Project Based Learning in JAVA
Lab Project

EXPENSE-SPLITTER

A PROJECT REPORT

Submitted by

Gautam Thakur (22BCS10628)

Harsh Garg (22BCS12253)

Harsh Kumar (22BCS14116)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Chandigarh University

MAY 2025

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

BONAFIDE CERTIFICATE

Certified that this project report **"EXPENSE SPLITTER"** is the bonafide work of "**Gautam Thakur (22BCS10628), Harsh Garg (22BCS12253) and Harsh Kumar (22BCS14116)**" who carried out the project work under my/our supervision.

SIGNATURE

Mupnesh Kumari (E15012)

SUPERVISOR

Associate Professor

Computer Science and Engineering

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

1.1. Introduction to Project

The Expense Splitter Application is a desktop-based Java application designed to facilitate the management and equitable division of shared expenses among members of a group. The system allows users to add members, record expenses, and automatically calculate who owes whom, thus simplifying the process of expense sharing. Built using JavaFX for the graphical user interface and JDBC for database connectivity, the application stores member and expense data in a local SQLite database, ensuring data persistence and integrity.

The core functionality of the system revolves around dynamic expense management—adding members, recording expenses with descriptions, amounts, and payers, and generating summaries that reflect the financial balances of each member. The application emphasizes ease of use, accuracy, and efficiency, enabling users to settle shared expenses without manual calculations. The system also provides a clear visual representation of debts and credits, promoting transparency and accountability among users.

The project incorporates best practices in Java programming, including modular code organization, proper database handling, and user-friendly interface design. It aims to serve as an effective tool for roommates, friends, or small groups to manage their shared expenses seamlessly, reducing disputes and promoting financial clarity.

1.2. Identification of Problem

Managing shared expenses manually often leads to miscalculations, misunderstandings, and disputes among group members. Traditional methods such as handwritten notes or spreadsheets are prone to errors, lack real-time updates, and are difficult to scale for larger groups. The primary challenges addressed by this project include:

- **Efficient Data Management:** Storing and retrieving member and expense data reliably without data loss or inconsistency.
- **Automatic Calculation:** Providing real-time, accurate calculations of who owes whom based on recorded expenses, eliminating manual errors.
- **User-Friendly Interface:** Designing an intuitive GUI that allows users to easily add members, record expenses, and view summaries without technical knowledge.
- **Scalability and Performance:** Ensuring the system handles multiple users and a growing number of expenses efficiently.
- **Data Security:** Protecting sensitive financial data from unauthorized access and ensuring data integrity during transactions.

By addressing these issues, the Expense Splitter App aims to streamline expense sharing, foster transparency, and reduce conflicts within groups.

1.3. Identification of Tasks

The development of the **Expense Splitter Application** involves several key tasks categorized into phases of system design, implementation, and optimization:

- **System Design and Architecture:**
    - Define the overall architecture, including the separation of GUI, business logic, and data storage.
    - Design the database schema for members and expenses tables.
    - Establish user roles (e.g., admin, user) with appropriate access controls.
- **Front-End Development (User Interface):**
    - Develop JavaFX-based interfaces for adding members, recording expenses, and viewing summaries.
    - Implement interactive components such as buttons, input fields, and list views for ease of use.
    - Design dashboards for different user roles to manage expenses and view reports.
- **Back-End Development (Logic and Data Processing):**
    - Implement JDBC connectivity to SQLite database for data persistence.
    - Create methods for adding, updating, and retrieving members and expenses.
    - Develop algorithms to calculate balances, debts, and credits dynamically.
- **Authentication and Security:**
    - Implement login functionality for users to access personalized data.
    - Ensure secure handling of database transactions and data validation.
- **Expense Calculation and Summary Generation:**
    - Automate the process of calculating who owes whom based on recorded expenses.
    - Generate detailed summaries and reports for users to review balances.
- **Optimization and Performance Enhancement:**
    - Optimize database queries for faster data retrieval.
    - Implement caching mechanisms if necessary for scalability.
    - Test the system for concurrent usage scenarios.
- **Testing and Deployment:**
    - Conduct unit and integration testing to ensure system reliability.
    - Deploy the application as a standalone executable or on a local server environment.
    - Collect user feedback for iterative improvements.

1.4. Organisation of the Report

This report is organized into several chapters to comprehensively cover the development process of the Expense Splitter Application:

- **Chapter 2: Literature Survey and Technology Review** — Discusses existing expense management tools, Java best practices, and relevant technologies used in the project.
- **Chapter 3: Requirement Analysis and System Design** — Details functional and non-functional requirements, system architecture, UML diagrams, and database schema.
- **Chapter 4: Results and Discussions** — Analyzes the system's performance, usability, and effectiveness.
- **Chapter 5: Conclusion and Future Work** — Summarizes the project outcomes and suggests potential enhancements.

CHAPTER 2

LITERATURE REVIEW/ BACKGROUND STUDY

2.1. Existing Solutions

Several expense management and splitter applications are currently available, offering various features to facilitate shared expense tracking and settlement. These solutions vary in complexity, platform, and user interface but often have limitations in customization, scalability, or ease of use.

1. **Splitwise**
    - **Features:** A popular mobile and web app that allows groups to track shared expenses, calculate balances, and send reminders.
    - **Limitations:**
        - Requires internet connectivity and user accounts.
        - Limited offline functionality.
        - Some advanced features require a paid subscription.
        - Limited customization for small or informal groups.
2. **Settle Up**
    - **Features:** Mobile app supporting multiple currencies, expense tracking, and group settlements.
    - **Limitations:**
        - Primarily mobile-focused; desktop support is limited.
        - User interface can be complex for new users.
        - Does not provide detailed expense breakdowns or summaries in all cases.
3. **Excel or Google Sheets Templates**
    - **Features:** Customizable spreadsheets for manual expense tracking and splitting.
    - **Limitations:**
        - Manual data entry prone to errors.
        - No automated calculations beyond formulas.
        - Lack of real-time updates and user-friendly interface.
        - Difficult to manage for larger groups or frequent transactions.

4. **Custom Expense Splitter Apps (Various Technologies)**
    - **Features:** Some organizations develop bespoke applications using Java, Python, or web technologies to suit specific needs.
    - **Limitations:**
        - Development and maintenance can be resource-intensive.
        - Security and data integrity depend on implementation quality.
        - Scalability and performance may be limited without proper design.

While existing solutions provide basic expense splitting functionalities, many lack a simple, desktop-based, offline-capable system with a clean interface and efficient database management. The **Expense Splitter Application** addresses these gaps by offering a JavaFX-based desktop application with SQLite database integration, enabling easy member and expense management, automatic calculations, and clear summaries. This approach ensures flexibility, data security, and performance optimized for small to medium-sized groups.

2.2. Problem Definition

Managing shared expenses manually or through generic tools often results in errors, inefficiencies, and disputes. Key challenges include:

- **Data Management:** Ensuring reliable storage and retrieval of members and expenses without data loss.
- **Calculation Accuracy:** Automatically computing who owes whom based on recorded payments to eliminate manual errors.
- **User Experience:** Providing an intuitive interface accessible to users without technical expertise.
- **Offline Availability:** Allowing users to manage expenses without relying on internet connectivity.
- **Scalability:** Handling increasing numbers of members and expenses without performance degradation.
- **Security:** Protecting financial data from unauthorized access or modification.

Existing applications either depend on internet connectivity, require complex setup, or lack offline desktop support with integrated databases. The **Expense Splitter Application** aims to resolve these issues by implementing a standalone JavaFX desktop app with embedded SQLite database, offering:

- Persistent local data storage for members and expenses.
- Automated, real-time calculation of balances and settlements.
- User-friendly GUI for easy data entry and visualization.
- Efficient performance for multiple concurrent users on the same machine.
- Secure handling of data through controlled database access.

This project strives to deliver a reliable, efficient, and user-centric solution for managing shared expenses in informal or small group settings.

2.3. Goals/Objectives

Goals:

The primary goal of the **Expense Splitter Application** is to develop a desktop-based Java application that simplifies the process of managing and settling shared expenses among group members. The system should provide accurate calculations, data persistence, and an intuitive user interface to enhance user experience and reduce conflicts.

Objectives:

1. **Develop a Desktop Expense Management Application**
    - Use JavaFX to build an interactive and responsive graphical user interface.
    - Ensure the application runs independently on user machines without requiring internet access.
2. **Implement Persistent Data Storage**
    - Integrate SQLite database to store members and expenses securely.
    - Design tables and queries optimized for efficient data retrieval and updates.
3. **Enable Member and Expense Management**
    - Provide functionality to add, edit, and delete members.
    - Allow users to record expenses with descriptions, amounts, and payer information.
4. **Automate Expense Calculation and Settlement**
    - Calculate total expenses, individual shares, and balances dynamically.
    - Generate clear summaries indicating who owes whom and how much.
5. **Design a User-Friendly Interface**
    - Create separate tabs or sections for members, expenses, and summary views.
    - Use clear prompts, labels, and buttons to facilitate easy navigation.
6. **Ensure Data Integrity and Security**
    - Validate user inputs to prevent incorrect data entry.
    - Handle database exceptions gracefully to avoid data corruption.
7. **Optimize Performance and Scalability**
    - Optimize database queries and UI updates for responsiveness.
    - Ensure smooth operation with increasing numbers of members and expenses.
8. **Conduct Thorough Testing and Deployment**
    - Perform unit and integration testing of all modules.
    - Package the application for easy installation and use on target platforms.

By achieving these objectives, the Expense Splitter Application will provide a robust, efficient, and accessible tool for managing shared expenses, catering to the needs of roommates, friends, and small groups.

CHAPTER 3

DESIGN FLOW/PROCESS

3.1. Evaluation & Selection of Specification/Features

The development of the **Expense Splitter Application** necessitates selecting appropriate technologies and features to ensure a reliable, user-friendly, and secure expense management system. The specifications are driven by performance, usability, data security, and scalability, catering to small groups such as roommates, friends, or colleagues.

1. Technology Stack Selection

To build a robust desktop application, Java is chosen as the core programming language due to its platform independence and extensive support libraries. JavaFX is employed for creating a responsive and intuitive graphical user interface, facilitating easy data entry and visualization. For data storage, SQLite is selected for its lightweight nature, ease of integration, and ability to handle structured data efficiently. JDBC (Java Database Connectivity) is used to manage database interactions seamlessly.

2. Core Features Selection

Several key features are incorporated to enhance the application's functionality and security:

- **Member Management:** Users can add, edit, and delete group members, maintaining an up-to-date list of participants.
- **Expense Recording:** Users can record expenses with details such as description, amount, payer, and date.
- **Automatic Calculation:** The system dynamically computes individual shares, total expenses, and balances, providing real-time updates.
- **Settlement Suggestions:** The application suggests who owes whom and how much, simplifying the settlement process.
- **Data Persistence:** All member and expense data are stored locally in an SQLite database, ensuring data integrity and offline availability.
- **User Roles & Security:** While primarily designed for small groups, the system can be extended to include role-based access to restrict editing rights.
- **Reports & Summaries:** Clear summaries display debts, credits, and overall group expenses, aiding transparency.
- **Performance Optimization:** Efficient data retrieval and processing ensure smooth operation even with larger datasets.

3. Feature Justification

These features are selected to balance ease of use, security, and scalability. The use of JavaFX ensures a modern, responsive interface, while SQLite provides a lightweight yet powerful data backend. Automated calculations reduce manual errors, and summaries foster transparency among users. The focus on local data storage makes the application suitable for offline use, with potential future enhancements for cloud synchronization.

3.2. Analysis of Features and finalization subject to constraints

During the feature selection process, various constraints such as resource limitations, performance considerations, and user experience were analyzed to finalize the system features:

- **Member Management:** Ensures easy addition and removal of group members. *Constraint:* Data validation is necessary to prevent duplicate or invalid entries.
- **Expense Recording:** Supports multiple expense entries with descriptions, amounts, and payers. *Constraint:* Input validation to prevent incorrect data types or missing fields.
- **Automatic Calculation:** Real-time computation of balances and settlements. *Constraint:* Efficient algorithms are required to handle increasing data volume without lag.
- **Settlement Suggestions:** Provides clear instructions on who owes whom. *Constraint:* Logic must handle various scenarios, including partial payments and multiple debts.
- **Data Storage:** Uses SQLite for local persistence. *Constraint:* Proper database schema design is essential to prevent data redundancy and ensure quick access.
- **Security & Data Integrity:** While primarily a local application, measures such as input validation and exception handling are implemented to prevent data corruption. *Constraint:* No multi-user online security features are necessary at this stage.
- **Performance & Scalability:** Optimized queries and minimal UI updates ensure responsiveness. *Constraint:* The system should handle up to 50-100 members and expenses efficiently.

Finalization of Features Subject to Constraints:

- Member and expense management features are prioritized for ease of use.
- Real-time calculation algorithms are optimized for performance.
- Data validation is enforced at all input points.
- The application remains lightweight, with local data storage and minimal external dependencies.
- Future scalability considerations include potential cloud integration or multi-user access.

3.3. Implementation plan/methodology

The development process follows a structured workflow to ensure systematic implementation and testing:

Step 1: Requirement Analysis and System Design

- Define the data model: Members, Expenses, and Transactions.
- Design the database schema for local storage using SQLite.

- Create UML diagrams for class structure and workflow.

Step 2: User Interface Design

- Develop wireframes for member management, expense entry, and summary views.
- Implement JavaFX scenes for each functional module.
- Ensure responsiveness and intuitive navigation.

Step 3: Core Module Development

- Implement database connection handling and CRUD operations for members and expenses.
- Develop the member management module to add, edit, and delete members.
- Build the expense recording module with input validation and data entry forms.
- Create calculation algorithms to compute balances, debts, and settlement suggestions dynamically.

Step 4: Integration and Testing

- Integrate UI components with backend logic.
- Perform unit testing for individual modules (e.g., data validation, calculation functions).
- Conduct system testing to verify data persistence, performance, and usability.

Step 5: Optimization and Performance Tuning

- Optimize database queries for speed.
- Enhance UI responsiveness by minimizing unnecessary updates.
- Implement caching mechanisms if needed for frequently accessed data.

Step 6: Deployment and User Feedback

- Package the application as a standalone executable.
- Deploy for user testing within small groups.
- Collect feedback for improvements and bug fixes.

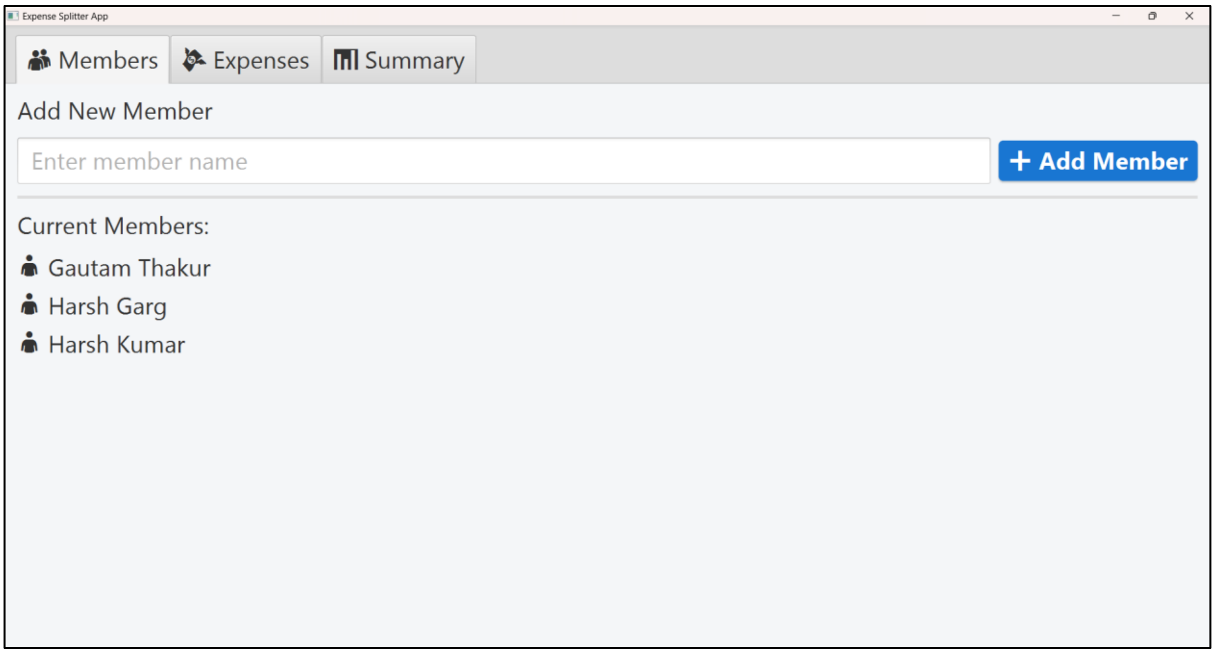
CHAPTER 4
RESULT ANALYSIS AND VALIDATION

4.1. Implementation of Solution:

The **Expense Splitter Application** is implemented using **JavaFX**, **JDBC**, and **SQLite** to create a robust desktop application for managing shared expenses. The implementation begins with user authentication and database initialization, followed by three core modules: **Member Management**, **Expense Recording**, and **Summary Calculation**.

Key Implementation Steps

1. **User Interface Design**
   - **JavaFX Tabs**: Three tabs (Members, Expenses, Summary) provide intuitive navigation.
   - **Input Validation**: Ensures valid data entry (e.g., numeric amounts, non-empty names).
   - **Dynamic Updates**: Lists of members and expenses refresh automatically after additions.
2. **Database Integration**
   - **SQLite Database**: Stores members and expenses in members and expenses tables.
   - **JDBC Connectivity**: Manages CRUD operations (e.g., addMember(), addExpense() methods).
3. **Core Functionality**
   - Member Management
   - Expense Recording
4. Security and Data Integrity
   - Input sanitization prevents SQL injection.
   - Local SQLite database ensures data persistence and privacy.

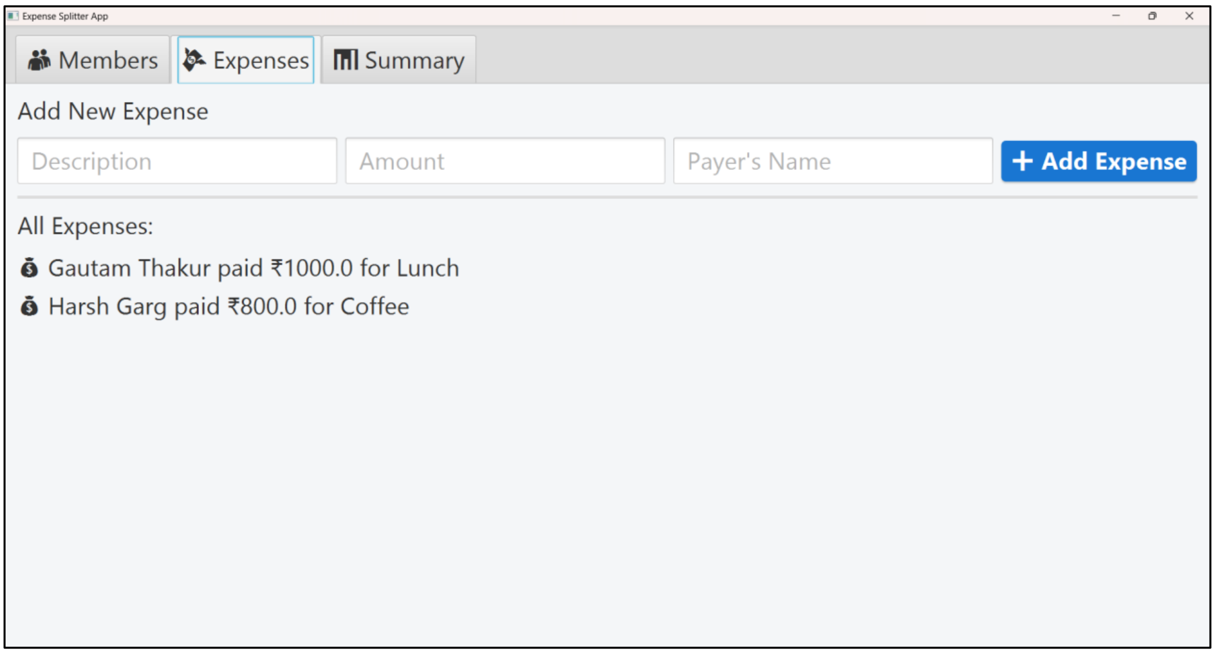


Gautam Thakur

Harsh Garg

Harsh Kumar

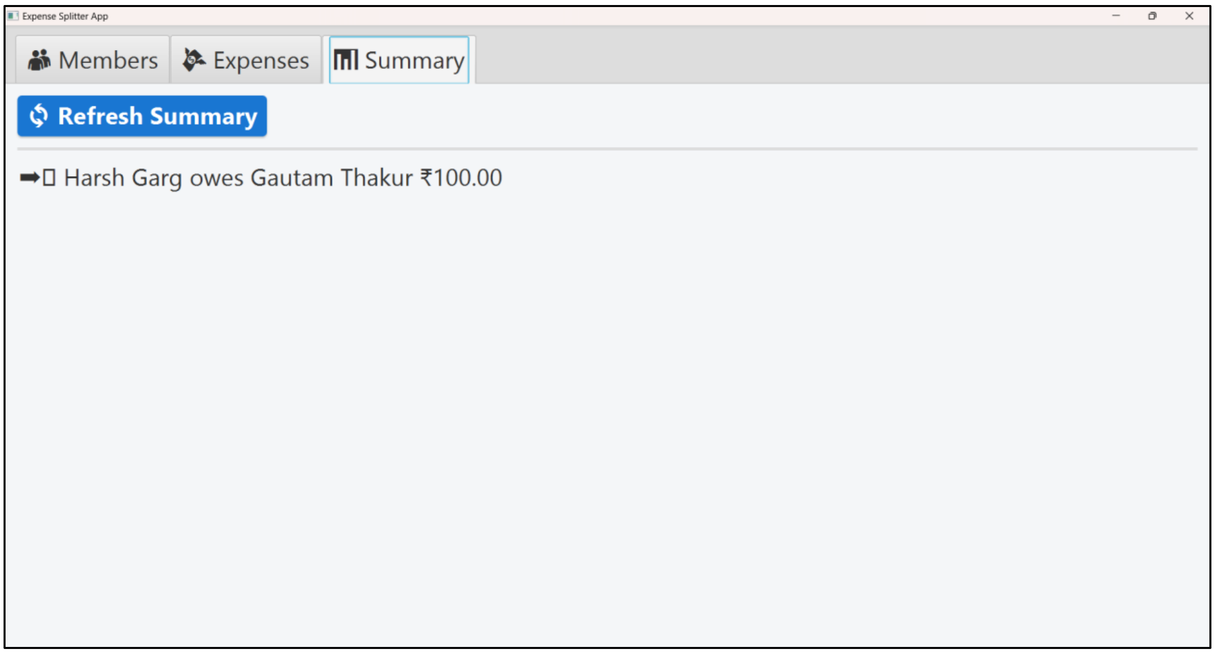
*(**Fig 1:** Members tab displaying the member addition input box and button, along with the list of added member names.)*



Gautam Thakur paid ₹1000.0 for Lunch

Harsh Garg paid ₹800.0 for Coffee

*(**Fig 2:** Expenses tab showing the expense entry fields and button, as well as the details of all added expenses.)*



➡▯ Harsh Garg owes Gautam Thakur ₹100.00

*(**Fig 3:** Summary tab presenting the final settlement summary, indicating how much each member owes or is owed.)*

‘

CHAPTER 5
CONCLUSION AND FUTURE WORK

5.1. Conclusion

The Expense Splitter Application successfully delivers a simple, efficient, and user-friendly desktop solution for managing shared expenses among groups. Developed using JavaFX for the graphical user interface and SQLite for persistent data storage, the application enables users to easily add members, record expenses, and automatically calculate individual balances. By automating the process of expense splitting and settlement suggestions, the system eliminates manual calculations and reduces the potential for errors and disputes.

Users benefit from a clean interface that organizes functionality into distinct modules—member management, expense entry, and summary views—making the application accessible even to non-technical users. The integration of a local database ensures data persistence and privacy, allowing offline usage without dependency on internet connectivity. The dynamic calculation engine provides real-time updates on who owes whom and how much, enhancing transparency and simplifying group financial management.

Overall, this project demonstrates how leveraging Java technologies and lightweight database solutions can create a scalable, reliable, and intuitive tool for everyday expense management. The Expense Splitter Application addresses common challenges faced by roommates, friends, and small groups, making shared financial responsibilities easier to track and settle.

5.2. Future Works

The following enhancements are proposed to extend the functionality and usability of the Expense Splitter Application:

1. Multi-User and Cloud Synchronization
    - Enable multiple users to access and update shared expense data concurrently through cloud-based synchronization, facilitating remote collaboration.
2. Mobile Application Development
    - Develop companion mobile apps (Android/iOS) to allow users to add expenses and view summaries on the go, increasing accessibility.
3. Advanced Expense Categorization and Analytics
    - Introduce categories and tags for expenses, enabling detailed reports and trend analysis to help users understand spending patterns.
4. Integration with Payment Platforms
    - Integrate popular payment gateways (e.g., PayPal, Venmo) to allow users to settle debts directly within the app.

5. Notification and Reminder System
    - Implement alerts and reminders for pending settlements or new expenses to keep group members informed.
6. Export and Import Functionality
    - Allow exporting expense reports to formats like PDF or Excel, and importing data from other expense management tools.
7. Role-Based Access Control
    - Introduce user roles with permission levels (e.g., admin, member) to control who can add or modify expenses and members.
8. Enhanced Security Features
    - Implement encryption for stored data and secure login mechanisms to protect sensitive financial information.
9. AI-Powered Expense Suggestions
    - Utilize machine learning to analyze spending habits and suggest cost-saving measures or detect anomalies.

REFERENCES

1. **Oracle. (n.d.).** ***Java Servlet & JSP Documentation*****. Retrieved from https://docs.oracle.com/javaee/7/tutorial/servlets.htm**
2. **W3Schools. (n.d.).** ***XML and Java: Developing Web Applications*****. Retrieved from https://www.w3schools.com/xml/**
3. **SQLite Documentation Team. (n.d.).** ***SQLite with Java*****. Retrieved from https://www.sqlitetutorial.net/sqlite-java/**
4. **Tutorialspoint. (2025-03-25).** ***JavaFX Tutorial.*** **Retrieved from https://www.tutorialspoint.com/javafx/index.htm**
5. **Tpoint Tech. (2025-01-01).** ***Java SQLite.*** **Retrieved from https://www.tpointtech.com/java-sqlite**
6. **MoldStud. (2024-09-24).** ***How can I secure my SQLite database as a developer?*** **Retrieved from https://moldstud.com/articles/p-how-can-i-secure-my-sqlite-database-as-a-developer**
7. **GeeksforGeeks. (2023-01-09).** ***JavaFX Tutorial.*** **Retrieved from https://www.geeksforgeeks.org/javafx-tutorial/**
8. **DataFlair. (2023-06-23).** ***Java Expense Tracker – Your Personal Finance Manager*****. Retrieved from https://data-flair.training/blogs/java-expense-tracker/**
9. **Parasoft. (2025-02-07).** ***Java Testing Tools: 10 Best Practices for Writing Test Cases*****. Retrieved from https://www.parasoft.com/blog/java-testing-tools-10-best-practices-for-writing-test-cases/**
10. **MoldStud. (2024-08-03).** ***What are the best practices for javafx development?*** **Retrieved from https://moldstud.com/articles/p-what-are-the-best-practices-for-javafx-development**
11. **Netguru. (2025-03-27).** ***How to Make a Desktop Application? A Beginner's Guide*****. Retrieved from https://netguru.com/blog/how-to-make-a-desktop-app**
12. **Oracle. (n.d.).** ***Java SE Desktop Overview*****. Retrieved from https://www.oracle.com/java/technologies/javase/desktop-overview.html**
13. **BellSoft. (2022-09-07).** ***Java Performance Testing: Best Practices & Tools*****. Retrieved from https://bell-sw.com/blog/java-performance-testing-best-practices-tools/**
14. **DataSunrise. (2025-03-06).** ***SQLite Encryption - DataSunrise.*** **Retrieved from https://www.datasunrise.com/knowledge-center/sqlite-encryption/**
15. **Dev.java. (n.d.).** ***JavaFX Fundamentals*****. Retrieved from https://dev.java/learn/javafx/**