NORTH SOUTH UNIVERSITY
Center of Excellence in Higher Education

CSE299: Junior Design

Week 04 Progress Report

Submitted to: Tanzilah Noor Shabnam
Submitted on: 24/10/2024
Team Name: Machine Minds
Project Title: Flood Guard

Task completed: Machine Learning, Login Functionality, Donation list

Team members

Name	ID
Sabbir Hossain	2131272042
Md. Misbah Khan	2132086642
Aritra Islam Saswato	2132629642

Languages and Tools

Programming language: Python, Javascript

Framework:

Name	Language
Jupiter Notebook	Python
MySQL	SQL
React	Javascript

Middleware:

Name	**Use**	**Links**
Leaflet	for map location and direction	https://leafletjs.com/
Firebase	User Authentication	https://firebase.google.com/

Collaboration and storing code: GitHub

Contribution:

Name	**Contribution**
Md. Misbah Khan	Login functionality, modal, Data collection and scraping
Sabbir Hossain	Data Reading, Data Prepossessing, Feature Engineering, Data visualization, Showing Correlation Between Cleaned Data, Model Training, Scaling, Testing, and Evaluating and Checking Accuracy.
Aritra Islam Saswato	Update Profile UI and Functionality, Backend bug fix for creating user

External API:

Name	**Use**	**Links**
Open Cage	converting latitude and longitude to location name	https://opencagedata.com/api
Open Street Map API v0.6	Accessing and editing map data	https://wiki.openstreetmap.org/wiki/API_v0.6

Snippets:

```
[1] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
    from sklearn.linear_model import LogisticRegression
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.svm import SVC
```

```
flood_data = pd.read_csv('/content/FloodPrediction.csv')
```

```
[3] print(flood_data.head())
    print(flood_data.info())
    print(flood_data.describe())
```

```
   Sl Station_Names  Year  Month  Max_Temp  Min_Temp  Rainfall  \
0   0       Barisal  1949      1      29.4      12.3       0.0
1   1       Barisal  1949      2      33.9      15.2       9.0
2   2       Barisal  1949      3      36.7      20.2       8.0
3   3       Barisal  1949      4      33.9      23.9     140.0
4   4       Barisal  1949      5      35.6      25.0     217.0

   Relative_Humidity  Wind_Speed  Cloud_Coverage  Bright_Sunshine  \
0               68.0    0.453704             0.6         7.831915
1               63.0    0.659259             0.9         8.314894
2               59.0    1.085185             1.5         8.131915
3               71.0    1.772222             3.9         8.219149
4               76.0    1.703704             4.1         7.046809

   Station_Number     X_COR     Y_COR  LATITUDE  LONGITUDE  ALT   Period  \
0           41950  536809.8  510151.9      22.7      90.36    4  1949.01
1           41950  536809.8  510151.9      22.7      90.36    4  1949.02
2           41950  536809.8  510151.9      22.7      90.36    4  1949.03
3           41950  536809.8  510151.9      22.7      90.36    4  1949.04
4           41950  536809.8  510151.9      22.7      90.36    4  1949.05

   Flood?
0     NaN
1     NaN
2     NaN
3     NaN
4     NaN
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20544 entries, 0 to 20543
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Sl                 20544 non-null  int64
 1   Station_Names      20544 non-null  object
 2   Year               20544 non-null  int64
 3   Month              20544 non-null  int64
 4   Max_Temp           20544 non-null  float64
 5   Min_Temp           20544 non-null  float64
 6   Rainfall           20544 non-null  float64
 7   Relative_Humidity  20544 non-null  float64
 8   Wind_Speed         20544 non-null  float64
 9   Cloud_Coverage     20544 non-null  float64
 10  Bright_Sunshine    20544 non-null  float64
 11  Station_Number     20544 non-null  int64
 12  X_COR              20544 non-null  float64
 13  Y_COR              20544 non-null  float64
 14  LATITUDE           20544 non-null  float64
 15  LONGITUDE          20544 non-null  float64
 16  ALT                20544 non-null  int64
 17  Period             20544 non-null  float64
 18  Flood?             4493 non-null   float64
dtypes: float64(13), int64(5), object(1)
memory usage: 3.0+ MB
None
                 Sl          Year         Month      Max_Temp      Min_Temp  \
count  20544.000000  20544.000000  20544.000000  20544.000000  20544.000000
mean   10271.500000   1985.332944      6.500000     33.450739     21.166872
std     5930.686301     17.610799      3.452137      2.956401      4.949587
min        0.000000   1948.000000      1.000000     21.600000      6.200000
25%     5135.750000   1972.000000      3.750000     31.700000     16.900000
50%    10271.500000   1987.000000      6.500000     33.900000     23.400000
75%    15407.250000   2000.000000      9.250000     35.400000     25.400000
max    20543.000000   2013.000000     12.000000     44.000000     28.100000

            Rainfall  Relative_Humidity    Wind_Speed  Cloud_Coverage  \
count  20544.000000       20544.000000  20544.000000    20544.000000
mean     198.776621          79.497375      1.415049        3.485827
std      240.693197           7.667925      1.042454        2.083791
min        0.000000          34.000000      0.000000        0.000000
25%        8.000000          75.000000      0.700000        1.600000
50%      111.000000          81.000000      1.200000        3.300000
75%      312.000000          85.000000      1.900000        5.500000
max     2072.000000          97.000000     11.200000        7.900000

       Bright_Sunshine  Station_Number          X_COR          Y_COR  \
count     20544.000000    20544.000000   20544.000000   20544.000000
mean          6.419056    41935.098131  549703.189176  579280.955958
std           1.747959       36.516932  116032.076255  130616.053201
min           0.000000    41859.000000       0.000000       0.000000
25%           4.965517    41909.000000  435303.700000  504500.300000
50%           6.800000    41941.000000  540098.600000  561770.300000
75%           7.800000    41963.000000  650012.100000  687095.900000
max          11.000000    41998.000000  734765.400000  844822.300000

            LATITUDE     LONGITUDE           ALT        Period       Flood?
count  20544.000000  20544.000000  20544.000000  20544.000000  4493.000000
mean      23.326647     90.493193     13.357477   1985.397944     0.919653
std        1.155241      1.108720     13.529530     17.610832     0.271860
min       20.870000     88.560000      0.000000   1948.010000     0.000000
25%       22.640000     89.550000      4.000000   1972.050000     1.000000
50%       23.170000     90.410000      7.000000   1987.040000     1.000000
75%       24.290000     91.460000     19.000000   2000.092500     1.000000
max       25.720000     92.260000     63.000000   2013.120000     1.000000
```

Machine Learning 2 [Data Reading]

```
[4] print(flood_data.shape)
```

```
(20544, 19)
```

```
[5] flood_data_cleaned = flood_data.copy()
```

```
[6] print("NaN values in 'Flood?':", flood_data['Flood?'].isna().sum())
    flood_data_cleaned = flood_data_cleaned.fillna(0)
    print("Data shape after cleaning:", flood_data_cleaned.shape)
    print("NaN values in 'Flood?' after replacing:", flood_data_cleaned['Flood?'].isna().sum())
```

```
NaN values in 'Flood?': 16051
Data shape after cleaning: (20544, 19)
NaN values in 'Flood?' after replacing: 0
```

```
[7] flood_data_cleaned.head()
```

	Sl	Station_Names	Year	Month	Max_Temp	Min_Temp	Rainfall	Relative_Humidity	Wind_Speed	Cloud_Coverage	Bright_Sunshine	Station_Number	X_COR	Y_C(
0	0	Barisal	1949	1	29.4	12.3	0.0	68.0	0.453704	0.6	7.831915	41950	536809.8	510151
1	1	Barisal	1949	2	33.9	15.2	9.0	63.0	0.659259	0.9	8.314894	41950	536809.8	510151
2	2	Barisal	1949	3	36.7	20.2	8.0	59.0	1.085185	1.5	8.131915	41950	536809.8	510151
3	3	Barisal	1949	4	33.9	23.9	140.0	71.0	1.772222	3.9	8.219149	41950	536809.8	510151
4	4	Barisal	1949	5	35.6	25.0	217.0	76.0	1.703704	4.1	7.046809	41950	536809.8	510151

Machine Learning 3 [Data Preprocessing]

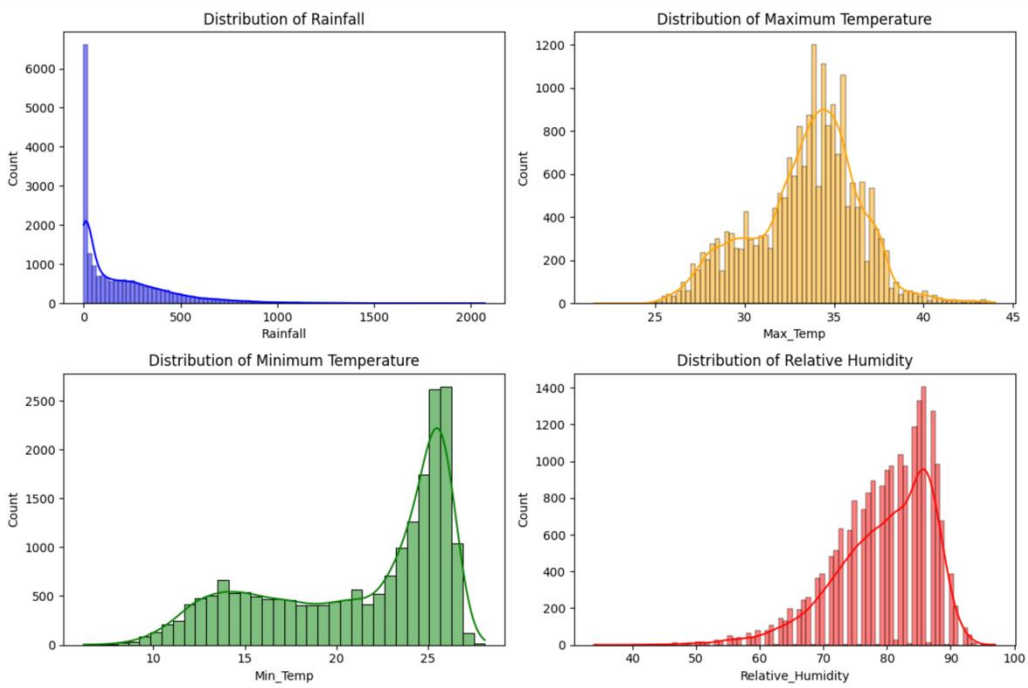
```
[8] plt.figure(figsize=(12, 8))
    plt.subplot(2, 2, 1)
    sns.histplot(flood_data_cleaned['Rainfall'], kde=True, color='blue')
    plt.title('Distribution of Rainfall')

    plt.subplot(2, 2, 2)
    sns.histplot(flood_data_cleaned['Max_Temp'], kde=True, color='orange')
    plt.title('Distribution of Maximum Temperature')

    plt.subplot(2, 2, 3)
    sns.histplot(flood_data_cleaned['Min_Temp'], kde=True, color='green')
    plt.title('Distribution of Minimum Temperature')

    plt.subplot(2, 2, 4)
    sns.histplot(flood_data_cleaned['Relative_Humidity'], kde=True, color='red')
    plt.title('Distribution of Relative Humidity')

    plt.tight_layout()
    plt.show()
```



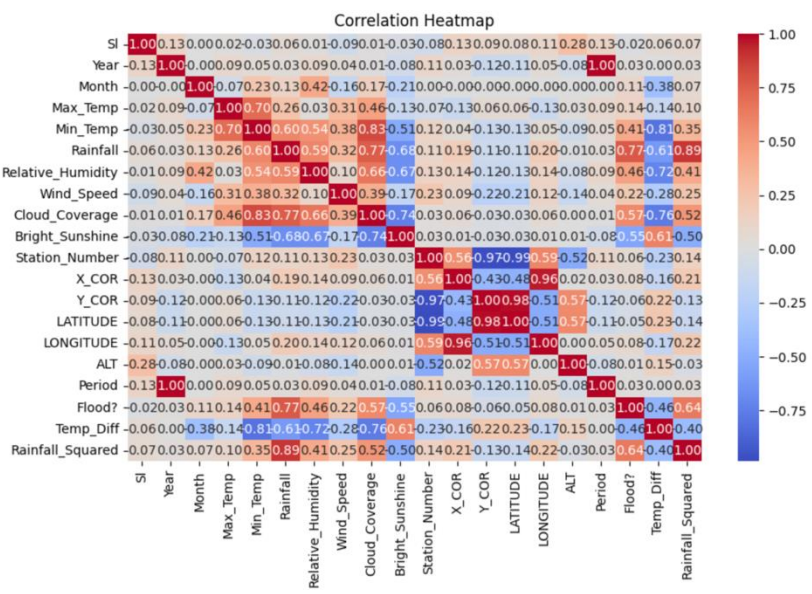
Machine Learning 4 [Data Visualization after Preprocess]

Feature Engineering

```
[9] flood_data_cleaned['Temp_Diff'] = flood_data_cleaned['Max_Temp'] - flood_data_cleaned['Min_Temp']
    flood_data_cleaned['Rainfall_Squared'] = flood_data_cleaned['Rainfall'] ** 2
```

```
[10] numerical_data = flood_data_cleaned.select_dtypes(include=['number'])

     plt.figure(figsize=(10, 6))
     correlation_matrix = numerical_data.corr()
     sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
     plt.title('Correlation Heatmap')
     plt.show()
```



Machine Learning 5 [Feature Engineering, and showing the Correlation between the data]

```
[11] X = flood_data_cleaned[['Max_Temp', 'Min_Temp', 'Rainfall', 'Relative_Humidity',
                             'Wind_Speed', 'Cloud_Coverage', 'Bright_Sunshine',
                             'Temp_Diff', 'Rainfall_Squared']]
     y = flood_data_cleaned['Flood?']
```

```
[12] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[13] scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)
```

```
[14] def plot_confusion_matrix(y_true, y_pred, model_name):
         cm = confusion_matrix(y_true, y_pred)
         plt.figure(figsize=(6, 4))
         sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
         plt.title(f"Confusion Matrix for {model_name}")
         plt.ylabel('Actual')
         plt.xlabel('Predicted')
         plt.show()
```

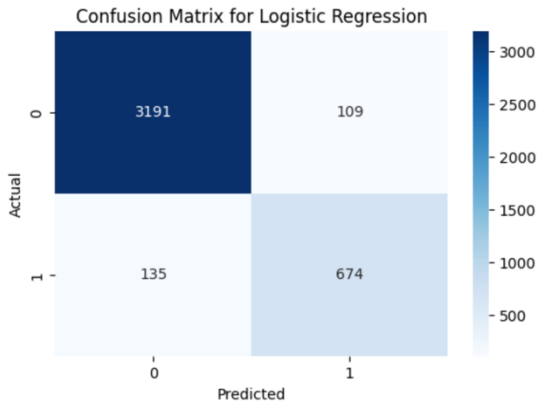
```
[15] def evaluate_model(y_true, y_pred, model_name):
         print(f"---{model_name}---")
         print(f"Accuracy: {accuracy_score(y_true, y_pred):.4f}")
         print(classification_report(y_true, y_pred))
         plot_confusion_matrix(y_true, y_pred, model_name)
```

```
logistic_model = LogisticRegression()
logistic_model.fit(X_train_scaled, y_train)
y_pred_logistic = logistic_model.predict(X_test_scaled)
evaluate_model(y_test, y_pred_logistic, "Logistic Regression")
```

```
---Logistic Regression---
Accuracy: 0.9406
              precision    recall  f1-score   support

         0.0       0.96      0.97      0.96      3300
         1.0       0.86      0.83      0.85       809

    accuracy                           0.94      4109
   macro avg       0.91      0.90      0.90      4109
weighted avg       0.94      0.94      0.94      4109
```



Machine Learning 6 [Logistic Regression Model Training, Scaling, Testing, and checking accuracy]

```python
import React, { useEffect, useState } from "react";
import { FaEdit } from "react-icons/fa"; // Import an edit icon from react-icons

const Profile = () => {
  const [userData, setUserData] = useState({});
  const [editableData, setEditableData] = useState({});
  const [isEditing, setIsEditing] = useState(false);

  useEffect(() => {
    const storedData = sessionStorage.getItem("userData");
    if (storedData) {
      const parsedData = JSON.parse(storedData);
      setUserData(parsedData);
      setEditableData(parsedData); // Initialize editableData with userData
    }
  }, []);

  // Prepare profile picture URL
  const profilePictureUrl = `https://avatar.iran.liara.run/username?username=${userData.name || "User"}`;

  // Function to handle input change
  const handleChange = (key, value) => {
    setEditableData((prevData) => ({
      ...prevData,
      [key]: value,
    }));
  };

  // Function to handle edit toggle
  const handleEditToggle = () => {
    setIsEditing((prev) => !prev);
  };

  // Function to handle update API call
  const handleUpdate = async () => {
    const apiUrl =
      userData.role === "victim"
        ? "http://localhost:3000/api/victims/"
        : "http://localhost:3000/api/rescuers/";

    try {
      const response = await fetch(apiUrl, {
        method: "PUT", // Use PUT or PATCH as necessary
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify(editableData),
      });
      if (response.ok) {
        // Handle successful update (e.g., show a success message)
        console.log("Profile updated successfully");
        setUserData(editableData); // Update the displayed userData
        handleEditToggle(); // Close the edit mode
      } else {
        console.error("Failed to update profile");
      }
    } catch (error) {
      console.error("Error updating profile:", error);
    }
  };

  return (
    <div className="w-10/12 mx-auto my-12 p-6 rounded-xl shadow-lg bg-secondary">
      <h2 className="text-3xl font-light text-center my-6">Profile</h2>

      <div className="flex flex-col items-center mb-6">
        <img
          src={profilePictureUrl}
          alt="Profile"
          className="w-32 h-32 rounded-full border-4 border-primary mb-4"
        />
        <span className="text-3xl font-semibold text-primary capitalize">
          {userData.name || "N/A"}
        </span>
      </div>

      {/* User data displayed in a table */}
      <table class="w-full table-auto border-collapse">
        <thead>
          <tr>
            <th class="p-2 text-left text-primary">Field</th>
            <th class="p-2 text-right text-primary">Value</th>
          </tr>
        </thead>
        <tbody>
          {Object.entries(userData).map(([key, value]) => (
            <tr key={key} class="border-b">
              <td class="p-2 text-primary capitalize">
                {key.replace(/_/g, " ")}
              </td>
              <td class="p-2 text-right text-primary">
                {userData.role !== "victim" && key === "id" ? (
                  <span>{value}</span>
                ) : isEditing ? (
                  <input
                    type="text"
                    value={editableData[key] || ""}
                    onChange={(e) => handleChange(key, e.target.value)}
                    class="border rounded p-1 text-gray-800"
                  />
                ) : (
                  <>
                    {value || "N/A"}
                    {key !== "id" && key !== "role" && (
                      <FaEdit
                        class="inline ml-2 cursor-pointer text-primary hover:text-blue-500"
                        onClick={handleEditToggle}
                      />
                    )}
                  </>
                )}
              </td>
            </tr>
          ))}
        </tbody>
      </table>

      {isEditing && (
        <div class="mt-6">
          <button
            onClick={handleUpdate}
            class="bg-primary text-white px-4 py-2 rounded hover:bg-blue-700"
          >
            Update Profile
          </button>
        </div>
      )}
    </div>
  );
};

export default Profile;
```

1:Update Profile

```python
"use client";
import React, { useContext, useState } from "react";
import { Label } from "../../../components/ui/label";
import { Input } from "../../../components/ui/input";
import { Button } from "../../../components/ui/button";
import { LabelInputContainer } from "../../../components/ui/LabelInputContainer";
import { BottomGradient } from "../../../components/ui/BottomGradient";
import { cn } from "@/lib/utils";
import { inputFields } from "./InputFields";
import { AuthContext } from "../../../context/AuthProvider";
import logger from "../../../utils/logger";
import { toast } from "sonner";
import {
  Select,
  SelectContent,
  SelectGroup,
  SelectItem,
  SelectLabel,
  SelectTrigger,
  SelectValue,
} from "../../../components/ui/select";
import { useNavigate } from "react-router-dom";
import axios from "axios"; // Add Axios for API calls

const Login = () => {
  const { login } = useContext(AuthContext);
  const [role, setRole] = useState("");
  const navigate = useNavigate();

  const handleSelectChange = (value) => {
    setRole(value);
  };

  const fetchUserDataByEmail = async (email) => {
    let apiUrl = "";
    if (role === "rescuer") {
      apiUrl = `http://localhost:3000/api/rescuers/${email}`; // Replace with your actual API route
    } else if (role === "victim") {
      apiUrl = `http://localhost:3000/api/victims/${email}`; // Replace with your actual API route
    }

    try {
      const response = await axios.get(apiUrl);
      return response.data; // Return the user data
    } catch (error) {
      console.error(error);
      toast.error("Error fetching user data");
      return null;
    }
  };

  const handleLogin = async (event) => {
    event.preventDefault();
    const form = event.target;
    const email = form.email.value;
    const password = form.password.value;

    if (!role) {
      toast.error("Please select a role.");
      return;
    }

    try {
      const result = await login(email, password);
      const user = result.user;

      if (user) {
        const userData = await fetchUserDataByEmail(email); // Fetch user data
        if (userData) {
          userData.role = role;
          sessionStorage.setItem("userData", JSON.stringify(userData));
          toast.success("User logged in successfully");

          if (role === "rescuer") {
            navigate("/rescue-portal");
          } else if (role === "victim") {
            navigate("/victim-portal");
          } else {
            navigate("/login");
          }
        }
      }
    } catch (err) {
      console.error(err);
      toast.error("Login failed");
      logger(`Error during login: ${err.message}`, "error", {
        color: "#ff0000",
        backgroundColor: "#f8d7da",
      });
    }
  };

  return (
    <div className="max-w-md w-full mx-auto rounded-none md:rounded-2xl my-12 p-4 md:p-8 shadow-input bg-primary bg-opacity-15">
      <h2 className="font-light text-3xl text-center font-museo my-6 pb-6 text-primary">
        Welcome to flood guard
      </h2>
      <p className="text-center text-primary my-6 pb-6 text-xl font-light">
        Log in to your account{" "}
      </p>
      <form onSubmit={handleLogin}>
        <div className="flex flex-col space-y-2 mb-4">
          {inputFields
            .filter(({ id }) => id === "email" || id === "password")
            .map({ id, label, type, placeholder }) => (
              <LabelInputContainer key={id}>
                <Label htmlFor={id}>{label}</Label>
                <Input id={id} placeholder={placeholder} type={type} />
              </LabelInputContainer>
            ))}
        </div>
        <LabelInputContainer className="mb-4">
          <Label htmlFor="role">Select Role</Label>
          <Select onValueChange={handleSelectChange}>
            <SelectTrigger className="w-full">
              <SelectValue
                placeholder="Select a role"
                className="placeholder:text-primary"
              />
            </SelectTrigger>
            <SelectContent>
              <SelectGroup>
                <SelectLabel>--Select Role--</SelectLabel>
                <SelectItem value="victim">Victim</SelectItem>
                <SelectItem value="rescuer">Rescuer</SelectItem>
              </SelectGroup>
            </SelectContent>
          </Select>
        </LabelInputContainer>
        <Button
          variant="submit"
          className="w-full text-lg font-light text-primary"
          type="submit"
        >
          Log in
          <BottomGradient />
        </Button>

        <div className="bg-gradient-to-r from-transparent via-primary dark:via-primary to-transparent my-10 h-[1px] w-full" />

        <div className="flex flex-col space-y-4">
          <div>
            <p className="text-center text-primary font-light">
              Don't have an account?{" "}
              <a
                href="/signup"
                className={cn(
                  "text-primary font-light underline",
                  "text-primary font-light underline"
                )}
              >
                Sign up
              </a>
            </p>
          </div>
        </div>
      </form>
    </div>
  );
};

export default Login;
```

2: Login

```
import React, { createContext, useEffect, useState } from "react";
import {
  getAuth,
  createUserWithEmailAndPassword,
  onAuthStateChanged,
  signOut,
  signInWithEmailAndPassword,
} from "firebase/auth";
import app from "../firebase/firebase.config";
export const AuthContext = createContext();
const auth = getAuth(app);

export const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(null);
  const createUser = async (email, password) => {
    setLoading(true);
    try {
      const result = await createUserWithEmailAndPassword(
        auth,
        email,
        password
      );


      return result;
    } finally {
      setLoading(false);
    }
  };

  const login = async (email, password) => {
    setLoading(true);
    try {
      const result = await signInWithEmailAndPassword(auth, email, password);

      return result;
    } finally {
      setLoading(false);
    }
  };

  const logOut = async () => {
    setLoading(true);
    try {
      await signOut(auth);
    } finally {
      setLoading(false);
    }
  };

  useEffect(() => {
    const unsubscribe = onAuthStateChanged(auth, (currentUser) => {
      setUser(currentUser);
      setLoading(false);
    });

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

  const authInfo = {
    user,
    loading,
    createUser,
    login,
    logOut,
  };

  return (
    <AuthContext.Provider value={authInfo}>{children}</AuthContext.Provider>
  );
};

export default AuthProvider;
```

3: AuthProvider

```jsx
import React, { useState, useEffect } from "react";
import axios from "axios";
import { debounce } from "lodash";
import Pagination from "../../../../components/ui/Pagination";
import Table from "../../../../components/ui/table";
import DonationDetails from "./DonationDetails";

const Donations = () => {
  const [donations, setDonations] = useState([]);
  const [totalDonations, setTotalDonations] = useState(0);
  const [pageNumber, setPageNumber] = useState(0);
  const [rowsPerPage, setRowsPerPage] = useState(10);
  const [searchTerm, setSearchTerm] = useState("");
  const [searchField, setSearchField] = useState("name");
  const [sortField, setSortField] = useState("name");
  const [sortOrder, setSortOrder] = useState("asc");

  const headers = [
    { label: "Donor Name", field: "donor_name", sortable: true },
    { label: "Donation Type", field: "donation_type", sortable: true },
    { label: "Quantity", field: "quantity", sortable: true },
    { label: "Date Received", field: "date_received", sortable: true },
  ];

  const handleChangePage = (newPage) => {
    setPageNumber(newPage);
  };

  const handleChangeRowsPerPage = (event) => {
    setRowsPerPage(+event.target.value);
    setPageNumber(0);
  };

  const handleSearchInput = debounce((event) => {
    setSearchTerm(event.target.value);
    setPageNumber(0);
  }, 300);

  const handleSort = (field) => {
    const newSortOrder =
      sortField === field && sortOrder === "asc" ? "desc" : "asc";
    setSortField(field);
    setSortOrder(newSortOrder);
  };

  useEffect(() => {
    const fetchDonations = async () => {
      try {
        const offset = pageNumber * rowsPerPage;
        const response = await axios.get(
          "http://localhost:3000/api/donations",
          {
            params: {
              search: searchTerm || "",
              searchField,
              sortField,
              sortOrder,
              limit: rowsPerPage,
              offset,
            },
          }
        );
        setDonations(response.data.donations);
        setTotalDonations(response.data.totalCount);
      } catch (error) {
        console.error("Error fetching donations:", error);
      }
    };

    fetchDonations();
  }, [searchTerm, searchField, sortField, sortOrder, pageNumber, rowsPerPage]);

  const renderVictimRow = (victim) => (
    <DonationDetails key={victim.id} row={victim} />
  );

  return (
    <div className="w-full p-4">
      <div className="flex flex-col justify-between mb-4">
        <p className="text-center my-4 text-lg">Search by Name and Item</p>
        <input
          type="text"
          onChange={handleSearchInput}
          placeholder="Enter query"
          className="px-4 py-2 border w-1/2 mx-auto bg-secondary rounded-lg"
        />
      </div>

      <Table
        headers={headers}
        data={donations}
        sortField={sortField}
        sortOrder={sortOrder}
        handleSort={handleSort}
        renderRow={renderVictimRow}
      />
      <Pagination
        pageNumber={pageNumber}
        rowsPerPage={rowsPerPage}
        totalItems={totalDonations}
        onPageChange={handleChangePage}
        onRowsPerPageChange={handleChangeRowsPerPage}
      />
    </div>
  );
};

export default Donations;
```

4: Donations table