PROJECT REPORT- Financely

Table of Contents

Introduction

Project Overview

Financely is a personal expense tracking application designed to help individuals manage their finances efficiently. It enables users to record transactions, categorize expenses, set budgets, and visualize financial trends using dynamic charts. The platform ensures a smooth user experience with features like pagination for large datasets, personalized recommendations, and **real-time anomaly detection** to prevent fraudulent transactions.

Purpose and Motivation

In today's digital world, managing personal finances is crucial. Traditional methods such as manual logs or spreadsheets are inefficient, error-prone, and lack intelligent insights. Financely provides a modern, automated, and interactive tool to simplify financial management while incorporating fraud detection mechanisms for enhanced security.

Objectives

- Provide a user-friendly platform for **expense tracking**.
- Enable users to **categorize expenses** and set budgets.
- **Visualize spending trends** through interactive graphs.
- Implement **fraud detection using machine learning** with an accuracy of 94.63%.

- Ensure **data security** and smooth performance.

Key Features

1. Secure User Authentication

- Users register and log in securely using **JWT authentication**.
- Passwords are stored in an **encrypted format** to prevent security breaches.

2. Transaction Management

- Users can **add, edit, delete, and view** transactions.
- Transactions are **categorized** into different labels such as "Food," "Entertainment," etc.
- **Pagination** ensures smooth handling of large datasets.

3. Budget Tracking

- Users can set budgets for specific categories.
- Alerts notify users when spending approaches or exceeds the set budget.
- Budget recommendations are generated based on spending patterns.

4. Data Visualization and Insights

- **Pie Charts** display category-wise spending.
- **Bar Graphs** show monthly spending trends.
- **Statistical Insights** like total spend, average spend, and standard deviation are provided.

5. Real-Time Anomaly Detection System

- Uses **Isolation Forest** (Machine Learning) to detect unusual transactions with an accuracy of 94.63%.
- Flags suspicious transactions based on spending behaviour.
- Notifies users before adding a transaction for confirmation.
- API integration allows real-time fraud detection.

Technology Stack

Frontend:

- **HTML5, CSS3, JavaScript** - Provides a clean and responsive UI.

- **Chart.js** - Used for interactive financial visualizations.
- **Font Awesome** - Enhances UI elements with icons.

Backend:

- **Node.js with Express.js** - Handles API requests and user authentication.
- **C++ Core Engine** - Manages transactions, budget tracking, and storage.
- **Python Flask** - Used for **anomaly detection and fraud prevention**.

Database:

- **File-based storage (C++)** - Efficiently manages user transaction data.

Data Structures Used

- **Doubly Linked List** - Stores and retrieves transactions efficiently.
- **Priority Queue** - Tracks overspending alerts in real-time.
- **Hash Maps** - Stores categorized data for faster retrieval.
- **KMP Algorithm** - Optimized search for specific transaction details.

Functional Workflow

1. **Login/Register Page**: As the user enters our website, he/she encounters the login page which has the link for the register page for the first time users. The user logins or registers securely managed by the Login_system.cpp file
2. **Main Dashboard**: After the login, the user encounters the main dashboard which features few last transactions, spending pie chart, monthly spend bar graph which is managed by the get_data.cpp file
3. **Add Transactions**: This tab on click displays a form which allows the user to add a new transaction, **the transactions details are then checked for anomaly using the anomaly detection model** before getting added to the doubly linked list for further storing. This is managed by post_data.cpp file
4. **Show History**: This tab on click displays the user transactions till date. Since the number of transactions can be huge, the transactions are paginated. This is maintained by get_data.cpp file
5. **Set Budget**: This tab allows the user to set new budget which is stored and further used in the priority queue to generate user recommendations. This is managed by the BudgetNRecommender.cpp file

6. **Get Transactions**: This tab helps the user to get the transactions as per his/her needs. The user can get the data filtered by category or in a particular range of dates as per the user needs. This is managed by the get_data.cpp file
7. **Monthly Summary**: This tab helps the user to get monthly summary of his/her spending. The user can see total spend, average spend and a standard deviation of his/her spend for the month. It also features a bar graph for category wise spend. This is managed by the Statistics.cpp file

---