

**Team Members: Zayn Zaidi (615), Feilian Huang (615)**  
**zzaidi3@jhu.edu, fhuang13@jhu.edu**

**The Personal Finance Tracker is a financial management application designed to help users monitor their income, expenses, budgets, and savings goals. It enables tracking of transactions across multiple accounts, categorization of expenses, and visualization of progress toward financial targets. The system aims to provide actionable insights for better financial planning and decision-making.**

[https://github.com/EDGAhab/Database\\_Project](https://github.com/EDGAhab/Database_Project)

## Platform Details

- **Database:** MySQL
- **Server:** Node.js with Express.js
- **Frontend:** React.js

## User Guide

### 1. Setup Instructions:

- **MySQL Database:** Run `pfsql.sql` on local MySQL, named the database as `finance_track`
- Run `npm install` to install dependencies.
- Backend Server: First `cd` into the backend folder, then do the command `node index.js`.
- Start the React frontend: Use the command `npm start`.

### 2. Specific Instructions:

- This project runs on local MySQL, remember to change this field at `/backend/index.js`

```
const db = mysql.createConnection({  
  host: "localhost",  
  user: "root",  
  password: "Feilian990725",  
  database: "finance_track",  
});
```

- 
- Replace with your openai key at `src/components/NaturalLanguageSearch.js` for the natural language search.
- If the project could not run or experienced any issues, I am very happy to explain and help!! ([fhuang13@jh.edu](mailto:fhuang13@jh.edu), 4103443108) I just don't want to get point deducted if something accidentally happens to make the project not run. It works very well on my laptop...

### 3. Main Features:

- View accounts, transactions, budgets, and savings goals.
- Search data using **natural language queries**.
- Generate SQL commands dynamically.

### 4. For more information, check readme.md at the Github:

[https://github.com/EDGAhab/Database\\_Project/blob/main/README.md](https://github.com/EDGAhab/Database_Project/blob/main/README.md)

## Areas of Specialization

### 1. Natural Language Interface:

- Integration with GPT-4 API for translating user queries into SQL commands.

### 2. Particularly advanced GUI form interface :

- Direct UI Interface as a web application

## Data Extraction:

Daily transaction data was imported from a csv file. Users, accounts, budgets, investments, and savings goals data were inserted into their respective tables.

Daily transaction data was found on Kaggle

(<https://www.kaggle.com/datasets/prasad22/daily-transactions-dataset/data>) and preprocessed using the Python script preprocessing.py.

Tables, views, stored procedures, and other data were inserted into the database using the script pfsql.sql.

## Project Strengths

### 1. User friendly Front End Design:

Implemented a comprehensive, user friendly front-end website that interacts with our SQL database. The frontend could directly helps the user to

- View their personal finance analysis, including goals, progress, and the categories spent analysis.
- Edit personal information such as savings goals, amounts, and deadlines.
- Do a natural language search to make users intuitively know their information.

### 2. Natural Language Search Function:

Implemented a Natural Language search function which allows users to input a statement in english and receive results by mapping the statement to a SQL query. Then print a table based on that SQL query.

### 3. Comprehensive Data Organization:

The database schema covers essential aspects of personal finance, including users, accounts, transactions, budgets, savings goals, and investments.

### 4. Extensibility:

The use of structured foreign key relationships and modular design allows easy extension of functionality, such as adding new financial instruments or tracking currencies.

### 5. User-Focused Features:

- Tracks budgets and provides insights into category-wise spending limits.

- Supports savings goals with progress monitoring to motivate users toward financial targets.
- 6. **Automation-Ready:**  
The database structure and stored procedures facilitate automation of common financial calculations, such as monthly budget rollovers, recurring transactions, or savings goal projections.
- 7. **Performance Optimization:**  
Indexing on critical columns like TransactionDate ensures faster retrieval of transaction history for reporting and analytics.
- 8. **Financial Transparency:**  
Tracks both credit and debit transactions, providing a clear distinction between income and expenses, while supporting multi-currency accounts.
- 9. **Investment Tracking:**  
The inclusion of investment-specific data (e.g., stock symbols, current values) enhances its utility for users managing diversified portfolios.

## **Project Limitations**

1. **Multi-User Roles or Access Levels:**  
The system assumes all users have the same level of access, with no differentiation for shared accounts or restricted views.
2. **No Currency Conversion Support:**  
Although multi-currency accounts are supported, the system lacks integration with live exchange rates for conversion or comparison.
3. **Lack of Notifications or Alerts:**  
Users are not notified when they approach budget limits or deadlines for savings goals, reducing its usability for proactive financial planning.
4. **Static Budgets:**  
Budgets are fixed and cannot automatically adjust based on actual spending trends or income variations over time.
5. **No Fraud Detection or Security Features:**  
There are no mechanisms to flag unusual transactions or enforce multi-factor authentication for enhanced security.

## **Project Output:**

### **Frontend Functionalities: Will be shown in the screenshot in next section**

1. Personal Finance Analysis:
  - a. Show the saving goal, current savings, and a progress bar above to show the saving progress towards target
  - b. Show an analysis on which category that the user spend the most over the last year

2. Edit Personal Information: Users can edit or add the personal saving goals and deadlines
3. Natural Language Search: Users could search any information he wants in the search bar, as long as it is in the database

### Database Management:

1. User Management: Stores users information about all their current financial status
2. Financial Data Tracking: Track user's transactions, budgets, accounts, and savings, etc.
3. Views:
  - a. UserBalances: Aggregates total balance across all accounts
  - b. CategorySpending: Analyzes spending patterns by category
  - c. SavingsProgress: Tracks progress towards savings goals
  - d. MonthlyTransactions: Summarizes monthly financial activity
4. Stored procedure for generating monthly reports

### Screenshots:

The screenshot displays the 'Personal Finance Tracker' web application. At the top, there is a navigation bar with a logo on the left and a user profile icon labeled 'User1' on the right. Below the navigation bar, three main action buttons are visible: 'Personal Finance Analysis', 'Edit Personal Information', and 'Natural Language Search'. The 'Natural Language Search' section is currently active, showing a search input field with the text 'Does Alice Johnson meet her saving goal?' and a green 'Search' button. Below the search input, the 'Generated SQL Command' is displayed in a light gray box. The SQL query is as follows:

```
Generated SQL Command:
SELECT
  SavingsGoals.GoalName,
  SavingsGoals.TargetAmount,
  SavingsGoals.CurrentAmount,
  (SavingsGoals.TargetAmount <= SavingsGoals.CurrentAmount) AS IsGoalMet
FROM
  Users
JOIN
  SavingsGoals ON Users.UserID = SavingsGoals.UserID
WHERE
  Users.Name = 'Alice Johnson';
```

Below the SQL command, the 'Results' section shows a table with the following data:

GoalName	TargetAmount	CurrentAmount	IsGoalMet
New Car	10000.00	1500.00	0



# Personal Finance Tracker

[Personal Finance Analysis](#)[Edit Personal Information](#)[Natural Language Search](#)

## Natural Language Search

[Search](#)

Generated SQL Command:

```
SELECT InvestmentType FROM Investments
JOIN Users ON Investments.UserID = Users.UserID
WHERE Users.Name = 'Alice Johnson';
```

Results:

InvestmentType
Stock - AAPL



# Personal Finance Tracker

[Personal Finance Analysis](#)[Edit Personal Information](#)[Natural Language Search](#)

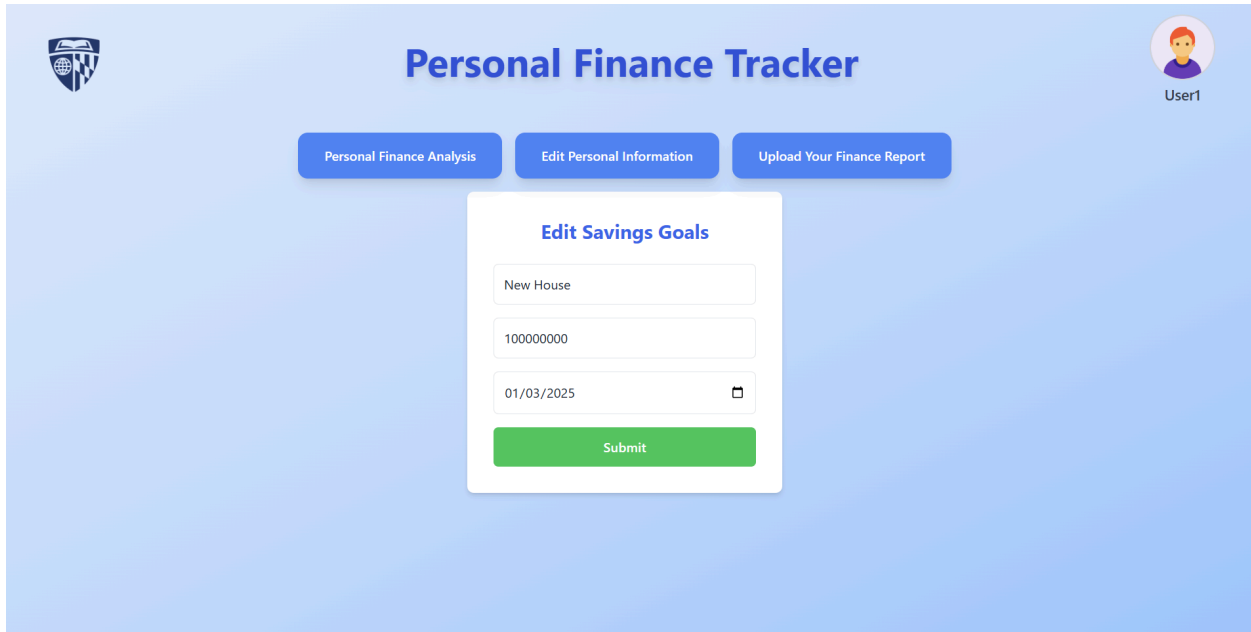
Goal: New Car (\$10000.00)

15%

User Name	Current Amount	Goal Name	Target Amount
Alice Johnson	\$1500.00	New Car	\$10000.00

## Spending Habits

Category	Spent Amount	Month
Groceries	\$50.25	6
Salary	\$300.00	12
Travel	\$200.00	12
Entertainment	\$195.00	12
Groceries	\$50.00	12



The image shows a web application titled "Personal Finance Tracker". In the top left corner is a shield-shaped logo with a globe and a building. In the top right corner is a user profile icon labeled "User1". Below the title, there are three blue buttons: "Personal Finance Analysis", "Edit Personal Information", and "Upload Your Finance Report". The "Edit Personal Information" button is highlighted, and a modal form titled "Edit Savings Goals" is displayed in the center. This form contains three input fields: the first has the text "New House", the second has "100000000", and the third has a date "01/03/2025" with a calendar icon to its right. At the bottom of the form is a green "Submit" button.

## Relational data model

- **Entities and Attributes**

- Users:
  - UserID: Primary key.
  - Name: User's full name.
  - Email: User's email address.
  - PasswordHash: Encrypted password for secure login.
  - CreatedAt: Timestamp when the user account was created.
- Accounts:
  - AccountID: Primary key.
  - UserID: Foreign key linking to the Users entity.
  - AccountName: Descriptive name of the account (e.g., "Savings Account").
  - AccountType: Type of account (e.g., Bank, Credit Card, Wallet, Investment).
  - Balance: Current balance in the account.
  - Currency: Currency type.
  - CreatedAt: Timestamp when the account was added.
- Categories:
  - CategoryID: Primary key.
  - Name: Name of the category.
  - Description: Optional description of the category.
- Transactions:
  - TransactionID: Primary key.

- AccountID: Foreign key linking to the Accounts entity.
- CategoryID: Foreign key linking to the Categories entity.
- Amount: Transaction amount.
- TransactionType: Type of transaction.
- TransactionDate: Date of the transaction.
- Description: Optional description of the transaction.
- CreatedAt: Timestamp when the transaction was added.
- Budgets:
  - BudgetID: Primary key.
  - UserID: Foreign key linking to the Users entity.
  - CategoryID: Foreign key linking to the Categories entity.
  - BudgetAmount: Budgeted amount for a specific category.
  - StartDate: Start date for the budget period.
  - EndDate: End date for the budget period.
  - CreatedAt: Timestamp when the budget was created.
- Savings Goals:
  - GoalID: Primary key.
  - UserID: Foreign key linking to the Users entity.
  - GoalName: Name of the savings goal.
  - TargetAmount: Total amount the user aims to save.
  - CurrentAmount: Amount saved so far.
  - Deadline: Target date for achieving the savings goal.
  - CreatedAt: Timestamp when the savings goal was added.
- Investments:
  - InvestmentID: Primary key.
  - UserID: Foreign key linking to the Users entity.
  - InvestmentType: Type of investment.
  - InitialValue: Value of the investment at the time of purchase.
  - CurrentValue: Current value of the investment.
  - PurchaseDate: Date the investment was purchased.
  - Notes: Optional notes about the investment.
  - CreatedAt: Timestamp when the investment was added.
- **Relationship**
  - Users ↔ Accounts: One-to-many.
  - Accounts ↔ Transactions: One-to-many.
  - Categories ↔ Transactions: One-to-many.
  - Users ↔ Budgets: One-to-many.
  - Users ↔ SavingsGoals: One-to-many.
  - Users ↔ Investments: One-to-many.



