Software Requirements Specification

For

PROJECT TITLE: BUDGET TRACKER

Prepared By

S. No	Students Name	Roll Number	Sap Id	Specialization
1.	Yashans Joshi	R2142220299	500105296	AIML NH(B6)
2.	Priyanshi Bhatt	R2142220277	500105809	AIML NH(B5)
3.	Ayush Dwivedi	R2142220780	500106868	AIML NH(B6)
4.	Ria Singhal	R2142220618	5001056141	AIML NH(B6)

Department of Artificial Intelligence of Computer Science
UNIVERSITY OF PETROLEUM & ENERGY STUDIES,
DEHRADUN- 248007. Uttarakhand

Dr. Mohammad Ahsan
Project Mentor

Table of Contents

I. INTRODUCTION

A budget tracker is a website that helps us keep track of our expenses. It can be used to track our spending on a daily, weekly, monthly, or yearly basis. Budget trackers typically allow us to categorize our expenses, so we can see where our money is going. They can also help us to set budgets and track our progress toward our financial goals.

1.1 Purpose of the Project

The project, Budget Tracker, aims to create a website that helps users track their expenses on a daily, weekly, monthly, or yearly basis. Users can categorize expenses to see where their money goes, set budgets, and monitor progress toward financial goals, promoting better financial management and awareness.

1.2 Target Beneficiary

The primary beneficiaries of the Budget Tracker project are individuals and households seeking effective ways to manage their finances. These could include:

- Students tracking daily expenses and setting small budgets.
- Young professionals managing monthly spending and savings goals.
- Families organizing household expenses across categories like groceries, utilities, and entertainment.
- Freelancers or small business owners who want to monitor and categorize expenses for financial tracking and planning.

This project aims to aid users in gaining better financial insight and control, fostering responsible spending habits.

1.3 Project Scope

- Allow users to log expenses with descriptions and amounts.
- Provide functionality to view all expenses in a structured list, graphs, and pie charts.
- Include a simple, user-friendly interface that facilitates easy input and tracking of expenses.
- Use SQLite for local storage of user and expense data.

1.4 References

1) Michael J. Hernandez, “Database Design for Mere Mortals” Published by Addison-Wesley, 2013.
2) "Java Documentation" - Oracle: The official Java documentation is a valuable resource for Java development and best practices.
3) "JavaSwing Documentation" - Oracle: we can refer to the official JavaSwing documentation for insights into JavaSwing development and user interface design.

II. PROJECT DESCRIPTION

2.1 Tech Stack

- **NetBeans:** The Integrated Development Environment (IDE) used for developing the budget tracker website.
- **Java[2]:** The programming language used to implement the core logic of the application.
- **MySQL:** The database used for storing income, expenses, and user information.
- **Java Swing[3]**: Used for designing the graphical user interface, including forms and buttons.
- **JDBC (Java Database Connectivity)[1]**: The library used for connecting Java applications to the MySQL database.
- **JTable:** Used within Java Swing to display lists of transactions and reports in a tabular format.
- **Custom Icons and Fonts**: Used for enhancing UI aesthetics, such as button icons and fonts.

2.2 SWOT Analysis

A SWOT (**Strengths, Weaknesses, Opportunities, Threats**) analysis serves as a criticalcomponent of project planning and strategy development. In the context of the Budget Tracker App:

- **Strengths**: The app's strengths lie in its user-friendly interface, real-time expense tracking, and powerful data analysis capabilities. These features position it as a valuable tool for usersseeking to manage their finances efficiently.

- **Weaknesses**: Weaknesses may include potential security vulnerabilities and the need for ongoing maintenance and updates to keep the app responsive and bug-free.

- **Opportunities**: Opportunities for growth and improvement can be identified through user feedback and evolving financial management trends. The app can seize opportunities to expand its user base and add new features based on emerging financial technologies.

- **Threats**: Threats may arise from competitors offering similar apps, changing user preferences, or evolving security threats. The SWOT analysis will help in developing strategies to mitigate these threats.

.

2.3 Project Features

- **Expense Tracking**: Users can input expenses with details such as description, amount, category (e.g., groceries, entertainment, transportation), payer, and participants. The app automates the distribution of costs based on chosen settings.
- **Automated Calculations**: The app calculates the amount spent by each user on different categories over a selected period, allowing users to track and manage their unnecessary expenses effectively.
- **Reporting and Analytics:** Provides visual reports and insights, such as spending trends and savings progress, often through graphs and charts.

2.4 User Classes and Characteristics

Students:

- Track your everyday costs and create tiny budgets.
- Interface requirements are simple, allowing for rapid and straightforward input.

Young professionals:

- Manage your monthly spending and savings goals.
- Need analytics such as spending patterns and progress reports.

Families:

- Organize home spending into categories (such as groceries, utilities, and entertainment).
- Require collaboration features or the ability to track many users.

Freelancers and small business owners:

- Monitor and categorize expenses for financial tracking.

Characteristics:

- Can view their spending history.
- Can add new expenses.
- Can edit existing expenses.
- Can delete expenses.
- Can view reports and charts.
- Can customize the application's settings.

2.5 Design and Implementation Constraints

Hardware Boundary Conditions:

- **Processor:** Minimum 1.6 GHz; recommended for optimal performance.
- **RAM:** Minimum 4GB, 8GB recommended.
- **Storage:** 500MB free space.
- **Display:** Minimum resolution of 1024x768.

Software Constraints:

- **Operating System:** Compatible with Windows 10 or higher, macOS 10.12, and Linux.
- **Development Environment:** NetBeans IDE 12.0 with JDK 11 or higher.
- **Database:** MySQL 8.0 for storing user data, income, and expenses.

Technologies and Tools:

- **Programming Language:** Java (backend and UI using Java Swing).
- **Libraries:** JDBC for database connectivity, Java Swing components for UI design.
- JFreeChart or a similar library for advanced data visualization.

Interfaces:

- Integration with SQLite and MySQL for data storage.
- Graphs and charts generated using Java Swing components.
- Intuitive and user-friendly design with clear navigation.

Security:

- Encrypted user data storage (both at rest and in transit).
- Role-based access control and session management to ensure secure operations.
- Thoroughly test the application to ensure it meets functional and non-functional requirements.

2.6 Design Diagrams

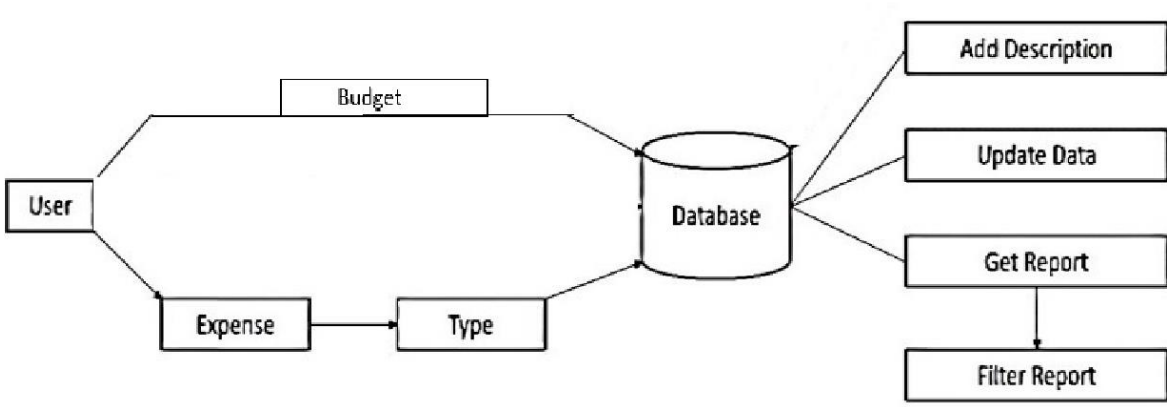

Fig. 1 Data Flow Diagram

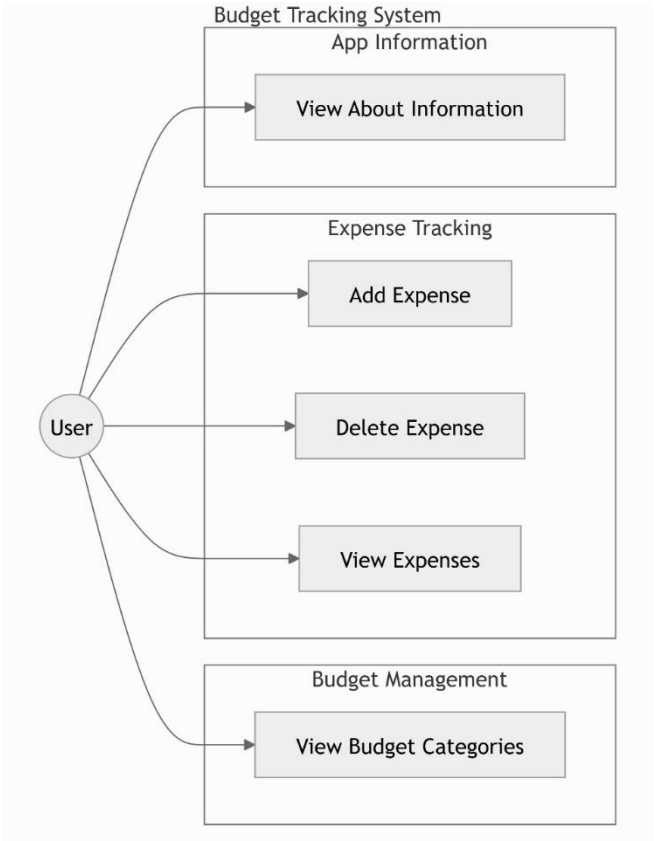

Fig 2: Use Case

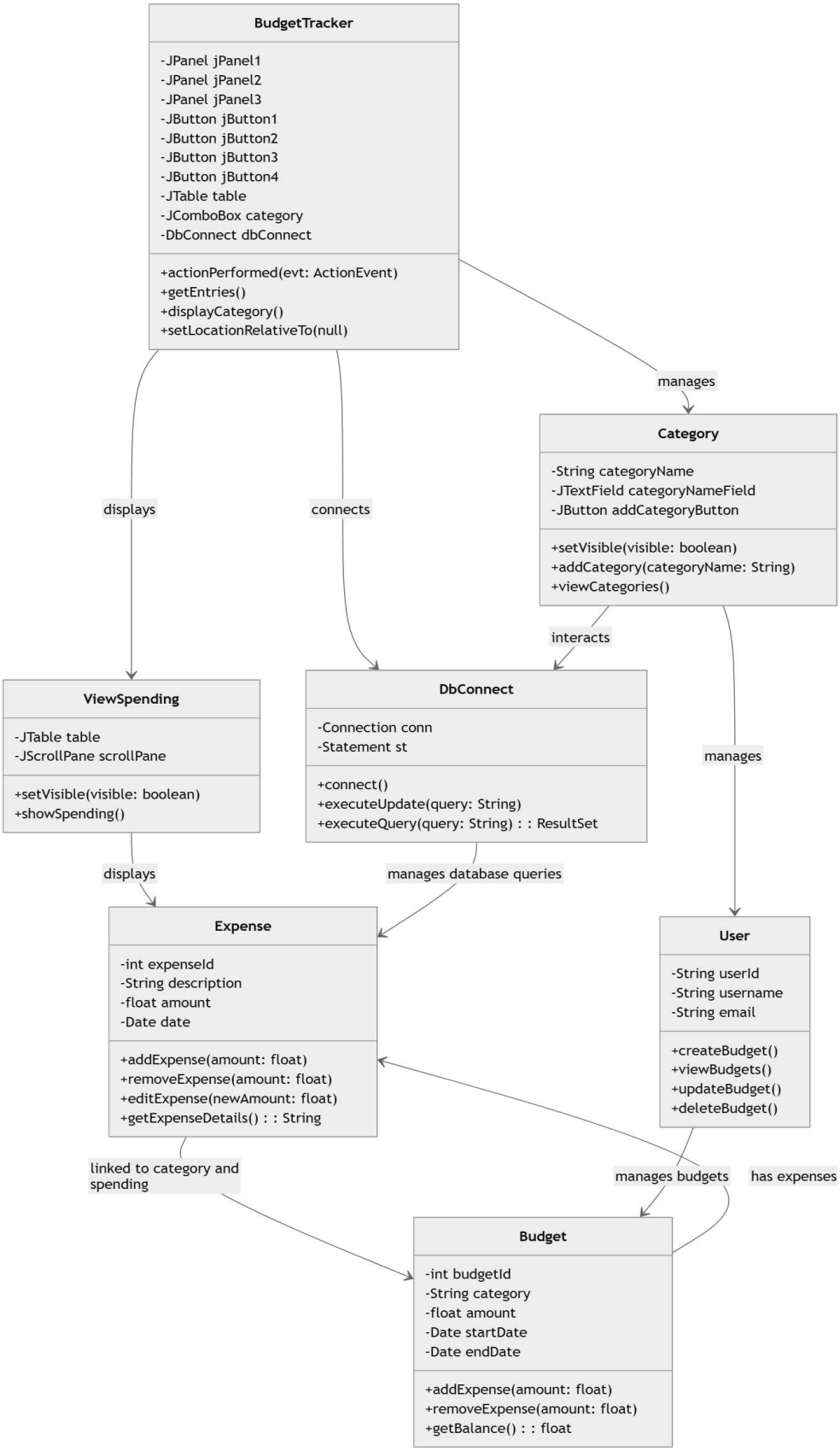

Fig. 3 Class Diagram

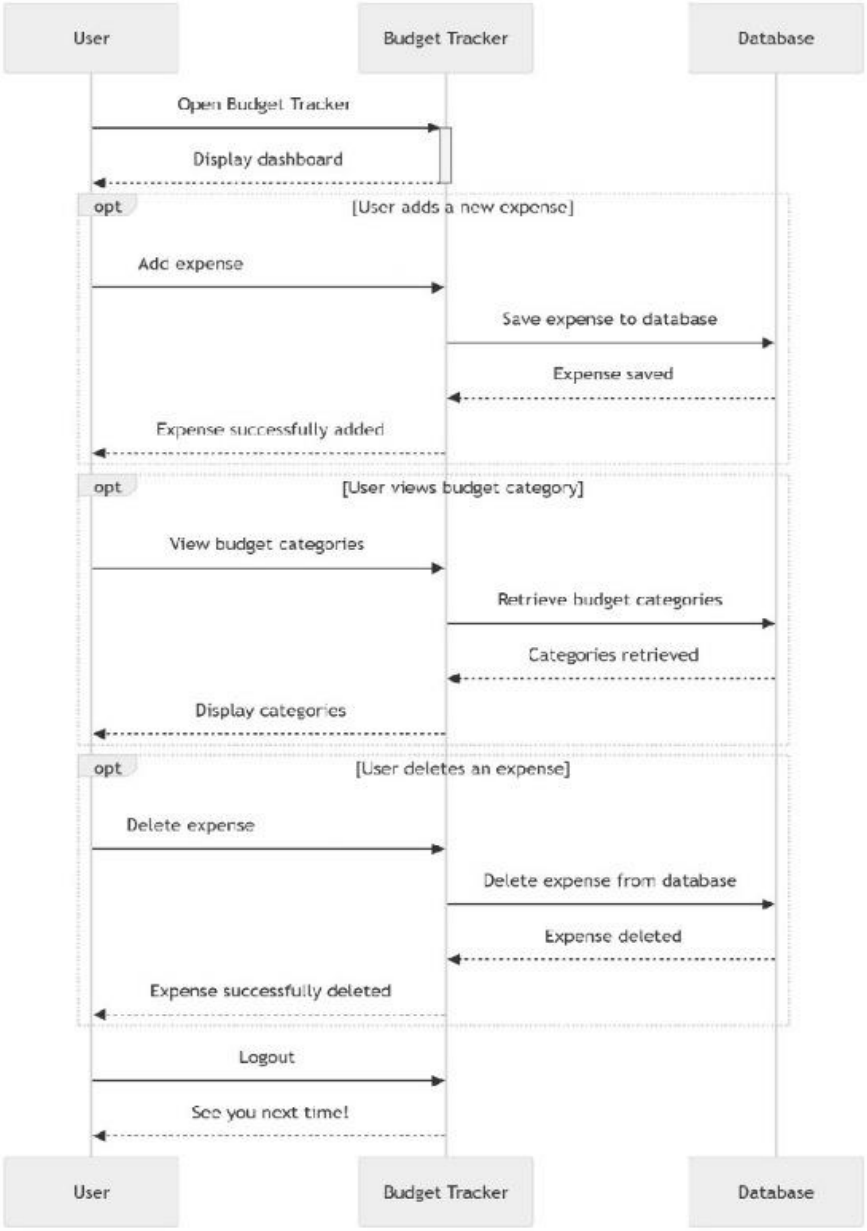

Fig. 4 Sequence Diagram

2.7 Assumptions and Dependencies

Assumptions:

- **Database Connectivity:**
  - The database (MySQL or SQLite) is correctly configured and accessible.
  - The database schema is compatible with the application's data model.

- **User Interface:**
  - The user interface components (e.g., buttons, text fields, labels) will function as expected.
  - The user interface will be compatible with different screen resolutions and operating systems.

- **Operating System:**
  - The application will run on a compatible operating system (Windows, macOS, Linux).
  - The necessary system libraries and frameworks (e.g., Java Runtime Environment, Java Swing) are installed.

- **User Input:**
  - Users will provide valid input (e.g., correct date formats, numeric values for amounts).
  - Users will not intentionally or unintentionally input malicious data.

Dependencies:

- **Database:** The application relies on the database for storing and retrieving user data.
- **Operating System:** The application requires a compatible operating system to run.
- **Java Runtime Environment:** The application requires the Java Runtime Environment to execute.
- **Java Swing Library:** The application uses the Java Swing library for creating the user interface.
- **Third-party Libraries:** If any third-party libraries (e.g., JFreeChart for charting) are used, the application depends on their availability and compatibility.

III. System Requirements

3.1 User Interface

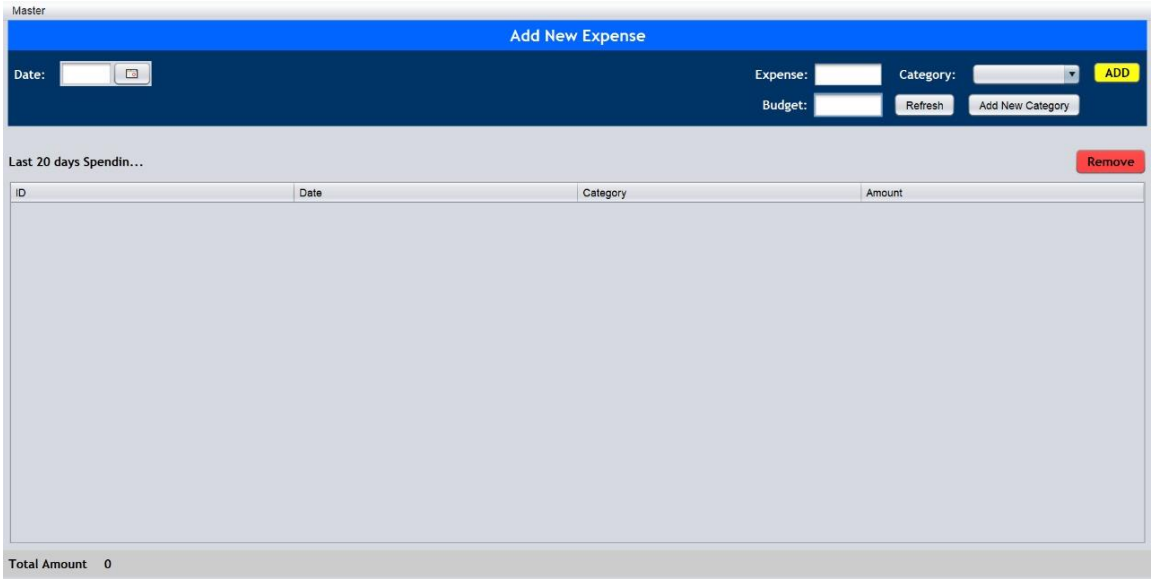

Fig. 1

Main Window:

- **Header:** Displays the title "Add New Expense."
- **Input Fields:** Allows users to input the date, amount, and category of an expense.
- **Buttons:** Includes buttons for adding a new expense, refreshing the table, and adding a new category.
- **Expense Table:** Displays a list of recent expenses, including ID, date, category, and amount.
- **Total Amount:** Shows the total amount spent based on the entries in the table.
- **Remove Button:** Removes the selected expense from the table.

Category Window:

- **Title:** "Add New Category"
- **Input Field:** A text field to enter the name of a new category.
- **"ADD" Button:** Adds the entered category to the list.
- **Table:** Displays a list of existing categories with their serial numbers.
- **"Delete" Button:** Deletes the selected category from the list.

Fig. 2

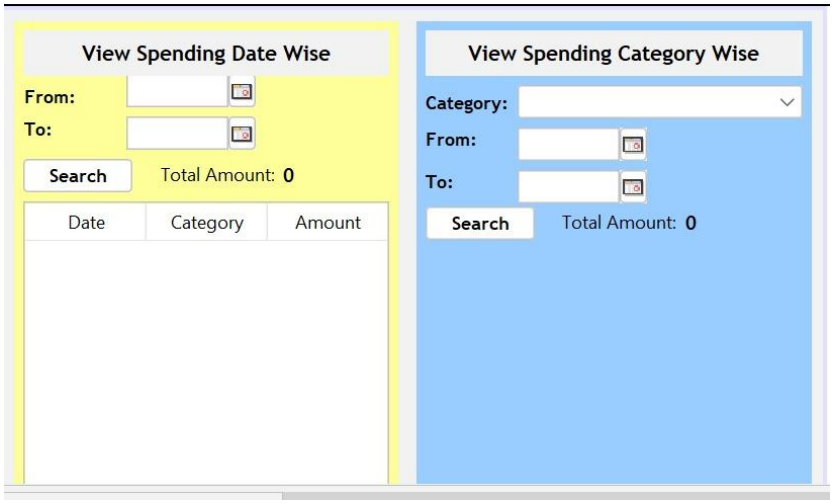

Fig. 3

View Spending Date Wise

- **Date Range Selection:** Users can select a date range (from and to) to filter expenses.
- **Search Button:** Searches for expenses within the specified date range.
- **Total Amount:** Displays the total amount spent within the selected date range.
- **Table:** Displays a list of expenses within the selected date range, showing the date, category, and amount of each expense.

View Spending Category Wise

- **Category Selection:** A dropdown menu to select a specific category.
- **Date Range Selection:** Users can select a date range to filter expenses within the chosen category.
- **Search Button:** Searches for expenses within the selected category and date range.
- **Total Amount:** Displays the total amount spent within the selected category and date range.

3.2 Hardware Requirements:

- **Processor**: Minimum 1.6 GHz or faster processor.
- **RAM**: 4GB of RAM (8GB recommended for better performance).
- **Storage**: At least 200MB of free storage space for project files, dependencies, and application data.
- **Display**: Monitor with a resolution of 1024x768 or higher.

3.3 Software Requirements:

- **Operating System**: Windows 7 or above, macOS, or Linux.
- **Development Environment**: NetBeans IDE with JDK 8 or higher installed.
- **Database**: MySQL (storing user income, expenses, Wishlist, etc.).
- **Languages**: Java (for backend logic using Java Swing for UI).
- **Libraries/Dependencies**: JDBC (for database connectivity), Java Swing components (for UI design).

IV. Non-functional Requirements

4.1 Performance Requirements

- **Responsiveness:** The Budget Tracker should provide instant feedback to users when they enter or update expenses, ensuring no significant delays in loading or updating data.
- **Scalability:** The application should handle a growing number of user entries over time without significant impact on performance. This includes accommodating potentially large datasets of transaction history and user data.
- **Resource Usage:** The application is optimized for devices with a minimum of 4GB RAM but performs best with 8GB. It should operate smoothly without excessive CPU or memory consumption.
- **Data Processing:** All expense-related calculations (e.g., budget summaries, and analytics) should be processed in under two seconds to provide a seamless experience.

4.2 Security Requirements

- **User Authentication and Authorization:** Ensure secure login credentials by encrypting user passwords in the database, requiring strong password policies, and enforcing role-based access where only authenticated users can view, modify, or delete personal financial data.
- **Data Encryption:** Sensitive user data, such as financial records and login credentials, should be encrypted both in transit (e.g., using HTTPS) and at rest in the database.
- **Data Integrity and Recovery:** Implement data integrity checks and database backups to prevent data loss or corruption, providing mechanisms for data recovery in case of system failure.
- **Session Management:** Enforce automatic session expiration after periods of inactivity to prevent unauthorized access, especially on shared devices.

4.3 Software Quality Attributes

- **Usability:** The application should have an intuitive, user-friendly interface, enabling users to quickly log expenses, set budgets, and view reports with minimal learning curve.
- **Reliability:** Ensure that the application is free from critical bugs and runs consistently under normal conditions. It should be capable of continuous operation without crashes.
- **Maintainability:** The codebase should be modular, well-documented, and structured for easy updates and maintenance, allowing for new features or improvements to be implemented with minimal disruption.
- **Portability:** The application should be compatible with major operating systems (Windows, macOS, Linux) and adaptable to various screen resolutions, particularly for users on different device types.
- **Efficiency:** Minimize the resource footprint of the application by using efficient algorithms for database access and UI rendering, ensuring quick response times and low memory usage.

V. Miscellaneous

- Appendix A: Glossary

Term	Definition
SRS	Software Requirements Specification
UI	User Interface
IDE	Integrated Development Environment
JDK	Java Development Kit
SQL	Structured Query Language
JDBC	Java Database Connectivity
HTTPS	Hypertext Transfer Protocol Secure
SWOT	**S**trengths, Weaknesses, Opportunities, Threats

- Appendix B: Analysis Model

Use Case

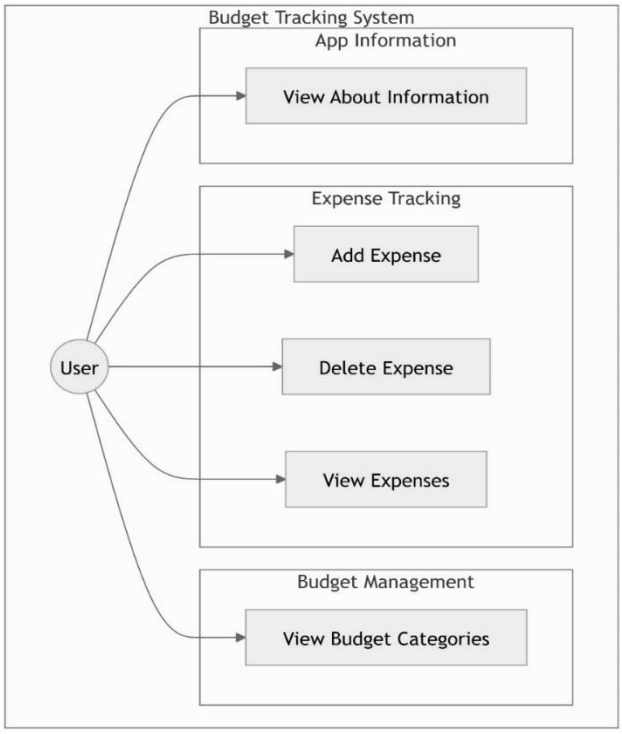

Fig. 1

Class Diagram

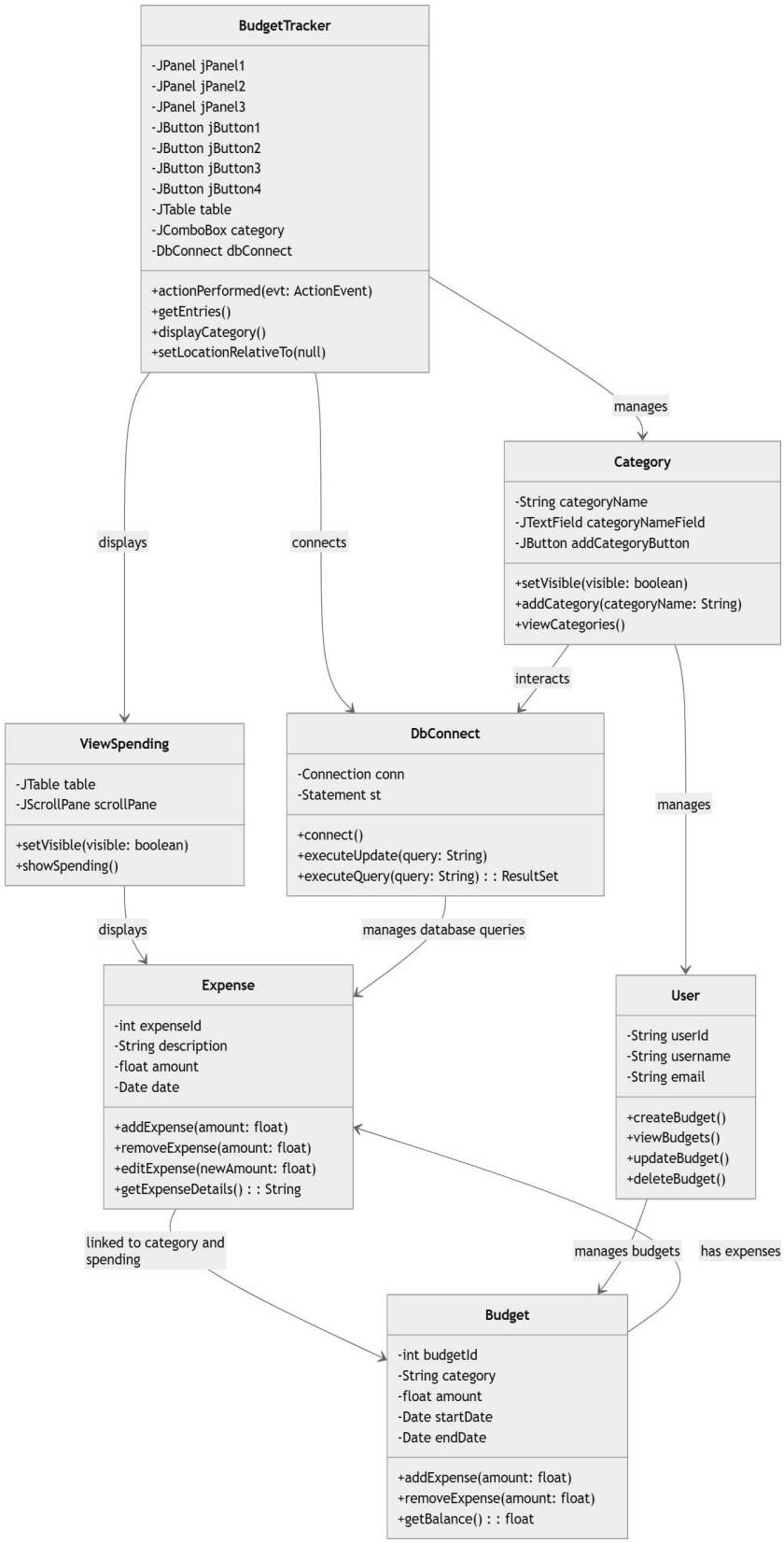

Fig. 2

Sequence Diagram

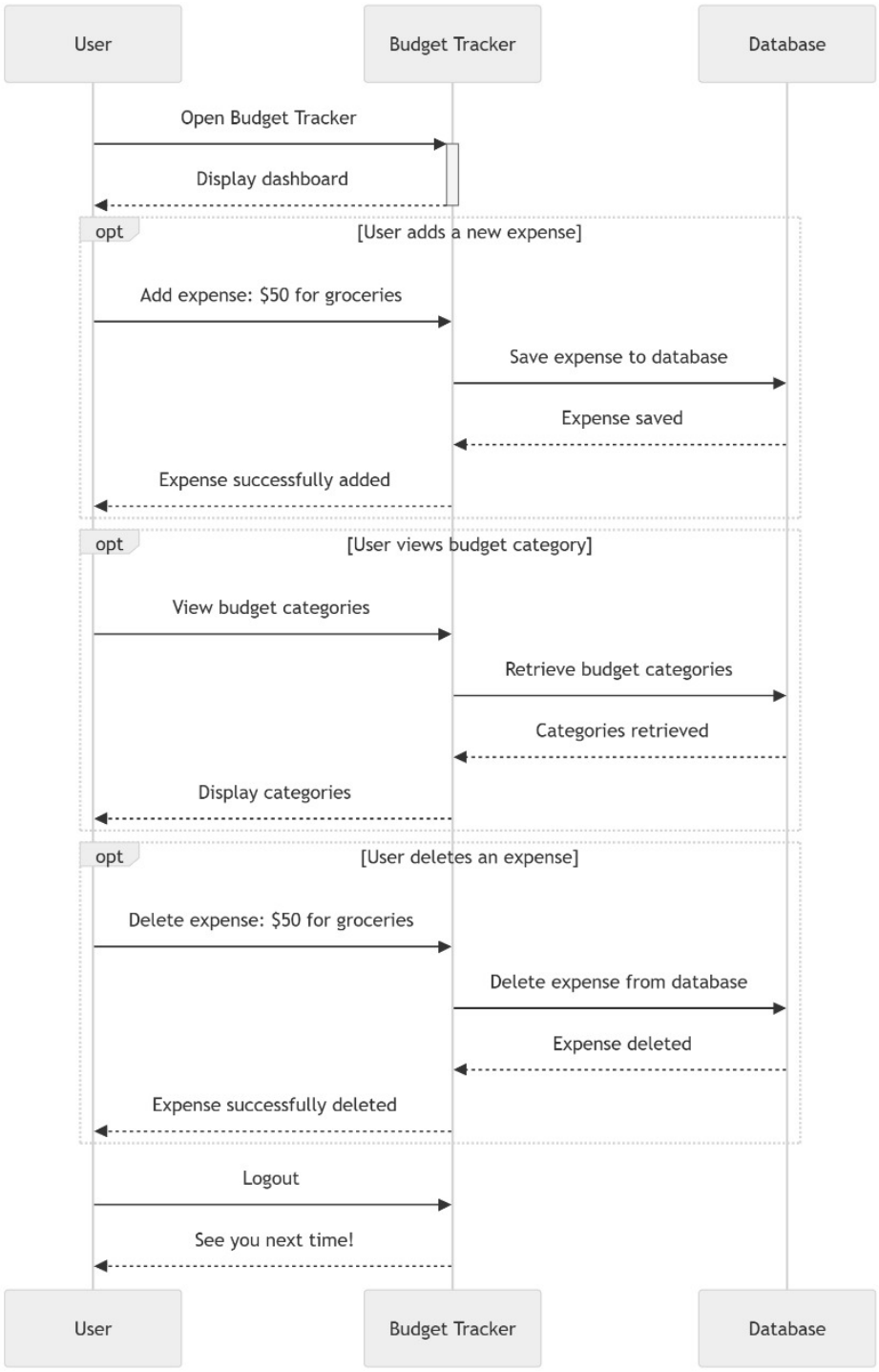

Fig. 3

- Appendix C: Issues List

Issue ID	Description	Status	Priority	Assigned to
ISSUE-1	Implement user authentication and authorization.	Open	High	Developer A
ISSUE-2	Add a feature to categorize expenses.	Close	Medium	Developer B
ISSUE-3	Improve the user interface for better user experience.	Open	Low	Developer C
ISSUE-4	Optimize database queries for faster performance.	Open	Medium	Developer A