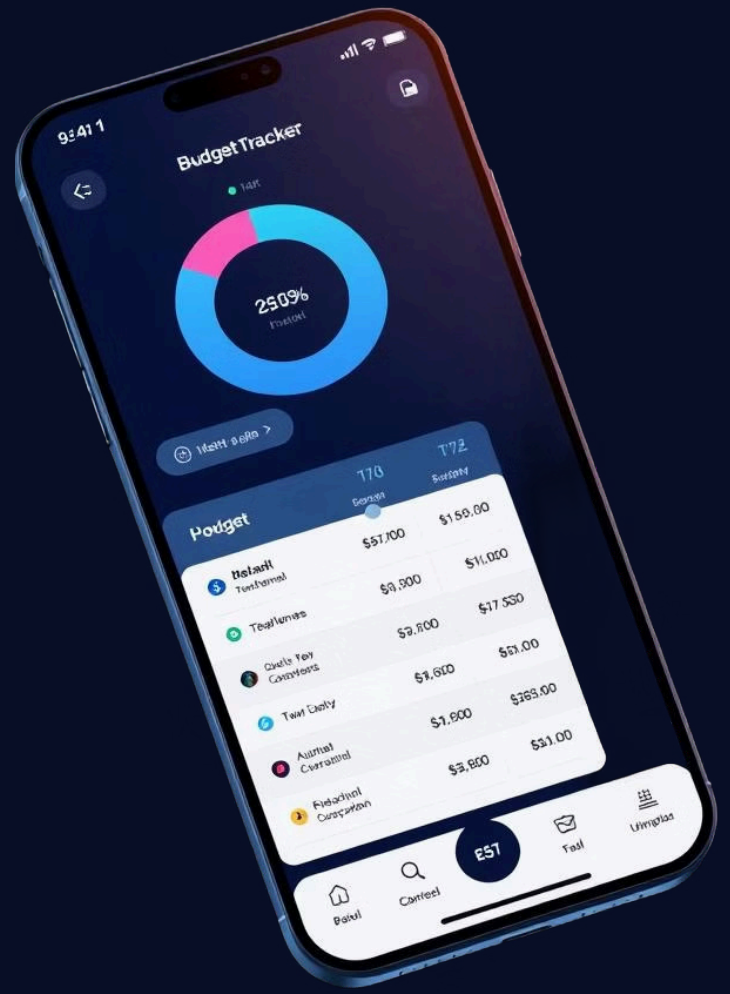
Budget Tracker with Auto-Fetch and Drag-and-Drop Categorization

You can easily keep track of your expenses and income with our budget tracker. The auto-fetch feature automatically imports your financial data from your bank accounts, making it simple to see where your money is going. In addition, the drag-and-drop categorization feature allows you to easily organize your transactions into different categories for better budgeting.

This presentation outlines the design and development plan for a budget tracker application that automates transaction fetching, integrates a drag-and-drop categorization system, and offers interactive visualization through a user-friendly GUI.



Key Features and Functionality

1 Auto-fetch Transactions

Automate transaction fetching from SMS messages using Selenium, targeting Google Messages for efficient data extraction.

2 Drag-and-Drop Categorization

Implement a drag-and-drop interface for intuitive categorization of transactions, directly influencing the Pie Chart visualization.

3 Database Integration

Integrate SQLite for storing transaction details, ensuring persistence and retrieval of financial data.

4 Interactive GUI

Create an interactive GUI using JavaFX to provide users with a clear visual representation of their budget and transactions.

Design Patterns for Enhanced Structure and Scalability

To enhance the structure and scalability of the budget tracker application, the following design patterns will be implemented:

Singleton Pattern

Utilize the Singleton pattern for the database integration, ensuring a single instance of the SQLite connection throughout the application.

Observer Pattern

Implement the Observer pattern to allow components to receive updates when transaction data or categorization changes, enabling real-time visualization updates.

Singleton Pattern

Manage database connections and user sessions effectively, ensuring a single instance of critical objects.

Observer Pattern

Enable real-time updates of the Pie Chart in response to new transactions, providing dynamic visualization.

Factory Pattern

Create objects for different transaction types (Income/Expense), promoting flexible and scalable code.

Command Pattern

Implement Undo/Redo functionality for user actions, allowing for easy correction of categorization mistakes.

Expected Application Flow: A Step-by-Step Breakdown

1. The user initiates the application by entering their login credentials. The Singleton pattern manages the session for seamless user interaction.
2. Selenium automatically extracts transaction data from SMS messages. Regular expressions are used for parsing and extracting relevant information.
3. Users can then visually categorize transactions by dragging them to specific sections of the Pie Chart, providing interactive data manipulation.
4. Categorized transactions are stored in the SQLite database, ensuring data persistence and accessibility.
5. The Observer pattern is triggered by new transactions, dynamically updating the Pie Chart to reflect the latest budget breakdown.
6. The Command pattern facilitates Undo/Redo functionality, allowing users to easily correct categorization mistakes, ensuring accuracy and control.

Simplified Roadmap: Milestones for Project Development

1

Build GUI

Design the GUI using JavaFX, focusing on a user-friendly interface for the login page, transaction table, and interactive Pie Chart.

2

Implement Patterns

Incorporate Singleton, Observer, and Factory patterns into key components, ensuring a structured and scalable codebase.

3

Integrate Automation

Set up Selenium for automated transaction fetching from SMS messages, leveraging Google Messages for efficiency.

4

Database Integration

Connect SQLite to store and retrieve categorized transaction data, ensuring data persistence and retrieval.

5

Polish UI

Add drag-and-drop functionality and real-time updates to the Pie Chart, enhancing user experience.