Finance Expense Tracker and Prediction Website with ARIMA Model

This website allows users to make time-series predictions using a pre-trained ARIMA model via a web-based interface. The backend is built with **Node.js** and **Flask** (for Python ARIMA model integration), and the frontend is built with **React**. MongoDB is used to store historical data and user credentials. This project is deployed on AWS cloud, and you can access the live application through the provided URL.

Table of Contents

Prerequisites

Before you begin, make sure that the following software is installed on your system:

1. Node.js (v14.0 or higher)

- Install from Node.js (https://nodejs.org/en/download/).

2. npm (Node Package Manager)

- Comes pre-installed with Node.js.

3. Python 3.x (v3.8 or higher)

- Install from Python.org (https://www.python.org/downloads/).

4. MongoDB Atlas (Cloud-based MongoDB)

- MongoDB is hosted on the cloud (MongoDB Atlas). You don't need to install it locally; just use MongoDB connection string for connecting it with your backend application.

5. Other Dependencies

- For **Flask** application (Python ARIMA Model), the required packages are listed in the `requirements.txt` file.

Project Structure

The project is structured as follows:

For NodeJS Application-

```
Folder PATH listing
Volume serial number is 005C0044 1AE2:D868
C:\USERS\CHECKOUT\FinalCodeSubmission\NodeJsBackEndApplication
|   .env
|   .gitignore
|   db.js
|   package-lock.json
|   package.json
|   README.md
|   server.js
|   userdata.py
|
+---config
|       db.js
|
+---controllers
|       arimaController.js
|       authController.js
|       reportController.js
|       settingsController.js
|       transactionController.js
|
+---middleware
|       authMiddleware.js
|
+---models
|       transactionModel.js
|       userModel.js
|
+---routes
|       arimaRoutes.js
|       authRoutes.js
|       reportRoutes.js
|       settingsRoutes.js
|       transactionRoutes.js
|
+---scripts
|       csv_to_mongo.js
|       transactions.csv
|
+---services
|       arimaService.js
|       authService.js
|       model.pkl
|       reportService.js
|       settingsService.js
|       transactionService.js
|
\---utils
        errorHandler.js
```

For React Application-

```
Folder PATH listing
Volume serial number is 0044004E 1AE2:D868
C:\USERS\CHECKOUT\FinalCodeSubmission\ReactFrontEndApplication
|   .gitignore
|   eslint.config.js
|   index.html
|   package-lock.json
|   package.json
|   README.md
|   vite.config.js
|
+---public
|       vite.svg
|
\---src
    |   App.css
    |   App.jsx
    |   index.css
    |   main.jsx
    |
    +---api
    |       index.js
    |
    +---assets
    |       react.svg
    |
    +---components
    |   |   ErrorBoundary.jsx
    |   |
    |   +---Auth
    |   |       ProtectedRoute.jsx
    |   |       SignIn.jsx
    |   |       SignUp.jsx
    |   |
    |   +---Charts
    |   |       CategoryTrendsChart.jsx
    |   |
    |   +---Dashboard
    |   |       Dashboard.jsx
    |   |
    |   +---Layout
    |   |       AppHeader.jsx
    |   |       AppNavbar.jsx
    |   |
    |   +---Reports
    |   |       Reports.jsx
    |   |
    |   +---Settings
    |   |       Settings.jsx
    |   |
    |   \---Transactions
    |           AddTransactionModal.jsx
    |           Transactions.jsx
    |
    +---context
    |       AuthContext.jsx
    |
    +---data
    |       Data_Cleaned.csv
    |       mockTransactions.js
    |
    \---utils
            dashboardUtils.js
```

For Flask Application -

```
Folder PATH listing
Volume serial number is 1AE2-D868
C:.
    arima_api.py
    arima_model.pkl
    best_arima_model_main.pkl
    output.txt
    README.md
    requirements.txt
```

For ARIMA Model Training -

```
Arima_implemetation.ipynb
```

Setup Instructions

1. Clone the Repository

Clone this repository to your local machine:

```
git clone <repository-url>
```

2. Install Node.js Dependencies (Backend)

Navigate to the `react code` directory and install the dependencies:

```
npm install
```

3. Install React Dependencies (Frontend)

Navigate to the `frontend code` directory and install the dependencies:

```
npm install
```

4. Install Python Dependencies (Flask and ARIMA Model)

Navigate to the `flask code` directory and install the Python dependencies from `requirements.txt`:

```
pip install -r requirements.txt
```

This will install all the necessary Python libraries, including Flask and `statsmodels` for the ARIMA model.

5. Setup MongoDB Atlas

The MongoDB database is hosted on **MongoDB Atlas**. You don't need to install MongoDB locally, but you need the connection string to connect your application to the cloud database.

1. Create an account on MongoDB Atlas (https://www.mongodb.com/cloud/atlas).
2. Create a cluster and a database.
3. Create a MongoDB user with appropriate access rights.
4. Create a new MongoDB Cluster
5. Copy the **connection string**
6. Add the connection string to the `.env` file in your project to ensure proper database access.

Running the Application

1. Start the Flask Application (Python Backend)

To start the Flask application (which interacts with the ARIMA model), run the following command from the `flask code` directory:

```
python arima_api.py
```

This will start the Flask server, typically running on `http://localhost:5000`, but we have configured port 5002.

2. Start the Node.js Backend

In the `nodejs code` directory, run the following command to start the Node.js server:

```
npm start
```

This will start the Node.js server, typically running on `http://localhost:3000`.

3. Start the React Application (Frontend)

In the `react code` directory, run the following command to start the React frontend:

```
npm run dev
```

This will start the React development server, typically running on `http://localhost:5001`.

Challenges and Solutions

1. **Database Connectivity**:
   MongoDB was hosted on MongoDB Atlas, which required proper connection handling and retry mechanisms. This was resolved by adding the connection string to the `.env` file and using the Mongoose library for efficient database interaction.

2. **API Communication between Flask and Node.js**:
   Initially, there were issues with correctly handling API requests from Node.js to Flask. This was resolved by ensuring proper configuration in the Flask `arima_api.py` file and setting up CORS headers to allow communication between different servers.

3. **Model Deployment**:
   Running the ARIMA model within Flask and ensuring it works as expected with Node.js to manage communication between the frontend (React) and backend (Flask).

Technologies Used

- **Backend**:

  - **Node.js** with **Express.js** for routing and API handling.
  - **Flask** (Python) to serve the ARIMA model and handle predictions.
  - **MongoDB Atlas** for cloud-based NoSQL database.
  - **Mongoose** for MongoDB interaction in Node.js.

- **Frontend**:

  - **React** for building the frontend user interface.
  - **Axios** for making HTTP requests from React to the backend.

- **Machine Learning**:

  - **Python** with **statsmodels** for ARIMA time-series prediction.

- **Version Control**:

  - **Git** for code management.

Final Deployment

The application has been deployed on the cloud and can be accessed via the following URL:

[**Access the live application here**](https://main.d1ehtloqw9iwe5.amplifyapp.com/) (https://main.d1ehtloqw9iwe5.amplifyapp.com/)

You can interact with the web interface, see the dashboard, reports, finance expenses and predictions based on number of days you select from UI.
