Application Development(APD545NDD.12037.2251)
Final Project

Full Name: Radmehr Behzadfar
Id: 148786221
Seneca email: rbehzadfar@myseneca.ca

Personal Finance Management Application

Project Report

1. Introduction

The Personal Finance Management Application is a financial desktop application that enables users to record their transactions and manage budgets efficiently. It is developed with Java, JavaFX, and SQLite. This program allows users to input their income and expenditure and subsequently set financial goals or limitations, viewing their data represented in various dynamic forms through charts. This project is developed using the Model-View-Controller (MVC) architectural pattern which promotes separation of concerns making the application easier to maintain. This application uses JavaFX's Task framework for executing concurrency in order to keep the UI responsive during heavy processing tasks.

2. Application Features and Functionality

The application incorporates several key features aimed at providing comprehensive personal finance management:

- **Transaction Management:**
  Users can add, edit, and delete financial transactions. Each entry includes a date, amount, category, description, and a type indicator (either "income" or "expense"). The application validates all user inputs to ensure data integrity.

- **Budget Management:**
  A dedicated module allows users to set monthly budget limits for different categories such as food, utilities or even entertainment. The application monitors expenses and notifies users when their spending is close or exceeds the set limits.

- **Summary Reports:**
  The Summary module provides visual representations of the user's financial data. It features a pie chart displaying expense distribution by category and a bar chart summarizing monthly net totals (calculated as income minus expenses). These visual aids help users quickly understand their financial trends.

- **Data Validation and Error Handling:**
  Input data is rigorously validated—dates must follow the "yyyy-MM-dd" format, amounts must be positive, and mandatory fields cannot be left blank. The application uses alerts and logs errors to guide users and maintain robustness.

- **Database Integration:**
  An SQLite database saves all data the is related to transactions or budget. The schema will be automatically created on its first run of the application using a database initializer. This ensures that users can immediately start managing their finance without any additional setup.

3. Architecture and MVC Implementation

The project is architected following the Model-View-Controller (MVC) pattern, which will divide the application into three components related to each other:

- **Model:**
  The model layer comprises classes that represent the data entities of the application. Key classes include Transaction and Budget, which encapsulate the properties of financial records and budget limits. The Database class is responsible for all interactions with the SQLite database, including executing CRUD operations.

- **View:**
  The view layer is defined by JavaFX FXML files, which specify the layout and visual elements of the application. The main user interface is structured with a BorderPane layout, featuring a navigation pane (tabs) for accessing the Transactions, Budget, and Summary sections. Each section provides interactive controls, such as forms and charts, for data entry and visualization.

- **Controller:**
  Controllers handle user actions and serve as the bridge between the view and model layers. For instance, the TransactionsController processes user inputs for creating or modifying transactions and then communicates with the database to

update records. The SummaryController manages the generation of visual reports and utilizes background tasks to process data without hindering the user experience.

This separation of concerns not only improves code maintainability but also facilitates independent testing of individual components.

4. Concurrency Features and Threading

To ensure a smooth and responsive user experience, the application incorporates concurrency techniques:

- **Background Processing:**
  The Summary module utilize a JavaFX Task to perform resource-intensive operations such as data aggregation and chart data preparation. This task runs on a separate thread so that the main UI thread remains free for user interaction.

- **Safe UI Updates:**
  Due to JavaFX requirements of interfacing updates happening on the primary thread, the application incorporates “Platform.runLater” to safely update chart data after the background processing is complete. This design reduces application hangs during long operations.

- **Thread Management:**
  By offloading heavy computations to background threads, the application maintains high responsiveness even when handling large data sets. This approach exemplifies good use of concurrency in desktop applications.

5. Known Issues and Limitations

While the application meets its primary objectives, there are several areas for future enhancement:

- **Input Duplication:**
  Currently, the application does not actively check for duplicate entries. Future iterations might include logic to identify and prevent duplicate transactions or budget entries.

- **Error Handling:**
  The current error handling strategy primarily relies on printing stack traces and displaying alerts. Implementing a more robust logging framework would improve diagnostics and error management.

- **User Interface Enhancements:**
  The user interface serves its purpose, but the addition of visual polish, supplementary tools, dynamic notifications, and more proactive assistance could enhance the overall experience.

- **Scalability:**
  The system is tailored for a single user and small data sets, but will likely need an architectural shift in design to facilitate a multi-user system with high data capacity.

- **Extended Features:**
  Additional features such as recurring transactions, data export options (CSV, PDF), and advanced budgeting tools can also enhance more to further functionalities of the application.

6. Conclusion

The application for managing personal finances is a powerful tool for balancing personal finances. It tracks spendings by providing budget, and summary of reports that can help users understand their spendings. The construction of application according to MVC design enhances the sustainability of the codebase, and use of JavaFX concurrency features improves user experience. While there are possibilities to improve the application, the implementation has already met the project requirements and it can serve as a starting point for further enhancements.