Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

\_\_\_\_\_\_\_\_ Ілля АХАЛАДЗЕ

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 р.

**Багатокористувацька веб-книга рецептів**

**Текст програми**

КПІ. ІП-1324.045440.03.12

“ПОГОДЖЕНО”

Керівник роботи:

\_\_\_\_\_\_\_\_\_\_\_\_ Ілля Ахаладзе

|  |  |
| --- | --- |
| Консультант: | Виконавець: |
| \_\_\_\_\_\_\_\_\_\_\_ Максим ГОЛОВЧЕНКО | \_\_\_\_\_\_\_\_\_\_ Ігор ПЕТРОВ |

Київ – 2024

**Файл backend/functions/index.js**

const functions = require("firebase-functions");

const cors = require("cors")({ origin: true });

const admin = require("firebase-admin");

const algoliasearch = require("algoliasearch");

const APP\_ID = functions.config().algolia.app;

const ADMIN\_KEY = functions.config().algolia.key;

const client = algoliasearch(APP\_ID, ADMIN\_KEY);

const index = client.initIndex("recipes");

admin.initializeApp();

exports.addToIndex = functions.firestore

.document("recipes/{recipeId}")

.onCreate((snapshot) => {

({

name,

imageLink,

category,

cookingComplexity,

cookingTime,

servingsNum,

description,

ingredients,

userUid,

} = snapshot.data());

const objectID = snapshot.id;

return index.saveObject({

name,

imageLink,

category,

cookingComplexity,

cookingTime,

servingsNum,

description,

ingredients,

userUid,

objectID,

});

});

exports.deleteFromIndex = functions.firestore

.document("recipes/{recipeId}")

.onDelete((snapshot) => index.deleteObject(snapshot.id));

exports.manageSavedRecipes = functions.https.onRequest(async (req, res) => {

cors(req, res, async () => {

try {

const { userId, recipeId } = req.body;

if (!userId || !recipeId) {

return res.status(400).json({ error: "No userId or recipeId" });

}

const userRef = admin.firestore().collection("users").doc(userId);

const userDoc = await userRef.get();

if (!userDoc.exists) {

return res.status(404).json({ error: "No user found" });

}

const userData = userDoc.data();

const savedRecipes = userData.saved || [];

const recipeIndex = savedRecipes.indexOf(recipeId);

const batch = admin.firestore().batch();

if (recipeIndex !== -1) {

savedRecipes.splice(recipeIndex, 1);

} else {

savedRecipes.push(recipeId);

}

batch.update(userRef, { saved: savedRecipes });

await batch.commit();

if (recipeIndex !== -1) {

return res.status(200).json({ message: "Recipe deleted from saved" });

} else {

return res.status(200).json({ message: "Recipe added to saved" });

}

} catch (error) {

console.error(error);

return res.status(500).json({ error: "Error" });

}

});

});

**Файл frontend/src/main.jsx**

import React from "react";

import ReactDOM from "react-dom/client";

import App from "./App.jsx";

import { BrowserRouter } from "react-router-dom";

import "./index.css";

import { Provider } from "react-redux";

import store from "./storage/store.js";

ReactDOM.createRoot(document.getElementById("root")).render(

<Provider store={store}>

<React.StrictMode>

<BrowserRouter>

<App />

</BrowserRouter>

</React.StrictMode>

</Provider>

);

**Файл frontend/src/App.jsx**

import { Routes, Route } from "react-router-dom";

import HomePage from "./pages/HomePage/HomePage";

import NoPage from "./pages/NoPage/NoPage";

import SignIn from "./pages/SignInUp/SignIn";

import SignUp from "./pages/SignInUp/SignUp";

import RecipeConstructor from "./pages/RecipeConstructor/RecipeConstructor";

import Footer from "./components/Footer/Footer";

import Header from "./components/Header/Header";

import Profile from "./pages/Profile/Profile";

import RecipesByCategory from "./pages/RecipesByCategory/RecipesByCategory";

import RecipePage from "./pages/RecipePage/RecipePage";

import { Sugar } from "react-preloaders2";

import { useDispatch } from "react-redux";

import { auth, firestoreDb } from "./firebase";

import { setUser, removeUser } from "./reducers/userReducer";

import { useEffect, useState } from "react";

import { ToastContainer } from "react-toastify";

import { onSnapshot, doc, getDoc } from "firebase/firestore";

import "react-toastify/dist/ReactToastify.css";

import "./Toastify.css";

import ProfileSettings from "./pages/ProfileSettings/ProfileSettings";

import SavedRecipes from "./pages/SavedRecipes/SavedRecipes";

import MyRecipes from "./pages/MyRecipes/MyRecipes";

import UserPage from "./pages/UserPage/UserPage";

import { setSaved } from "./reducers/userReducer";

import SearchResult from "./pages/SearchResult/SearchResult";

function App() {

const dispatch = useDispatch();

const [loading, setLoading] = useState(true);

useEffect(() => {

const unsubscribe = auth.onAuthStateChanged((user) => {

if (user) {

const userRef = doc(firestoreDb, "users", user.uid);

const getSaved = async () => {

try {

const userDoc = await getDoc(userRef);

if (userDoc.exists()) {

const userData = userDoc.data();

dispatch(setSaved(userData.saved ? userData.saved : []));

}

} catch (error) {

console.error(error);

}

};

getSaved();

onSnapshot(userRef, (snapshot) => {

const data = snapshot.data();

dispatch(

setUser({

nickname: data.nickname,

fullName: data.fullName,

description: data.description,

email: user.email,

uid: user.uid,

})

);

});

setLoading(false);

} else {

dispatch(removeUser());

setLoading(false);

}

});

return () => {

unsubscribe();

};

}, [dispatch]);

return (

<>

<Header />

<Routes>

<Route path="/" element={<HomePage />} />

<Route path="/profile" element={<Profile />} />

<Route path="/profile/settings" element={<ProfileSettings />} />

<Route path="/profile/saved" element={<SavedRecipes />} />

<Route path="/profile/myrecipes" element={<MyRecipes />} />

<Route path="/constructor" element={<RecipeConstructor />} />

<Route path="/login" element={<SignIn />} />

<Route path="/signup" element={<SignUp />} />

<Route path="/:category" element={<RecipesByCategory />} />

<Route path="/recipes/:id" element={<RecipePage />} />

<Route path="/users/:uid" element={<UserPage />} />

<Route path="/search" element={<SearchResult />}/>

<Route path="\*" element={<NoPage />} />

</Routes>

<Footer />

<Sugar

customLoading={loading}

background="var(--dark-grey)"

color="var(--white)"

/>

<ToastContainer

style={{ maxWidth: "45vw" }}

position="bottom-left"

autoClose={1500}

hideProgressBar={false}

newestOnTop={false}

closeOnClick

rtl={false}

draggable

pauseOnHover

theme="colored"

/>

</>

);

}

export default App;

**Файл frontend/src/storage/store.js**

import { configureStore } from "@reduxjs/toolkit";

import { combineReducers } from "@reduxjs/toolkit";

import thunk from "redux-thunk";

import constructorReducer from "../reducers/constructorReducer";

import userReducer from "../reducers/userReducer";

const rootReducer = combineReducers({

recipeConstructor: constructorReducer,

user: userReducer

});

const store = configureStore({

reducer: rootReducer,

middleware: [thunk],

devTools: true,

});

export default store;

**Файл frontend/src/reducers/ConstructorReducer.js**

import { createSlice } from "@reduxjs/toolkit";

const initialState = {

id: "",

name: "",

image: null,

cookingTime: 10,

servingsNum: 1,

cookingComplexity: "easy",

category: "soups",

commentary: "",

ingredients: [{ weight: "", unit: "g", name: "" }],

steps: [{ text: "" }],

};

const constructorReducer = createSlice({

name: "constructor",

initialState,

reducers: {

changeName: (state, action) => {

state.name = action.payload;

},

changeMainImg: (state, action) => {

state.image = action.payload;

},

changeTime: (state, action) => {

state.cookingTime = action.payload;

},

changeServingsNum: (state, action) => {

state.servingsNum = action.payload;

},

changeComplexity: (state, action) => {

state.cookingComplexity = action.payload;

},

changeCategory: (state, action) => {

state.category = action.payload;

},

changeCommentary: (state, action) => {

state.commentary = action.payload;

},

addIngredient: (state) => {

state.ingredients.push({ weight: "", unit: "g", name: "" });

},

removeIngredient: (state, action) => {

state.ingredients.splice(action.payload, 1);

},

changeIngredientWeight: (state, action) => {

const { index, weight } = action.payload;

state.ingredients[index].weight = weight;

},

changeIngredientUnit: (state, action) => {

const { index, unit } = action.payload;

state.ingredients[index].unit = unit;

},

changeIngredientName: (state, action) => {

const { index, name } = action.payload;

state.ingredients[index].name = name;

},

addStep: (state) => {

state.steps.push({ text: "" });

},

removeStep: (state, action) => {

state.steps.splice(action.payload, 1);

},

changeStepText: (state, action) => {

const { index, text } = action.payload;

state.steps[index].text = text;

},

clearConstructor: () => initialState,

},

});

export const {

changeName,

changeMainImg,

changeTime,

changeServingsNum,

changeComplexity,

changeCategory,

changeCommentary,

addIngredient,

removeIngredient,

changeIngredientWeight,

changeIngredientUnit,

changeIngredientName,

addStep,

removeStep,

changeStepText,

clearConstructor,

} = constructorReducer.actions;

export default constructorReducer.reducer;

**Файл backend/src/reducers/userReducer.js**

import { createAsyncThunk, createSlice } from "@reduxjs/toolkit";

import { toast } from "react-toastify";

import { manageSavedRecipes } from "../utils/firebaseUtils";

const initialState = {

nickname: null,

fullName: null,

email: null,

uid: null,

description: null,

saved: [],

};

export const addToSaved = createAsyncThunk(

"user/addToSavedAsync",

async (payload, thunkAPI) => {

const { uid } = thunkAPI.getState().user;

try {

await manageSavedRecipes(uid, payload);

return payload;

} catch (error) {

throw error;

}

}

);

export const removeFromSaved = createAsyncThunk(

"user/removeFromSavedAsync",

async (payload, thunkAPI) => {

const { uid } = thunkAPI.getState().user;

try {

await manageSavedRecipes(uid, payload);

return payload;

} catch (error) {

throw error;

}

}

);

const userSlice = createSlice({

name: "user",

initialState: initialState,

reducers: {

setUser(state, action) {

const { nickname, fullName, email, uid, description } = action.payload;

state.nickname = nickname;

state.fullName = fullName;

state.email = email;

state.uid = uid;

state.description = description;

},

setSaved(state, action) {

state.saved = action.payload

},

removeUser(state) {

state.nickname = null;

state.fullName = null;

state.email = null;

state.uid = null;

state.description = null;

state.saved = [];

},

},

extraReducers: (builder) => {

builder

.addCase(addToSaved.fulfilled, (state, action) => {

state.saved.push(action.payload);

})

.addCase(addToSaved.rejected, (state, action) => {

console.error(action.error);

toast.error("Error!");

})

.addCase(removeFromSaved.fulfilled, (state, action) => {

state.saved = state.saved.filter((id) => id !== action.payload);

})

.addCase(removeFromSaved.rejected, (state, action) => {

console.error(action.error);

toast.error("Error!");

});

},

});

export const { setUser, removeUser, setSaved } = userSlice.actions;

export default userSlice.reducer;

**Файл frontend/src/pages/HomePage.jsx**

import { useEffect } from "react";

import styles from "./HomePage.module.css";

import { Link } from "react-router-dom";

import { collection, getDocs, limit, query, orderBy } from "firebase/firestore";

import { firestoreDb } from "../../firebase";

import { useState } from "react";

import ClosedRecipeCard from "../../components/ClosedRecipeCard/ClosedRecipeCard";

import { useBreakpoint } from "../../hooks/useBreakpoint";

const HomePage = () => {

const recipesRef = collection(firestoreDb, "recipes");

const [newRecipes, setNewRecipes] = useState([]);

const [popularRecipes, setPopularRecipes] = useState([]);

const [isMoreNewRecipes, setIsMoreNewRecipes] = useState(false);

const [isMorePopularRecipes, setisMorePopularRecipes] = useState(false);

const breakPoint = useBreakpoint();

async function getNewRecipes() {

const q = query(recipesRef, orderBy("timestamp", "desc"), limit(6));

try {

const snapshot = await getDocs(q);

if (!snapshot.empty) {

return snapshot.docs.map((doc) => ({

recipeID: doc.id,

data: doc.data(),

}));

} else {

return [];

}

} catch (e) {

console.error(snapshot);

}

}

async function getPopularRecipes() {

const q = query(

recipesRef,

orderBy("userCommentsLength", "desc"),

limit(6)

);

try {

const snapshot = await getDocs(q);

if (!snapshot.empty) {

return snapshot.docs.map((doc) => ({

recipeID: doc.id,

data: doc.data(),

}));

} else {

return [];

}

} catch (e) {

console.error(snapshot);

}

}

useEffect(() => {

getNewRecipes().then((data) => {

setNewRecipes(data);

});

getPopularRecipes().then((data) => {

setPopularRecipes(data);

});

}, []);

return (

<section className={styles.homePage}>

<div className={styles.intro}>

<div className={styles.introInnerDiv}>

<h1>

Welcome to <br /> Taste Bud Tales!

</h1>

<p>

A place where flavor meets creativity! Share your passion for

cooking by contributing your finest recipes.

</p>

</div>

</div>

<div>

<h2 className={styles.sectionHeader}>

<hr />

Most recent

<hr />

</h2>

<div className={styles.recipesSection}>

{newRecipes.slice(0, breakPoint === "small" ? 4 : 3).map((recipe) => (

<ClosedRecipeCard

recipeID={recipe.recipeID}

key={recipe.recipeID}

imgLink={recipe.data.imageLink}

name={recipe.data.name}

cookingTime={recipe.data.cookingTime}

servingsNum={recipe.data.servingsNum}

complexity={recipe.data.cookingComplexity}

userID={recipe.data.userUid}

/>

))}

{isMoreNewRecipes &&

newRecipes

.slice(breakPoint === "small" ? 4 : 3, 6)

.map((recipe) => (

<ClosedRecipeCard

recipeID={recipe.recipeID}

key={recipe.recipeID}

imgLink={recipe.data.imageLink}

name={recipe.data.name}

cookingTime={recipe.data.cookingTime}

servingsNum={recipe.data.servingsNum}

complexity={recipe.data.cookingComplexity}

userID={recipe.data.userUid}

/>

))}

</div>

{!isMoreNewRecipes && (

<button

type="button"

className={styles.viewMoreButton}

onClick={() => {

setIsMoreNewRecipes(true);

}}

>

View more

</button>

)}

</div>

<div>

<h2 className={styles.sectionHeader}>

<hr />

Most Popular

<hr />

</h2>

<div className={styles.recipesSection}>

{popularRecipes.slice(0, breakPoint === "small" ? 4 : 3).map((recipe) => (

<ClosedRecipeCard

recipeID={recipe.recipeID}

key={recipe.recipeID}

imgLink={recipe.data.imageLink}

name={recipe.data.name}

cookingTime={recipe.data.cookingTime}

servingsNum={recipe.data.servingsNum}

complexity={recipe.data.cookingComplexity}

userID={recipe.data.userUid}

/>

))}

{isMorePopularRecipes &&

popularRecipes

.slice(breakPoint === "small" ? 4 : 3, 6)

.map((recipe) => (

<ClosedRecipeCard

recipeID={recipe.recipeID}

key={recipe.recipeID}

imgLink={recipe.data.imageLink}

name={recipe.data.name}

cookingTime={recipe.data.cookingTime}

servingsNum={recipe.data.servingsNum}

complexity={recipe.data.cookingComplexity}

userID={recipe.data.userUid}

/>

))}

</div>

{!isMorePopularRecipes && (

<button

type="button"

className={styles.viewMoreButton}

onClick={() => setisMorePopularRecipes(true)}

>

View more

</button>

)}

</div>

<div>

<h2 className={styles.sectionHeader}>

<hr />

What to cook today?

<hr />

</h2>

<nav className={styles.navigation}>

<ul className={styles.navList}>

<li className={`${styles.navItem} ${styles.li1}`}>

<Link to={"/soups"}>Soups</Link>

</li>

<li className={`${styles.navItem} ${styles.li2}`}>

<Link to={"/appetizers"}>Appetizers</Link>

</li>

<li className={`${styles.navItem} ${styles.li3}`}>

<Link to={"/bakery"}>Bakery</Link>

</li>

<li className={`${styles.navItem} ${styles.li4}`}>

<Link to={"/beverages"}>Beverages</Link>

</li>

<li className={`${styles.navItem} ${styles.li5}`}>

<Link to={"/salads"}>Salads</Link>

</li>

<li className={`${styles.navItem} ${styles.li6}`}>

<Link to={"/side\_dishes"}>Side dishes</Link>

</li>

<li className={`${styles.navItem} ${styles.li7}`}>

<Link to={"/main\_dishes"}>Main dishes</Link>

</li>

</ul>

</nav>

</div>

</section>

);

};

export default HomePage;

**Файл frontend/src/pages/RecipeConstructor.jsx**

import styles from "./RecipeConstructor.module.css";

import AddDishName from "../../components/AddDishName/AddDishName";

import AddImg from "../../components/AddImg/AddImg";

import AddIngredients from "../../components/AddIngredients/AddIngredients";

import AddDescription from "../../components/AddDescription/AddDescription";

import AddSteps from "../../components/AddSteps/AddSteps";

import SubmitConstructor from "../../components/SubmitConstructor/SubmitConstructor";

import AddCharacteristics from "../../components/AddСharacteristics/AddCharacteristics";

import { useAuth } from "../../hooks/useAuth";

const RecipeConstructor = () => {

const user = useAuth();

return (

<section className={styles.wrapper}>

{user.isAuth && (

<>

<h1 className={styles.constructorMainHeader}>

Creating your awesome recipe

</h1>

<form name="recipeConstructor" onSubmit={(e) => handleSubmit(e)}>

<ul>

<li className={styles.constructorElement}>

<h2 className={styles.constructorSectionHeader}>Name</h2>

<AddDishName />

</li>

<li className={styles.constructorElement}>

<h2 className={styles.constructorSectionHeader}>Main image</h2>

<AddImg />

</li>

<li className={styles.constructorElement}>

<h2 className={styles.constructorSectionHeader}>

Characteristics

</h2>

<AddCharacteristics />

</li>

<li className={styles.constructorElement}>

<h2 className={styles.constructorSectionHeader}>

Description (optional)

</h2>

<AddDescription />

</li>

<li className={styles.constructorElement}>

<h2 className={styles.constructorSectionHeader}>Ingredients</h2>

<AddIngredients />

</li>

<li className={styles.constructorElement}>

<h2 className={styles.constructorSectionHeader}>Steps</h2>

<AddSteps />

</li>

</ul>

<SubmitConstructor />

</form>

</>

)}

</section>

);

};

export default RecipeConstructor;

**Файл frontend/src/components/AddDescription.jsx**

import { useSelector, useDispatch } from "react-redux";

import styles from "./AddDescription.module.css";

import { changeCommentary } from "../../reducers/constructorReducer";

import TextareaAutosize from "react-textarea-autosize";

const AddDescription = () => {

const commentary = useSelector((state) => state.recipeConstructor.commentary);

const dispatch = useDispatch();

function handleInput(input) {

dispatch(changeCommentary(input));

}

return (

<div className={styles.commentaryWrapper}>

<TextareaAutosize

value={commentary}

onChange={(event) => handleInput(event.target.value)}

className={styles.commentary}

placeholder="Enter short description"

spellCheck={false}

maxLength={250}

/>

<span className={styles.counter}>{commentary.length}/250</span>

</div>

);

};

export default AddDescription;

**Файл frontend/src/components/AddDishName.jsx**

import { useDispatch, useSelector } from "react-redux";

import { changeName } from "../../reducers/constructorReducer";

import styles from "./AddDishName.module.css";

const AddDishName = () => {

const name = useSelector((state) => state.recipeConstructor.name);

const dispatch = useDispatch();

function handleInput(value) {

dispatch(changeName(value));

}

return (

<input

className={styles.input}

placeholder="Dish Name..."

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

value={name}

type="text"

maxLength={50}

/>

);

};

export default AddDishName;

**Файл frontend/src/components/AddImg.jsx**

import { useRef } from "react";

import styles from "./AddImg.module.css";

import { useDispatch, useSelector } from "react-redux";

import { changeMainImg } from "../../reducers/constructorReducer";

import camera from "@/assets/icons/camera.svg"

const AddImg = () => {

const file = useSelector((state) => state.recipeConstructor.image);

const dispatch = useDispatch();

const fileInput = useRef(null);

const wrapper = useRef(null);

const handleFileChange = (event) => {

event.preventDefault();

const file = event.target.files ? event.target.files[0] : null;

if (file) {

dispatch(changeMainImg(file));

}

};

const handleDropFileChange = (event) => {

event.preventDefault();

handleDrag(event, false);

const file = event.dataTransfer.files ? event.dataTransfer.files[0] : null;

if (file) {

dispatch(changeMainImg(file));

}

};

const handleDrag = (event, isDragging) => {

event.preventDefault();

if (wrapper.current) {

wrapper.current.style.backgroundColor = isDragging ? "#383838" : "var(--dark-grey)";

}

};

const handleDragOver = (event) => {

handleDrag(event, true);

};

const handleDragLeave = (event) => {

handleDrag(event, false);

};

const clearImg = () => {

if (fileInput.current) {

fileInput.current.value = "";

}

dispatch(changeMainImg(null));

return null;

};

return (

<div

onDrop={handleDropFileChange}

onDragOver={handleDragOver}

onDragLeave={handleDragLeave}

className={styles.imgWrapper}

ref={wrapper}

>

<input

ref={fileInput}

className={styles.fileInput}

type="file"

accept="image/\*"

onChange={handleFileChange}

/>

{!file ? (

<>

<img

className={styles.dragImage}

src={camera}

alt="addImg"

/>

<p className={styles.dragText}>Drag photo here</p>

</>

) : (

<>

{file.type.startsWith("image/") ? (

<a target="\_blank" href={URL.createObjectURL(file)}>

<img

className={styles.mainImg}

src={URL.createObjectURL(file)}

alt="Image"

/>

</a>

) : (

clearImg()

)}

</>

)}

<button

type="button"

className={styles.addImageButton}

onClick={() => fileInput.current?.click()}

>

{!file ? "Add Image" : "Change Image"}

</button>

</div>

);

};

export default AddImg;

**Файл frontend/src/components/AddIngredients.jsx**

import { useDispatch, useSelector } from "react-redux";

import styles from "./AddIngredients.module.css";

import { useRef } from "react";

import {

addIngredient,

removeIngredient,

changeIngredientWeight,

changeIngredientUnit,

changeIngredientName,

} from "../../reducers/constructorReducer";

import plus from "@/assets/icons/plus.svg";

import cross from "@/assets/icons/cross.svg";

const units = ["g", "kg", "ml", "l", "-", "tsp", "tbsp", "cup"];

const AddIngredients = () => {

const ingredientsList = useSelector(

(state) => state.recipeConstructor.ingredients

);

const dispatch = useDispatch();

const list = useRef(null);

function handleWeightChange(value, index) {

if (/^[0-9/.,\\-\s]\*$/.test(value)) {

dispatch(changeIngredientWeight({ weight: value, index: index }));

}

}

function handleUnitChange(value, index) {

dispatch(changeIngredientUnit({ unit: value, index: index }));

}

function handleNameChange(value, index) {

dispatch(changeIngredientName({ name: value, index: index }));

}

function handleRemove(index) {

dispatch(removeIngredient(index));

}

function addingIngredient() {

dispatch(addIngredient());

}

return (

<>

<ul ref={list} className={styles.ingredientsUl}>

{ingredientsList.map((ingredient, index) => (

<li key={index} className={styles.ingredientLi}>

<input

className={styles.weightInput}

onChange={(e) => handleWeightChange(e.target.value, index)}

type="text"

value={ingredient.weight}

placeholder="Weight"

/>

<div className={styles.unitSelect}>

<select

onChange={(e) => handleUnitChange(e.target.value, index)}

defaultValue="g"

className={styles.innerSelect}

>

{units.map((unit) => (

<option key={unit} value={unit}>

{unit}

</option>

))}

</select>

</div>

<input

className={styles.nameInput}

onChange={(e) => handleNameChange(e.target.value, index)}

type="text"

value={ingredient.name}

placeholder="Name"

/>

<button

type="button"

onClick={() => handleRemove(index)}

className={styles.crossBtn}

>

<img

className={styles.crossIcon}

src={cross}

alt="cross"

/>

</button>

</li>

))}

</ul>

<button

type="button"

onClick={() => addingIngredient()}

className={styles.addIngredientButton}

>

<img src={plus} alt="plus" />

Add Ingredient

</button>

</>

);

};

export default AddIngredients;

**Файл frontend/src/components/AddSteps.jsx**

import { useDispatch, useSelector } from "react-redux";

import styles from "./AddSteps.module.css";

import {

addStep,

removeStep,

changeStepText,

} from "../../reducers/constructorReducer";

import TextareaAutosize from "react-textarea-autosize";

import plus from "@/assets/icons/plus.svg";

const AddSteps = () => {

const steps = useSelector((state) => state.recipeConstructor.steps);

const dispatch = useDispatch();

function handleInput(index, value) {

dispatch(changeStepText({ index: index, text: value }));

}

function handleDelete(index) {

dispatch(removeStep(index));

}

return (

<>

<ul>

{steps.map((step, index) => (

<li key={index} className={styles.stepContainer}>

<h3 className={styles.stepHeader}>Step {index + 1}</h3>

<div className={styles.stepTextWrapper}>

<TextareaAutosize

value={step.text}

onChange={(event) => handleInput(index, event.target.value)}

className={styles.stepText}

placeholder="Enter step text"

spellCheck={false}

maxLength={500}

/>

<span className={styles.counter}>{step.text.length}/500</span>

</div>

<button

type="button"

onClick={() => handleDelete(index)}

className={styles.removeBtn}

>

Delete step

</button>

</li>

))}

</ul>

<button

type="button"

onClick={() => dispatch(addStep())}

className={styles.addStepButton}

>

<img src={plus} alt="plus" />

Add Step

</button>

</>

);

};

export default AddSteps;

**Файл frontend/src/components/AddCharacteristics.jsx**

import { useDispatch, useSelector } from "react-redux";

import styles from "./AddCharacteristics.module.css";

import {

changeTime,

changeServingsNum,

changeComplexity,

changeCategory,

} from "../../reducers/constructorReducer";

import clock from "@/assets/icons/clock.svg";

import serving from "@/assets/icons/portion.svg";

import complexitySvg from "@/assets/icons/complexity.svg";

import categorySvg from "@/assets/icons/category.svg";

const AddCharacteristics = () => {

const time = useSelector((state) => state.recipeConstructor.cookingTime);

const portions = useSelector((state) => state.recipeConstructor.servingsNum);

const complexity = useSelector(

(state) => state.recipeConstructor.cookingComplexity

);

const category = useSelector((state) => state.recipeConstructor.category);

const dispatch = useDispatch();

function handleTimeChange(value) {

dispatch(changeTime(value));

}

function handlePortionsChange(value) {

dispatch(changeServingsNum(value));

}

function handleComplexityChange(value) {

dispatch(changeComplexity(value));

}

function handleCategoryChange(value) {

dispatch(changeCategory(value));

}

return (

<ul className={styles.wrapper}>

<li className={styles.characteristic}>

<img src={clock} alt="clock" />

<h3>Time in min.</h3>

<input

value={time}

type="number"

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

/>

</li>

<li className={styles.characteristic}>

<img src={serving} alt="portions" />

<h3>Servings</h3>

<input

value={portions}

type="number"

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

/>

</li>

<li className={styles.characteristic}>

<img src={complexitySvg} alt="complexity" />

<h3>Complexity</h3>

<div className={styles.unitSelect}>

<select

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

className={styles.innerSelect}

value={complexity}

>

<option value="easy">easy</option>

<option value="medium">medium</option>

<option value="hard">hard</option>

</select>

</div>

</li>

<li className={styles.characteristic}>

<img src={categorySvg} alt="category" />

<h3>Category</h3>

<div className={styles.unitSelect}>

<select

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

className={styles.innerSelect}

value={category}

>

<option value="soups">soup</option>

<option value="salads">salad</option>

<option value="side\_dishes">side dish</option>

<option value="main\_dishes">main dish</option>

<option value="appetizers">appetizer</option>

<option value="bakery">bakery</option>

<option value="beverages">beverage</option>

</select>

</div>

</li>

</ul>

);

};

export default AddCharacteristics;

**Файл frontend/src/components/SubmitConstructor.jsx**

import styles from "./SubmitConstructor.module.css";

import { getCurrentDate } from "../../utils/dateFormatting";

import { useDispatch, useSelector } from "react-redux";

import { toast } from "react-toastify";

import { storage } from "../../firebase";

import {

ref as storageRef,

uploadBytes,

getDownloadURL,

} from "firebase/storage";

import { areAllFieldsFilled } from "../../utils/checkObjects";

import { useAuth } from "../../hooks/useAuth";

import { useNavigate } from "react-router-dom";

import { useState } from "react";

import { clearConstructor } from "../../reducers/constructorReducer";

import { doc, setDoc, serverTimestamp } from "firebase/firestore";

import { firestoreDb } from "../../firebase";

const SubmitConstructor = () => {

const newRecipe = useSelector((state) => state.recipeConstructor);

const dispatch = useDispatch();

const [isLoading, setIsLoading] = useState(false);

const navigate = useNavigate();

const user = useAuth();

const cleanName = newRecipe.name.replace(/\s+/g, " ").trim();

const cleanIngredients = newRecipe.ingredients.map((ingredient) => {

return {

weight: ingredient.weight.replace(/\s+/g, " ").trim(),

unit: ingredient.unit,

name: ingredient.name.replace(/\s+/g, " ").trim()

};

});

const cleanSteps = newRecipe.steps.map((step) => {

return {

text: step.text.replace(/\s+/g, " ").trim()

}

});

function checkRecipe() {

switch (true) {

case !cleanName:

return "No dish name";

case cleanName.length <= 3:

return "Dish name length must be at least 3";

case !newRecipe.image:

return "No dish image";

case !newRecipe.cookingTime:

return "No cooking time";

case Number(newRecipe.cookingTime) < 1:

return "Cooking time must be >1";

case Number(newRecipe.cookingTime) > 10000:

return "Cooking time must be <10000";

case !newRecipe.servingsNum:

return "No servings number";

case Number(newRecipe.servingsNum) < 1:

return "Recipe must be for at least one serving";

case Number(newRecipe.servingsNum) > 100:

return "Recipe must be for <100 servings";

case cleanIngredients.length <= 1:

return "Recipe must contain at least 2 ingredients";

case !areAllFieldsFilled(cleanIngredients):

return "Some ingredients are invalid";

case cleanSteps.length < 1:

return "Recipe must contain at least 1 step";

case !areAllFieldsFilled(cleanSteps):

return "Some steps are invalid";

default:

return null;

}

}

async function handleSubmit(e) {

e.preventDefault();

const checkRecipeResult = checkRecipe();

if (checkRecipeResult) {

toast.error(checkRecipeResult);

} else {

setIsLoading(true);

const currentDate = getCurrentDate();

const uniqueKey = user.uid + Date.now();

const imageRef = storageRef(

storage,

`recipeImages/${uniqueKey}/mainImg.${

newRecipe.image.type.split("/")[1]

}`

);

const recipesRef = doc(firestoreDb, "recipes", uniqueKey);

try {

await uploadBytes(imageRef, newRecipe.image);

const imageLink = await getDownloadURL(imageRef);

await setDoc(

recipesRef,

{

name: cleanName,

imageLink: imageLink,

cookingTime: newRecipe.cookingTime,

servingsNum: newRecipe.servingsNum,

cookingComplexity: newRecipe.cookingComplexity,

category: newRecipe.category,

description: newRecipe.commentary.trim(),

ingredients: cleanIngredients,

steps: cleanSteps,

updateDate: currentDate,

userUid: user.uid,

nickname: user.nickname,

timestamp: serverTimestamp(),

userCommentsLength: 0,

comments: [],

},

{

maxAttempts: 1,

backoffMillis: 3000,

}

);

dispatch(clearConstructor());

toast.success("New recipe created");

navigate(`/recipes/${uniqueKey}`);

} catch (e) {

console.error(e);

}

}

}

return (

<button

onClick={(e) => handleSubmit(e)}

type="submit"

className={styles.submit}

disabled={isLoading}

>

{isLoading ? "Creating..." : "Submit recipe"}

</button>

);

};

export default SubmitConstructor;