Budgeteers

Nehang Patel
Computer Science

Tharun Swaminathan
Ravi Kumar
Computer Science

Shruti Asolkar
Computer Science

Project Advisor - Dr. Nitin Nitin

Tech Stack

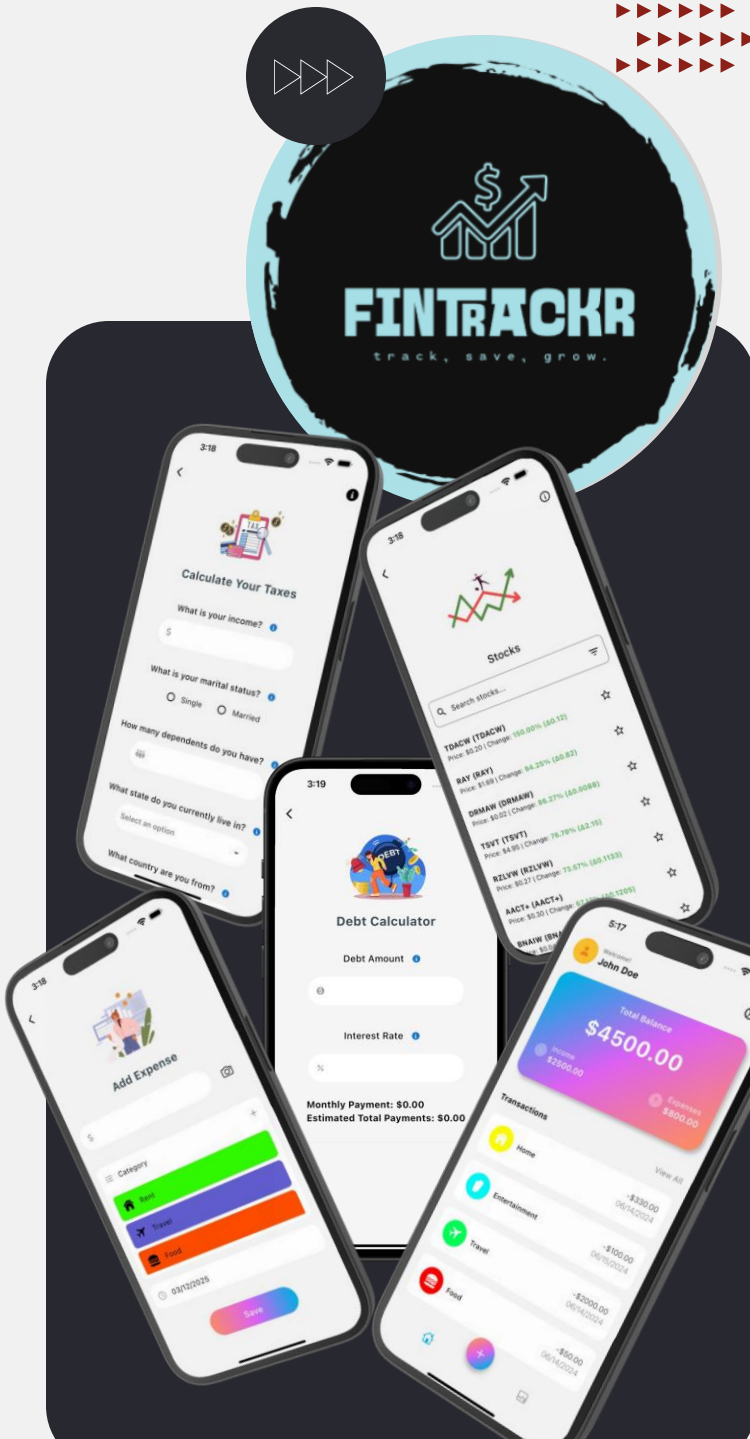
Dart bloc Firebase Flutter

Manage your expenses with confidence!

Senior Design Presentation

FinTrackr

GitHub Repo



FinTrackr – Project Purpose

Goal:

Develop an all-in-one financial solution that integrates daily money management with long-term investment opportunities.

Overview:

This project focuses on creating a comprehensive personal finance application that enables users to:

- Track expenses and income efficiently
- Monitor stock investments in real-time
- Improve budgeting and debt management
- Simplify tax preparation

Team Members & Faculty Advisor

Nehang Patel

Email:
patel3ng@mail.uc.edu

Phone:
513-726-8552

Shruti Asolkar

Email:
asolkasy@mail.uc.edu

Phone:
513-237-3457

Tharun Swaminathan Ravi Kumar

Email:
ravikutn@mail.uc.edu

Phone:
513-834-2011

Dr. Nitin Nitin

Email:
nitinfu@ucmail.uc.edu

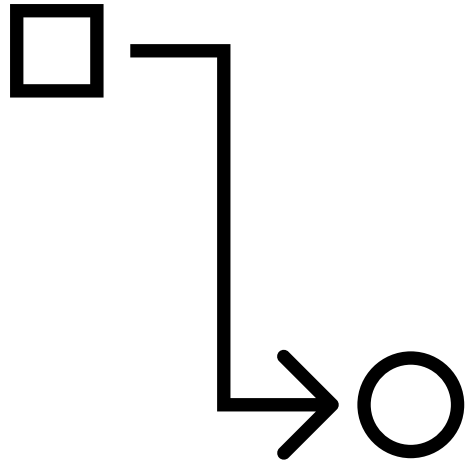
Phone:
513-556-7619

Intellectual Merits

- **Real-Time Expense Tracking** – Dynamic transaction list with Firestore sync
- **OCR Expense Entry** – Extracts data from receipts for automatically detecting expense amount
- **Stock Management** – Live stock updates with search & favorites
- **Debt & Tax Calculator** – Computes debt repayments with interest-based formulas and taxes with accurate "IRS" calculations
- **Seamless Flutter & Firebase Integration** – Ensures smooth real-time updates

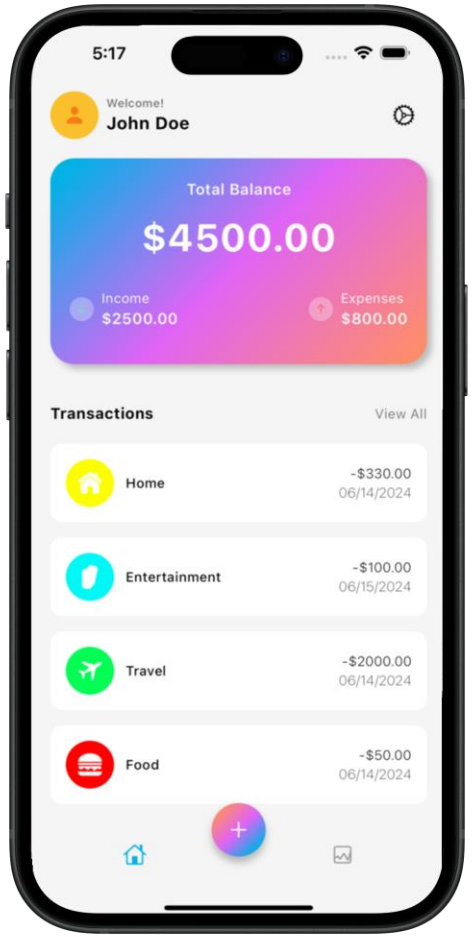
Broader Impacts

- Enhancing Financial Literacy
- Encouraging Financial Responsibility
- Reducing Financial Stress
- Enhancing Economic Stability
- Improved Financial Well-being
- Encouraging Smart Investments
- Financial Sustainability and Long-term Growth



System Overview

- **FinTrackr**: A Flutter-based expense tracking app with Firebase backend.
- **Architecture**:
  - **UI Layer**: Flutter widgets for seamless user interaction.
  - **State Management**: Bloc pattern for reactive state updates.
  - **Repository Layer**: Handles Firestore database operations.
  - **Data Layer**: Stores expenses & categories in Firestore.
- **Key Features**:
  - Add & categorize expenses.
  - Real-time data sync with Firestore.



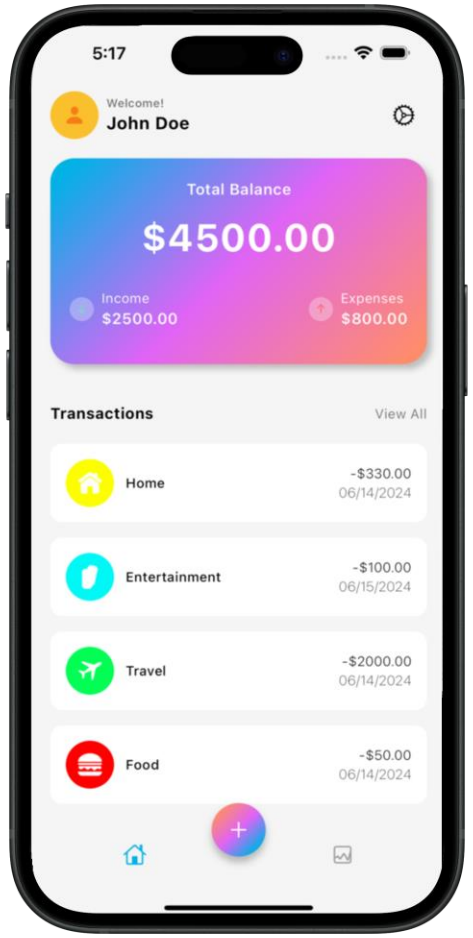
System Overview – Landing Page

Overview:

- Displays all expenses categorized by **type, date, and category**.
- Provides a **detailed breakdown** of each expense.
- Dynamically update the values in the balance card with every added expense.

Implementation:

- Uses Firestore to fetch expenses dynamically.
- Displays expenses in a **scrollable list**.



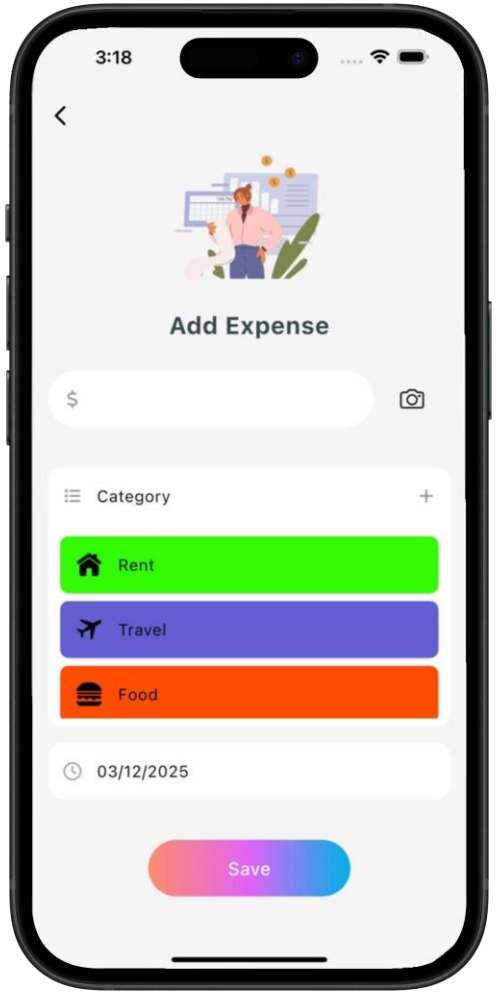
System Overview – Add Expense

Overview:

- Allows users to manually add expenses or use OCR scanning for detecting expenses from receipts.
- Users can create/select categories, add dates and enter amounts for each expense.

Implementation:

- The form tracks the page's state dynamically when users enter details.
- Expenses are stored in Firestore with real-time updates to the Transactions/Landing Page.



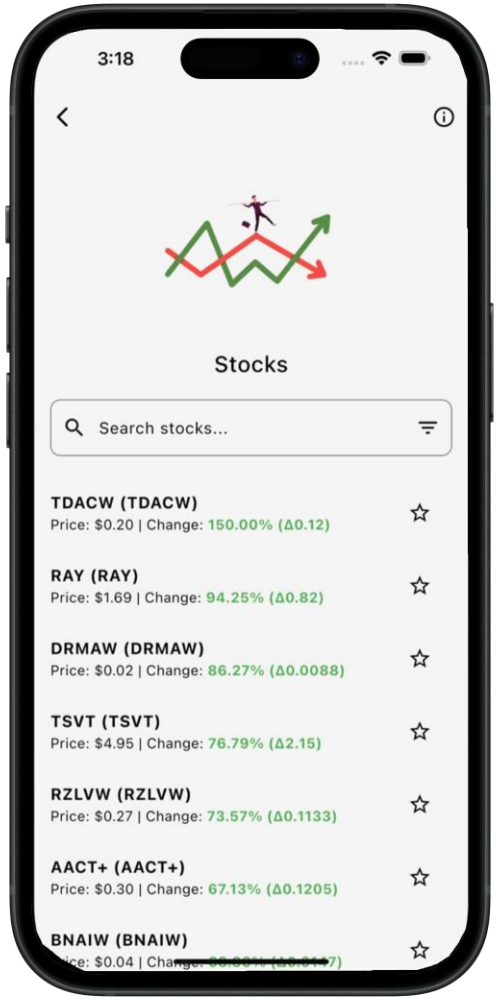
System Overview – Stocks Page

Overview:

- Tracks real-time stock market data.
- Displays portfolio performance and stock price changes.
- Allows users to favorite and monitor stocks.
- Displays more information about selected stock like price graph, company information, and so on, when a stock is clicked.

Implementation:

- Fetches stock prices from an external API.
- Dynamically updates stock information in the UI.



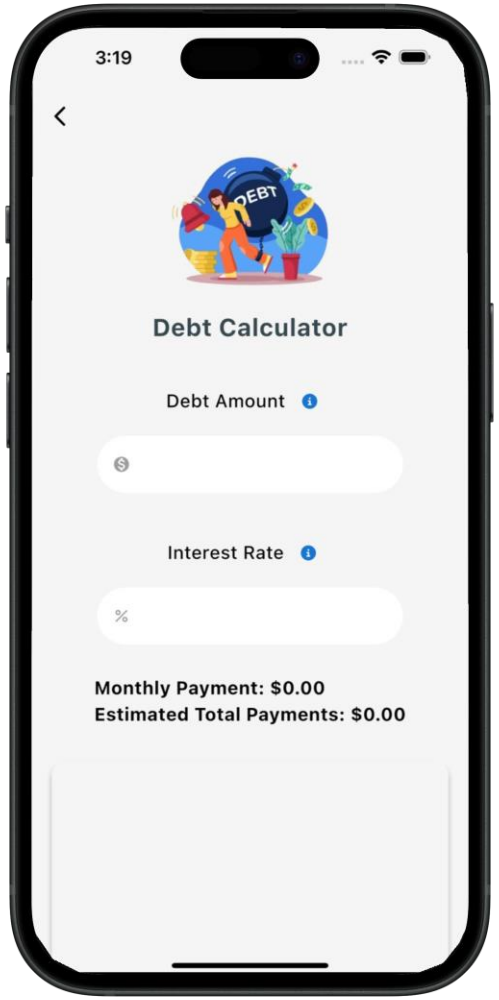
System Overview – Debt Page

Overview:

- Allows users to calculate loan repayments based on interest rates.
- Provides a debt breakdown, showing principal, interest, and total payable amount.
- Helps users estimate monthly payments using a debt calculator.
- Displays a pie chart to help users visualize their debt.

Implementation:

- Uses a mathematical formula to calculate loan payments.
- Uses a pie chart visualization to view the debt breakdown.



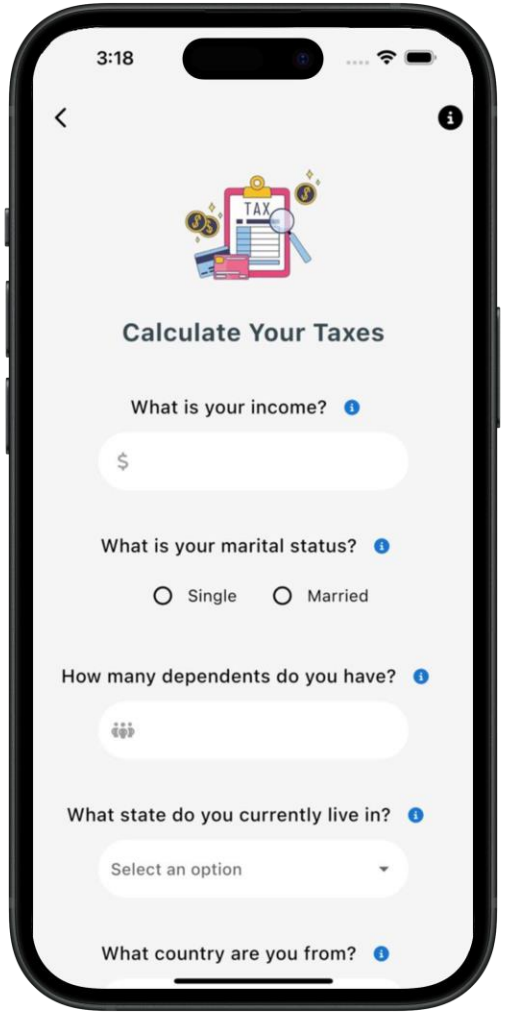
System Overview – Taxes Page

Overview:

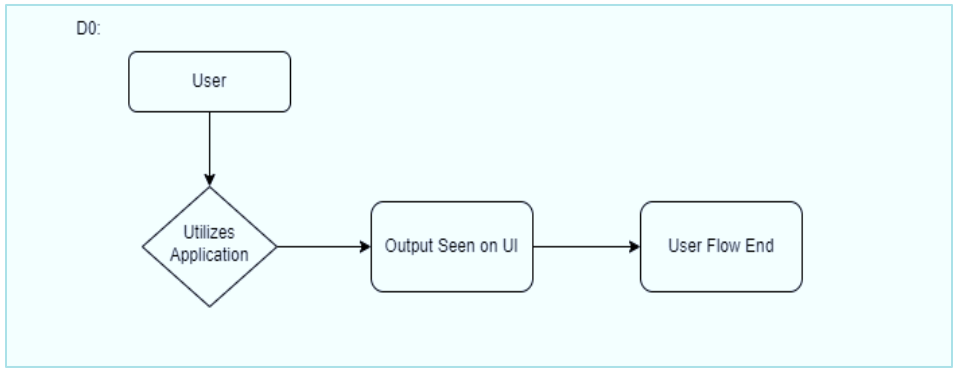
- Calculates tax estimates based on various factors, such as income, number of dependents, marital status, etc.
- Helps users understand potential tax liabilities.
- Allows users to input deductions and exemptions.

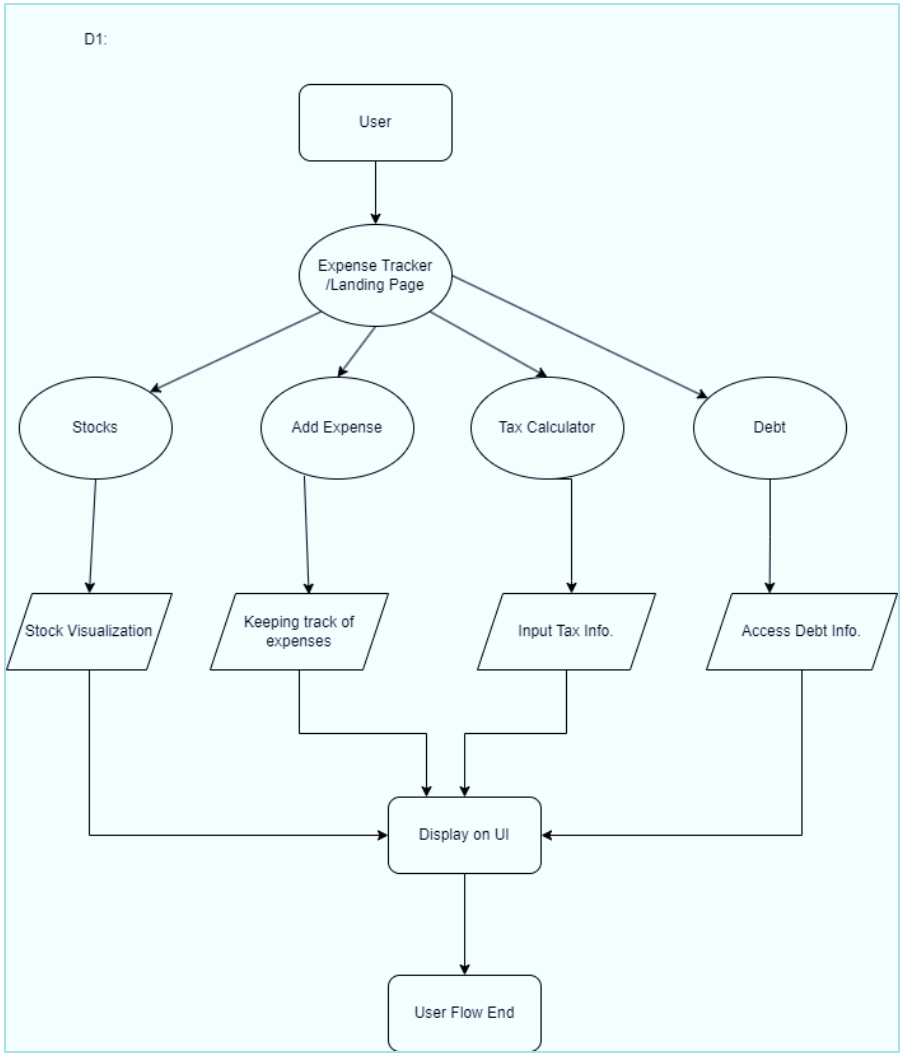
Implementation:

- Uses a simple tax calculation formula.
- Dynamically updates tax estimates as user input for the various factors change.

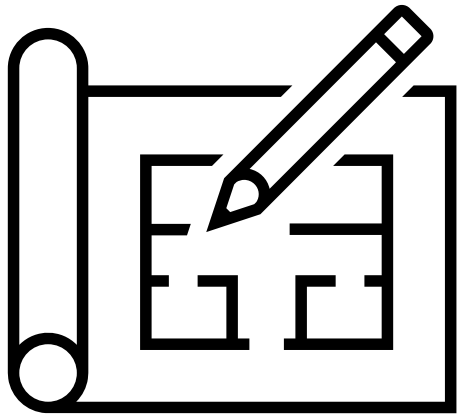


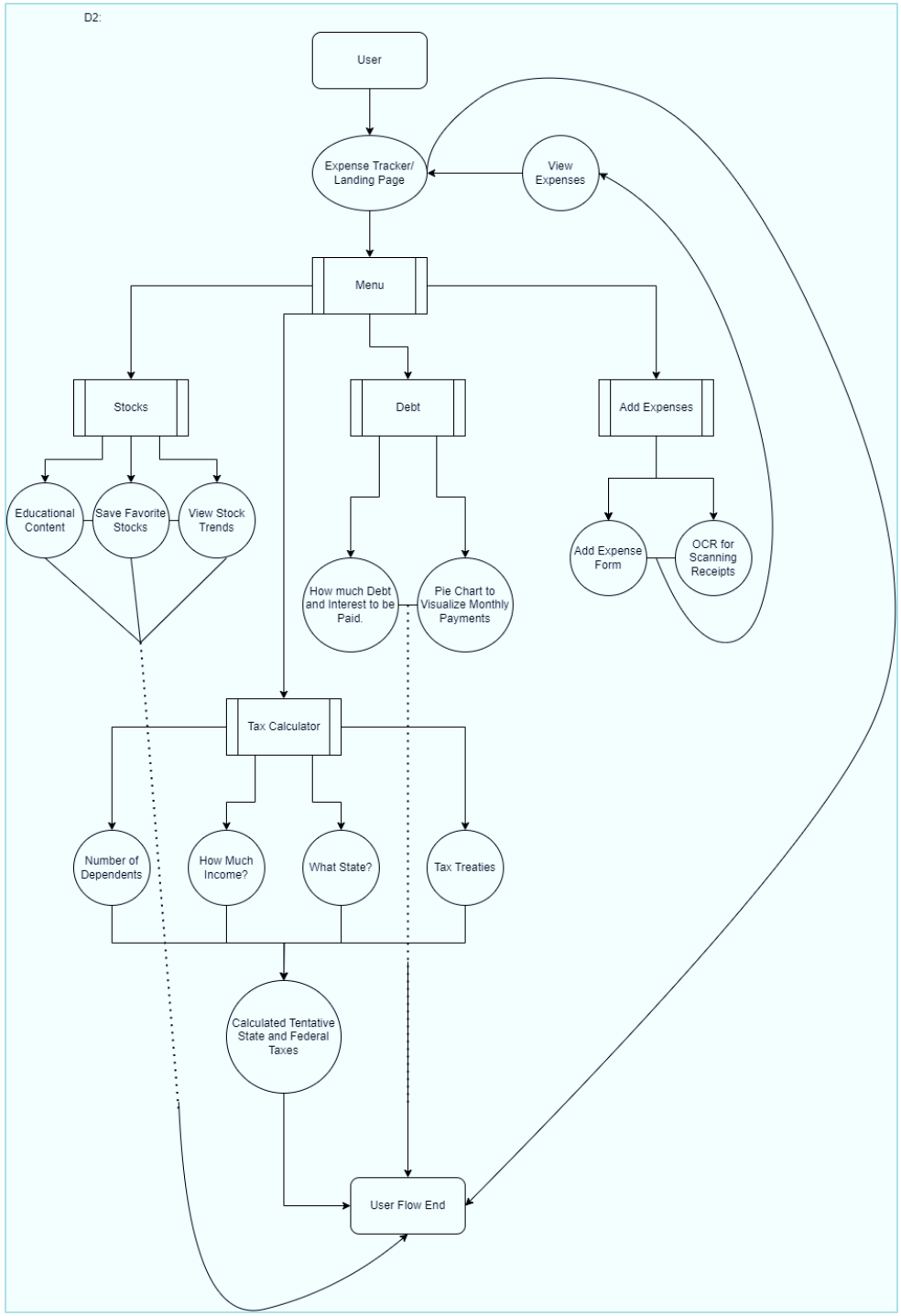
Design Diagrams





Design Diagram





Technologies

- **Backend: *Firebase*** used for data storage and retrieval.
- **Frontend: *Flutter*** (Dart) for User Interface & ***bloc*** used for state management.
- **APIs:**
  - AlphaVantage for Stocks - This API extracts requested stock information as json files. We have used several different endpoints to get different kinds of data like price fluctuations, time series data for graphs, company and revenue information, etc.
  - OCRSpace - This api extracts text data from the images providing the text as blob of text. We then use our function manipulation to read specific phrases like total amount to extract the amount next to it to input the expense directly through the process.

Major Project Constraints

Economic

We face financial limitations as our development relies on free and open-source tools to manage our budget effectively. This can hinder the development of advanced features. However, leveraging free resources could also promote accessibility for our target users.

Legal

The project must comply with various regulations concerning intellectual property and user data privacy. We use third-party APIs for stock recommendations and OCR which mean we need to make sure the usage of these APIs adheres to licensing agreements and avoids infringing on intellectual property.

Social

We aim to serve a significant public interest by providing users with the tools to improve their financial literacy and therefore, overall quality of life. In light of this, one of our constraints is impactful outreach to users who can benefit from our solution. We need to make sure that we are brainstorming effective outreach techniques.

Milestones

Task	Start	End
Refine the "Add Expenses" page and add a section to make notes about a transaction.	20 October 2024	23 October 2024
Research, design, and develop the logic and code for OCR under "Add Expenses" section.	21 October 2024	31 October 2024
Research stock trends and educational content for the "Stocks" page.	20 October 2024	23 October 2024
Design and develop the "Debt" page UI.	23 October 2024	31 October 2024
Develop the mathematical formula and visualizations for the "Debt" page.	01 November 2024	10 November 2024
Design and implement pages for the different pending parts of the app.	24 October 2024	01 November 2024
Design and develop the UI for the "Stocks" page.	01 November 2024	11 November 2024
Design the form for the "Tax Calculator" page.	02 November 2024	12 November 2024
Develop the database API to pull data about tax percentages based on user inputs on the "Taxes" page.	13 November 2024	17 November 2024
Research and develop list of debt-repayment educational content for the "Debt" page.	15 January 2025	19 January 2025
Research and implement in-app notifications.	15 January 2025	30 January 2025
Research chatbot integrations for the app.	01 February 2025	10 February 2025
Implement user authentication for logging in/out or creating user accounts.	20 January 2025	25 January 2025
Implement user account linking from backend and update frontend based on it.	25 January 2025	15 February 2025
Testing and Quality Assurance during development.	01 November 2024	28 February 2025

Results

- Implemented all the pages for all planned functionalities.
- Implemented user authentication using Firestore.
- Implemented OCR to input transactions.
- Added some unplanned features like filtering stocks, a settings page, educational content on Debt and Taxes pages, etc.
- Created a clean and seamless UI for a smooth user experience and potential customer retention.
- Seamless backend integration with Firebase to handle data storage and fetching.

Challenges

- OCR implementation required some extensive research on the models and API's available and the functions to access and implement it. After further research *OCRSpace* was the most apt and useable OCR implementation API which we could use for our receipt-scanning feature.
- We could not implement a feature to add an income in the app due to time constraints and the extensive changes required in the codebase to do so.
- During Alpha Testing, there was a delay in expense population and duplication of the record while adding new records in the expense list, which overwrote each newly updated records after each expense update. This was fixed by implementing real-time data retrieval in the Firebase data retrieval functions and enhancing the UI's state management.

Thank You!

We hope you liked our presentation. Please feel free to reach out in case you have any questions, comments, or feedback regarding our project!