



**INSTITUTE FOR ADVANCED COMPUTING AND  
SOFTWARE DEVELOPMENT, AKURDI, PUNE**

**Personal Finance Tracker**

PG-DAC February 2025

Submitted By:

Group - 19

**Roll No.**

252129

252020

**Student Name**

Aniruddha Bywar

Atul Singh

Mrs. Monika Sindhikar

**Project Coordinator**

Mr. Prashant Deshpande

**Centre Coordinator**

## **Abstract**

The Personal Finance Tracker is a comprehensive web-based application designed to help individuals manage their financial activities efficiently and intuitively. It provides a centralized platform for tracking income, expenses, budgets, and transactions while offering clear insights into spending patterns and savings goals. Built with a modern technology stack comprising a React-based frontend and a Spring Boot backend, the system ensures seamless user experience with real-time interaction and secure data handling.

Key features include account creation and management, categorization of transactions, budget planning, and detailed summaries for financial analysis. The application is designed with modular architecture, separating concerns into distinct components for accounts, budgets, transactions, and user authentication, ensuring maintainability and scalability for future enhancements.

From a user perspective, the platform emphasizes simplicity without compromising functionality—offering responsive forms, dynamic data tables, and visual summaries to make financial decision-making more informed. From a technical perspective, the project demonstrates integration of RESTful APIs, robust backend validation, secure data storage, and clear separation between client and server layers.

The Personal Finance Tracker aims not only to digitize personal finance management but also to empower users to take control of their financial well-being through an accessible, interactive, and reliable application. It stands as a practical example of applying full-stack development principles to solve real-world problems with efficiency, usability, and scalability in mind.

## ***Acknowledgement***

We bow with utmost reverence to **Goddess Saraswati**, the divine embodiment of wisdom, knowledge, and learning, whose blessings have guided us with clarity of thought, perseverance, and inspiration throughout the course of this project. Without Her grace, the successful completion of this work would not have been possible.

We express our sincere gratitude to the **Institute for Advanced Computing and Software Development (IACSD)** for providing us with the opportunity, resources, and a conducive environment to conceptualize, develop, and refine our *Personal Finance Tracker* project.

We are deeply indebted to our project guide, **Mrs. Monika Sindhikar**, for her invaluable guidance, constructive feedback, and continuous encouragement. Her expertise and insightful suggestions have been instrumental in shaping this project to its present form. I sincerely thank our respected Centre Co-Ordinator, **Mr. Prasant Deshpande**, for allowing us to use the available facilities

Our heartfelt thanks are also extended to all the faculty members and mentors at IACSD, whose teaching, motivation, and technical expertise have significantly contributed to our professional growth and development.

Lastly, we acknowledge the unwavering support and cooperation of our peers, friends, and well-wishers, whose encouragement and assistance helped us overcome challenges and remain committed to the successful completion of this project.

Atul Singh (250241220040)

Aniruddha Bywar(250241220029)

***Table of Contents***

<b>Sr. No.</b>	<b>Section</b>	<b>Page No.</b>
1	Introduction	4
2	SRS	5
3	Diagrams	7
3.1	ER Diagram	15
3.2	Use Case Diagram	16
3.3	Data Flow Diagram	17
3.4	Activity Diagram	18
3.5	Class Diagram	21
3.6	Sequence Diagram	23
4	Database Design	24
5	Snapshots	27
6	Conclusion	32
7	References	33

## ***Introduction***

In the contemporary digital era, where data and automation permeate every aspect of daily life, personal financial management has undergone a profound transformation. The traditional modes of tracking income, expenses, and savings—once managed through manual logs and spreadsheets—are now inadequate to address the complexities of modern financial behavior. Individuals grapple with multifaceted challenges: managing recurring transactions, forecasting budgets, classifying expenditures, and making informed decisions about investment and debt reduction. Simultaneously, financial literacy remains an area where accessibility and personalization are paramount, especially as younger demographics and tech-savvy professionals seek intuitive tools tailored to their lifestyle and habits.

The proliferation of mobile applications and web platforms has catalyzed a shift toward self-service financial management. Yet, many of these solutions suffer from fragmented interfaces, limited feature sets, or opacity in their data handling mechanisms. This underscores the necessity for systems that offer not only functional utility but also architectural robustness, transparency, and adaptability. Bridging this gap calls for an application that harmonizes backend intelligence with frontend usability, grounded in sound software engineering principles and responsive design philosophies.

The *Personal Finance Manager* system arises as a direct response to these imperatives. It is conceived as a comprehensive solution that enables users to record, organize, and interpret their financial transactions with ease and precision. The system architecture is defined by a layered, modular approach, where maintainability, testability, and scalability are primary design goals. Through its components, the application supports granular data control and real-time insights, fostering greater financial awareness and autonomy.

The *Personal Finance Manager* system is developed with a clear purpose: to empower individuals with a personalized and structured tool for managing their financial activities. Its primary objective is to bridge the gap between financial data and actionable decision-making, presenting users with a platform that is not only intuitive to navigate but also academically and technically robust in its design.

The system enables granular tracking of income and expenses, classification of transactions, visualization of monthly spending patterns, and projection of budget forecasts. These features are built to ensure that users can not only record their financial

behavior but also extract meaningful insights over time. The goal is to foster financial literacy, encourage informed choices, and help users set tangible financial goals that reflect their lifestyle and constraints.

From a technical standpoint, the application is architected using a layered, full-stack paradigm. The backend leverages **Spring Boot** for its robustness in enterprise-grade application development, RESTful communication, and modular service handling. **MySQL** serves as the data store, chosen for its relational consistency and query optimization across transactional data. **React** powers the frontend interface, designed with component modularity and responsive layouts to ensure cross-device accessibility.

The rationale behind this technology stack is grounded in several principles:

- **Scalability:** Ensuring the system can support increased user load and future feature expansion.
- **Maintainability:** Facilitating clean separation of concerns through domain-driven design and structured component hierarchy.
- **Security:** Implementing **JWT-based authentication**, encryption of sensitive data, and role-based access control to safeguard user information.
- **Adaptability:** Offering flexibility for integration with external financial APIs, mobile responsiveness, and potential data analytics enhancements.

To reinforce system transparency and defensibility, the documentation includes a formally structured **Software Requirements Specification (SRS)**, complete **Entity Relationship Diagrams (ERD)** to model relational integrity, and **Class Diagrams** for object-oriented design clarity. These artifacts provide a holistic view of system behavior, supporting both academic validation and practical deployment.

The development of the *Personal Finance Manager* is guided by a methodological framework that combines **structured software engineering processes** with academic documentation principles. The design lifecycle adheres to established standards—starting from requirement analysis, progressing through architectural modeling, and culminating in implementation and validation. Each step is supported by formal artifacts: a Software Requirements Specification (SRS) that lays out functional and non-functional requirements; Unified Modeling Language (UML) diagrams such as use case, class, and sequence diagrams that illustrate system behavior and interaction; and database modeling tools to ensure relational coherence.

To enhance academic rigor, the project emphasizes traceability, modularity, and defensibility of design choices. For instance, data access layers are abstracted for testability

and clean separation of concerns; domain models are defined with validation constraints to safeguard integrity; and controller-service-repository interactions are documented with clear behavioral mapping. These practices not only improve real-world applicability but also anticipate scrutiny from reviewers evaluating system cohesion and maintainability.

Moreover, this system serves as an example for the intersection of theory and practice in the domain of personal finance technology. By grounding its development in architectural principles like RESTful service design, relational database normalization, and component-based UI composition, the application demonstrates how **academic constructions can yield practical solutions**. It becomes not just a software tool, but a pedagogical case study in modern full-stack design.

Beyond the scope of individual usage, the broader impact of the *Personal Finance Manager* lies in its potential scalability and adaptability to other financial domains—such as small business bookkeeping, expense forecasting, or integration with digital wallets. With future enhancements, such as machine learning-driven analytics, multilingual support, or open banking API integration, the system could evolve into a comprehensive financial hub.

In summary, the *Personal Finance Manager* project is a synthesis of structured software engineering, user-centric design, and academic integrity. It reflects a conscientious effort to build not just a functional application, but a robust, scalable, and defensible system that embodies both technical sophistication and scholarly excellence.

## ***Software Requirements Specification (SRS)***

### **Purpose**

The purpose of this document is to define the Software Requirements Specification (SRS) for the *Personal Finance Tracker* application. This system aims to provide users with a simple yet effective web-based platform to manage their financial activities, including accounts, budgets, categories, and transactions. The document outlines functional and non-functional requirements, system constraints, and hardware/software specifications to guide the development and deployment process.

### **Scope**

The **Personal Finance Tracker** is a full-stack web application developed with a **React.js** frontend and a **Spring Boot** backend, integrated with a **MySQL** relational database.

The system is designed to help users manage their personal finances efficiently, providing an intuitive interface for:

- **Account Management:** Create, update, and delete multiple user accounts.
- **Budget Management:** Add, modify, and remove budgets for specific time periods.
- **Expense & Income Tracking:** Record and categorize transactions for better financial insight.
- **Transaction Monitoring:** View detailed transaction logs and summaries for better decision-making.

The target audience includes individual users seeking a personal financial management solution that operates entirely locally without relying on third-party banking APIs or authentication services.

By offering complete local control over financial data, the Personal Finance Tracker ensures privacy, flexibility, and reliability for personal use.



## **Functional Requirements**

### **User Authentication**

- **FR-1:** Users shall be able to sign up with an email and password.
- **FR-2:** Users shall be able to log in with valid credentials.
- **FR-3:** Users shall be able to log out of the system.

### **Account Management**

- **FR-4:** Users shall be able to create new accounts.
- **FR-5:** Users shall be able to update account details.
- **FR-6:** Users shall be able to delete accounts.
- **FR-7:** Users shall be able to view all their accounts.

### **Budget Management**

- **FR-8:** Users shall be able to create new budgets.
- **FR-9:** Users shall be able to edit existing budgets.
- **FR-10:** Users shall be able to delete budgets.
- **FR-11:** Users shall be able to view budgets and their remaining allocations.

### **Transaction Management**

- **FR-12:** Users shall be able to record income transactions.
- **FR-13:** Users shall be able to record expense transactions.
- **FR-14:** Users shall be able to edit transactions.

- **FR-15:** Users shall be able to delete transactions.
- **FR-16:** Users shall be able to filter transactions by account, date range, or category.

### Category Management

- **FR-17:** Users shall be able to select categories when adding transactions.
- **FR-18:** Users shall be able to view all available categories.

### Non-Functional Requirements

#### Performance Requirements

- The system should respond to user actions within **2 seconds** under normal load.
- The database should handle up to **10,000 transactions per user** without noticeable slowdown.

#### Security Requirements

- Passwords shall be stored in **hashed format** using industry-standard hashing algorithms.
- The application shall enforce basic input validation to prevent SQL injection and XSS attacks.

#### Usability Requirements

- The interface shall be intuitive and consistent across all pages.
- The application should be mobile-friendly and responsive.

### **Reliability and Availability**

- The system should have at least **99% uptime** in production environments.
- The system should allow for data backup and restore operations.

### **Hardware and Software Requirements**

#### **Hardware Requirements (Minimum)**

- **Client Machine:**
  - Processor: Dual-core 2.0 GHz or higher
  - RAM: 4 GB
  - Storage: 2 GB free space
  - Display: 1366×768 resolution or higher
- **Server Machine:**
  - Processor: Quad-core 2.5 GHz or higher
  - RAM: 8 GB
  - Storage: 10 GB free space
  - Network: Stable internet connection for deployment

### **Software Requirements**

- **Frontend:** React.js
- **Backend:** Java 17+, Spring Boot 3+
- **Database:** MySQL 8+
- **Tools:** Maven, npm, Postman, Spring Tool Suite 4, Visual Studio Code
- **Browser:** Chrome, Firefox, or Edge (latest versions)

### **Constraints**

- The system does not integrate with external bank APIs.
- The system assumes a single-user mode without multi-tenancy in the current version.
- Internet connection required for deployment on remote servers.

### **Assumptions and Dependencies**

- Users have basic knowledge of using a web browser.
- A functioning MySQL database is available before the backend server starts.
- The system depends on modern web technologies (React, Spring Boot, MySQL).

## Diagrams

### Entity-Relationship Diagram:

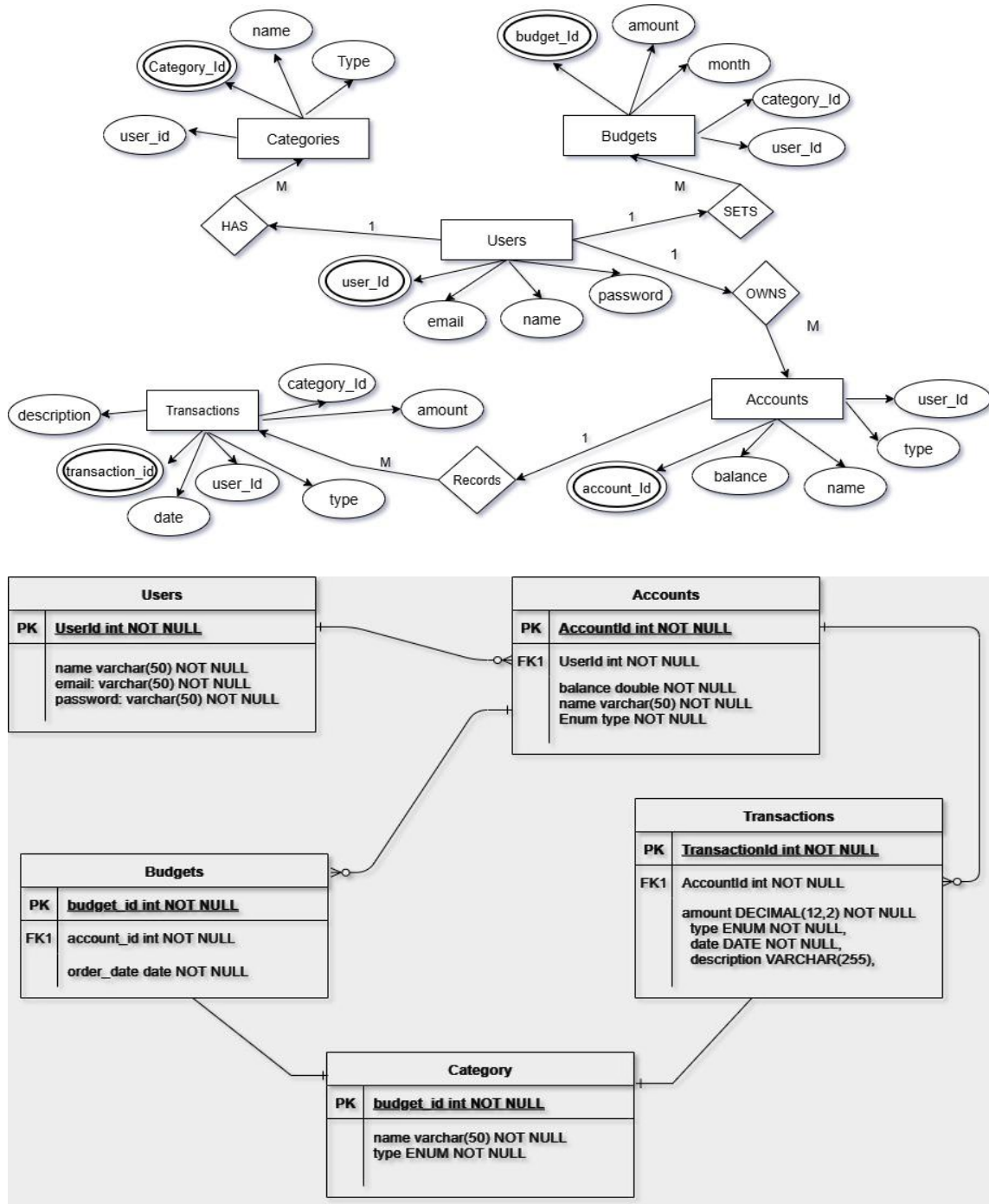


Fig.1 ER Diagram

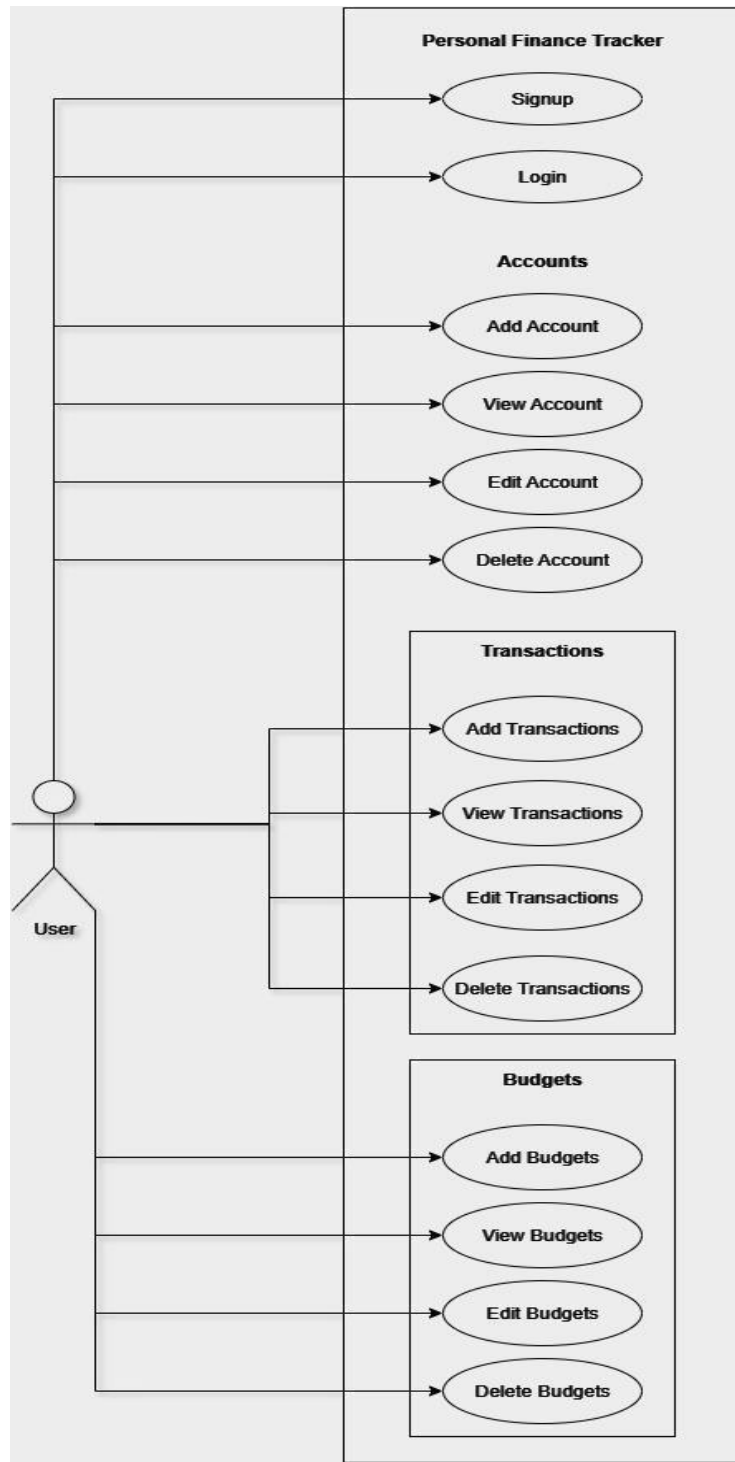
**Use Case Diagram:**

Fig.2 Use Case Diagram

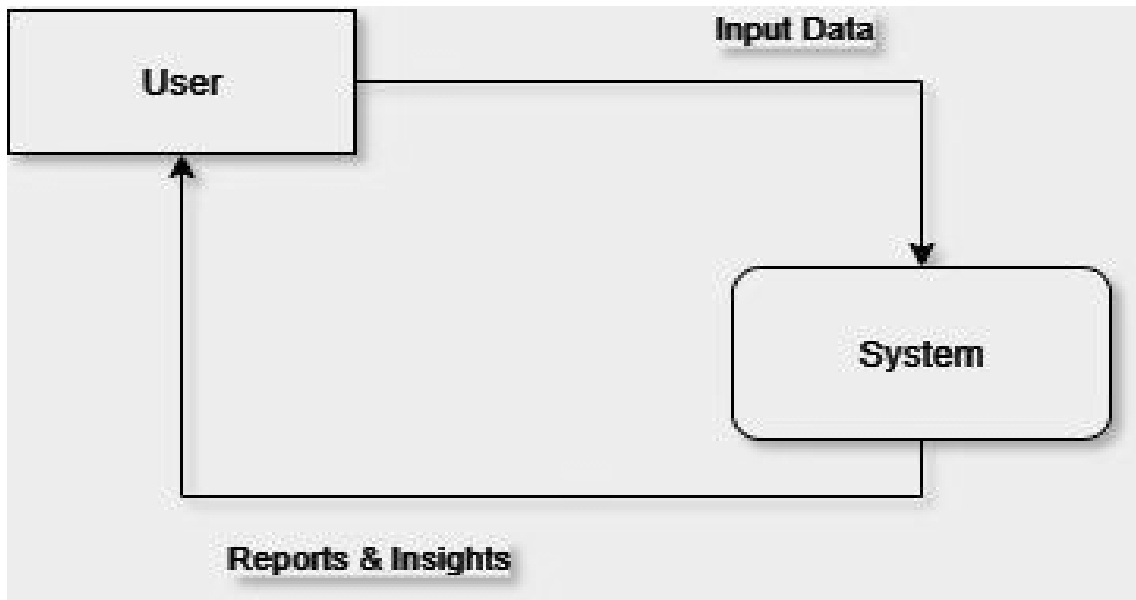
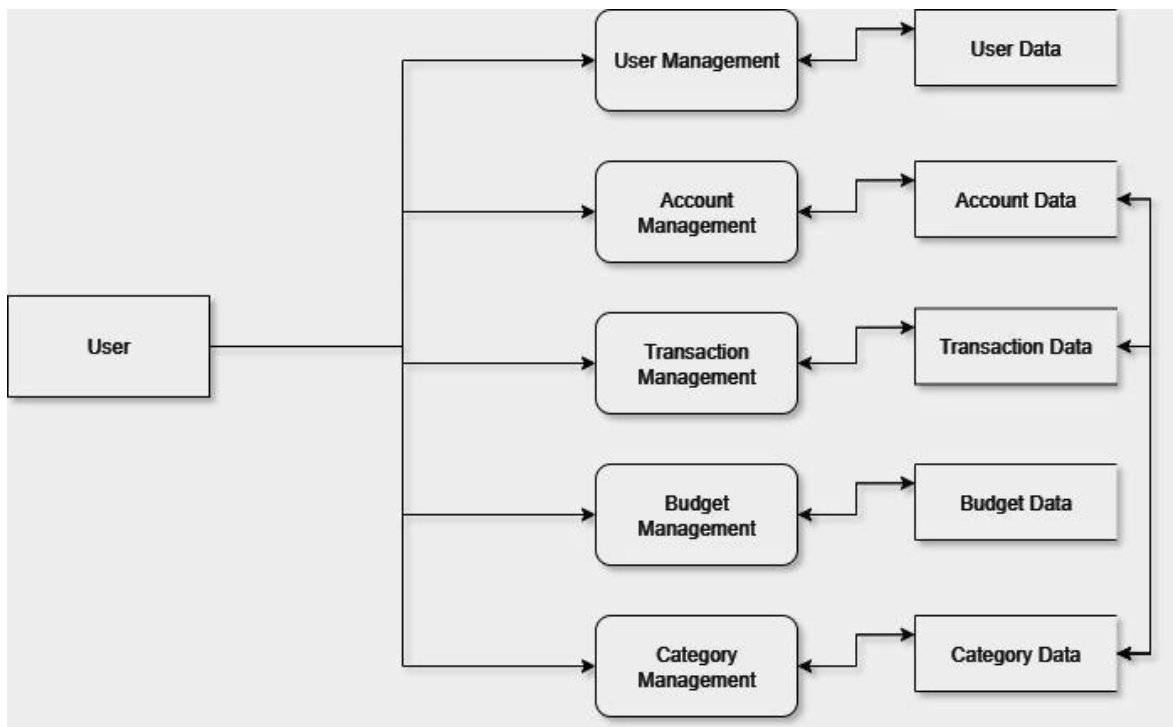
**Data Flow Diagram:****DFD - Level 0**

Fig.3 Data Flow Diagram

**DFD- Level 1**

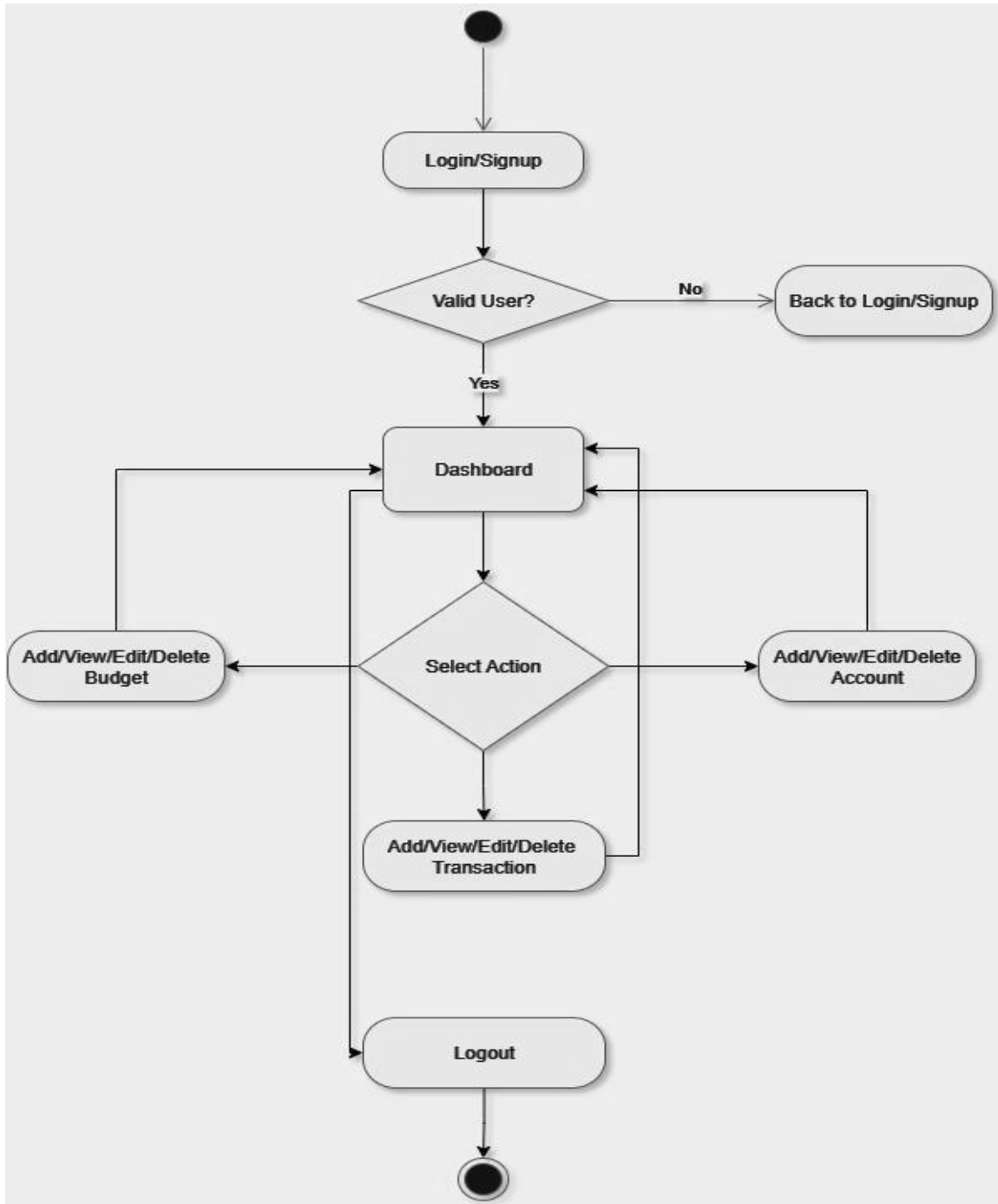
**Activity Diagrams**

Fig. 4 Activity Diagram



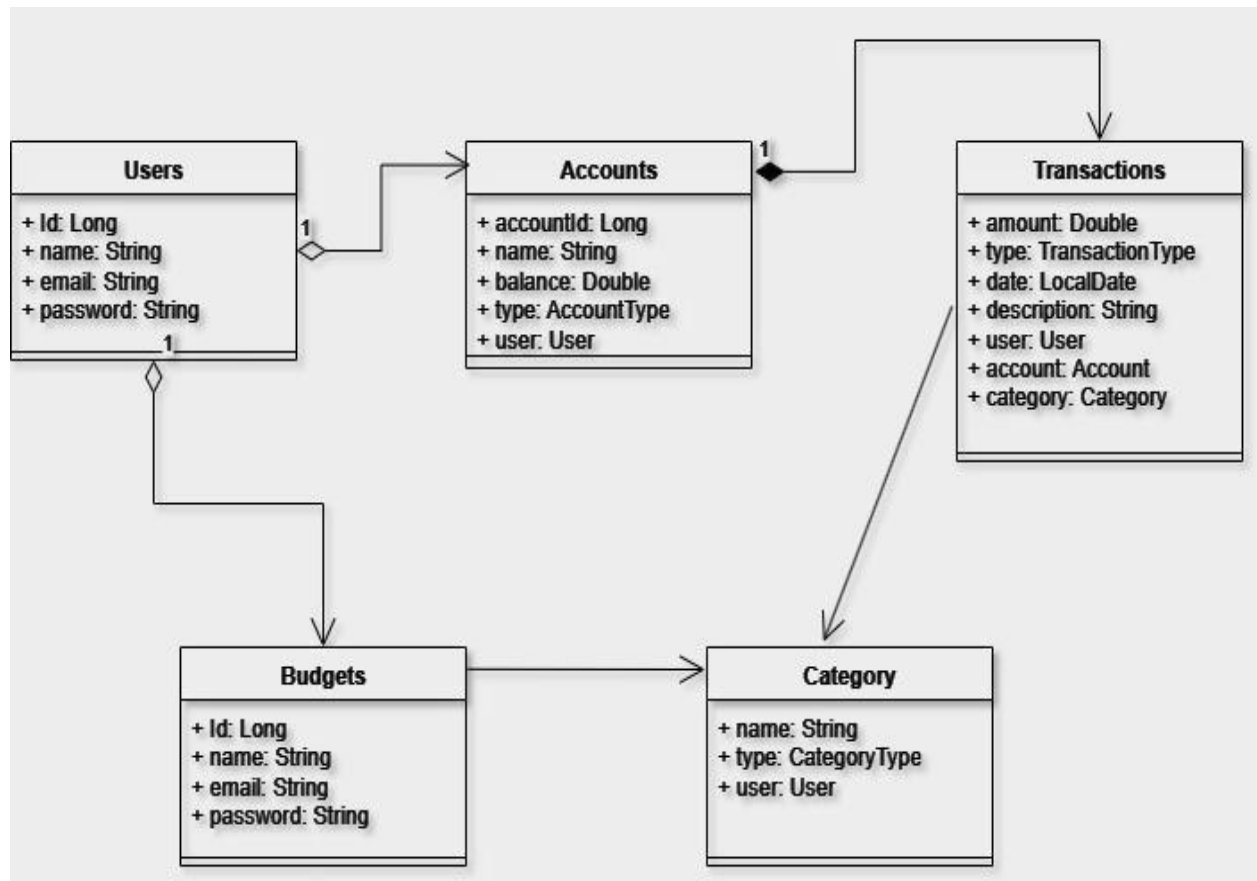
**Class Diagram:**

Fig.5 Class Diagram

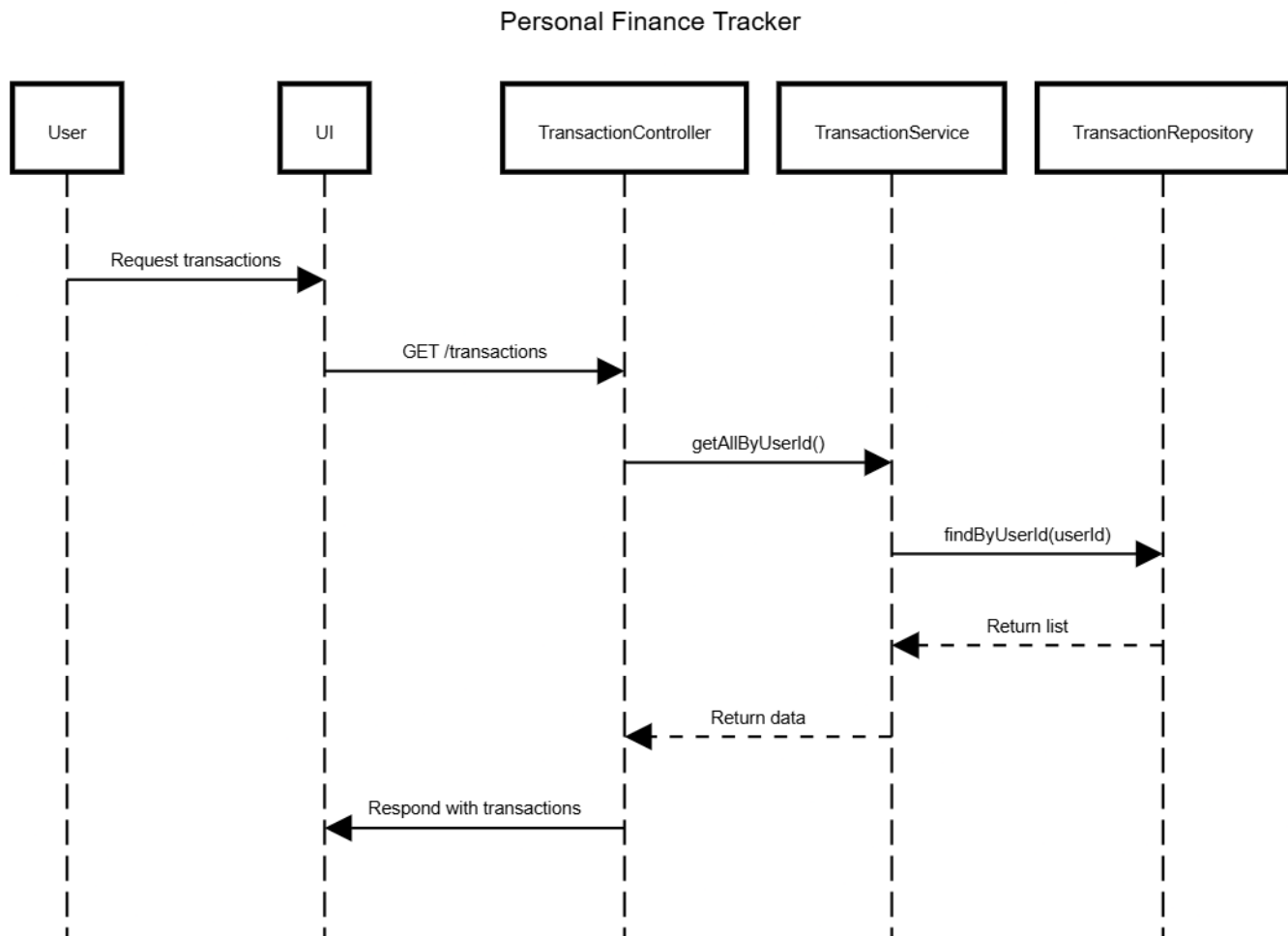
**Sequence Diagram:**

Fig.6 Sequence Diagram

## Database Design:

### 4.1 Design:

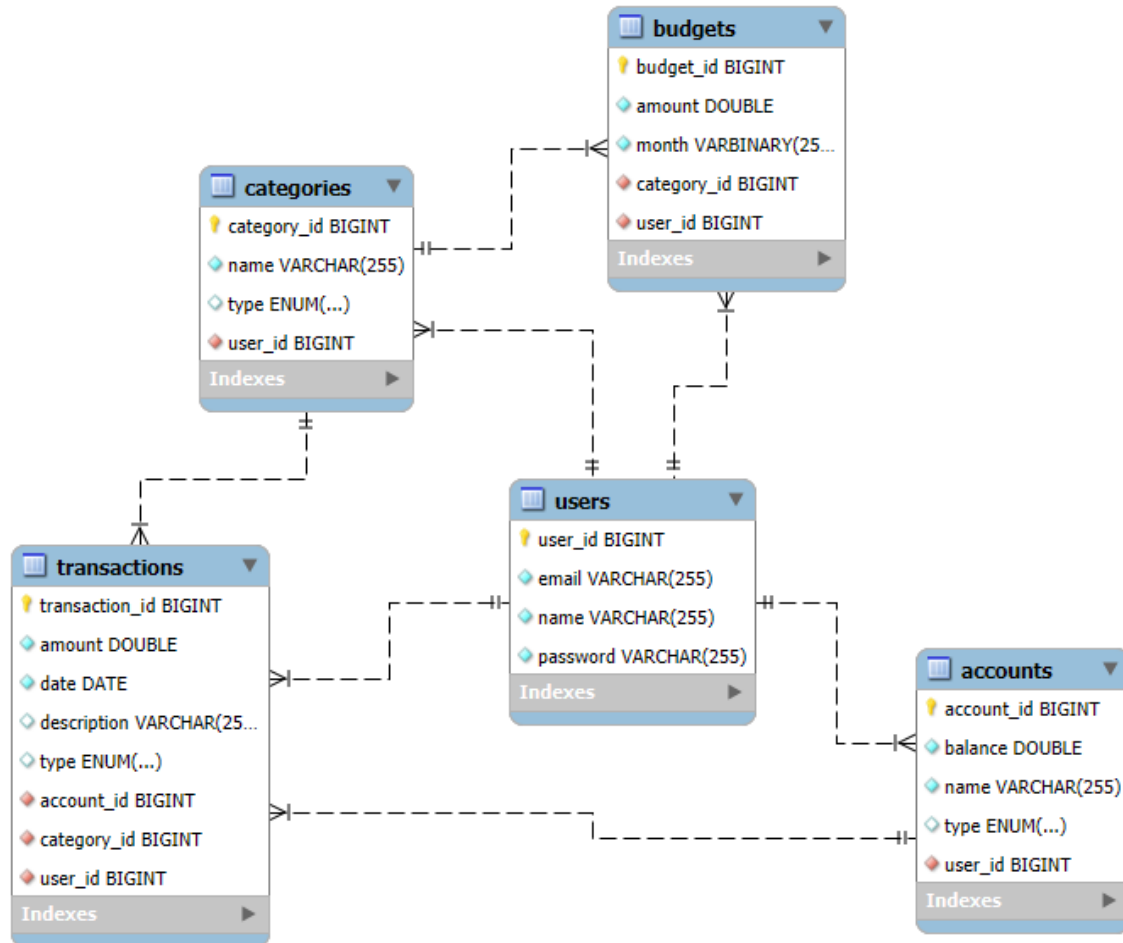


Fig.6 Database Design

## 4.2 Tables:

The following table structures depict the database design.

```
mysql> show tables;
+-----+
| Tables_in_finacncedb |
+-----+
| accounts              |
| budgets               |
| categories             |
| transactions           |
| users                  |
+-----+
5 rows in set (0.00 sec)
```

```
mysql> desc users;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| user_id    | bigint        | NO   | PRI | NULL    | auto_increment |
| email      | varchar(255)  | NO   | UNI | NULL    |                |
| name       | varchar(255)  | NO   |     | NULL    |                |
| password   | varchar(255)  | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)
```

Table 1: users

```
mysql> desc accounts;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| account_id | bigint        | NO   | PRI | NULL    | auto_increment |
| balance    | double        | NO   |     | NULL    |                |
| name       | varchar(255)  | NO   |     | NULL    |                |
| type       | enum('CREDIT','CURRENT','INVESTMENT','SAVINGS','WALLET') | YES |     | NULL    |                |
| user_id    | bigint        | NO   | MUL | NULL    |                |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Table 2: accounts

```
mysql> desc categories;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| category_id | bigint | NO | PRI | NULL | auto_increment |
| name | varchar(255) | NO | | NULL | |
| type | enum('EXPENSE','INCOME') | YES | | NULL | |
| user_id | bigint | NO | MUL | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Table 3: categories

```
mysql> desc transactions;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| transaction_id | bigint | NO | PRI | NULL | auto_increment |
| amount | double | NO | | NULL | |
| date | date | NO | | NULL | |
| description | varchar(255) | YES | | NULL | |
| type | enum('EXPENSE','INCOME') | YES | | NULL | |
| account_id | bigint | NO | MUL | NULL | |
| category_id | bigint | NO | MUL | NULL | |
| user_id | bigint | NO | MUL | NULL | |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

Table 4: transactions

```
mysql> desc budgets;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| budget_id | bigint | NO | PRI | NULL | auto_increment |
| amount | double | NO | | NULL | |
| month | varbinary(255) | NO | | NULL | |
| category_id | bigint | NO | MUL | NULL | |
| user_id | bigint | NO | MUL | NULL | |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Table 5: budgets

## Snapshots:

Login Page:

### Login

Email \*

Password \*

LOGIN

SIGN UP

Sign up Page:

### Sign up

Name \*

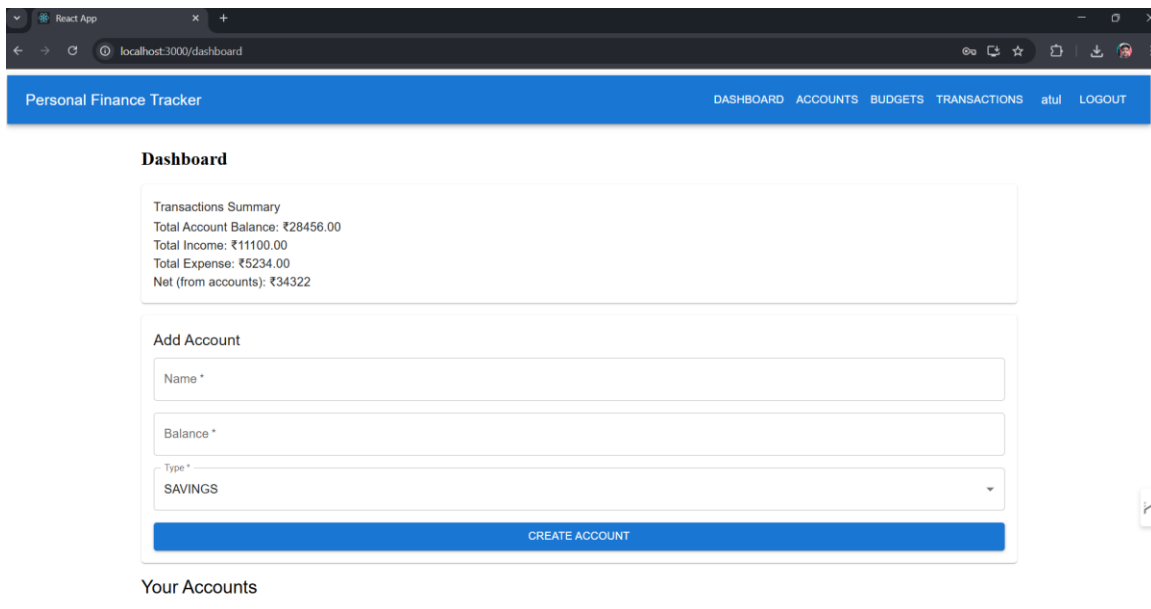
Email \*

Password \*

SIGN UP

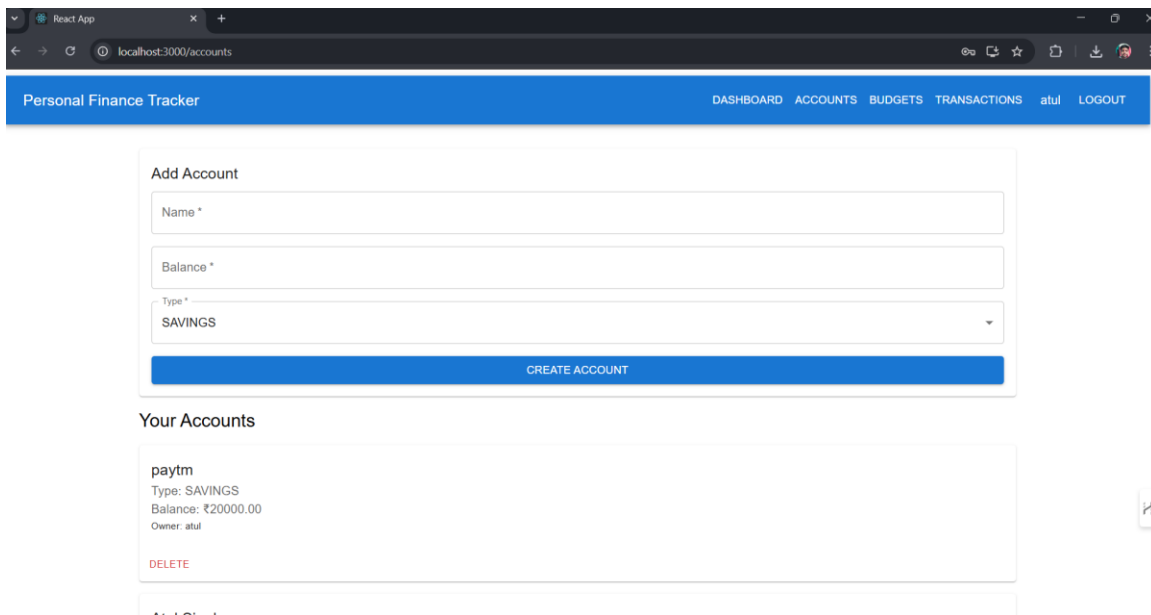
BACK TO LOGIN

Dashboard Page:



The screenshot shows the 'Dashboard' page of the 'Personal Finance Tracker' application. The browser address bar indicates the URL is 'localhost:3000/dashboard'. The application has a blue header bar with the title 'Personal Finance Tracker' and navigation links: 'DASHBOARD', 'ACCOUNTS', 'BUDGETS', 'TRANSACTIONS', 'atul', and 'LOGOUT'. The main content area is titled 'Dashboard' and contains two sections. The first section, 'Transactions Summary', displays the following data: 'Total Account Balance: ₹28456.00', 'Total Income: ₹11100.00', 'Total Expense: ₹5234.00', and 'Net (from accounts): ₹34322'. The second section, 'Add Account', is a form with three input fields: 'Name \*', 'Balance \*', and 'Type \*' (a dropdown menu currently showing 'SAVINGS'). Below the form is a blue button labeled 'CREATE ACCOUNT'. At the bottom of the dashboard, there is a section titled 'Your Accounts'.

### Account Page:



The screenshot shows the 'Account Page' of the 'Personal Finance Tracker' application. The browser address bar indicates the URL is 'localhost:3000/accounts'. The application has the same blue header bar as the dashboard. The main content area contains two sections. The first section, 'Add Account', is identical to the one in the dashboard, with input fields for 'Name \*', 'Balance \*', and 'Type \*' (dropdown showing 'SAVINGS'), and a 'CREATE ACCOUNT' button. The second section, 'Your Accounts', displays a list of accounts. The first account listed is 'paytm' with details: 'Type: SAVINGS', 'Balance: ₹20000.00', and 'Owner: atul'. Below these details is a red 'DELETE' button. Below the list, there is a partially visible 'Add Account' button.

### Transaction Page:

The screenshot shows a web browser window with the URL `localhost:3000/transactions`. The application has a blue header bar with the title "Personal Finance Tracker" and navigation links: DASHBOARD, ACCOUNTS, BUDGETS, TRANSACTIONS, atul, and LOGOUT. The main content area features a form titled "Add Transaction" with the following fields: Amount \*, Type \* (set to EXPENSE), Date \* (placeholder dd-mm-yyyy), Description, Account \*, and Category \*. A blue button labeled "ADD TRANSACTION" is at the bottom of the form.

## Transaction History:

Transactions						
Date	Amount	Type	Account	Category	Description	Actions
2025-08-08	₹1000	INCOME	1	1	xyz	
2025-08-09	₹234	EXPENSE	3	1	aqw	
2025-08-09	₹10000	INCOME	4	1	Income	
2025-08-09	₹4000	EXPENSE	6	1	food	
2025-08-09	₹100	INCOME	6	3		
2025-08-09	₹1000	EXPENSE	6	3		



## Budget Page:

Personal Finance Tracker

DASHBOARDACCOUNTSBUDGETSTRANSACTIONSatulLOGOUT

Add Budget

Amount \*

Month (YYYY-MM) \*

Category \*

ADD BUDGET

My Budgets

Month	Amount	Category	Actions
2025-03	₹15000	salary	

## Conclusion

The **Personal Finance Tracker** successfully provides a simple yet powerful platform for managing personal finances effectively. By combining a **Spring Boot backend** with a **React.js frontend**, the application delivers a smooth user experience while maintaining secure and efficient data handling.

Users can easily track income and expenses, set budgets, categorize transactions, and monitor account balances in real-time. The system's modular architecture and clean UI make it adaptable for future enhancements, such as integrating bank APIs, adding data visualization charts, or implementing AI-powered financial recommendations.

Overall, this project not only meets its primary goal of simplifying financial management but also demonstrates the practical integration of **full-stack development**, database management, and responsive UI design — making it both a useful tool and a strong example of modern software engineering.

## References

1. Spring Boot Documentation. Retrieved from <https://spring.io/projects/spring-boot>
2. React Documentation. Retrieved from <https://react.dev>
3. MDN Web Docs. Retrieved from <https://developer.mozilla.org>
4. MySQL Documentation. Retrieved from <https://dev.mysql.com/doc>
5. Martin, R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Pearson Education, 2017.
6. Kleppmann, M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, 2017.
7. Sommerville, I. Software Engineering, 10th Edition. Pearson, 2015.
8. Xu, A. System Design Interview – An Insider's Guide, Volume 1. ByteByteGo, 2020.
9. Fowler, M. Refactoring: Improving the Design of Existing Code, 2nd Edition. Addison-Wesley, 2018.