FSD Project

Panel-C

EXPENSE TRACKER

1032221904- AMAN

1032221964-RONAK

1032221864 -HARSH

1032221979- KRİSHNA

Problem Statement & Proposed Solution

Problem Statement:

• People often struggle with tracking their **expenses**, leading to **poor financial** planning and budgeting.
• Lack of a consolidated platform for managing **all categories** of expenses like utilities, entertainment, housing, etc.
• Difficulties in **monitoring** trends over time and **predicting** future expenses.

Proposed Solution:

• A full-stack expense tracker and manager that allows users to input, track, categorize, and visualize their expenses.
• Integration of prediction models for estimating future expenses.
• User authentication and secure data handling through the MERN stack.

1

2

3

4

5

6

7

8

9

Process Flow Diagram & System Architecture

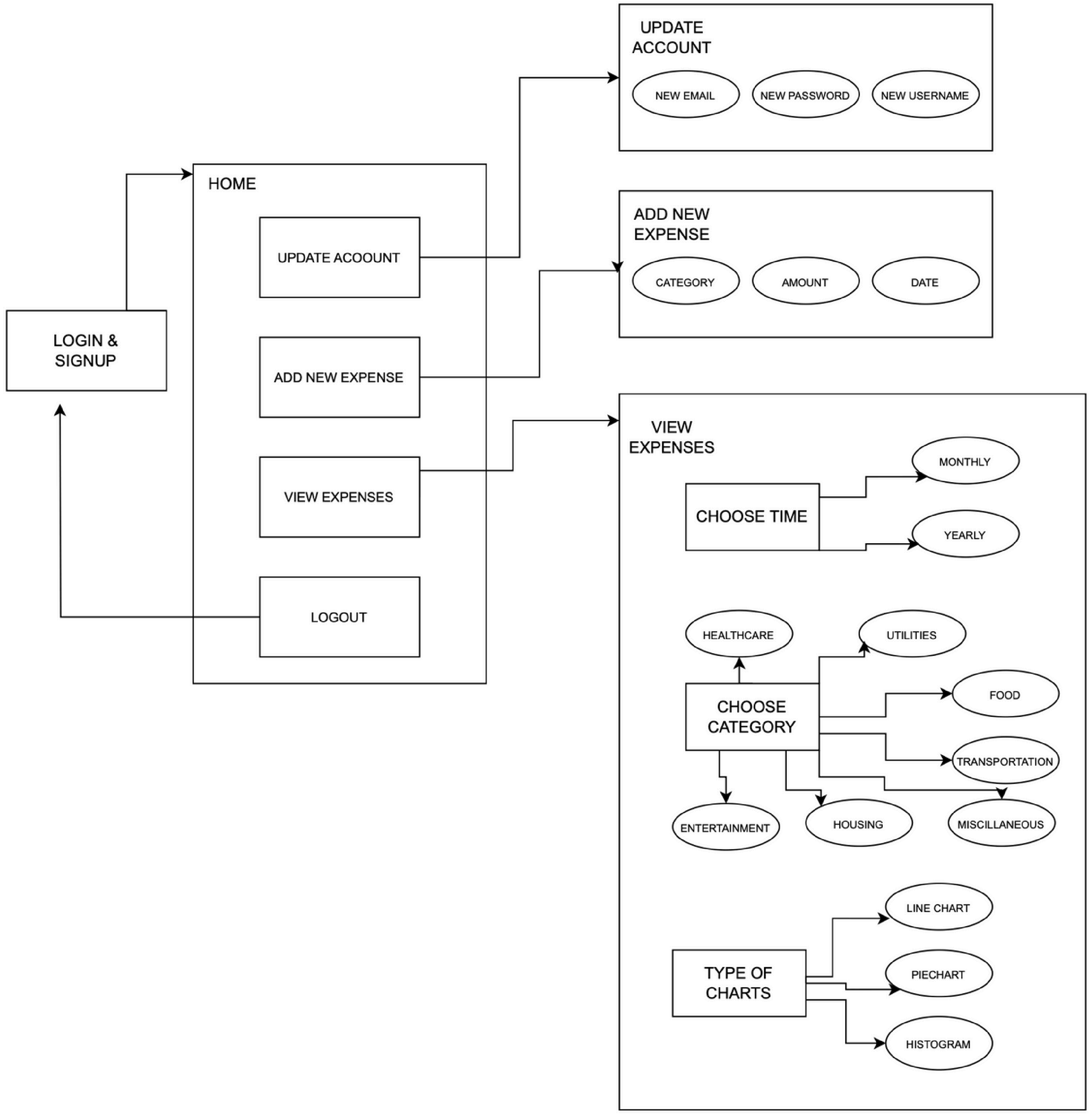
Backend

```
backend
      models
          |- expense.js
          |- User.js
      routes
          |- expenseRoutes.js
          |- userRoutes.js

      .env
      .server.js
```

Frontend

```
src
     |- components
           |- addExpensejs
           |- home.js
           |- login.js
           |- Navbar.js
           |- signup.js
           |- updateAccount.js
           |- viewExpense.js
      |- App.css
      |- App.js
```



Technical Stack with Reasoning

MongoDB:

- Flexible Storage: Handles changing data structures, perfect for dynamic expense data.
- Scalable: Easily grows with user and data needs.
- Real-Time Analytics: Fast aggregation for instant insights.

Express.js:

- Simple API Building: Quick and easy setup for REST APIs.
- Secure Middleware: Manages authentication and validation.
- Organized Routing: Keeps code clean and scalable.

React.js:

- Reusable Components: Modular design for efficient UI.
- Dynamic Updates: Real-time data handling.
- Efficient Hooks: Streamlines interactive features.

Node.js:

- Fast, Non-Blocking: Handles multiple requests at once.
- JavaScript Everywhere: Unifies frontend and backend.
- Rich Libraries: Extensive tools for easy feature integration.

1
2
3
4
5
6
7
8
9

Challenges Faced During Development

- Authentication Issues:

• The current version only tracks expenses and predicts based on categories without deep integration of user-specific or external factors (like inflation).
• Limited user interface with basic chart visualizations; more complex charts and reports could be added.

- Data Validation & Handling:

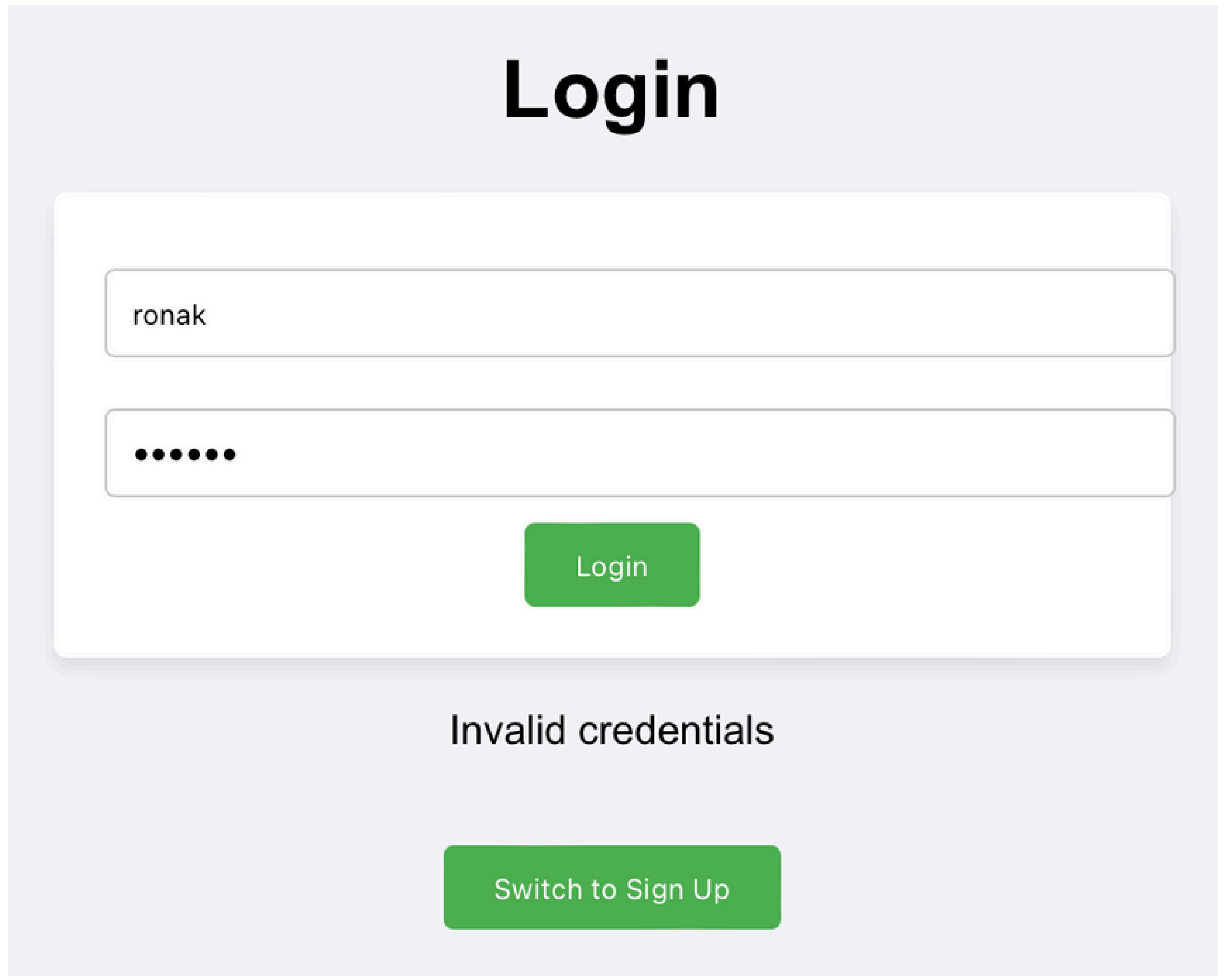
Ensuring the correct validation and sanitization of input data to prevent errors and attacks (e.g., SQL injection) was time-consuming.

- Integrating Predictions:

Creating a model that predicts future expenses based on historical data was complex. Ensuring accurate predictions based on diverse categories of expenses required careful data preprocessing.

- Responsive Design:

Ensuring the UI was responsive and accessible across devices required additional styling and testing.


Login

ronak

••••••

Invalid credentials

Limitations and Scope of the Project with Future Projection & Extension

Limitations:

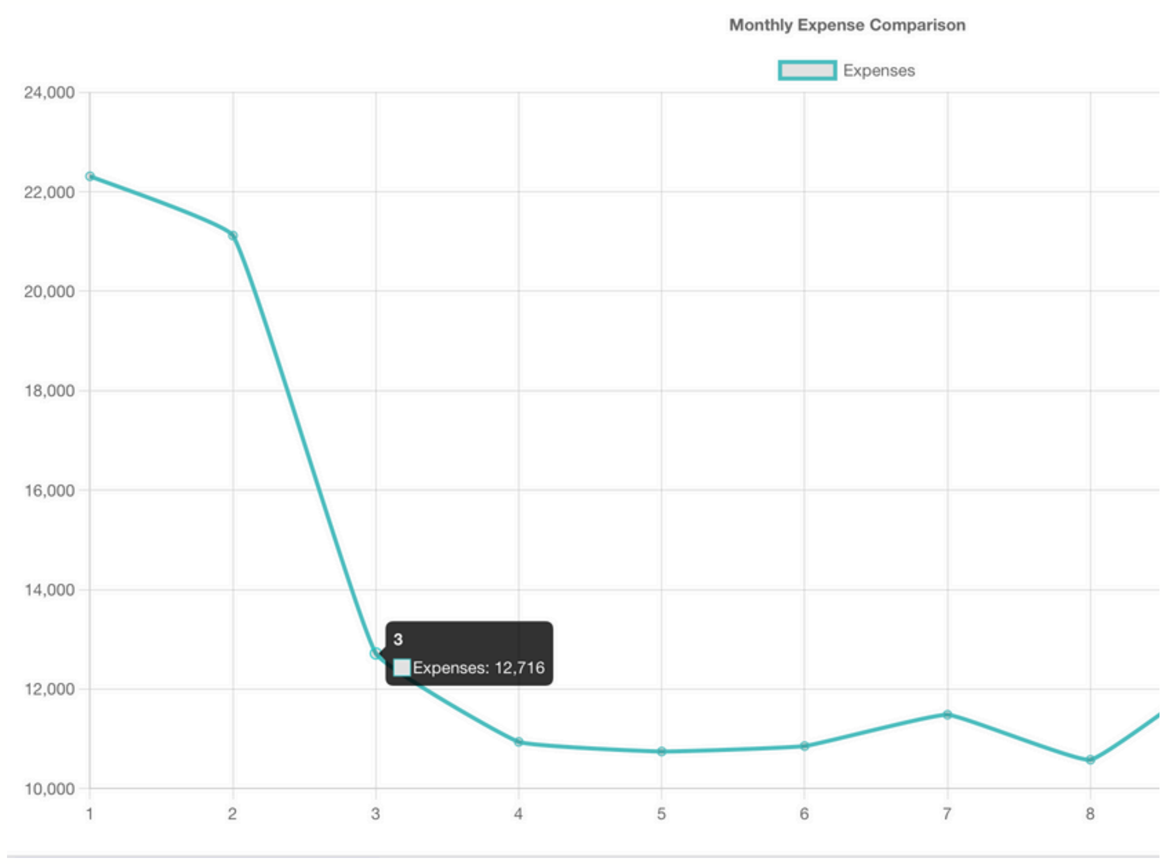
• The current version only tracks expenses and predicts based on categories without deep integration of user-specific or external factors (like inflation).
• Limited user interface with basic chart visualizations; more complex charts and reports could be added.

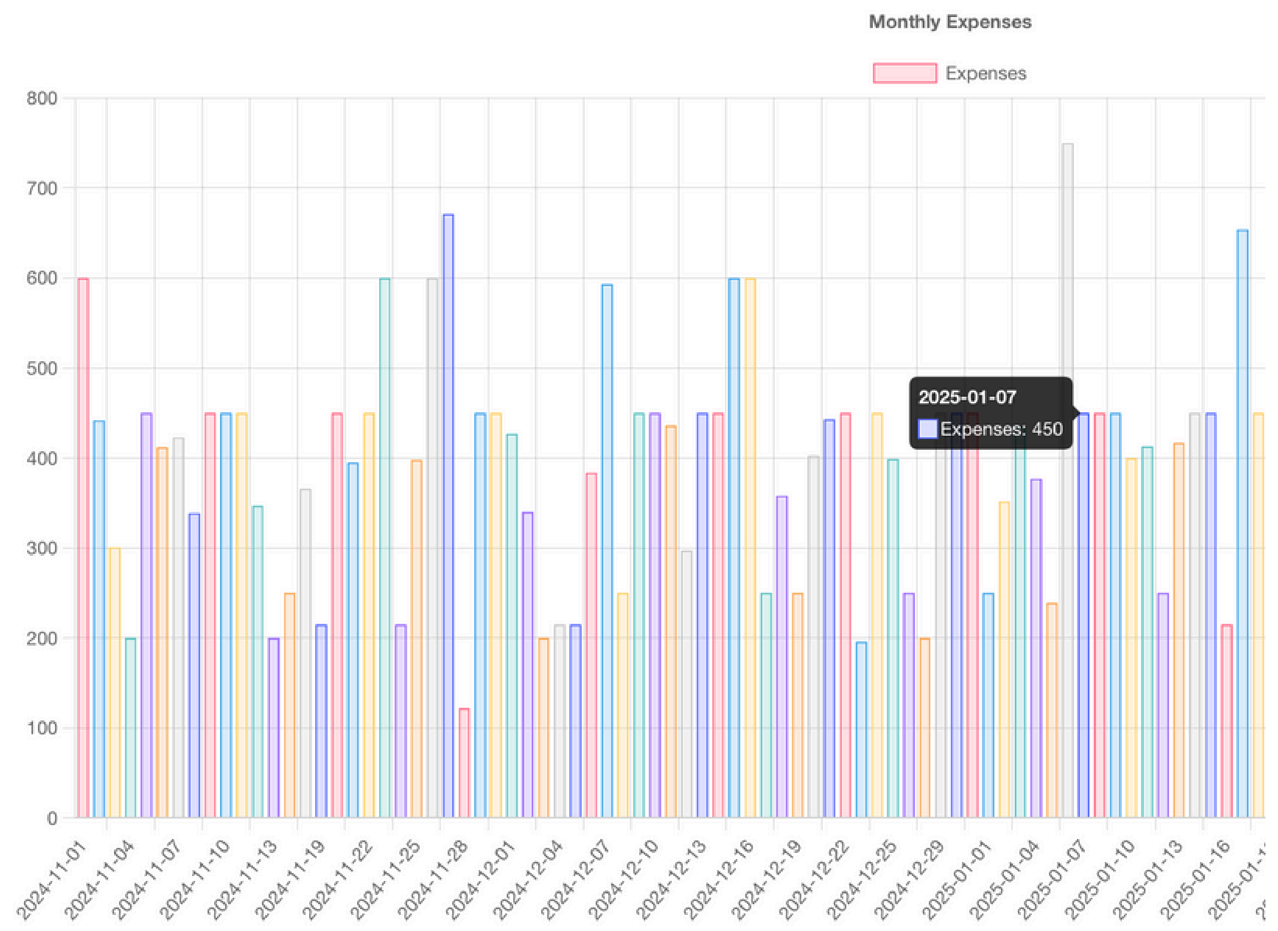
Scope of Future Extensions:

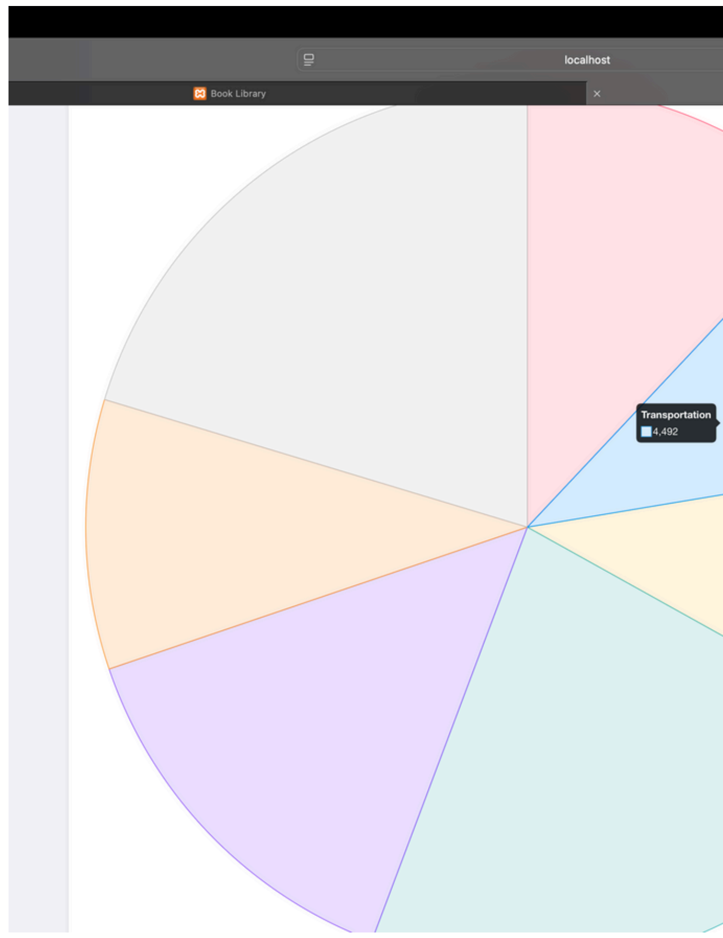
• Integration with bank APIs for automatic expense import.
• Adding a mobile app version using React Native.
• Incorporating advanced machine learning models for more accurate predictions.
• Allowing users to set monthly budgets and receive alerts for overspending.
• As a future enhancement, we plan to integrate an expense prediction feature powered by an AI model. By leveraging a Long Short-Term Memory (LSTM) neural network, the system will analyze historical expense data and identify spending patterns unique to each user. This predictive model will provide personalized insights, offering users proactive suggestions and helping them anticipate upcoming expenses based on past behavior. Such a feature would elevate the application from a basic tracker to a comprehensive financial planning tool, making it invaluable for users aiming to optimize their budgeting and financial habits.

Working Demo

- Line Graph: Shows spending trends over time, allowing users to track how their expenses vary month-to-month or week-to-week. This helps identify patterns, such as increased spending in certain months or seasons.
- Bar Graph: Offers a clear, comparative view of expenses across different categories or time periods, making it easy for users to see where most of their budget goes and compare spending levels across months or categories.
- Pie Chart: Breaks down expenses by category, providing a quick visual of how spending is distributed. This helps users understand what portion of their budget is allocated to different areas like food, entertainment, or utilities.







Project Structure

Backend

```
backend
     models
          |- expense.js
          |- User.js
     routes
          |- expenseRoutes.js
          |- userRoutes.js

     .env
     .server.js
```

Frontend

```
src
    |- components
         |- addExpensejs
         |- home.js
         |- login.js
         |- Navbar.js
         |- signup.js
         |- updateAccount.js
         |- viewExpense.js
    |- App.css
    |- App.js
```

Backend Structure:

- Middleware: authMiddleware.js (handles authentication logic)
- Models: expense.js, user.js (defines the schema for user and expense data)
- Routes: expenseRoutes.js, userRoutes.js (routes for expense-related operations and user authentication)
- Environment: .env (holds sensitive data like database URL, JWT secret)
- Server: .server.js (entry point to the backend, sets up Express server and middleware)

Frontend Structure:

- Components:
- AddExpense.js: Form to add new expenses.
- Home.js: Dashboard showing expense trends.
- Login.js: User login form.
- Navbar.js: Navigation bar.
- Signup.js: User signup form.
- UpdateAccount.js: Form for users to update account details.
- ViewExpense.js: Shows all expenses with sorting and categorization.
- Styling: App.css, App.js.

Q/A

Q1: What is the primary purpose of this expense tracker project?

A: The primary purpose is to help users manage and monitor their expenses efficiently by categorizing spending and providing visual insights. This empowers users to understand their spending patterns and make informed financial decisions.

Q2: Why did you choose the MERN stack for this project?

A: The MERN stack (MongoDB, Express, React, Node.js) was chosen for its flexibility, scalability, and efficiency. MongoDB's NoSQL format is ideal for dynamic expense data, Express and Node.js create a fast, secure backend, and React provides a responsive, component-based frontend for a smooth user experience.

Q3: How does the system ensure data security and user authentication?

A: Data security is managed using JWT (JSON Web Tokens) for user authentication, ensuring that only authenticated users can access and manage their expenses. Additionally, passwords are securely hashed before storing them in the database.

Q4: How are expenses visualized, and why were these specific charts chosen?

A: Expenses are visualized through line graphs, bar graphs, and pie charts. Line graphs show trends over time, bar graphs allow comparisons between months or categories, and pie charts provide a quick overview of expense distribution. These charts were chosen for their clarity and effectiveness in conveying financial insights.

Thank you

Harsh	*PC-42*
Aman	*PC-51*
Ronak	*PC-56*
Krishna	*PC-61*