Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Керівник роботи
_____ Ілля АХАЛАДЗЕ
“___” _____ 2024 р.

ВЕБ-ЗАСТОСУНОК ПІДТРИМКИ РОБОТИ ПУНКТУ ОБМІНУ ВАЛЮТ

Текст програми

КПІ.ІП-1304.045440.05.13

“ПОГОДЖЕНО”
Керівник роботи:
_____ Ілля АХАЛАДЗЕ

Консультант:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Максим БОНДАРЕНКО

Київ – 2024

Файл main.jsx

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App.jsx";
import "./index.css";
import { BrowserRouter } from "react-router-dom";
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 App.jsx

```
import { Routes, Route } from "react-router-dom";
import SignUp from "./pages/SignUp/SignUp";
import Header from "./components/Header/Header";
import Home from "./pages/Home/Home";
import SignIn from "./pages/SignIn/SignIn";
import StartExchange from "./pages/StartExchange/StartExchange";
import { useDispatch } from "react-redux";
import { setUser, removeUser } from "./reducers/userReducer";
import { useEffect } from "react";
import { auth } from "./firebase";
```

```
import { db } from "./firebase";
import { doc, onSnapshot } from "firebase/firestore";
import Profile from "./pages/Profile/Profile";
import AdminExchangeRequests from
"./pages/AdminExchangeRequests/AdminExchangeRequests";
import Footer from "./components/Footer/Footer";
import { useState } from "react";
import classes from "./App.module.css";

function App() {
  const dispatch = useDispatch();
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const unsubscribe = auth.onAuthStateChanged((user) => {
      if (user) {
        const userRef = doc(db, "users", user.uid);
        onSnapshot(userRef, (snapshot) => {
          if (snapshot.exists()) {
            const data = snapshot.data();
            dispatch(
              setUser({
                username: data.username,
                email: user.email,
                currencyDiff: data.currencyDiff,
                isAdmin: data.isAdmin,
                uid: user.uid,
              })
            );
          } else {
```

```
          console.log("User does not exist in Firestore.");
        }
      });
      setLoading(false);
    } else {
      dispatch(removeUser());
      setLoading(false);
    }
  });
  return () => {
    unsubscribe();
  };
}, [dispatch]);

return (
  <>
    {loading ? (
      <div className={classes.ldsDualRing}></div>
    ) : (
      <>
        <Header />
        <Routes>
          <Route path="/signup" element={<SignUp />} />
          <Route path="/" element={<Home />} />
          <Route path="/signin" element={<SignIn />} />
          <Route path="/profile" element={<Profile />} />
          <Route path="/start-exchange" element={<StartExchange />} />
          <Route
            path="/exchange-requests"
            element={<AdminExchangeRequests />}
```

```
          />
        </Routes>
        <Footer />
      </>
    )}
  </>
 );
}

export default App;
```

Файл store.js

```
import { configureStore } from "@reduxjs/toolkit";
import userReducer from "../reducers/userReducer";
import { combineReducers } from "redux";

const rootReducer = combineReducers({
 user: userReducer,
});

const store = configureStore({
 reducer: rootReducer,
 devTools: true,
});

export default store;
```

Файл userReducer.js

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

```
const initialState = {
  username: null,
  email: null,
  currencyDiff: {
    USD: null,
    EUR: null,
    PLN: null,
    GBP: null,
  },
  isAdmin: false,
  uid: null,
};

const authSlice = createSlice({
  name: "user",
  initialState,
  reducers: {
    setUser: (state, action) => {
      state.username = action.payload.username;
      state.email = action.payload.email;
      state.currencyDiff = action.payload.currencyDiff;
      state.isAdmin = action.payload.isAdmin;
      state.uid = action.payload.uid;
    },
    removeUser: (state) => {
      state.username = null;
      state.email = null;
      state.currencyDiff = {
        USD: null,
        EUR: null,
```

```
        PLN: null,
        GBP: null,
      };
      state.isAdmin = false;
      state.uid = null;
    },
  },
});

export const { setUser, removeUser } = authSlice.actions;
export default authSlice.reducer;
```

Файл useAuth.jsx

```
import { useSelector } from "react-redux";

export function useAuth() {
  const { username, email, currencyDiff, isAdmin, uid } =
    useSelector((state) => state.user);

  return {
    isAuth: !!uid,
    username,
    email,
    currencyDiff,
    isAdmin,
    uid,
  };
}
```

Файл SignUpForm.jsx

```
import { useState } from "react";
// import classes from './SignUpForm.module.css';
import { getAuth, createUserWithEmailAndPassword } from "firebase/auth";
import { db } from "../../firebase";
// import { collection, addDoc } from 'firebase/firestore';
import { sendEmailVerification } from "firebase/auth";
import { useEffect } from "react";
import { doc, setDoc } from "firebase/firestore";
import classes from "../SignInForm/SignInForm.module.css";
import { useNavigate } from "react-router-dom";

const SignUpForm = () => {
  const [formData, setFormData] = useState({
    username: "",
    email: "",
    password: "",
  });

  const [errors, setErrors] = useState({
    username: "",
    email: "",
    password: "",
  });

  const navigate = useNavigate();

  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setFormData((prevData) => ({ ...prevData, [name]: value }));
```

```
    setErrors((prevErrors) => ({ ...prevErrors, [name]: "" }));
  };

  const validateForm = () => {
    let isValid = true;
    const newErrors = {};

    const usernameRegex = /^[a-zA-Z0-9_.]+$/;
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    const validateName = (fieldName, regex) => {
      if (!formData[fieldName].trim() || !regex.test(formData[fieldName])) {
        newErrors[fieldName] = `Valid ${fieldName.toLowerCase()} is required`;
        isValid = false;
      }
    };

    validateName("username", usernameRegex);
    validateName("email", emailRegex);

    const passwordRegex = /^[a-zA-Z0-9!@#$%^&*]{8,16}$/;

    if (!passwordRegex.test(formData.password)) {
      newErrors.password =
        "Password must be 8-16 characters long, contain only numbers, letters and
special characters";
      isValid = false;
    }

    setErrors(newErrors);
```

```
  return isValid;
};

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

  if (validateForm()) {
    try {
      const auth = getAuth();

      const userCredential = await createUserWithEmailAndPassword(
        auth,
        formData.email,
        formData.password
      );

      const user = userCredential.user;

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

      await sendEmailVerification(auth.currentUser);

      await setDoc(
        userRef,
        {
          username: formData.username,
          email: formData.email,
          currencyDiff: {
            USD: null,
            EUR: null,
```

```
            PLN: null,
            GBP: null,
          },
          isAdmin: false,
        },
        {
          maxAttempts: 1,
          backoffMillis: 3000,
        }
      );

      console.log("Verification email sent. Please verify your email.");
      navigate("/");
    } catch (error) {
      console.error("Error registering user:", error.message);
      if (error.code === "auth/email-already-in-use") {
        setErrors((prevErrors) => ({
          ...prevErrors,
          general: "Email is already in use",
        }));
      } else {
        setErrors((prevErrors) => ({
          ...prevErrors,
          general: error.message,
        }));
      }
    }
  }
};
```

```
return (
  <div className={classes.signInContainer}>
    <div className={classes.signInForm}>
      <h2 className={classes.text}>Sign Up</h2>
      <form onSubmit={handleSubmit}>
        <div className={classes.formGroup}>
          <input
            className={classes.input}
            type="text"
            name="username"
            value={formData.username}
            onChange={handleInputChange}
            placeholder="Username"
          />
          <div className={classes.error}>{errors.username}</div>
        </div>
        <div className={classes.formGroup}>
          <input
            className={classes.input}
            type="email"
            name="email"
            value={formData.email}
            onChange={handleInputChange}
            placeholder="Email"
            autoComplete="username"
          />
          <div className={classes.error}>{errors.email}</div>
        </div>
        <div className={classes.formGroup}>
          <input
```

```
            className={classes.input}
            type="password"
            name="password"
            value={formData.password}
            onChange={handleInputChange}
            placeholder="Password"
            autoComplete="current-password"
          />
          <div className={classes.error}>{errors.password}</div>
        </div>

        <div className={classes.myBtnCont}>
          <button type="submit" className={classes.myBtn}>
            Sign Up
          </button>
          <div className={classes.errorGeneral}>{errors.general}</div>
        </div>
      </form>
    </div>
  </div>
 );
};

export default SignUpForm;
```

Файл SignInForm.jsx

```
import { useState } from "react";
import classes from "./SignInForm.module.css";
import { getAuth, signInWithEmailAndPassword } from "firebase/auth";
import { useNavigate } from "react-router-dom";
```

```
import { db } from "../../firebase";
import { doc, updateDoc } from "firebase/firestore";

const SignInForm = () => {
  const [formData, setFormData] = useState({
    email: "",
    password: "",
  });

  const [errors, setErrors] = useState("");

  const navigate = useNavigate();

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData((prevData) => ({ ...prevData, [name]: value }));
  };

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

    try {
      const auth = getAuth();
      const { user } = await signInWithEmailAndPassword(
        auth,
        formData.email,
        formData.password
      );
      await updateDoc(doc(db, "users", auth.currentUser.uid), {
        email: auth.currentUser.email,
```

```
    });
    if (user) {
      navigate("/");
    }
  } catch (error) {
    console.error("Error signing in:", error.message);
    switch (error.code) {
      case "auth/invalid-credential":
        setErrors("Invalid email or password");
        break;
      case "auth/too-many-requests":
        setErrors("Many failed login attempts. Please try again later");
        break;
      case "auth/invalid-email":
        setErrors("Invalid email");
        break;
      case "auth/missing-password":
        setErrors("Missing password");
        break;
      default:
        setErrors("An error occurred while signing in");
        break;
    }
  }
};

return (
  <div className={classes.signInContainer}>
    <div className={classes.signInForm}>
      <h2 className={classes.text}>Login</h2>
```

```
<form onSubmit={handleSubmit}>
  <div className={classes.formGroup}>
    <input
      className={classes.input}
      type="email"
      name="email"
      value={formData.email}
      onChange={handleChange}
      placeholder="Email"
      autoComplete="username"
    />
  </div>
  <div className={classes.formGroup}>
    <input
      className={classes.input}
      type="password"
      name="password"
      value={formData.password}
      onChange={handleChange}
      placeholder="Password"
      autoComplete="current-password"
    />
  </div>

  <div className={classes.formGroup}>
    <button className={classes.signin} type="submit">
      Sign In
    </button>
    {errors && <div className={classes.error}>{errors}</div>}
  </div>
```

```
      </form>
    </div>
  </div>
 );
};

export default SignInForm;
```

Файл RequestsHistory.jsx

```
import { useEffect, useState } from "react";
import { db } from "../../firebase";
import { collection } from "firebase/firestore";
import RequestRow from "../RequestRow/RequestRow";
import classes from "./RequestsHistory.module.css";
import { query, where, orderBy } from "firebase/firestore";
import { useAuth } from "../../hooks/useAuth";
import { format } from "date-fns";
import { onSnapshot } from "firebase/firestore";

const RequestsHistory = ({ showAdmin }) => {
 const [requestsHistory, setRequestsHistory] = useState([]);
 const user = useAuth();

 useEffect(() => {
  const fetchRequestHistory = async () => {
   try {
    const requestsCollection = collection(db, "requests");

    let requestsQuery = null;
    if (showAdmin) {
```

```
  requestsQuery = query(
    requestsCollection,
    orderBy("timestamp", "desc")
  );
} else {
  requestsQuery = query(
    requestsCollection,
    where("userId", "==", user.uid),
    orderBy("timestamp", "desc")
  );
}

const unsubscribe = onSnapshot(requestsQuery, (querySnapshot) => {
  const historyData = querySnapshot.docs.map((doc) => {
    const { timestamp, ...rest } = doc.data();

    const formattedDate = format(timestamp.toDate(), "yyyy-MM-dd");
    const formattedTime = format(timestamp.toDate(), "HH:mm:ss");

    const shortId = doc.id.slice(-4);

    return {
      id: doc.id,
      shortId: shortId,
      date: formattedDate,
      time: formattedTime,
      ...rest,
    };
  });
```

```
        setRequestsHistory(historyData);
      });
      return () => {
        unsubscribe();
      };
    } catch (error) {
      console.error("Error fetching request history:", error.message);
    }
  };

  fetchRequestHistory();
}, [user.uid]);

return (
  <div className={classes.requestsHistory}>
    <table className={classes.table}>
      <thead>
        {requestsHistory.length > 0 ? (
          <tr className={classes.header}>
            <th className={classes.label}>ID</th>
            <th className={classes.label}>From</th>
            <th className={classes.label}>To</th>
            <th className={classes.label}>From</th>
            <th className={classes.label}>To</th>
            <th className={classes.label}>Date</th>
            <th className={classes.label}>Time</th>
            <th className={classes.status}>Status</th>
          </tr>
        ) : null}
      </thead>
```

```
        <tbody className={classes.tbody}>
          {requestsHistory.map((request) => (
            <RequestRow key={request.id} request={request} />
          ))}
        </tbody>
      </table>
    </div>
  );
};

export default RequestsHistory;
```

Файл RequestRow.jsx

```
import classes from "./RequestRow.module.css";
import { useState } from "react";
import { doc, updateDoc } from "firebase/firestore";
import { db } from "../../firebase";
import { useAuth } from "../../hooks/useAuth";

const RequestRow = ({ request }) => {
  const user = useAuth();
  const [status, setStatus] = useState(request.status);

  const handleStatusChange = async (event) => {
    const newStatus = event.target.value;

    console.log(user.isAdmin);
    if (!user.isAdmin && newStatus === "rejected") {
      const requestDocRef = doc(db, "requests", request.id);
      await updateDoc(requestDocRef, { status: newStatus });
```

```
    setStatus(newStatus);
  } else if (user.isAdmin) {
    const requestDocRef = doc(db, "requests", request.id);
    await updateDoc(requestDocRef, { status: newStatus });

    setStatus(newStatus);
  }
};

const getStatusColor = () => {
  switch (status) {
    case "completed":
      return "#6CAA8C";
    case "rejected":
      return "#c48a8a";
    case "take away":
      return "#cbba9c";
    case "pending":
      return "#9ca1cb";
    default:
      return "white";
  }
};

return (
  <tr className={classes.requestRow}>
    <td className={classes.info}>{request.shortId}</td>
    <td className={classes.info}>{request.sourceCurrency}</td>
    <td className={classes.info}>{request.targetCurrency}</td>
```

```
      <td className={classes.info}>{request.amount}</td>
      <td className={classes.info}>{request.convertedAmount}</td>
      <td className={classes.info}>{request.date}</td>
      <td className={classes.info}>{request.time}</td>
      <td className={classes.select}>
       <select
        className={classes.select}
        value={status}
        onChange={handleStatusChange}
        disabled={status === "completed" || status === "rejected"}
        style={{ backgroundColor: getStatusColor() }}
       >
        <option className={classes.option} value="pending">
         pending
        </option>
        <option className={classes.option} value="completed">
         completed
        </option>
        <option className={classes.option} value="rejected">
         rejected
        </option>
        <option className={classes.option} value="take away">
         take away
        </option>
       </select>
      </td>
    </tr>
  );
};
```

```
export default RequestRow;
```

Файл ProfileInfo.jsx

```
import { useState } from "react";
import { useAuth } from "../../hooks/useAuth";
import classes from "./ProfileInfo.module.css";
import { useNavigate } from "react-router-dom";
import { db } from "../../firebase";
import { deleteDoc, doc } from "firebase/firestore";
import { updateDoc } from "firebase/firestore";
import { useEffect } from "react";
import {
  EmailAuthProvider,
  getAuth,
  updatePassword,
  verifyBeforeUpdateEmail,
} from "firebase/auth";
import { reauthenticateWithCredential } from "firebase/auth";


const ProfileInfo = () => {
  const navigate = useNavigate();
  const auth = getAuth();
  const user = auth.currentUser;
  const currentEmail = user.email;

  const { username, email, currencyDiff, uid, isAuth } = useAuth();
  const [formData, setFormData] = useState({
    username,
    email,
```

```
  currencyDiff,
  curPassword: "",
  newPassword: "",
});

const [errors, setErrors] = useState({
  username: "",
  email: "",
  curPassword: "",
  newPassword: "",
  usd: "",
  eur: "",
  pln: "",
  gbp: "",
});

const [showPopup, setShowPopup] = useState(false);

useEffect(() => {
  if (showPopup) {
    const timeoutId = setTimeout(() => {
      setShowPopup(false);
    }, 5000);

    return () => clearTimeout(timeoutId);
  }
}, [showPopup]);

useEffect(() => {
  if (uid) {
```

```
      setFormData({
        username,
        email,
        currencyDiff,
      });
    }
  }, [uid]);

  const handleChange = (e) => {
    const { name, value } = e.target;

    if (name === "USD" || name === "EUR" || name === "PLN" || name ===
"GBP") {
      setFormData((prevData) => ({
        ...prevData,
        currencyDiff: { ...prevData.currencyDiff, [name]: value },
      }));
      return;
    } else if (name === "curPassword" || name === "newPassword") {
      setFormData((prevData) => ({ ...prevData, [name]: value }));
    }
    setFormData((prevData) => ({ ...prevData, [name]: value }));
    setErrors((prevErrors) => ({ ...prevErrors, [name]: "" }));
  };

  const validateForm = () => {
    let isValid = true;
    const newErrors = {};
    const usernameRegex = /^[a-zA-Z0-9_.]+$/;
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
```

```
  const passwordRegex = /^[a-zA-Z0-9!@#$%^&*]{8,16}$/;

  const validateName = (fieldName, regex) => {
    if (!formData[fieldName].trim() || !regex.test(formData[fieldName])) {
      newErrors[fieldName] = `Valid ${fieldName.toLowerCase()} is required`;
      isValid = false;
    }
  };

  const validateCurrency = (currencyName) => {
    if (
      isNaN(formData.currencyDiff[currencyName]) ||
      formData.currencyDiff[currencyName] < 0 ||
      formData.currencyDiff[currencyName] > 10 ||
      formData.currencyDiff[currencyName] === "-0"
    ) {
      newErrors[currencyName.toLowerCase()] = "Must be from 0 to 10.0";
      isValid = false;
    }
  };

  const validatePassword = (fieldName, regex) => {
    if (!regex.test(formData[fieldName]) && formData[fieldName].trim()) {
      newErrors[fieldName] =
        "Password must be 8-16 characters long, contain only numbers, letters and special characters";
      isValid = false;
    }
  };
```

```
  validateName("username", usernameRegex);
  validateName("email", emailRegex);

  ["USD", "EUR", "PLN", "GBP"].forEach((currency) =>
    validateCurrency(currency)
  );

  validatePassword("curPassword", passwordRegex);
  validatePassword("newPassword", passwordRegex);

  setErrors(newErrors);
  return isValid;
};

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

  if (validateForm()) {
    try {
      await updateDoc(doc(db, "users", uid), {
        username: formData.username,
        currencyDiff: {
          USD: formData.currencyDiff.USD || null,
          EUR: formData.currencyDiff.EUR || null,
          PLN: formData.currencyDiff.PLN || null,
          GBP: formData.currencyDiff.GBP || null,
        },
      });

      if (currentEmail !== formData.email) {
```

```
    const credential = EmailAuthProvider.credential(
      currentEmail,
      formData.curPassword
    );
    await reauthenticateWithCredential(user, credential);
    navigate("/signin");

    await verifyBeforeUpdateEmail(user, formData.email);
    await auth.signOut();
  }

  if (formData.newPassword) {
    const credential = EmailAuthProvider.credential(
      currentEmail,
      formData.curPassword
    );
    await reauthenticateWithCredential(user, credential);
    await updatePassword(user, formData.newPassword);
  }

  setShowPopup(true);
} catch (error) {
  console.error("Error updating profile:", error.message);
  if (
    error.code === "auth/invalid-credential"
  ) {
    setErrors((prevErrors) => ({
      ...prevErrors,
      curPassword: "Invalid password",
    }));
```

```
      } else if (error.code === "auth/too-many-requests") {
        setErrors((prevErrors) => ({
          ...prevErrors,
          curPassword: "Too many requests. Try again later",
        }));
      } else if (error.code === "auth/missing-password")
        setErrors((prevErrors) => ({
          ...prevErrors,
          curPassword: "Enter your password",
        }));
    }
  }
};

const handleSignOut = async () => {
  try {
    navigate("/");
    await auth.signOut();
  } catch (error) {
    console.error("Error signing out:", error.message);
  }
};

const handleDeleteAccount = async () => {
  try {
    const credential = EmailAuthProvider.credential(
      currentEmail,
      formData.curPassword
    );
    await reauthenticateWithCredential(user, credential);
```

```
    await deleteDoc(doc(db, "users", uid));
    navigate("/");
    await user.delete();
  } catch (error) {
    console.error("Error deleting account:", error.message);
    if (
      error.code === "auth/invalid-credential"
    ) {
      setErrors((prevErrors) => ({
        ...prevErrors,
        curPassword: "Invalid password",
      }));
    } else if (error.code === "auth/missing-password") {
      setErrors((prevErrors) => ({
        ...prevErrors,
        curPassword: "Enter your password",
      }));
    } else if (error.code === "auth/too-many-requests") {
      setErrors((prevErrors) => ({
        ...prevErrors,
        curPassword: "Too many requests. Try again later",
      }));
    }
  }
}

return (
  <div className={classes.profileInfoContainer}>
    {!auth.currentUser.emailVerified ? (
      <div className={classes.verifyEmail}>
```

```
    <p>
      Verify your email to get all features
    </p>
  </div>
) : null}
<div className={classes.profileInfoHeader}>
  <h2>Your Profile</h2>
</div>
<form className={classes.profileInfoForm} onSubmit={handleSubmit}>
  <label className={classes.profileInfoLabel}>
    Username:
    <input
      className={classes.profileInfoInput}
      type="text"
      name="username"
      value={formData.username}
      onChange={handleChange}
      placeholder={username}
      autoComplete="username"
    />
    <p className={classes.error}>{errors.username}</p>
  </label>

  <label className={classes.profileInfoLabel}>
    Email:
    <input
      className={classes.profileInfoInput}
      type="email"
      name="email"
      value={formData.email}
```

```
    onChange={handleChange}
    placeholder={email}
    autoComplete="username"
   />
   <p className={classes.error}>{errors.email}</p>
  </label>

  <div className={classes.password}>
   <label className={classes.passwordInfoLabel}>
    Current Password:
    <input
     className={classes.passwordInput}
     type="password"
     name="curPassword"
     value={formData.curPassword}
     onChange={handleChange}
     placeholder={"Enter your current password"}
     autoComplete="current-password"
    />
    <p className={classes.error}>{errors.curPassword}</p>
   </label>
   <label className={classes.passwordInfoLabel}>
    New Password:
    <input
     className={classes.passwordInput}
     type="password"
     name="newPassword"
     value={formData.newPassword}
     onChange={handleChange}
     placeholder={"Enter your new password"}
```

```
      autoComplete="new-password"
    />
    <p className={classes.error}>{errors.newPassword}</p>
  </label>
</div>

<p className={classes.paragraph}>
  Would you like to be notified when the exchange rate changes? Enter
  the minimum values next to the required currencies, when changing to
  which the email will be sent
</p>
<div className={classes.currBlock}>
  <label className={classes.currencyInfoLabel}>
    USD:
    <input
      className={classes.currency}
      type="text"
      name="USD"
      value={formData.currencyDiff.USD}
      onChange={handleChange}
      disabled={!user.emailVerified}
    />
    <p className={classes.errorCur}>{errors.usd}</p>
  </label>

  <label className={classes.currencyInfoLabel}>
    EUR:
    <input
      className={classes.currency}
      type="text"
```

```
    name="EUR"
    value={formData.currencyDiff.EUR}
    onChange={handleChange}
    disabled={!user.emailVerified}
   />
   <p className={classes.errorCur}>{errors.eur}</p>
  </label>

  <label className={classes.currencyInfoLabel}>
   PLN:
   <input
    className={classes.currency}
    type="text"
    name="PLN"
    value={formData.currencyDiff.PLN}
    onChange={handleChange}
    disabled={!user.emailVerified}
   />
   <p className={classes.errorCur}>{errors.pln}</p>
  </label>

  <label className={classes.currencyInfoLabel}>
   GBP:
   <input
    className={classes.currency}
    type="text"
    name="GBP"
    value={formData.currencyDiff.GBP}
    onChange={handleChange}
    disabled={!user.emailVerified}
```

```
    />
    <p className={classes.errorCur}>{errors.gbp}</p>
  </label>
</div>

<div className={classes.btnDiv}>
  <button className={classes.profileInfoSubmitBtn} type="submit">
    Save Changes
  </button>

  {showPopup && (
    <div className={classes.popup}>
      <p>Profile info successfully updated.</p>
    </div>
  )}
</div>

<div className={classes.profileInfoFooter}>
  <button
    className={classes.profileInfoSignoutBtn}
    onClick={handleSignOut}
    type="button"
  >
    Sign Out
  </button>
  <button
    className={classes.profileInfoDeleteBtn}
    onClick={handleDeleteAccount}
    type="button"
  >
```

```
            Delete account
          </button>
        </div>

      </form>
    </div>
  );
};

export default ProfileInfo;
```

Файл Header.jsx

```
import { Link } from "react-router-dom";
import classes from "./Header.module.css";
import logo from "../../assets/icons/logo.svg";
import { useAuth } from "../../hooks/useAuth";
import { getAuth } from "firebase/auth";
import { useState } from "react";
import { useEffect } from "react";
import { useNavigate } from "react-router";

const Header = () => {
  const user = useAuth();
  const auth = getAuth();
  const userAuth = auth.currentUser;
  const navigate = useNavigate();

  const [error, setError] = useState("");
  const [showPopup, setShowPopup] = useState(false);
```

```
useEffect(() => {
  if (showPopup) {
    const timeoutId = setTimeout(() => {
      setShowPopup(false);
    }, 5000);

    return () => clearTimeout(timeoutId);
  }
}, [showPopup]);

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

  if (!userAuth.emailVerified) {
    setError("Please verify your email first!");
    setShowPopup(true);
  } else {
    setError("");
    navigate("/start-exchange");
  }
};

return (
  <header className={classes.header}>
    {showPopup && <div className={classes.popup}>{error}</div>}

    {user.isAuth ? (
      user.isAdmin ? (
        <div className={classes.leftSide}>
          <Link to="/">
```

```
        <img className={classes.logo} src={logo} />
      </Link>
      <Link to="/">Home</Link>
      <Link to="/start-exchange" onClick={handleStartExchange}>
        Start Exchange
      </Link>
      <Link to="/exchange-requests">Exchange Requests</Link>
    </div>
  ) : (
    <div className={classes.leftSide}>
      <Link to="/">
        <img className={classes.logo} src={logo} />
      </Link>
      <Link to="/">Home</Link>
      <Link to="/start-exchange" onClick={handleStartExchange}>
        Start Exchange
      </Link>
    </div>
  )
) : (
  <div className={classes.leftSide}>
    <Link to="/">
      <img className={classes.logo} src={logo} />
    </Link>
    <Link to="/">Home</Link>
  </div>
)}

{user.isAuth ? (
  <div className={classes.rightSide}>
```

```
          <Link to="/profile">Profile</Link>
        </div>
      ) : (
        <div className={classes.rightSide}>
          <Link to="/signin">Sign In</Link>
          <Link to="/signup">Sign Up</Link>
        </div>
      )}
    </header>
  );
};

export default Header;
```

Файл Footer.jsx

```
import classes from "./Footer.module.css";

const Footer = () => {
  return (
    <footer className={classes.footer}>
      <div className={classes.footerContent}>
        <div className={classes.footerSection}>
          <h3>About Us</h3>
          <p>
            CashFlow Exchange offers the ultimate in simplified interaction
            between the individual and the point of exchange. You can track the
            latest and best exchange rates, create exchange orders and receive
            notifications of rate updates
          </p>
        </div>
```

```
        <div className={classes.footerSection}>
          <h3>Contact Us</h3>
          <p>Email: cash.flow.exchange.bsn@gmail.com</p>
          <p>Phone: +38 (095) 68-34-322</p>
        </div>
      </div>

      <div className={classes.footerBottom}>
        <p>&copy; 2023 CashFlow Exchange. All rights reserved.</p>
      </div>
    </footer>
  );
};

export default Footer;
```

Файл CurrencyRates.jsx

```
import { useEffect, useState } from "react";

import classes from "./CurrencyRates.module.css";
import { db } from "../../firebase";
import { doc } from "firebase/firestore";
import { getDoc } from "firebase/firestore";

import usdIcon from "../../assets/icons/US.svg";
import eurIcon from "../../assets/icons/EU.svg";
import plnIcon from "../../assets/icons/PL.svg";
import gbpIcon from "../../assets/icons/UK.svg";
```

```
const currencyIcons = {
  USD: usdIcon,
  EUR: eurIcon,
  PLN: plnIcon,
  GBP: gbpIcon,
};

const CurrencyRates = () => {
  const [rates, setRates] = useState("");

  useEffect(() => {
    const fetchRates = async () => {
      try {
        const ratesDoc = doc(db, "rates", "latestRates");
        const docSnapshot = await getDoc(ratesDoc);

        if (docSnapshot.exists()) {
          const ratesData = docSnapshot.data();
          setRates(ratesData);
        }
      } catch (error) {
        console.log(`Error fetching currency rates: ${error.message}`);
      }
    };

    fetchRates();
  }, []);

  return (
    <section className={classes.mainCon}>
```

```
      <h2 className={classes.header}>Exchange rates</h2>
      <table className={classes.table}>
        <thead>
          <tr className={classes.tableRow}>
            <th className={classes.tableHeader}>Currency</th>
            <th className={classes.tableHeader}>Rate</th>
          </tr>
        </thead>
        <tbody>
          {Object.entries(rates).map(([currency, rate]) => (
            <tr className={classes.tableRow} key={currency}>
              <td className={classes.tableFields}>
                <div className={classes.currencyInfo}>
                  <img
                    className={classes.flag}
                    src={currencyIcons[currency]}
                    alt={currency}
                  />
                  <span>{currency}</span>
                </div>
              </td>
              <td className={classes.tableFields}>{rate.toFixed(2)}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </section>
  );
};
```

```
export default CurrencyRates;
```

Файл CurrencyCalculator.jsx

```
import { useEffect, useState } from "react";
import { db } from "../../firebase";
import { doc, getDoc } from "firebase/firestore";
import classes from "./CurrencyCalculator.module.css";
import { serverTimestamp } from "firebase/firestore";
import { useAuth } from "../../hooks/useAuth";
import { collection, addDoc } from "firebase/firestore";
import exchangeSvg from "../../assets/icons/exchange.svg";

const CurrencyCalculator = ({ showCreateButton }) => {
  const [currencies, setCurrencies] = useState([]);
  const [sourceCurrency, setSourceCurrency] = useState("");
  const [targetCurrency, setTargetCurrency] = useState("");
  const [amount, setAmount] = useState("");
  const [convertedAmount, setConvertedAmount] = useState("");
  const [rates, setRates] = useState({});

  const user = useAuth();

  useEffect(() => {
    const fetchCurrencies = async () => {
      const ratesDoc = doc(db, "rates", "latestRates");
      const docSnapshot = await getDoc(ratesDoc);

      if (docSnapshot.exists()) {
        const ratesData = docSnapshot.data();
        const availableCurrencies = Object.keys(ratesData);
```

```
      if (!availableCurrencies.includes("UAH")) {
        availableCurrencies.unshift("UAH");
        ratesData["UAH"] = 1;
      }

      setCurrencies(availableCurrencies);
      setSourceCurrency(availableCurrencies[0]);
      setTargetCurrency(availableCurrencies[1]);
      setRates(ratesData);
    }
  };

  fetchCurrencies();
}, []);

useEffect(() => {
  if (sourceCurrency && targetCurrency && amount !== "") {
    const sourceRate = rates[sourceCurrency];
    const targetRate = rates[targetCurrency];
    const convertedValue = ((amount * sourceRate) / targetRate).toFixed(2);
    setConvertedAmount(convertedValue);
  } else {
    setConvertedAmount("");
  }
}, [sourceCurrency, targetCurrency, amount, rates]);

const handleAmountChange = (e) => {
  const inputAmount = e.target.value;
```

```
  if (inputAmount === "" || /^[1-9]\d*\.?\d*$/.test(inputAmount)) {
    setAmount(inputAmount);
  }
};

const handleCurrencyChange = (e, type) => {
  const selected = e.target.value;
  if (type === "source") {
    setSourceCurrency(selected);
  } else {
    setTargetCurrency(selected);
  }
};

const handleExchangeClick = () => {
  setSourceCurrency(targetCurrency);
  setTargetCurrency(sourceCurrency);
};

const handleCreateRequest = async () => {
  if (amount !== "" && sourceCurrency !== targetCurrency) {
    try {
      const requestsCollection = collection(db, "requests");

      await addDoc(requestsCollection, {
        sourceCurrency,
        targetCurrency,
        amount: (parseFloat(amount) + parseFloat(amount / 100)).toFixed(2),
        convertedAmount: parseFloat(convertedAmount),
        userId: user.uid,
```

```
        timestamp: serverTimestamp(),
        status: "pending",
      });

      console.log("Request created successfully!");
    } catch (error) {
      console.error("Error creating request:", error.message);
    }
  }
};

return (
  <div className={classes.currencyCalculator}>
    <div className={classes.calc}>
      <div className={classes.currencyBlock}>
        <select
          className={classes.currencySelectLeft}
          value={sourceCurrency}
          onChange={(e) => handleCurrencyChange(e, "source")}
        >
          {currencies.map((currency) => (
            <option key={currency} value={currency}>
              {currency}
            </option>
          ))}
        </select>
        <input
          className={classes.currencyInput}
          type="text"
          value={amount}
```

```
    onChange={handleAmountChange}
    placeholder="Enter amount"
   />
  </div>
  <img
   src={exchangeSvg}
   className={classes.exchangeBtn}
   alt="Exchange"
   onClick={handleExchangeClick}
  />
  <div className={classes.currencyBlock}>
   <select
    className={classes.currencySelectRight}
    value={targetCurrency}
    onChange={(e) => handleCurrencyChange(e, "target")}
   >
    {currencies.map((currency) => (
     <option className={classes.option} key={currency} value={currency}>
      {currency}
     </option>
    ))}
   </select>
   <input
    className={classes.currencyResult}
    type="text"
    value={convertedAmount}
    readOnly
    placeholder="You receive"
   />
  </div>
```

```
        <div className={classes.currencyBlock}>
          {showCreateButton ? (
            <button
              className={classes.createRequestButton}
              onClick={handleCreateRequest}
            ></button>
          ) : null}
        </div>
      </div>
      <div>
        <p className={classes.info}>
          Comission for exchange is 1% of the amount: {(amount / 100).toFixed(2)}
{sourceCurrency}
        </p>
      </div>
    </div>
  );
};

export default CurrencyCalculator;
```

Файл index.js

```
require("dotenv").config();
const functions = require("firebase-functions");
const admin = require("firebase-admin");
const axios = require("axios");
const nodemailer = require("nodemailer");
const smtpTransport = require("nodemailer-smtp-transport");

admin.initializeApp();
```

```
exports.fetchRatesAndWriteToFirestore = functions.pubsub
  .schedule("0 12 * * *")
  .timeZone("UTC")
  .onRun(async (context) => {
    try {
      const endpoint = "latest";
      const baseCurrency = "UAH";
      const response = await axios.get(
        `https://v6.exchangerate-api.com/v6/${endpoint}/${baseCurrency}`,
        {
          headers: {
            Authorization: `Bearer
${process.env.VITE_EXCHANGE_RATES_API_KEY}`,
          },
        }
      );
      const data = response.data;

      const requiredCurrencies = ["USD", "EUR", "PLN", "GBP"];

      const filteredRates = Object.fromEntries(
        Object.entries(data.conversion_rates).filter(([currency]) =>
          requiredCurrencies.includes(currency)
        )
      );

      const reciprocalRates = Object.fromEntries(
        Object.entries(filteredRates).map(([currency, rate]) => [
          currency,
```

```
    1 / rate,
  ])
);

console.log(reciprocalRates);
console.log(reciprocalRates);
console.log(reciprocalRates);

// reciprocalRates = fetchTest();

const ratesDoc = admin.firestore().collection("rates").doc("latestRates");
const docSnapshot = await ratesDoc.get();

const {
  USD = null,
  EUR = null,
  PLN = null,
  GBP = null,
} = docSnapshot.data() || {};

const usersSnapshot = await admin.firestore().collection("users").get();

const sendEmailPromises = [];

usersSnapshot.forEach(async (userDoc) => {
  const userData = userDoc.data();
  const currencyDiff = userData.currencyDiff || {};

  console.log(currencyDiff);
```

```
const newRates = {
  USD: reciprocalRates.USD || null,
  EUR: reciprocalRates.EUR || null,
  PLN: reciprocalRates.PLN || null,
  GBP: reciprocalRates.GBP || null,
};

const rateDifferences = {
  USD: newRates.USD - USD,
  EUR: newRates.EUR - EUR,
  PLN: newRates.PLN - PLN,
  GBP: newRates.GBP - GBP,
};

console.log(rateDifferences);

const emailMessages = Object.entries(rateDifferences)
  .filter(
    ([currency, diff]) =>
      Math.abs(diff) >= currencyDiff[currency] &&
      currencyDiff[currency] !== null
  )
  .map(([currency, diff]) => {
    return `${currency} rate changed by ${diff.toFixed(
      2
    )}\nOld rate: ${docSnapshot
      .data()
      [currency].toFixed(2)}\nNew rate: ${newRates[currency].toFixed(
      2
    )}\n`;
```

```
        });

        const message = emailMessages.join("");

        console.log(message);

        if (message) {
          sendEmailPromises.push(
            sendEmailNotification(userData.email, message)
          );
        }
      });

      await Promise.all(sendEmailPromises);

      const ratesRef = admin.firestore().collection("rates").doc("latestRates");
      await ratesRef.set(reciprocalRates);

      console.log("Rates fetched and written to Firestore successfully.");
      return null;
    } catch (error) {
      console.error("Error fetching and writing rates:", error.message);
      return null;
    }
  });

async function sendEmailNotification(email, message) {
  console.log(process.env.GMAIL_ADDRESS,
process.env.GMAIL_PASSWORD);
  const transporter = nodemailer.createTransport(
```

```
    smtpTransport({
      host: "smtp.gmail.com",
      port: 465,
      secure: true,
      service: "gmail",
      auth: {
        user: process.env.GMAIL_ADDRESS,
        pass: process.env.GMAIL_PASSWORD,
      },
    })
  );

  const mailOptions = {
    from: process.env.GMAIL_ADDRESS,
    to: email,
    subject: "Exchange Rate Notification",
    text: message,
  };

  await transporter.sendMail(mailOptions, (error, info) => {
    if (error) {
      return console.log(error);
    }
    console.log("Message sent: " + info.response);
  });
}

// function fetchTest()
// {
//   return {
```

```
//     USD: 38,
//     EUR: 42,
//     PLN: 9,
//     GBP: 46,
//   }
// }
```