),
}));

export const BootstrapInput = styled(InputBase)(({ theme }) => ({
    'label + &': {
        marginTop: theme.spacing(3),
    },
    '& .MuiInputBase-input': {
        borderRadius: 4,
        position: 'relative',
        backgroundColor: theme.palette.mode === 'light' ? '#fcfcfb' :
'#2b2b2b',
        border: '1px solid #ced4da',
        fontSize: 16,
        width: '100%',
        padding: '10px 10px 10px 10px',
        margin: '5px 0px 5px 0px',
        transition: theme.transitions.create([
            'border-color',
            'background-color',
            'box-shadow',
        ]),
        // Use the system font instead of the default Roboto font.
        fontFamily: [
            '-apple-system',
            'BlinkMacSystemFont',
            '"Segoe UI"',
            'Roboto',
            '"Helvetica Neue"',
            'Arial',
            'sans-serif',
            '"Apple Color Emoji"',
            '"Segoe UI Emoji"',
            '"Segoe UI Symbol"',
        ].join(','),
        '&:focus': {
            //transparent for #FA4616
            boxShadow: `${alpha('#FA4616', 0.25)} 0 0 0 0.2rem`,
            borderColor: '#FA4616',
        },
    },
}));

export const BootstrapButton = styled(Button)({
    boxShadow: 'none',
    textTransform: 'none',
    fontSize: 16,
    padding: '6px 12px',
    border: '1px solid #FA4616',
    width: '100%',
    lineHeight: 1.5,
    color: "#1d1d1d",
    backgroundColor: '#f5f5f5',
    borderColor: `${alpha('#f5f5f5', 0.25)} 0 0 0 0.2rem`,
    justifyContent: 'space-between',
    fontFamily: [
        '-apple-system',
        'BlinkMacSystemFont',
        '"Segoe UI"',
        'Roboto',
        '"Helvetica Neue"',
        'Arial',
        'sans-serif',
        '"Apple Color Emoji"',
        '"Segoe UI Emoji"',
        '"Segoe UI Symbol"',
    ].join(','),
    '&:hover': {
        backgroundColor: '#f5f5f5',
        borderColor: `${alpha('#f5f5f5', 0.25)} 0 0 0 0.2rem`,
        boxShadow: 'none',
    },
    '&:active': {
        boxShadow: 'none',
        backgroundColor: '#f5f5f5',
        borderColor: `${alpha('#f5f5f5', 0.25)} 0 0 0 0.2rem`,
    },
    '&:focus': {
        boxShadow: `${alpha('#f5f5f5', 0.25)} 0 0 0 0.2rem`,
    },
});
```

```
export const CardContentStyled = styled(CardContent) `
  padding: 0;
`;

export const CardStyles = {
    media: {
        paddingTop: '0',
        cursor: 'pointer',
        marginRight: "10px"
    },
    card: {
        position: 'relative',
    },
    overlay: {
        backgroundColor: 'rgba(0,0,0,0.1)',
        position: 'absolute',
        top: '75%',
        left: '50%',
        transform: 'translate(-50%, -50%)',
        color: '#FA4616',
        cursor: 'pointer',
        padding: '2px',
        borderRadius: '50%'
    }
};
```

Profile

Delete Profile Dialog JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */


import {deleteDoc, doc}      from "firebase/firestore";
import {database}            from "../../FirebaseConfig";
import PromptForCredentials from "./PromptForCredentials";
import React, {useState}     from 'react';
import Button                from 'react-bootstrap/Button';
import Dialog                from '@mui/material/Dialog';
import DialogActions         from '@mui/material/DialogActions';
import DialogContent         from '@mui/material/DialogContent';
import DialogContentText     from '@mui/material/DialogContentText';
import DialogTitle            from '@mui/material/DialogTitle';
import useMediaQuery          from '@mui/material/useMediaQuery';
import { useTheme }           from '@mui/material/styles';

import { getAuth, deleteUser, reauthenticateWithCredential,
EmailAuthProvider } from "firebase/auth";
import {useNavigate} from "react-router-dom";
import {useUserAuth} from "../../context/UserAuthContext";

const DeleteProfileDialog = () => {
    const theme = useTheme();
    const fullScreen = useMediaQuery(theme.breakpoints.down('md'));

    const {logOut} = useUserAuth();
    const navigate = useNavigate();
    const auth = getAuth();
    const user = auth.currentUser;

    const [open, setOpen] = useState(false);
    const [password, setPassword] = useState("");
    const [openPrompt, setOpenPrompt] = useState(false);
    const [credentials, setCredentials] = useState({});

    const handleOpenPrompt = async () => {
        setOpenPrompt(true);
    };

    const handleClosePrompt = () => {
        setOpenPrompt(false);
    };

    //Change password
    const handleChangePassword = (event) => {
        setPassword(event.target.value);
        setCredentials({...credentials, password: event.target.value});
    };

    const handleClickOpen = () => {
        setOpen(true);
    };

    const handleClose = () => {
        setOpen(false);
    };
    //Prompt to re authenticate
    const promptForCredentials = () => {
        handleOpenPrompt();
    };

    //Handle confirm button for re authentication with credentials
    const handleConfirm = () => {
        const credential = EmailAuthProvider.credential(
            auth.currentUser.email,
            password
        );
        reauthenticateWithCredential(user, credential).then(async () =>
{
            console.log("User re-authenticated.");
            await deleteDoc(doc(database, "users/" + user?.uid));
            deleteUser(user)
                .then(() => {
                        handleClosePrompt();
                        handleClose();
                        logOut();
                        sessionStorage.clear();
                        navigate("/");
                    })
                    .catch((error) => {
                        console.log("Error removing user: ", error);
                    });
                }).catch((error) => {
                    handleClosePrompt();
                    handleClose();
                    setPassword("");
                    console.log("User re-authentication failed.",
error);
        });
    }

    //Handle delete account
    const handleDeleteAccount = async () => {
        await deleteDoc(doc(database, "users/" + user?.uid));
        deleteUser(user).then(() => {
            handleClose();
            logOut();
            sessionStorage.clear();
            navigate("/");
        }).catch((error) => {
            //Prompt the user to re-provide their sign-in credentials
            promptForCredentials();
        });
    };

    return (
        <div>
            <div className="d-grid">
                <Button variant="danger" onClick={handleClickOpen}>
                    Delete your account
                </Button>
            </div>
            <Dialog
                fullScreen={fullScreen}
                open={open}
                onClose={handleClose}
                aria-labelledby="responsive-dialog-title"
            >
                <DialogTitle id="responsive-dialog-title">
                    {"Are you sure you want to delete your account?"}
                </DialogTitle>
                <DialogContent>
                    <DialogContentText>
                        All your data will be deleted.
                    </DialogContentText>
                </DialogContent>
                <DialogActions>
                    <Button variant="primary" autoFocus
onClick={handleClose}>
                        No
                    </Button>
                    <Button variant="danger"
onClick={handleDeleteAccount}>
                        Yes
                    </Button>
                </DialogActions>
            </Dialog>
            <PromptForCredentials
                openPrompt={openPrompt}
                handleClosePrompt={handleClosePrompt}
                handleChangePassword={handleChangePassword}
                handleConfirm={handleConfirm}
                password={password}
            />
        </div>
    );
}

export default DeleteProfileDialog;
```

Profile JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */


//MUI Icons
import EditIcon
from '@mui/icons-material/Edit';
//MUI material
import {Alert, Avatar, Container, IconButton, Snackbar, Typography}
from "@mui/material";
//Firebase
import React
from "react";
//React Bootstrap
import {Button, Col, Form, Row}
from "react-bootstrap";
import DeleteProfileDialog
from "./DeleteProfileDialog";
//Components
import NavigationBar
from "../NavigationBar/NavigationBar";
//Logic
import ProfileLogic
from "./ProfileLogic";

const Profile = () => {

    const {
        firstname,
        surname,
        email,
        photoURL,
        update,
        openSuccessSnackbar,
        openErrorSnackbar,
        successMessage,
        errorMessage,
        handleSubmit,
        resetPassword,
        updateImageFile,
        numberOfRecipes,
        setUpdate,
        setFirstname,
        setSurname,
        setEmail,
        setOpenSuccessSnackbar,
        setOpenErrorSnackbar,
    } = ProfileLogic();

  return (
    <div>
        <Container className="mt-5">
            <Row>
                <NavigationBar/>
            </Row>
        </Container>
        <Container className="mt-5">
                <Row className="justify-content-center float-right">
                    <Col lg={6} >
                        <Row className="float-end">
                            {/*Swap to update mode*/}
                            <IconButton  onClick={() =>
setUpdate(!update)}>
                                <EditIcon sx={{color:"#FA4616"}}/>
                            </IconButton>
                        </Row>
                    </Col>
                </Row>
                <Row className={"justify-content-center text-center"} >
                    <Col lg={5}>
                            <IconButton>
                                <label htmlFor="image-input">
                                    {/*Change profile picture*/}
                                    <Avatar alt="avatar"
srcSet={photoURL} sx={{padding: 0, width:"150px", height:"150px",
cursor: "pointer"}}/>
                                </label>
                            </IconButton>
                        <input id="image-input" type="file"
onChange={updateImageFile} hidden={true} required={true}/>
                        <Typography >{numberOfRecipes}
recipes</Typography>
                    </Col>
                </Row>
        </Container>
        <Container sx={{marginTop: "50px"}}>
            <Row className={"justify-content-center"}>
                <Col lg={6}>
                    {/*Update profile form*/}
                    <Form onSubmit={handleSubmit} >
                        <Row>
                            <Col lg={6} md={6} sm={6} mf={6}>
                                <Form.Group  className="mb-3"
controlId="formBasicFirstname">
                                    <Form.Label>Firstname</Form.Label>
                                    <Form.Control type="text"
placeholder="Enter firstname" onChange={(e) =>
setFirstname(e.target.value)} value={firstname ?? ""}
disabled={!update}/>
                                </Form.Group>
                            </Col>
                            <Col lg={6} md={6} sm={6} mf={6}>
                                <Form.Group className="mb-3"
controlId="formBasicSurname">
                                    <Form.Label>Surname</Form.Label>
                                    <Form.Control type="text"
placeholder="Enter surname" onChange={(e) =>
setSurname(e.target.value)} value={surname ?? ""} disabled={!update}/>
                                </Form.Group>
                            </Col>
                        </Row>
                            <Form.Group className="mb-3"
controlId="formBasicEmail">
                                <Form.Label>Email address</Form.Label>
                                <Form.Control type="email"
placeholder="Enter email" onChange={(e) => setEmail(e.target.value)}
value={email} readOnly />
                            </Form.Group>

                            <div className="d-grid">
                                <Button className="text-light"
variant="primary" onClick={resetPassword}>Reset password by
email</Button>
                            </div>

                            <div className="d-grid mt-3">
                                <Button className="text-light"
variant="primary" type="submit">Save</Button>
```

```
                            </div>

                            <div className="d-grid mt-3">
                                <DeleteProfileDialog />
                            </div>
                        </Form>
                    </Col>
                </Row>
                </Container>
                {/*Snackbars*/}
                <Snackbar open={openSuccessSnackbar}
autoHideDuration={6000} onClose={() => setOpenSuccessSnackbar(false)}>
                    <Alert onClose={() => setOpenSuccessSnackbar(false)}
severity="success" sx={{ width: '100%' }}>
                        {successMessage}
                    </Alert>
                </Snackbar>
                <Snackbar open={openErrorSnackbar} autoHideDuration={6000}
onClose={() => setOpenErrorSnackbar(false)}>
                    <Alert onClose={() => setOpenErrorSnackbar(false)}
severity="error" sx={{ width: '100%' }}>
                        {errorMessage}
                    </Alert>
                </Snackbar>
        </div>
  );
}

export default Profile;
```

Profile Logic JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */


import {useEffect, useState}                                          from
"react";
//React Router
import {useNavigate}                                                  from
"react-router-dom";
//Firebase
```

```
import {getAuth, sendPasswordResetEmail, updateProfile}         from
"firebase/auth";
import {collection, doc, getDoc, getDocs, query, setDoc, where} from
"firebase/firestore";
import {database}                                               from
"../../FirebaseConfig";
import * as firebaseServices                                    from
"../../services/FirebaseServices";

const ProfileLogic = () => {

    const auth = getAuth();
    const user = JSON.parse(sessionStorage.getItem('user'));
    const navigate = useNavigate();

    const [userData, setUserData] = useState({});
    const [numberOfRecipes, setNumberOfRecipes] = useState(0);
    const [recipes, setRecipes] = useState([]);
    const [photoURL, setPhotoURL] = useState('');
    const [firstname, setFirstname] = useState('');
    const [surname, setSurname] = useState('');
    const [email, setEmail] = useState('');
    const [update, setUpdate] = useState(false);
    const [openSuccessSnackbar, setOpenSuccessSnackbar] =
useState(false);
    const [openErrorSnackbar, setOpenErrorSnackbar] = useState(false);
    const [successMessage, setSuccessMessage] = useState('');
    const [errorMessage, setErrorMessage] = useState('');

    //Get user data from Firebase database
    const getUserData = async () => {
        const docRef = doc(database, "users", user.uid);
        const docSnap = await getDoc(docRef);
        if (docSnap.exists) {
            setUserData(docSnap.data());
            setFirstname(docSnap.data().displayName?.split(' ')[0]);
            setSurname(docSnap.data().displayName?.split(' ')[1]);
            setEmail(docSnap.data().email);
            setPhotoURL(user.photoURL);
        }
    }

    //Fetching recipes from Firebase database
    const getUserRecipes = async () => {
        const q = query(collection(database, "recipes"), where("uid",
"==", user.uid));
        const querySnapshot = await getDocs(q);
        querySnapshot.forEach((doc) => {
            let newRecipe = doc.data();
            //if the id of a recipe is not already in the array, add
the recipe
            newRecipe.id = doc.id;
            if(!recipes.some(recipe => recipe.id === newRecipe.id)){
                setNumberOfRecipes(numberOfRecipes => numberOfRecipes +
1);
                setRecipes(recipes => [...recipes, newRecipe]);
            }
        });
    }

    //Handle form submit
    const handleSubmit = async (e) => {
        e.preventDefault();
        await updateUserData();
        setSuccessMessage("Profile successfuly updated !");
        setOpenSuccessSnackbar(true);

    }

    //Update user data in Firebase database
    const updateUserData = async () => {
        const userRef = doc(database, "users", user.uid);
        await setDoc(userRef, {
            displayName: firstname + ' ' + surname,
            email: email,
            createdAt: new Date().getTime(),
            photoURL: photoURL
        }, {merge: true});
        await updateProfile(auth.currentUser, {
            displayName: firstname + ' ' + surname, photoURL: photoURL
        }).then(() => {
            const user = JSON.parse(sessionStorage.getItem('user'));
            user.photoURL = photoURL;
            sessionStorage.setItem('user', JSON.stringify(user));
            setUpdate(false);
        }).catch((error) => {
            console.log(error);
        });
    }

    //Reset password with an email
    const resetPassword = async () => {
        sendPasswordResetEmail(auth, user.email)
            .then(() => {
                setSuccessMessage("Email sent to " + user.email);
            }).catch(() => {
            setErrorMessage("Mail error");
        });
    }

    //Upload an image to the Firebase storage
    const uploadImage = (file) => {
        firebaseServices.uploadFile(file).then(url => {
            setPhotoURL(url);
        }).catch(error => {
            console.log(error);
        });
    }

    //Update the profile image
    const updateImageFile = e => {
        if(e.target.files[0]) {
            uploadImage(e.target.files[0])
        }
    }

    useEffect(async () => {
        //If the user is not logged in, redirect to the login page
        //To handle the case where the user is not logged in and tries
to access the page
        if(!user){
            navigate("/");
        } else {
            await getUserData();
            await getUserRecipes();
        }
    }, []);

    return {
        userData,
        firstname,
        surname,
        email,
        photoURL,
        update,
        openSuccessSnackbar,
        openErrorSnackbar,
        successMessage,
        errorMessage,
        handleSubmit,
        resetPassword,
        updateImageFile,
        uploadImage,
        updateUserData,
        numberOfRecipes,
        setUpdate,
        setFirstname,
        setSurname,
        setEmail,
        setOpenSuccessSnackbar,
        setOpenErrorSnackbar,
        setSuccessMessage,
    }

}

export default ProfileLogic;
```

Prompt For Credentials JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */


import Button            from '@mui/material/Button';
import TextField         from '@mui/material/TextField';
import Dialog            from '@mui/material/Dialog';
import DialogActions     from '@mui/material/DialogActions';
import DialogContent     from '@mui/material/DialogContent';
import DialogContentText from '@mui/material/DialogContentText';
import DialogTitle       from '@mui/material/DialogTitle';
```

```
const PromptForCredentials = ({password, handleChangePassword,
openPrompt, handleClosePrompt, handleConfirm}) => {

    return (
        <div>
            <Dialog open={openPrompt} onClose={handleClosePrompt}>
                <DialogTitle>Enter your credentials</DialogTitle>
                <DialogContent>
                    <DialogContentText>
                        To delete your account, please enter your
password.
                    </DialogContentText>
                    <TextField
                        autoFocus
                        margin="dense"
                        id="password"
                        label="Password"
                        type="password"
                        fullWidth
                        variant="standard"
                        value={password}
                        onChange={handleChangePassword}
                    />
                </DialogContent>
                <DialogActions>
                    <Button onClick={handleClosePrompt}>Cancel</Button>
                    <Button onClick={handleConfirm}>Confirm</Button>
                </DialogActions>
            </Dialog>
        </div>
    );
}

export default PromptForCredentials;
```

Protected Route

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */
 import React                     from "react";
 //React Router
 import {Navigate, useLocation} from "react-router-dom";
 //React User Context
 import { useUserAuth }          from "../../context/UserAuthContext";

 const ProtectedRoute = ({ children }) => {
    const { user } = useUserAuth();

    //async function to check if user is authenticated
    const isAuthenticated = async () => {
        if(!user){
            return true;
        }
        return false;
    }

    if (!isAuthenticated) {
        return <Navigate to="/" />;
    }
    return children;
};

 export default ProtectedRoute;
```

Recipe

Recipe JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */

import React, {useEffect, useState}
from "react";
import {Add, Done, FileUpload, MoreVert, Remove, Send, Star,
StarBorder} from "@mui/icons-material";
//MUI Icons
import ExpandMoreIcon
from '@mui/icons-material/ExpandMore';
import FavoriteIcon
from '@mui/icons-material/Favorite';
//MUI Material
```

```
import {
    Alert,
    Avatar,
    Box,
    Card,
    CardActionArea,
    CardActions,
    CardContent,
    CardHeader,
    CardMedia,
    Collapse,
    Divider,
    IconButton,
    Menu,
    MenuItem,
    Snackbar,
    Tooltip,
    Typography
} from '@mui/material';
//Firebase
import {arrayUnion, doc, updateDoc, arrayRemove, getDoc} from
"firebase/firestore";
import {database}                                           from
"../../FirebaseConfig";
import * as firebaseServices         from
"../../services/FirebaseServices";
//React Bootstrap
import {Col, Row}                    from "react-bootstrap";
//React Router Dom
import {useNavigate}                 from "react-router-dom";
//CSS
import "../Discover/Discover.css";
//Styles
import { BootstrapButton, BootstrapInput, CardStyles,
CardContentStyled, ExpandMore } from "./RecipeStyles";

//Recipe Component
 const Recipe = ({title, calories, image, method, ingredients, rating,
reviews, uid, removeRecipe, updateRecipe, id, photoURL,
disableAddToCookbook, userCookbooks, disableReview, displayName,
cookBookMode, cookbookId}) => {
     const user = JSON.parse(sessionStorage.getItem('user'));

     const navigate = useNavigate();
     const defaultImage =
"https://caer.univ-amu.fr/wp-content/uploads/default-placeholder.png";
     const [menu, setMenu] = useState(null);
     const [menuCookbook, setMenuCookbook] = useState(null);
     const [menuRemoveFromCookbook, setMenuRemoveFromCookbook] =
useState(null);
     const [messageMenu, setMessageMenu] = useState(null);
     const [update, setUpdate] = useState(false);
     const [expanded, setExpanded] = useState(false);
     const [expandedReviews, setExpandedReviews] = useState(false);
     const [expandedUpdate, setExpandedUpdate] = useState(true);
     const [expandedMethod, setExpandedMethod] = useState(false);
     const [expandedMethodUpdate, setExpandedMethodUpdate] =
useState(true);
     const [expandedUpdateIngredient, setExpandedUpdateIngredient] =
useState(true);
     const [titleState, setTitleState] = useState(title);
     const [imageState, setImageState] = useState(image);
     const [ingredientsState, setIngredientsState] =
useState(ingredients);
     const [caloriesState, setCaloriesState] = useState(calories);
     const [ingredient, setIngredient] = useState("");
     const [imageFile, setImageFile] = useState(null);
     const [methodState, setMethodState] = useState(method);
     const [ratingState, setRatingState] = useState(rating);
     const [review, setReview] = useState("");
     const [reviewsState, setReviewsState] = useState(reviews);
     const [openSuccessSnackbar, setOpenSuccessSnackbar] =
useState(false);
     const [renderRating, setRenderRating] = useState(rating);
     const [photoURLState, setPhotoURLState] = useState(photoURL);

     const settings = ['Modify', 'Remove'];
     const settingsCookbookMode = ['Remove from cookbook'];
     const settingsCookbook = userCookbooks;

     //add an ingredient to the ingredients list with the plus button
     const addIngredient = () => {
         if (ingredient !== "") {
             setIngredientsState(ingredientsState =>
[...ingredientsState, ingredient]);
             setIngredient("");
         }
```

```
    }

    //handle rating change with Firebase
    const handleRate = () => {
        const recipeRef = doc(database, "recipes", id);
        let newRating = 0;
        if (!renderRating) {
            newRating = ratingState;
        } else {
           newRating = (ratingState + renderRating) / 2;
        }
        updateDoc(recipeRef, {
            "rating": newRating
        }).then(() => {
            setRenderRating(newRating);
            setRatingState(newRating);
        }).catch(error => {
            console.log(error);
        });
    };

    //handle rating change with Firebase
    const sendReview = (event) => {
        event.preventDefault();
        if (review === "") return;
        const recipeRef = doc(database, "recipes", id);
        const newReview = {
            "text": review,
            "uid": user.uid,
            "rating": ratingState
        };
        if (user.photoURL) newReview.photoURL = user.photoURL;
        if (user.displayName) newReview.displayName =
user.displayName;

        updateDoc(recipeRef, {
            "reviews": arrayUnion(newReview)
        }).then(() => {
            if (reviewsState) {
                setReviewsState(reviewsState => [...reviewsState,
newReview]);
            } else {
                setReviewsState([newReview]);
            }
            setReview("");
            handleRate();
        }).catch(error => {
            console.log(error);
        });
    };

    //Add a new recipe to the cookbook
    const addRecipeToCookbook = (cookbook) => {
        const recipeRef = doc(database, "users/" + user?.uid +
"/cookbooks/" + cookbook.id);
        let recipe = {}
        if(title) recipe.title = title;
        if(image) recipe.image = image;
        if(ingredients) recipe.ingredients = ingredients;
        if(calories) recipe.calories = calories;
        if(rating) recipe.rating = rating;
        if(reviews) recipe.reviews = reviews;
        if(uid) recipe.uid = uid;
        if(id) recipe.id = id;
        if(photoURL) recipe.photoURL = photoURL;

        //Add a recipe in the recipe array of the cookbook
        updateDoc(recipeRef, {
            recipes: arrayUnion(recipe)
        }).then(() => {
            setOpenSuccessSnackbar(true);
            handleCloseMenuCookbook();
        }).catch(error => {
            console.log(error);
        });
    };

    //Remove a recipe from the cookbook
    const handleRemoveRecipeFromCookbook = () => {
        const recipeRef = doc(database, "users/" + user?.uid +
"/cookbooks/" + cookbookId);
        let recipe = {}
        if(title) recipe.title = title;
        if(image) recipe.image = image;
        if(ingredients) recipe.ingredients = ingredients;
        if(calories) recipe.calories = calories;
        if(rating) recipe.rating = rating;
        if(reviews) recipe.reviews = reviews;
        if(uid) recipe.uid = uid;
        if(id) recipe.id = id;
        if(photoURL) recipe.photoURL = photoURL;

        //Add a recipe in the recipe array of the cookbook
        updateDoc(recipeRef, {
            recipes: arrayRemove(recipe)
        }).then(() => {
            window.location.reload();
        }).catch(error => {
            console.log(error);
        });
    };

    //Refresh the photoURL
    const refreshPhotoURL = () => {
        if (!uid) return;
        const userRef = doc(database, "users/" + uid);
        getDoc(userRef).then(doc => {
            setPhotoURLState(doc.data().photoURL);
        }).catch(error => {
            console.log(error);
        });

        //Refresh the photoURL of the recipe in the reviews
        if (reviewsState) {
            reviewsState.forEach(review => {
                const userRef = doc(database, "users/" + review.uid);
                getDoc(userRef).then(doc => {
                    setReviewsState(reviewsState =>
reviewsState.map(review => {
                        if (review.uid === userRef.id) {
                            review.photoURL = doc.data().photoURL;
                        }
                        return review;
                    }));
                }).catch(error => {
                    console.log(error);
                });
            });
        }

    };
    //control accordion
    const handleExpandClick = () => {
        setExpanded(!expanded);
    };

    //control accordion
    const handleExpandMethod = () => {
        setExpandedMethod(!expandedMethod);
    };

    //control accordion
    const handleExpandUpdateIngredient = () => {
        setExpandedUpdateIngredient(!expandedUpdateIngredient);
    };

    //control accordion for reviews
    const handleExpandClickReviews = () => {
        setExpandedReviews(!expandedReviews);
    };

    //control accordion method update
    const handleExpandMethodUpdate = () => {
        setExpandedMethodUpdate(!expandedMethodUpdate);
    };

    //control accordion for update Card
    const handleExpandUpdateClick = () => {
        setExpandedUpdate(!expandedUpdate);
    };

    //delete an ingredient from the ingredients list
    const deleteIngredient = (index) => {
        const newIngredients = [...ingredientsState];
        newIngredients.splice(index, 1);
        setIngredientsState(newIngredients);
    }

    //check if the recipe is valid
    const checkRecipe = () => {
        return !(titleState === "" || imageState === defaultImage ||
ingredientsState.length === 0 || caloriesState === "" || methodState ==
"");

    }
    //upload an image to the Firebase storage
    const uploadImage = (file) => {
        firebaseServices.uploadFile(file).then(url => {
            setImageState(url);
        }).catch(error => {
            console.log(error);
        });
    }

    //update calories field
    const updateCalories = e => {
        if(isNaN(e.target.value)) {
            return;
        }
        setCaloriesState(e.target.value);
    }

    //update title field
    const updateTitle = e => {
        setTitleState(e.target.value);
    }

    //update image field
    const updateImageFile = e => {
        if(e.target.files[0]) {
            setImageFile(e.target.files[0]);
            uploadImage(e.target.files[0])
        }
    }

    //update method field
    const updateMethod = e => {
        setMethodState(e.target.value);
    }

    //update ingredients field
    const updateIngredient = e => {
        setIngredient(e.target.value);
    }

    //update review field
    const updateReview = e => {
        setReview(e.target.value);
    }

    //handle open actions menu
    const handleOpenMenu = (event) => {
        setMenu(event.currentTarget);
    };

    //handle close actions menu
    const handleCloseMenu = () => {
        setMenu(null);
    };

    //handle open messages menu
    const handleOpenMessageMenu = (event) => {
        if(user.uid !== uid) {
            setMessageMenu(event.currentTarget);
        }
    };

    //handle open menu cookbook
    const handleOpenMenuCookbook = (event) => {
        if(settingsCookbook.length > 0) {
           setMenuCookbook(event.currentTarget);
        }
    };

    //handle close menu cookbook
    const handleCloseMenuCookbook = (event) => {
        setMenuCookbook(null);
    };


    //handle open menu remove from cookbook
    const handleOpenMenuRemoveRecipeFromCookbook = (event) => {
        if(settingsCookbookMode.length > 0) {
            setMenuRemoveFromCookbook(event.currentTarget);
        }
    };

    //handle close menu remove from cookbook
    const handleCloseMenuRemoveRecipeFromCookbook = (event) => {
        setMenuRemoveFromCookbook(null);
    };
```

```
     //handle close all menus
     const handleCloseAllMenus = () => {
         setMenu(null);
         setMenuCookbook(null);
         setMessageMenu(null);
     };

     //handle remove recipe from cookbook
    const handleRemoveRecipe = (event) => {
        removeRecipe(event);
        handleCloseMenu();
    };

    //switch to update mode
    const controlUpdateMenu = () => {
        setUpdate(true);
        handleCloseMenu();
    }

    //handle close success snackbar
     const handleCloseSuccessSnackbar = () => {
         setOpenSuccessSnackbar(false);
     };

    //handle update recipe
    const handleUpdateRecipe = (event) => {
        event.preventDefault();
        updateRecipe(event, {title: titleState, image: imageState,
method: methodState, calories: caloriesState, id}, ingredientsState);
        handleCloseMenu();
        setUpdate(false);
    };

    //send message to the user, redirect to Messages page
    const sendMessage = () => {
        sessionStorage.setItem("interlocutorUID", uid);
        sessionStorage.setItem("interlocutorName", displayName);
        //redirect to Messages page with uid of the recipe in parameter
        navigate("/messages");
    }

    //update the rating state
     const changeRate = (event, newValue) => {
         setRatingState(newValue);
```

```
      }

      useEffect(() => {
          refreshPhotoURL();
      }, []);

    {/*card around each recipe */}
    return(
        <div>
                {/* update card recipe */}
            {update
                ? <Card className="mb-4" component="form">
                    <CardContentStyled>
                        <BootstrapInput multiline className="input-100
recipe-input" placeholder="Recipe" onChange={updateTitle}
required={true} value={titleState}/>
                        <BootstrapInput multiline
className="input-100" placeholder="Calories" onChange={updateCalories}
required={true} value={caloriesState}/>
                    </CardContentStyled>
                    <CardActionArea>
                        {/*Recipe image*/}
                        <label htmlFor="image-input">
                            <CardMedia component="img" height="300"
width="100%" image={imageState} alt="Recipe image"
style={CardStyles.media}/>
                        </label>
                        <label htmlFor="image-input">
                            <FileUpload className="upload-icon"
style={CardStyles.overlay} fontSize="large" color="primary"/>
                        </label>
                        {/*Hidden input to change the image file*/}
                        <input id="image-input" type="file"
onChange={updateImageFile} hidden={true} required={true}/>
                    </CardActionArea>
                    <CardActions disableSpacing>
                        <div>Ingredients</div>
                        <ExpandMore expand={expandedUpdateIngredient}
onClick={handleExpandUpdateIngredient}
aria-expanded={expandedUpdateIngredient} aria-label="show more">
                            <ExpandMoreIcon />
                        </ExpandMore>
                    </CardActions>
```

```
                     <Collapse in={expandedUpdateIngredient}
timeout="auto" unmountOnExit>
                         <CardContentStyled>
                             {/*List of ingredients*/}
                             {ingredientsState &&
ingredientsState.map((ingredient, index) =>(
                                 <BootstrapButton
                                     key={index}
                                     variant="contained"
                                     disableElevation
                                     aria-expanded={expanded}
                                     aria-label="show more"
                                     onClick={() =>
deleteIngredient(index)}
                                     endIcon={<Remove
color={"error"}/>}
                                 >
                                     {ingredient}
                                 </BootstrapButton>
                             ))}
                             <BootstrapInput multiline
className="input-100 recipe-input" placeholder="Ingredient"
value={ingredient} onChange={updateIngredient} />
                             {/*Add ingredient button*/}
                             <BootstrapButton
                                 variant="contained"
                                 disableElevation
                                 onClick={addIngredient}
                                 endIcon={<Add color={"success"}/>}
                             >
                                 Add an ingredient
                             </BootstrapButton>
                         </CardContentStyled>
                     </Collapse>
                     {/*Update recipe method*/}
                     <CardActions disableSpacing
expand={expandedMethodUpdate.toString()}
onClick={handleExpandMethodUpdate} aria-expanded={expandedMethodUpdate}
aria-label="show more" sx={{cursor:"pointer"}}>
                         <div>Method</div>
                         <ExpandMore>
                             <ExpandMoreIcon />
                         </ExpandMore>
                     </CardActions>
                     <Collapse in={expandedMethodUpdate} timeout="auto"
unmountOnExit>
                         <CardContent>
                             <BootstrapInput multiline
className="input-100 recipe-input" placeholder="Method"
onChange={updateMethod} required={true} value={methodState}/>
                         </CardContent>
                     </Collapse>
                     <CardContentStyled sx={{paddingBottom: '0'}}>
                         {/*Modify button*/}
                         <BootstrapButton
                             color="secondary"
                             variant="contained"
                             disableElevation
                             disabled={!checkRecipe()}
                             onClick={handleUpdateRecipe}
                             type="submit"
                             endIcon={<Done color={checkRecipe() ?
"success" : "disabled"}/>}
                         >
                             Modify
                         </BootstrapButton>
                     </CardContentStyled>
                 </Card>
            // Display card recipe
            : <Card className="mb-4 p-0" component="form">
                     {/*If it's a user recipe, display the user's
avatar*/}
                <CardHeader
                    avatar={
                        <>
                            {uid
                                ? <Avatar src={photoURLState}
alt={"photo"} onClick={handleOpenMessageMenu} sx={{cursor:"pointer"}}/>
                                : <Col sx={{width: 0, margin: 0,
padding: 0}}></Col>
                            }
                            {/*Send a message menu*/}
                            <Menu
                                sx={{ mt: '45px' }}
                                id="menu-appbar"
                                anchorEl={messageMenu}
                                anchorOrigin={{
                                    vertical: 'top',
                                    horizontal: 'right',
                                }}
                                transformOrigin={{
                                    vertical: 'top',
                                    horizontal: 'right',
                                }}
                                open={Boolean(messageMenu)}
                                onClose={handleCloseAllMenus}
                                onClick={handleCloseAllMenus}
                            >
                                {["Send a message"].map((setting,
index) => (
                                    <MenuItem key={index}
onClick={sendMessage}>
                                        <Typography
textAlign="center">{setting}</Typography>
                                    </MenuItem>
                                ))}
                            </Menu>
                        </>
                    }
                    title={
                        <Typography>{titleState}</Typography>
                    }
                    action = {
                            <>
                                {/*If it's the current user recipe,
display modification and remove button in menu*/}
                            {uid === user?.uid &&
                                // Menu to update or remove a recipe
                                <Box sx={{ flexGrow: 0 }}>
                                    <Tooltip title="Open settings">
                                        <IconButton
aria-label="settings" onClick={handleOpenMenu}>
                                            <MoreVert />
                                        </IconButton>
                                    </Tooltip>
                                    {/*Modify and remove menu*/}
                                    <Menu
                                        sx={{ mt: '45px' }}
                                        id="menu-appbar"
                                        anchorEl={menu}
                                        anchorOrigin={{
                                            vertical: 'top',
                                            horizontal: 'right',
                                        }}
                                        transformOrigin={{
                                            vertical: 'top',
                                            horizontal: 'right',
                                        }}
                                        open={Boolean(menu)}
                                        onClose={handleCloseMenu}
                                        onClick={handleCloseMenu}
                                    >
                                        {settings.map((setting, index)
=> (
                                            <MenuItem key={index}
onClick={setting === "Remove" ? handleRemoveRecipe : () =>
controlUpdateMenu()}>
                                                <Typography
textAlign="center">{setting}</Typography>
                                            </MenuItem>
                                        ))}
                                    </Menu>
                                </Box>
                            }
                            {/*If it's not the current user recipe,
display the "add to cookbook" button*/}
                            {
                                uid !== user?.uid &&
!disableAddToCookbook &&
                                <>
                                    <Tooltip
title={settingsCookbook?.length > 0 ? "Add to a Cookbook" : "You have
no Cookbook yet"}>
                                        <IconButton
aria-label={settingsCookbook?.length > 0 ? "Add to a Cookbook" : "You
have no Cookbook yet"} onClick={handleOpenMenuCookbook}>
                                            <FavoriteIcon sx={{color:
"#FA4616"}}/>
                                        </IconButton>
                                    </Tooltip>
                                    {/*Menu to add a recipe to a
cookbook*/}
                                    <Menu
                                        sx={{ mt: '45px' }}
                                        id="menu-appbar"
                                        anchorEl={menuCookbook}
                                        anchorOrigin={{
                                            vertical: 'top',
                                            horizontal: 'right',
                                        }}
                                        keepMounted
                                        transformOrigin={{
                                            vertical: 'top',
                                            horizontal: 'right',
                                        }}
                                        open={Boolean(menuCookbook)}

onClose={handleCloseMenuCookbook}
                                        >

{settingsCookbook?.map((setting, index) => (
                                            <MenuItem key={index}
onClick={() => addRecipeToCookbook(setting)}>
                                                <Typography
textAlign="center">{setting.name}</Typography>
                                            </MenuItem>
                                        ))}
                                    </Menu>
                                </>
                            }
                                {/*Enable to delete a recipe from the
current cookbook*/}
                                {
                                    cookBookMode &&
                                    <>
                                        <Box sx={{ flexGrow: 0 }}>
                                            <Tooltip title="Open
settings">
                                                <IconButton
aria-label="settings" onClick={handleOpenMenuRemoveRecipeFromCookbook}>
                                                    <MoreVert />
                                                </IconButton>
                                            </Tooltip>
                                            {/*Modify and remove
menu*/}
                                            <Menu
                                                sx={{ mt: '45px' }}
                                                id="menu-appbar"

anchorEl={menuRemoveFromCookbook}
```

```
                                                        anchorOrigin={{
                                                            vertical: 'top',
                                                            horizontal:
'right',
                                                        }}
                                                        transformOrigin={{
                                                            vertical: 'top',
                                                            horizontal:
'right',
                                                        }}

open={Boolean(menuRemoveFromCookbook)}

onClose={handleCloseMenuRemoveRecipeFromCookbook}

onClick={handleCloseMenuRemoveRecipeFromCookbook}
                                                    >

{settingsCookbookMode.map((setting, index) => (
                                                            <MenuItem
key={index} onClick={setting === "Remove from cookbook" ?
handleRemoveRecipeFromCookbook : null}>
                                                                <Typography
textAlign="center">{setting}</Typography>
                                                            </MenuItem>
                                                        ))}
                                                    </Menu>
                                                </Box>
                                            </>
                                        }
                                    </>
                                }
                        //recipe calories
                        subheader={`${parseFloat(calories).toFixed(2)}
kcal`}/>
                        {/*Display rating if it's a user recipe*/}
                        {uid &&
                            //Display rating if it's a user recipe
                            <CardContent sx={{padding: '5px'}}>
                                <Box sx={{flexGrow: 0}}
className={"stars"}>
                                    {
                                        Array(5).fill(5).map((_, index) =>
(
                                            <IconButton
                                                key={index}
                                                aria-label="star"
                                                disabled
                                            >
                                                {
                                                    index < renderRating
                                                        ? <Star
sx={{color:"orange"}} fontSize={"small"}/>
                                                        : <StarBorder
sx={{color:"orange"}}  fontSize={"small"}/>
                                                }
                                            </IconButton>
                                        ))
                                    }
                                </Box>
                            </CardContent>
                        }
                <CardMedia component="img" height="300" image={image}
alt=""/>
                {/*Display ingredients*/}
                <CardActions disableSpacing
expand={expanded.toString()} onClick={handleExpandClick}
aria-expanded={expanded} aria-label="show more"
sx={{cursor:"pointer"}}>
                    <div>Ingredients</div>
                    <ExpandMore>
                        <ExpandMoreIcon />
                    </ExpandMore>
                </CardActions>
                <Collapse in={expanded} timeout="auto" unmountOnExit>
                    <CardContent>
                        {/*List of ingredients*/}
                        {ingredients && ingredients.map((ingredient,
index) =>(
                            <div key={index}>
                                <hr/>
                                <Typography paragraph>{ingredient}
</Typography>
                            </div>
                        ))}
                    </CardContent>
                </Collapse>
```

```
                      {/*Display reviews collapse if it's a user
recipe*/}
                      { uid &&
                          <>
                              <CardActions disableSpacing
expand={expandedMethod.toString()} onClick={handleExpandMethod}
aria-expanded={expandedMethod} aria-label="show more"
sx={{cursor:"pointer"}}>
                                  <div>Method</div>
                                  <ExpandMore>
                                      <ExpandMoreIcon />
                                  </ExpandMore>
                              </CardActions>
                              <Collapse in={expandedMethod}
timeout="auto" unmountOnExit>
                                  <CardContent>
                                      <Typography
paragraph>{method}</Typography>
                                  </CardContent>
                              </Collapse>
                              <CardActions disableSpacing
expand={expandedReviews.toString()} onClick={handleExpandClickReviews}
aria-expanded={expandedReviews} aria-label="show more" sx={{cursor:
"pointer"}}>
                                  <div>Reviews</div>
                                  <ExpandMore>
                                      <ExpandMoreIcon />
                                  </ExpandMore>
                             </CardActions>
                              {/*Reviews of the recipe*/}
                              <Collapse in={expandedReviews}
timeout="auto" unmountOnExit>
                                  <CardContent>
                                      {!disableReview &&
                                          <>
                                              <Row
className={"justify-content-start align-items-center"}>
                                                  <Col lg={3}
md={"auto"} sx={{padding: "0px"}}>
                                                      <Avatar
src={user?.photoURL} alt="photo"/>
                                                  </Col>
                                                  <Col lg={"auto"}
md={"auto"} sx={{paddingLeft: 0}}
```

```
className={"justify-content-start align-items-center"}>
                                                <Typography
sx={{margin: 0, paddingLeft: "8px"}}
paragraph>{user?.displayName} </Typography>
                                                {

Array(5).fill(5).map((_, index) => (
<IconButton
key={index}
aria-label="star"
onClick={(event) => changeRate(index + 1, index + 1)}
                                                        >
                                                            {

index < ratingState
? <Star sx={{color: "orange"}}
fontSize={"small"}/>
: <StarBorder sx={{color: "orange"}}
fontSize={"small"}/>
                                                            }

</IconButton>
                                                    ))
                                                }
                                            </Col>
                                        </Row>
                                        <BootstrapInput
                                            multiline
                                            className="input-100
recipe-input"
                                            placeholder="Review"

onChange={updateReview}
```

```
                                                  required={true}
value={review}
                                              />
                                              <BootstrapButton
                                                  variant="contained"
                                                  type="submit"
                                                  disableElevation
                                                  aria-label="show more"
                                                  onClick={sendReview}
                                                  endIcon={<Send
sx={{color: "#FA4616"}}/>}
                                                  sx={{marginBottom:
"10px"}}
                                              >
                                                  Leave a review
                                              </BootstrapButton>
                                          </>
                                      }
                                      {/*Display user reviews */}
                                     {reviewsState?.map((reviewElement,
index) =>(
                                          <div key={index}>
                                              <Divider/>
                                              <Row
className={"justify-content-start align-items-center mt-2"}>
                                                  <Col lg={3}
md={"auto"} sx={{padding: "0px"}}>
                                                      <Avatar
src={reviewElement.photoURL} alt="photo"/>
                                                  </Col>
                                                  <Col lg={"auto"}
md={"auto"} sx={{paddingLeft: 0}} className={"justify-content-start
align-items-center"} >
                                                      <Typography
sx={{margin: 0, paddingLeft: "8px"}}
paragraph>{reviewElement.displayName}</Typography>
                                                      {/*Display rating
stars*/}
                                                      {

Array(5).fill(5).map((_, number) => (

<IconButton
key={number}

aria-label="star"

disabled

onClick={(event) => handleRate(number + 1, number + 1)}
                                                                         >
                                                                             {

number < reviewElement.rating

? <Star sx={{color:"orange"}} fontSize={"small"}/>

: <StarBorder sx={{color:"orange"}}  fontSize={"small"}/>
                                                                             }

</IconButton>
                                                                    ))
                                                               }
                                                          </Col>
                                                      </Row>
                                                     <Typography
paragraph>{reviewElement.text} </Typography>
                                                  </div>
                                              ))}
                                      </CardContent>
                                 </Collapse>
                              </>
                          }
            </Card>
             }
             {/*Snackbars*/}
             <Snackbar open={openSuccessSnackbar}
autoHideDuration={6000} onClose={handleCloseSuccessSnackbar}>
                 <Alert onClose={handleCloseSuccessSnackbar}
severity="success" sx={{ width: '100%' }}>
                     Recipe successfuly added to your cookbook !
                 </Alert>
             </Snackbar>
         </div>
     );
 }
```

```
export default Recipe;
```

Recipe Logic JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */

import React, { useState, useEffect } from 'react';
//User Auth context
import {useUserAuth}                    from
"../../context/UserAuthContext";
//Firebase
import {updateDoc, doc }                from "firebase/firestore";
import {database}                       from "../../FirebaseConfig"
import {uploadFile}                     from
"../services/FirebaseServices";
//CSS
import "../Discover/Discover.css";


//Recipe Logic Component
 const RecipeLogic = ({title, calories, image, method, ingredients,
rating, reviews, uid, removeRecipe, updateRecipe, id}) => {
     const { user } = useUserAuth();

     const settings = ['Modify', 'Remove'];
     const [menu, setMenu] = useState(null);
     const [update, setUpdate] = useState(false);
     const [expanded, setExpanded] = useState(false);
     const [expandedReviews, setExpandedReviews] = useState(false);
     const [expandedUpdate, setExpandedUpdate] = useState(true);
     const [titleState, setTitleState] = useState(title);
     const [imageState, setImageState] = useState(image);
     const [methodState, setMethodState] = useState(method);
     const [ingredientsState, setIngredientsState] =
useState(ingredients);
     const [caloriesState, setCaloriesState] = useState(calories);
     const [ingredient, setIngredient] = useState("");
     const [imageFile, setImageFile] = useState(null);
     const [ratingState, setRatingState] = useState(rating);
```

```
    const [review, setReview] = useState("");
    const [reviewsState, setReviewsState] = useState(reviews);

    //add an ingredient to the ingredients list with the plus button
    const addIngredient = () => {
        if (ingredient !== "") {
            setIngredientsState(ingredientsState =>
[...ingredientsState, ingredient]);
            setIngredient("");
        }
    }

    //add an ingredient to the ingredients list with key enter
    const handleKeyDownIngredient = (event) => {
        if (event.key === 'Enter') {
            addIngredient();
        }
    }

    //handle rating change with Firebase
    const handleRate = (event, newValue) => {
        const recipeRef = doc(database, "recipes", id);
        updateDoc(recipeRef, {
            "rating": newValue
        }).then(() => {
            setRatingState(newValue);
        }).catch(error => {
            console.log(error);
        });
    };

    //handle rating change with Firebase
    const sendReview = (event) => {
        event.preventDefault();
        if (review === "") {
            return;
        }
        const recipeRef = doc(database, "recipes", id);
        let reviewsArray = reviews ? reviews : [];
        reviewsArray.push({
            "text": review,
            "displayName": user.displayName,
            "photoURL": user.photoURL,
            "uid": user.uid,
            "rating": ratingState
        });
        updateDoc(recipeRef, {
            "reviews": reviewsArray
        }).then(() => {
            setReviewsState(reviewsArray);
            setReview("");
        }).catch(error => {
            console.log(error);
        });
    };

    //control accordion
    const handleExpandClick = () => {
        setExpanded(!expanded);
    };

    //control accordion for reviews
    const handleExpandClickReviews = () => {
        setExpandedReviews(!expandedReviews);
    };

    //control accordion for update Card
    const handleExpandUpdateClick = () => {
        setExpandedUpdate(!expandedUpdate);
    };

    //delete an ingredient from the ingredients list
    const deleteIngredient = (index) => {
        const newIngredients = [...ingredientsState];
        newIngredients.splice(index, 1);
        setIngredientsState(newIngredients);
    }

    //check if the recipe is valid
    const checkRecipe = () => {
        if (titleState === "" || imageState ===
"https://caer.univ-amu.fr/wp-content/uploads/default-placeholder.png"
|| ingredientsState.length === 0 || caloriesState === "") {
            return false;
        }
        return true;
    }
    //upload an image to the Firebase storage
    const uploadImage = (file) => {
        uploadFile(file).then(url => {
            setImageState(url);
        }).catch(error => {
            console.log(error);
        });
    }

    const updateCalories = e => {
        if(isNaN(e.target.value)) {
            return;
        }
        setCaloriesState(e.target.value);
    }

    const updateTitle = e => {
        setTitleState(e.target.value);
    }

    const updateImageFile = e => {
        if(e.target.files[0]) {
            setImageFile(e.target.files[0]);
            uploadImage(e.target.files[0])
        }
    }

    const updateMethod = e => {
        setMethodState(e.target.value);
    }

    const updateIngredient = e => {
        setIngredient(e.target.value);
    }

    const updateReview = e => {
        setReview(e.target.value);
    }


   const handleOpenMenu = (event) => {
       setMenu(event.currentTarget);
   };
```

```
     const handleCloseMenu = () => {
         setMenu(null);
     };

    const handleRemoveRecipe = (event) => {
        removeRecipe(event);
        handleCloseMenu();
    };

    const handleUpdateRecipe = (event) => {
        event.preventDefault();
        updateRecipe(event, {title: titleState, image: imageState,
method: methodState, calories: caloriesState, id}, ingredientsState);
        handleCloseMenu();
        // emptyFields();
        setUpdate(false);
    };

    return {
        menu,
        update,
        expanded,
        expandedReviews,
        expandedUpdate,
        titleState,
        imageState,
        methodState,
        ingredientsState,
        caloriesState,
        ingredient,
        imageFile,
        ratingState,
        review,
        reviewsState,
        uid,
        handleExpandClick,
        handleExpandClickReviews,
        handleExpandUpdateClick,
        deleteIngredient,
        checkRecipe,
        updateCalories,
        updateTitle,
        updateImageFile,
        updateMethod,
        updateIngredient,
        updateReview,
        handleOpenMenu,
        handleCloseMenu,
        handleRemoveRecipe,
        handleUpdateRecipe,
        sendReview,
        handleRate,
        user,
        settings
    }
}

export default RecipeLogic;
```

Recipe Style JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */


import {Button, CardContent, IconButton, InputBase} from
"@mui/material";
import {alpha, styled}                            from
"@mui/material/styles";
import React                                      from "react";

export const ExpandMore = styled((props) => {
    const { expand, ...other } = props;
    return <IconButton {...other} />;
})
(({ theme, expand }) => ({
    transform: !expand ? 'rotate(0deg)' : 'rotate(180deg)',
    marginLeft: 'auto',
    transition: theme.transitions.create('transform', {
        duration: theme.transitions.duration.shortest,
    }),
}));


export const BootstrapInput = styled(InputBase)(({ theme }) => ({
    'label + &': {
        marginTop: theme.spacing(3),
    },
    '& .MuiInputBase-input': {
        borderRadius: 4,
        multiline: "true",
        position: 'relative',
        backgroundColor: theme.palette.mode === 'light' ? '#fcfcfb' :
'#2b2b2b',
        border: '1px solid #ced4da',
        fontSize: 16,
        width: '100%',
        padding: '10px 10px 10px 10px',
        margin: '5px 0px 5px 0px',
        transition: theme.transitions.create([
            'border-color',
            'background-color',
            'box-shadow',
        ]),
        // Use the system font instead of the default Roboto font.
        fontFamily: [
            '-apple-system',
            'BlinkMacSystemFont',
            '"Segoe UI"',
            'Roboto',
            '"Helvetica Neue"',
            'Arial',
            'sans-serif',
            '"Apple Color Emoji"',
            '"Segoe UI Emoji"',
            '"Segoe UI Symbol"',
        ].join(','),
        '&:focus': {
            //transparent for #FA4616
            boxShadow: `${alpha('#FA4616', 0.25)} 0 0 0 0.2rem`,
            borderColor: '#FA4616',
        },
    },
}));

export const BootstrapButton = styled(Button)({
    boxShadow: 'none',
    textTransform: 'none',
    fontSize: 16,
    padding: '6px 12px',
    border: '1px solid #FA4616',
    width: '100%',
    lineHeight: 1.5,
    color: "#1d1d1d",
    backgroundColor: '#f5f5f5',
    borderColor: `${alpha('#f5f5f5', 0.25)} 0 0 0 0.2rem`,
    justifyContent: 'space-between',
    fontFamily: [
        '-apple-system',
        'BlinkMacSystemFont',
        '"Segoe UI"',
        'Roboto',
        '"Helvetica Neue"',
        'Arial',
        'sans-serif',
        '"Apple Color Emoji"',
        '"Segoe UI Emoji"',
        '"Segoe UI Symbol"',
    ].join(','),
    '&:hover': {
        backgroundColor: '#f5f5f5',
        borderColor: `${alpha('#f5f5f5', 0.25)} 0 0 0 0.2rem`,
        boxShadow: 'none',
    },
    '&:active': {
        boxShadow: 'none',
        backgroundColor: '#f5f5f5',
        borderColor: `${alpha('#f5f5f5', 0.25)} 0 0 0 0.2rem`,
    },
    '&:focus': {
        boxShadow: `${alpha('#f5f5f5', 0.25)} 0 0 0 0.2rem`,
    },
});

export const CardContentStyled = styled(CardContent) `
  padding: 0;
`;

export const CardStyles = {
    media: {
        paddingTop: '0',
        cursor: 'pointer',
        marginRight: "10px"
    },
    card: {
        position: 'relative',
        justifyContent: 'center',
    },
    overlay: {
        backgroundColor: 'rgba(0,0,0,0.1)',
        position: 'absolute',
        top: '75%',
        left: '50%',
        transform: 'translate(-50%, -50%)',
        color: '#FA4616',
        cursor: 'pointer',
        padding: '2px',
        borderRadius: '50%'
    }
};
```

Recipes

Recipes JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */

//MUI Icons
import {Add, Search}
from "@mui/icons-material";
//MUI Material
import {Alert, Card, CardActions, Collapse, IconButton, InputBase,
Paper, Snackbar,} from "@mui/material";
//React Material Styles
import React
from "react";
//React Bootstrap
import {Col, Container, Row}
from "react-bootstrap";
//CSS
import "../Discover/Discover.css";
//Header bar
import NavigationBar
from "../NavigationBar/NavigationBar";
import NewRecipeCard
from "../NewRecipeCard/NewRecipeCard";
```

```
import Recipe
from "../Recipe/Recipe";
import "./Recipes.css";
//Logic Import
import RecipesLogic
from "./RecipesLogic";
//Styles
import {CardStyles, BootstrapButton} from "./RecipesStyles";

//Recipes Component
const Recipes = () => {

    const {
        recipes,
        userCookbooks,
        search,
        updateSearch,
        removeRecipe,
        updateRecipe,
        openSuccessSnackbar,
        openModifySnackbar,
        openRemoveSnackbar,
        handleCloseSuccessSnackbar,
        handleCloseModifySnackbar,
        handleCloseRemoveSnackbar,
        addRecipe,
        expanded,
        handleExpandClick,
    } = RecipesLogic();

    return(
        <>
            {/*Header*/}
            <Container className="mt-5">
                <Row>
                    <NavigationBar/>
                </Row>
            </Container>

            <Container className="mt-4">
                <Row className={"justify-content-center w-100 mr-0"}>
                    <Col className="mt-4" lg={5} mf={7} sm={7} >
                        {/*Search Input*/}
                        <Paper className="mb-4" component="form"
sx={{display: 'flex', alignItems: 'center', maxWidth: '100%' }}   >
                            <InputBase sx={{ ml: 1, flex: 1 }}
placeholder="Search recipe or ingredient" value={search ?? ""}
onChange={updateSearch} />
                            <IconButton sx={{ p: '10px' }}
aria-label="search">
                                <Search className="searchIcon"/>
                            </IconButton>
                        </Paper>
                        {/*Add a new recipe button*/}
                        <Card className="mb-4" sx={{ boxShadow:
'none'}} style={CardStyles.card}>
                            <CardActions disableSpacing
sx={{justifyContent: "center", cursor: "pointer"}}
onClick={handleExpandClick}>
                                <BootstrapButton
                                    variant="contained"
                                    disableElevation
                                    aria-expanded={expanded}
                                    aria-label="show more"
                                    endIcon={<Add />}
                                >
                                    Add a new recipe
                                </BootstrapButton>
                            </CardActions>
                            <Collapse in={expanded} timeout="auto"
unmountOnExit >
                                <Row
className={"justify-content-center"}>
                                    <Col>
                                        <NewRecipeCard
addRecipe={addRecipe} />
                                    </Col>
                                </Row>
                            </Collapse>
                        </Card>
                        {/*List of recipes*/}
                        {recipes && recipes.map(recipeElement =>(
                            <Recipe
                                key={recipeElement.id}
                                title={recipeElement.title}
                                calories={recipeElement.calories}
                                image={recipeElement.image}
                                method={recipeElement.method}

ingredients={recipeElement.ingredients.map(ingredient =>
ingredient.text)}
                                rating={recipeElement.rating}
                                reviews={recipeElement.reviews}
                                uid={recipeElement.uid}
                                id={recipeElement.id}
                                photoURL={recipeElement.photoURL}
                                displayName={recipeElement.displayName}
                                userCookbooks={userCookbooks}
                                removeRecipe={(e) => removeRecipe(e,
recipeElement.id)}
                                updateRecipe={updateRecipe}
                            />
                        ))}
                    </Col>
                </Row>
            </Container>

            {/*Snackbars*/}
            <Snackbar open={openSuccessSnackbar}
autoHideDuration={6000} onClose={handleCloseSuccessSnackbar}>
                <Alert onClose={handleCloseSuccessSnackbar}
severity="success" sx={{ width: '100%' }}>
                    Recipe successfuly added !
                </Alert>
            </Snackbar>
            <Snackbar open={openModifySnackbar} autoHideDuration={6000}
onClose={handleCloseModifySnackbar}>
                <Alert onClose={handleCloseModifySnackbar}
severity="success" sx={{ width: '100%' }}>
                    Recipe successfuly modified !
                </Alert>
            </Snackbar>
            <Snackbar open={openRemoveSnackbar} autoHideDuration={6000}
onClose={handleCloseRemoveSnackbar}>
                <Alert onClose={handleCloseRemoveSnackbar}
severity="success" sx={{ width: '100%' }}>
                    Recipe successfuly removed !
                </Alert>
            </Snackbar>
        </>
```

```
    );
};

export default Recipes;
```

Recipes Logic JS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */

import React, {useEffect, useState}                              from
"react";
//React Router
import {useNavigate}                                             from
"react-router-dom";
//Firebase
import {database}
from "../../FirebaseConfig";
import {addDoc, updateDoc, collection, deleteDoc, doc, getDocs, getDoc,
orderBy} from "firebase/firestore";
//CSS
import "../Discover/Discover.css";

//Recipes Logic
const RecipesLogic = () => {
    const navigate = useNavigate();

    const user = JSON.parse(sessionStorage.getItem("user"));
    //Accordion controller
    const [expanded, setExpanded] = useState(false);
    const [openSuccessSnackbar, setOpenSuccessSnackbar] =
useState(false);
    const [openModifySnackbar, setOpenModifySnackbar] =
useState(false);
    const [openRemoveSnackbar, setOpenRemoveSnackbar] =
useState(false);
    const [recipes, setRecipes ] = useState ([]);
    const [search, setSearch] = useState("");
    const [userCookbooks, setUserCookbooks] = useState([]);
    const [firstLoad, setFirstLoad] = useState(true);

    useEffect(async () =>{
        //If the user is not logged in, redirect to the login page
        //To handle the case where the user is not logged in and tries
to access the page
        if(!user){
            navigate("/");
        } else {
            await getRecipes();
            if(search !== ""){
                filterRecipes();
            } else {
                if (firstLoad) {
                    setFirstLoad(false);
                } else {
                    window.location.reload();
                }
            }
            await getUserCookbooks();
            if (sessionStorage.getItem("added")) {
                setOpenSuccessSnackbar(true);
                sessionStorage.removeItem("added");
            }
        }
    }, [search]);

    //Fetching random recipes from Firebase database
    const getRecipes = async () => {
        const querySnapshot = await getDocs(collection(database,
"recipes"));

        querySnapshot.forEach((doc) => {
            let newRecipe = doc.data();
            //if the id of a recipe is not already in the array, add
the recipe
            newRecipe.id = doc.id;
            if(!recipes.some(recipe => recipe.id === newRecipe.id)){
                setRecipes(recipes => [...recipes, newRecipe]);
            } else {
                //if the id of a recipe is already in the array, update
the recipe
                const index = recipes.findIndex(recipe => recipe.id ===
newRecipe.id);
                const newRecipes = [...recipes];
                newRecipes[index] = newRecipe;
                setRecipes(newRecipes);
            }
        });
    }


    //Get the user's cookbooks from Firebase database
    const getUserCookbooks = async () => {
        const querySnapshot = await getDocs(collection(database,
"users/" + user?.uid + "/cookbooks"));
        querySnapshot.forEach((doc) => {
            let newCookbook = doc.data();
            newCookbook.id = doc.id;
            if(!userCookbooks.some(cookbook => cookbook.id ===
newCookbook.id)){
                setUserCookbooks(cookbooks => [...cookbooks,
newCookbook]);
            } else {
                let newUserCookBooks = userCookbooks;
                const index = newUserCookBooks.findIndex(cookbook =>
cookbook.id === newCookbook.id);
                newUserCookBooks[index] = newCookbook;
                setUserCookbooks(newUserCookBooks);
            }
        });
    }

    //Filter recipes by title and ingredients
    const filterRecipes = () => {
        getRecipes().then(() => {
            const filteredRecipesByTitle = recipes.filter(recipe =>
recipe?.title.toLowerCase().includes(search.toLowerCase()));
            const filteredRecipesByIngredients = recipes.filter(recipe
=> recipe?.ingredients.some(ingredient =>
ingredient.text.toLowerCase().includes(search.toLowerCase())));
            //merge the two arrays without same id
            const filteredRecipes = [...filteredRecipesByTitle,
...filteredRecipesByIngredients].filter((recipe, index, self) =>
                index === self.findIndex((t) => (
                    t.id === recipe.id
                ))
            );
            setRecipes(filteredRecipes);
        })
    }

    //add recipe to Firebase database
    const addRecipe = async (event, recipe, ingredients) => {
        //Get user ref
        const userRef = doc(database, "users/" + user?.uid);
        const userSnapshot = await getDoc(userRef);
        const userData = userSnapshot.data();
        const displayName = userData.displayName;
        event.preventDefault();
        recipe.ingredients = [];
        recipe.uid = user.uid;
        recipe.displayName = displayName;
        if (user.photoURL) recipe.photoURL = user.photoURL;
        ingredients.map(ingredient => {
            recipe.ingredients.push({text: ingredient});
        });
        addDoc(collection(database, "recipes"), recipe).then(() => {
            //add recipe to recipes
            setOpenSuccessSnackbar(true);
            setExpanded(false);
            sessionStorage.setItem("added", true);
            window.location.reload();
        }).catch(() => {
            console.log("Error adding recipe");
        });
    }

    //remove recipes by ID from Firebase database
    const removeRecipe = async (e, id) => {
        e.preventDefault();
        deleteDoc(doc(database, "recipes", id))
            .then(() => {
                //remove recipe from recipes
                setRecipes(recipes => recipes.filter(recipe =>
recipe.id !== id));
                setOpenRemoveSnackbar(true);
            }).catch(() => {
                console.log("Error removing recipe");
            });
    }

    //update recipe in Firebase database
    const updateRecipe = async (e, recipe, ingredients) => {
        e.preventDefault();
        recipe.ingredients = [];
        ingredients.map(ingredient => {
            recipe.ingredients.push({text: ingredient});
        });
        updateDoc(doc(database, "recipes", recipe.id), recipe).then(()
=> {
            getRecipes();
            setOpenModifySnackbar(true);
        }).catch(() => {
            console.log("Error updating recipe");
        });
    }

    const handleExpandClick = () => {
        setExpanded(!expanded);
    };

    const handleOpenSuccessSnackbar = () => {
        setOpenSuccessSnackbar(true);
    };

    const handleCloseSuccessSnackbar = () => {
        setOpenSuccessSnackbar(false);
    };

    const handleOpenModifySnackbar = () => {
        setOpenModifySnackbar(true);
    };

    const handleCloseModifySnackbar = () => {
        setOpenModifySnackbar(false);
    };

    const handleOpenRemoveSnackbar = () => {
        setOpenRemoveSnackbar(true);
    };

    const handleCloseRemoveSnackbar = () => {
        setOpenRemoveSnackbar(false);
    };

    const updateSearch = e => {
        setSearch(e.target.value);
    }

    const getSearch = (e) => {
        e.preventDefault();
    }
    return {
        recipes,
        userCookbooks,
        search,
        updateSearch,
        addRecipe,
        removeRecipe,
        updateRecipe,
        openSuccessSnackbar,
        openModifySnackbar,
        openRemoveSnackbar,
        handleCloseSuccessSnackbar,
        handleOpenSuccessSnackbar,
        handleCloseModifySnackbar,
        handleOpenModifySnackbar,
        handleCloseRemoveSnackbar,
        handleOpenRemoveSnackbar,
        expanded,
        handleExpandClick
    }
};

export default RecipesLogic;
```

Recipes Styles JS File

```
import {Button}         from "@mui/material";
import {alpha, styled} from "@mui/material/styles";

export const BootstrapButton = styled(Button)({
    boxShadow: 'none',
    textTransform: 'none',
    fontSize: 16,
    padding: '6px 12px',
    border: '1px solid',
    lineHeight: 1.5,
    backgroundColor: '#FA4616',
    borderColor: `${alpha('#FA4616', 0.25)} 0 0 0 0.2rem`,
    justifyContent: 'space-around',
    fontFamily: [
        '-apple-system',
        'BlinkMacSystemFont',
        '"Segoe UI"',
        'Roboto',
        '"Helvetica Neue"',
        'Arial',
        'sans-serif',
        '"Apple Color Emoji"',
        '"Segoe UI Emoji"',
        '"Segoe UI Symbol"',
    ].join(','),
    '&:hover': {
        backgroundColor: '#FA4616',
        borderColor: `${alpha('#FA4616', 0.25)} 0 0 0 0.2rem`,
        boxShadow: 'none',
    },
    '&:active': {
        boxShadow: 'none',
        backgroundColor: '#FA4616',
        borderColor: `${alpha('#FA4616', 0.25)} 0 0 0 0.2rem`,
    },
    '&:focus': {
        boxShadow: `${alpha('#FA4616', 0.25)} 0 0 0 0.2rem`,
    },
});


export const CardStyles = {
    media: {
        paddingTop: '0',
        cursor: 'pointer',
    },
    card: {
        position: 'relative',
        justifyContent: 'center',
    },
    overlay: {
        backgroundColor: 'rgba(0,0,0,0.1)',
        position: 'absolute',
        top: '75%',
        left: '50%',
        transform: 'translate(-50%, -50%)',
        color: '#FA4616',
        cursor: 'pointer',
        padding: '2px',
        borderRadius: '50%'
    }
```

```
};
```

Recipes CSS File

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */

.input-100 {
    width: 100% !important;
}

.upload-icon:hover {
    margin: 1px;
}

.recipe-input {
    margin-bottom: 5px;
}

.bg-grey {
    background-color: #f5f5f5;
}
```

Sign Up

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */

 import {sendEmailVerification}                               from
"firebase/auth";
import React, { useState }                                    from
"react";
 import { Container, Col, Row, Button, Form, Image, Alert }  from
"react-bootstrap";
 import { Link, useNavigate  }                               from
"react-router-dom";
 import '../Login/Login.css';
 import "bootstrap/dist/css/bootstrap.css";
 import "../Custom.scss";
 import { useUserAuth }                                   from
"../../context/UserAuthContext";



  const Signup = () => {
      const [email, setEmail] = useState("");
      const [firstname, setFirstname] = useState("");
      const [surname, setSurname] = useState("");
      const [password, setPassword] = useState("");
      const [error, setError] = useState("");
      const [success, setSuccess] = useState("");
      const {signUp} = useUserAuth();

      const navigate = useNavigate();

       //error handling
      const handleSubmit = async (e) => {
          e.preventDefault();
          setError("");

          if (!email || !password || !firstname || !surname) {
              setError("Please fill all the fields");
              return;
          }

          try {
              await signUp(email, password, firstname, surname);
              setSuccess("Check your email to verify your account");
          } catch (error) {
              switch (error.code) {
                  case "auth/email-already-in-use":
                      setError("Email already in use");
                      break;
                  case "auth/invalid-email":
                      setError("Invalid email");
                      break;
                  case "auth/weak-password":
                      setError("Password is too weak");
                      break;
                  default:
                      setError("An error occurred");
              }
          };
      }

      return (
          <>
             <Container className="mt-5">
                 <Row>
                     <Col lg={12} md={12} sm={12}>
                         <img className="mx-auto d-block img-fluid"
src="loginLogo.png"></img>
                     </Col>
                 </Row>
                 <Row>
                     <Col lg={3} mf={3} sm={0}></Col>
                     <Col lg={6} mf={6} sm={12}>

                         {error && <Alert variant="danger">{ error
}</Alert>}
                         {success && <Alert variant="success">{ success
}</Alert>}
                         <Form onSubmit={handleSubmit}>
                             <Form.Group className="mb-3"
controlId="formBasicEmail">
                                 <Form.Label>Email address</Form.Label>
                                 <Form.Control type="email"
placeholder="Enter email" onChange={(e) => setEmail(e.target.value)}
required />
                                 <Form.Text className="text-muted">
We'll never share your email with anyone else.</Form.Text>
                             </Form.Group>

                             <Form.Group className="mb-3"
controlId="formBasicFirstname">
                                 <Form.Label>Firstname</Form.Label>
                                 <Form.Control type="text"
placeholder="Enter firstname" onChange={(e) =>
setFirstname(e.target.value)} required/>
                                 <Form.Text className="text-muted">
We'll never share your firstname with anyone else.</Form.Text>
                             </Form.Group>

                             <Form.Group className="mb-3"
controlId="formBasicSurname">
                                 <Form.Label>Surname</Form.Label>
                                 <Form.Control type="text"
placeholder="Enter surname" onChange={(e) =>
setSurname(e.target.value)} required/>
                                 <Form.Text className="text-muted">
We'll never share your surname with anyone else.</Form.Text>
                             </Form.Group>

                             <Form.Group className="mb-3"
controlId="formBasicPassword">
                                 <Form.Label>Password</Form.Label>
                                 <Form.Control type="password"
placeholder="Password" onChange= {(e) => setPassword(e.target.value)}
required/>
                             </Form.Group>

                             <div className="d-grid">
                                   <Button className="text-light"
variant="primary" type="submit">Sign up</Button>
                             </div>
                         </Form>

                         <div className="mt-2">
                            <small >Already have an account? <Link
to="/" className="forgotPass">Login</Link></small>
                         </div>
                     </Col>


                 </Row>

             </Container>
          </>
      )
  }


   export default Signup;
```

Context

User Authentication Context

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */

import {doc, setDoc, updateDoc}                          from
"firebase/firestore";
import { createContext, useContext, useEffect, useState } from "react";
import {
    createUserWithEmailAndPassword,
    signInWithEmailAndPassword,
    signOut,
    onAuthStateChanged,
    GoogleAuthProvider,
    signInWithPopup,
    sendPasswordResetEmail,
    sendEmailVerification,
    updateProfile
}                        from "firebase/auth";
import {auth, database} from "../FirebaseConfig";

const userAuthContext = createContext(null);

//this function is used to authenticate the user
export function UserAuthContextProvider({ children }){
    const [user, setUser] = useState(null);

    //"auth" is from FirebaseConfig.js
    function logIn(email, password){
        return signInWithEmailAndPassword(auth, email,
password).then(user => {
            if (!auth.currentUser.emailVerified) {
                const error = {
                    code: "auth/email-not-verified",
                    message: "Please verify your email address before
logging in."
                };
                return Promise.reject(error);
            } else {
                return auth.currentUser;
            }
        });
    }

    //"auth" is from FirebaseConfig.js
    function signUp(email, password, firstname, surname){
        return createUserWithEmailAndPassword(auth, email,
password).then( async (user) => {
            const userRef = doc(database, "users", user.user.uid);
            return setDoc(userRef, {
                uid: user.user.uid,
                displayName: firstname + " " + surname,
                email: user.user.email,
                photoURL: user.user.photoURL,
                createdAt: new Date().getTime()
            }, {merge: true}).then(() => {
                return sendEmailVerification(auth.currentUser)
                    .then(() => {
                            return updateProfile(auth.currentUser, {
                                displayName: firstname + " " + surname,
                            })
                        }).catch(error => {
                            return error;
                });
            });
        });
    }

    function logOut(){
        setUser(null);
        sessionStorage.clear();
        return signOut(auth);//"auth" is from FirebaseConfig.js
    }

    function googleSignIn(){
        const googleAuthProvider = new GoogleAuthProvider();
        return signInWithPopup(auth, googleAuthProvider).then(user => {
            //Get firstname and surname from google
            const userRef = doc(database, "users", user.user.uid);
            return setDoc(userRef, {
                uid: user.user.uid,
                displayName: user.user.displayName,
                email: user.user.email,
                photoURL: user.user.photoURL,
                createdAt: new Date().getTime()
            }, {merge: true});
        });
    }

    function passwordReset(email){
        return sendPasswordResetEmail(auth, email);
    }


    //this function notifies when a user is created or a user has
logged in. Function runs only once.
    useEffect(() => {
        const unsubscribe = onAuthStateChanged(auth, (currentUser) => {
            if(currentUser) {
                sessionStorage.setItem("user",
JSON.stringify(currentUser));
                setUser(currentUser);
            }
        });

        //Clean up
        return () => {
            unsubscribe();
        };
    }, []);

    return(
        <userAuthContext.Provider value={{user, signUp, logIn, logOut,
googleSignIn, passwordReset, sendEmailVerification}}>
            {children}
        </userAuthContext.Provider>
    );
}

export function useUserAuth(){
    return useContext(userAuthContext);
}
```

App JS

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */



import { Container, Row, Col }      from "react-bootstrap";
import { Routes, Route }            from "react-router-dom";
import { UserAuthContextProvider } from "./context/UserAuthContext";
import Login                       from "./components/Login/Login";
import Signup                      from "./components/SignUp/Signup";
import Discover                    from
"./components/Discover/Discover";
import ForgotPass                  from
"./components/ForgotPass/ForgotPass";
import ProtectedRoute              from
"./components/ProtectedRoute/ProtectedRoute";
import Recipes                     from "./components/Recipes/Recipes";
import MyRecipes                   from
"./components/MyRecipes/MyRecipes";
import MyCookbooks from "./components/MyCookbooks/MyCookbooks";
import Messages    from "./components/Messages/Messages";
import Profile    from "./components/Profile/Profile";

import "./App.css";

function App(){

    return (
      <Container>
          <Row>
              <Col>
                    <UserAuthContextProvider>
                        <Routes>
                            <Route path="/" exact element={ <Login />
}/>
                            <Route path="/signup" element={ <Signup />
}/>
                            <Route path="/forgotpass" element={
<ForgotPass /> }/>
                            <Route path="/discover" element={
<ProtectedRoute> <Discover/> </ProtectedRoute> }/>
                            <Route path="/recipes" element={
<ProtectedRoute> <Recipes/> </ProtectedRoute> }/>
                            <Route path="/myCookbooks" element={
<ProtectedRoute> <MyCookbooks/> </ProtectedRoute> }/>
                            <Route path="/myRecipes" element={
<ProtectedRoute> <MyRecipes/> </ProtectedRoute> }/>
                            <Route path="/messages" element={
<ProtectedRoute> <Messages/> </ProtectedRoute> }/>
                            <Route path="/profile" element={
<ProtectedRoute> <Profile/> </ProtectedRoute> }/>
```

```
                        </Routes>
                    </UserAuthContextProvider>
              </Col>
          </Row>
      </Container>
    );
}

export default App;
```

Firebase Configuration

```
/**
 * @author Fillipe Soares
 * @StartDate 18/01/2022
 */

// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
import { getAuth } from "firebase/auth";
import { getFirestore } from "firebase/firestore";
import { getMessaging } from "firebase/messaging";


//Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyCLbnhSZXk1PGVBniz0FBxcItuAQnat6u0",
  authDomain: "lifereceitas-1f6c1.firebaseapp.com",
  projectId: "lifereceitas-1f6c1",
  storageBucket: "lifereceitas-1f6c1.appspot.com",
  messagingSenderId: "854861200208",
  appId: "1:854861200208:web:d55e6d271490b378ee4a55",
  measurementId: "G-WW0D58E38C"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const analytics = getAnalytics(app);
const database = getFirestore(app);
const messaging = getMessaging(app);
```

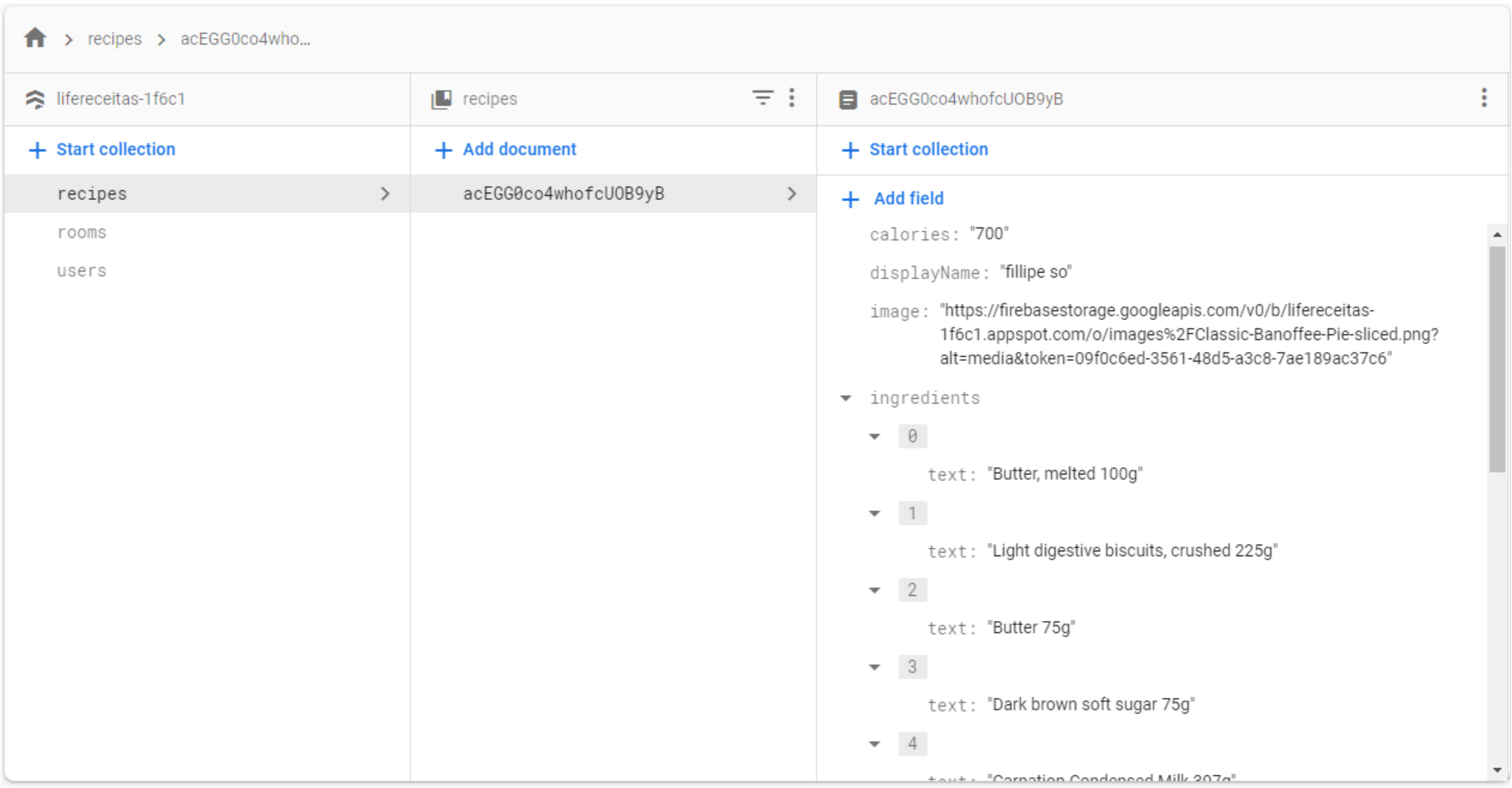
```
export { auth, database, messaging };

export default app;
```

Manifest JSON

```
{
  "name": "LifeReceitas",
  "short_name": "LifeReceitas",
  "theme_color": "#fa4616",
  "background_color": "#ffffff",
  "display": "standalone",
  "scope": "/",
  "start_url": "/",
  "icons": [
    {
      "src": "maskable_icon.png",
      "sizes": "196x196",
      "type": "image/png",
      "purpose": "any maskable"
    },
   {
     "src": "logo192.png",
     "sizes": "192x192",
     "type": "image/png"
   },
   {
    "src": "logo256.png",
    "sizes": "256x256",
    "type": "image/png"
  },
  {
    "src": "logo384.png",
    "sizes": "384x384",
    "type": "image/png"
  },
  {
    "src": "logo516.png",
    "sizes": "516x516",
    "type": "image/png"
  }
  ]
}
```

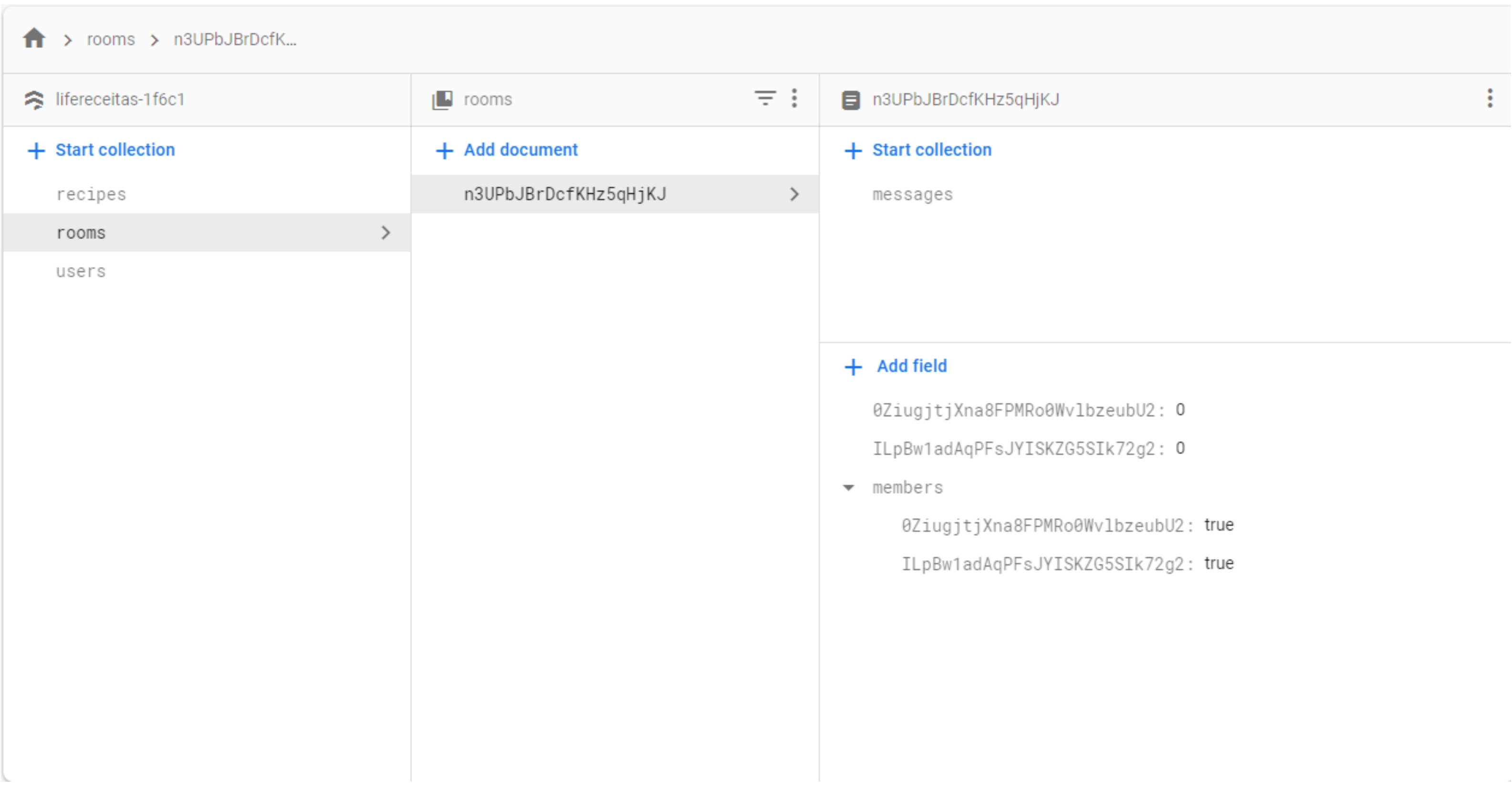
Database Layout

Recipes Layout



```
calories: "700"
displayName: "fillipe so"
image: "https://firebasestorage.googleapis.com/v0/b/lifereceitas-
       1f6c1.appspot.com/o/images%2FClassic-Banoffee-Pie-sliced.png?
       alt=media&token=09f0c6ed-3561-48d5-a3c8-7ae189ac37c6"
ingredients
  0
     text: "Butter, melted 100g"
  1
     text: "Light digestive biscuits, crushed 225g"
  2
     text: "Butter 75g"
  3
     text: "Dark brown soft sugar 75g"
  4
     text: "Carnation Condensed Milk 397g"
```

Messages Layout



rooms > n3UPbJBrDcfK...

lifereceitas-1f6c1

+ Start collection

recipes

rooms

users

rooms

+ Add document

n3UPbJBrDcfKHz5qHjKJ

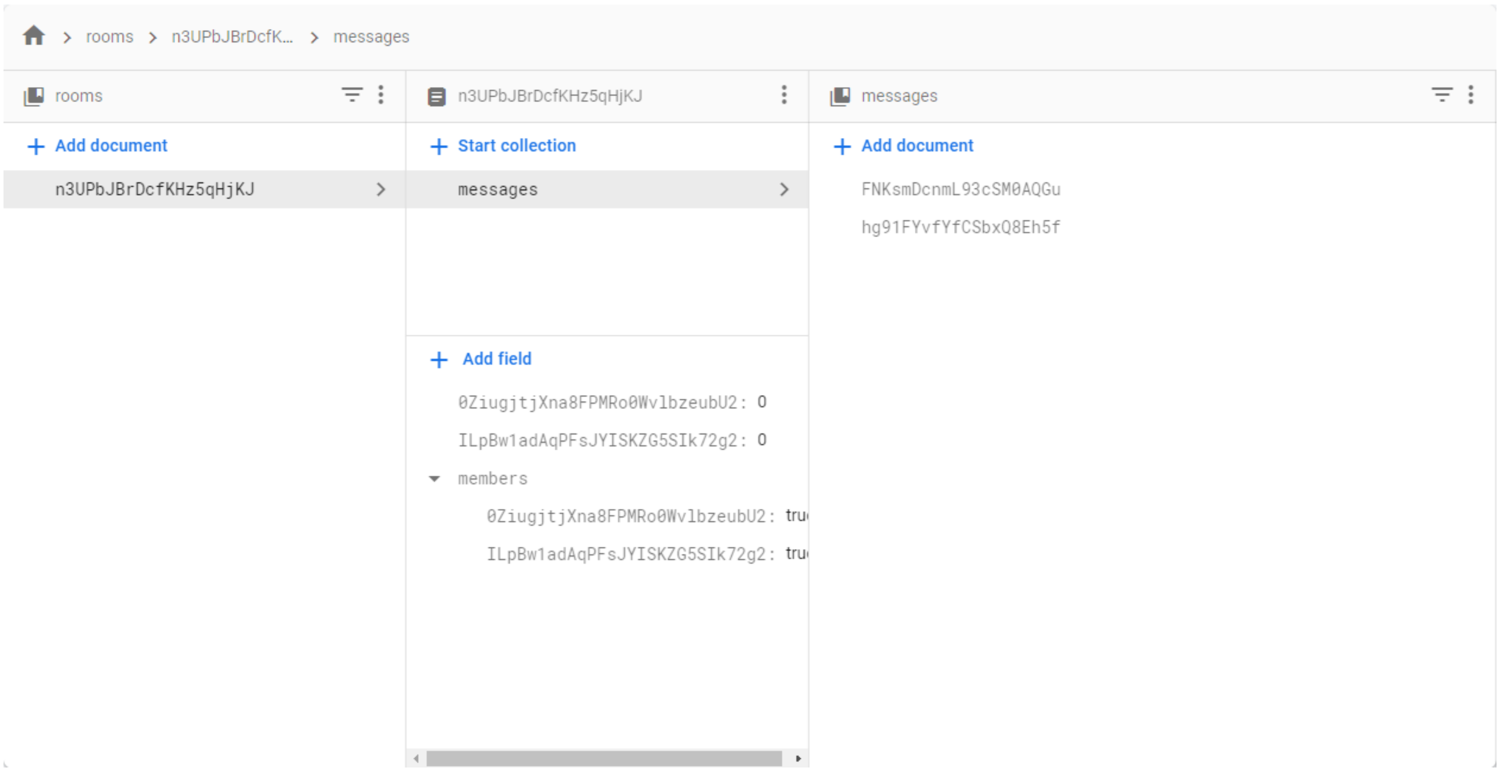
n3UPbJBrDcfKHz5qHjKJ

+ Start collection

messages

+ Add field

```
0ZiugjtjXna8FPMRo0WvlbzeubU2: 0
ILpBw1adAqPFsJYISKZG5SIk72g2: 0
members
    0ZiugjtjXna8FPMRo0WvlbzeubU2: true
    ILpBw1adAqPFsJYISKZG5SIk72g2: true
```



rooms > n3UPbJBrDcfK... > messages

rooms

+ Add document

n3UPbJBrDcfKHz5qHjKJ

n3UPbJBrDcfKHz5qHjKJ

+ Start collection

messages

+ Add field

```
0ZiugjtjXna8FPMRo0WvlbzeubU2: 0
ILpBw1adAqPFsJYISKZG5SIk72g2: 0
members
    0ZiugjtjXna8FPMRo0WvlbzeubU2: tru
    ILpBw1adAqPFsJYISKZG5SIk72g2: tru
```

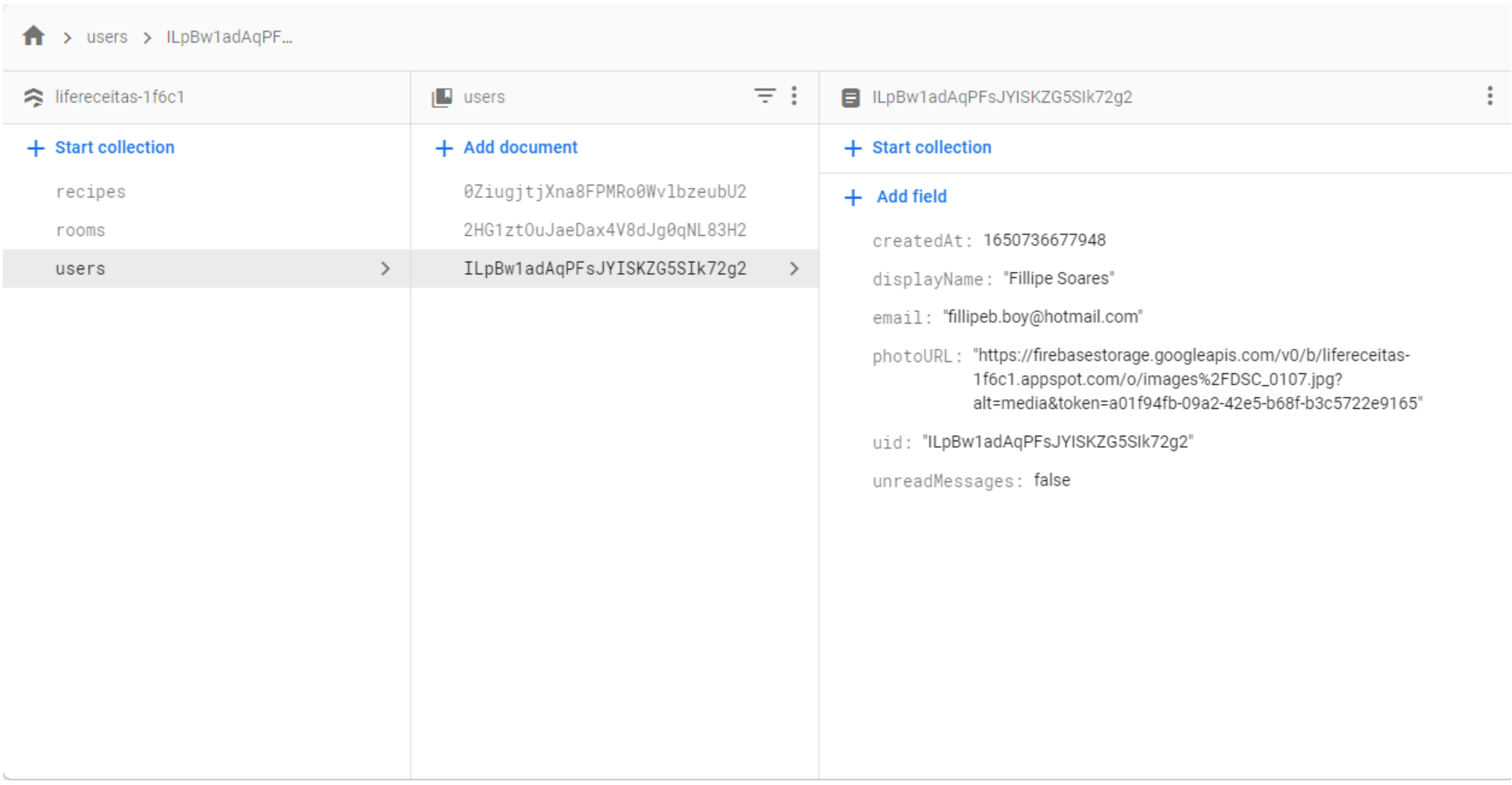
messages

+ Add document

FNKsmDcnmL93cSM0AQGu

hg91FYvfYfCSbxQ8Eh5f

User Details Layout



users > ILpBw1adAqPF...

lifereceitas-1f6c1

+ Start collection

recipes
rooms
users

users

+ Add document

0ZiugjtjXna8FPMRo0WvlbzeubU2
2HG1ztOuJaeDax4V8dJg0qNL83H2
ILpBw1adAqPFsJYISKZG5SIk72g2

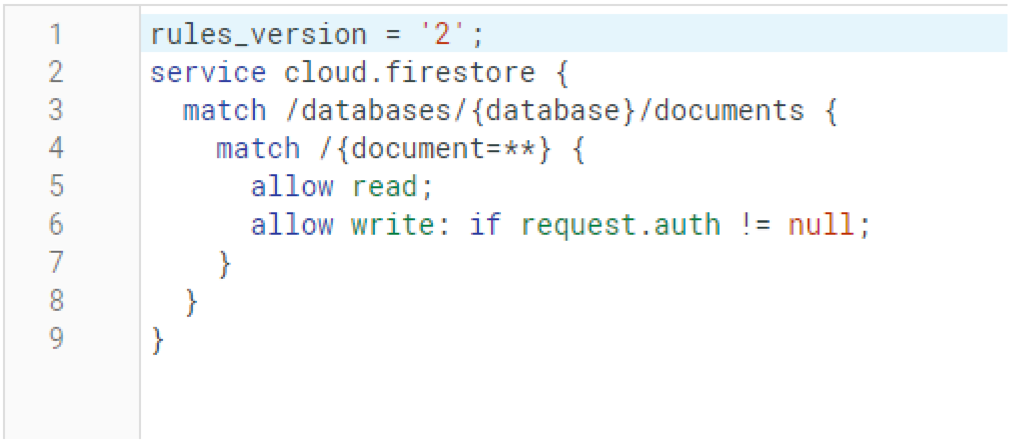
ILpBw1adAqPFsJYISKZG5SIk72g2

+ Start collection

+ Add field

createdAt: 1650736677948
displayName: "Fillipe Soares"
email: "fillipeb.boy@hotmail.com"
photoURL: "https://firebasestorage.googleapis.com/v0/b/lifereceitas-1f6c1.appspot.com/o/images%2FDSC_0107.jpg?alt=media&token=a01f94fb-09a2-42e5-b68f-b3c5722e9165"
uid: "ILpBw1adAqPFsJYISKZG5SIk72g2"
unreadMessages: false

Firebase Firestore Rules



```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read;
      allow write: if request.auth != null;
    }
  }
}
```