

FinEXTracker

Personal Expenses Tracker

Venkata Saileenath Reddy Jampala
Founqnigue Souleymane Hassan Coulibaly

Roles and Responsibilities



Venkata Saileenath Reddy Jampala :

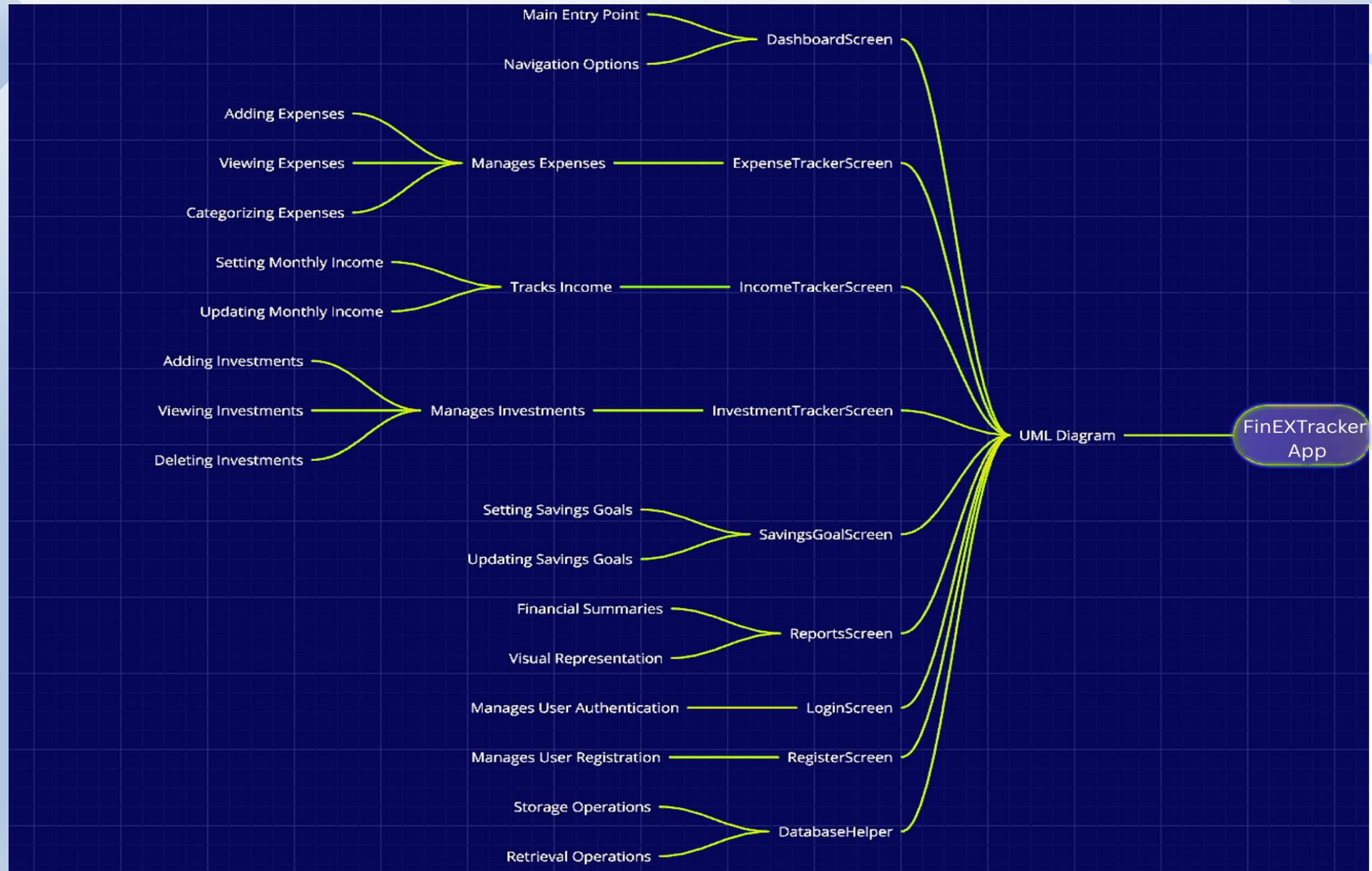
Backend Development, Database Management, Functionality Implementation, Project Management, Timelines and Documentation, Team Coordination.

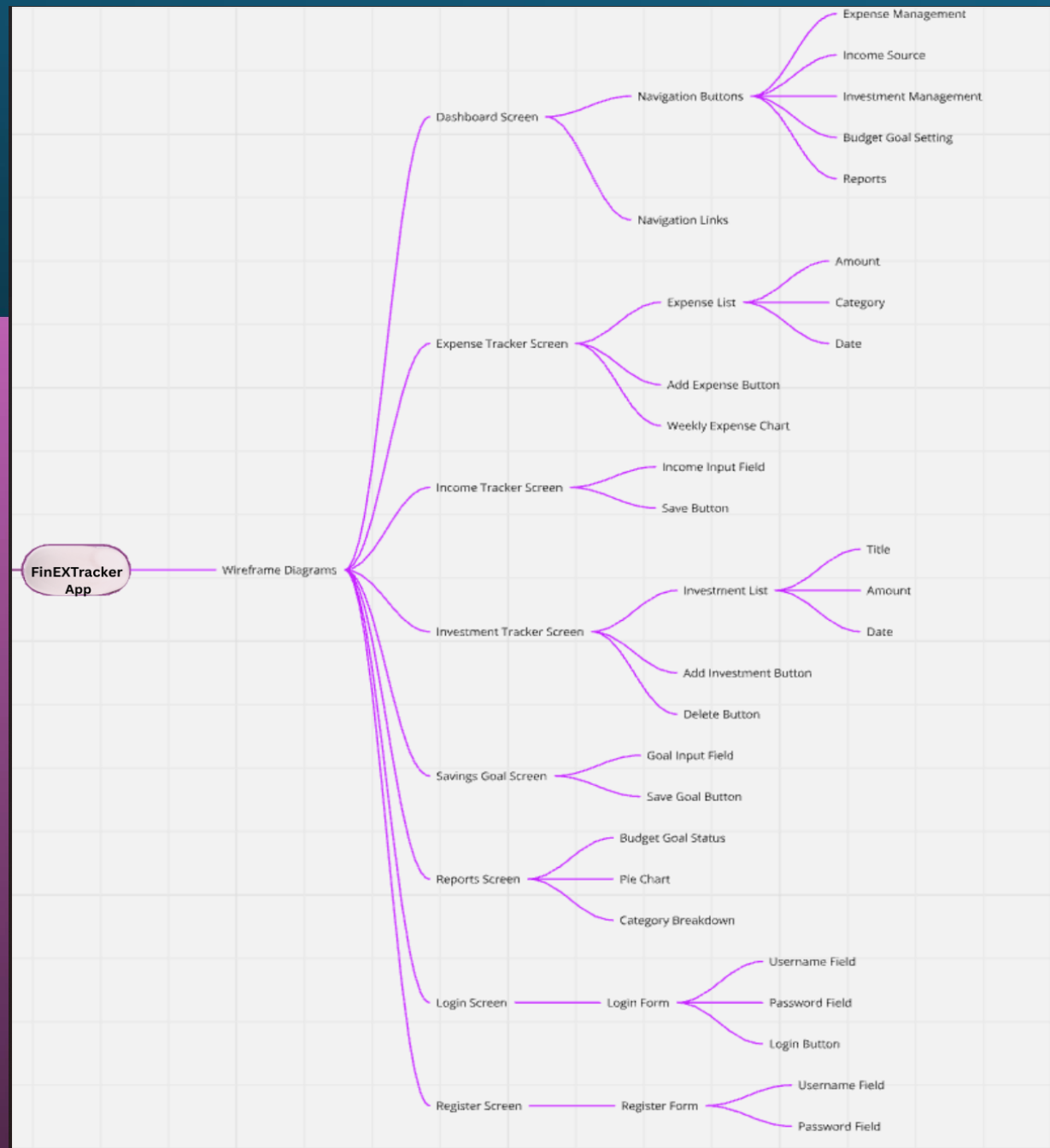
Foungnigue Souleymane Hassan Coulibaly :

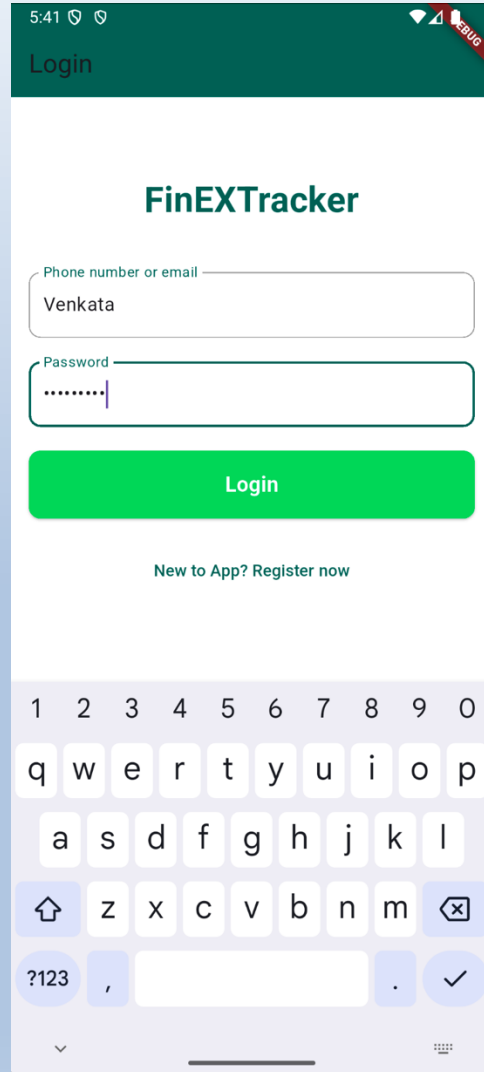
UI/UX Design, Frontend Development, Wireframes, Usability Tests, Implementing User Interface, Testing and Debugging, Analyzing Feedback, Iterative Design Adjustments.

Purpose

- The FinEXTracker app aims to help users manage and track their daily expenses efficiently. By providing a user-friendly interface and robust functionality, this app enables users to visualize their spending habits, categorize expenses, and make informed financial decisions.







The image shows a mobile application login screen. At the top, there is a dark green header with the word "Login" in white. Below the header, the app name "FinEXTracker" is displayed in a bold, dark green font. There are two input fields: the first is labeled "Phone number or email" and contains the text "Venkata"; the second is labeled "Password" and contains a series of dots. Below these fields is a large green button with the text "Login" in white. Underneath the button is a link that says "New to App? Register now" in a smaller, dark green font. At the bottom of the screen, a standard Android keyboard is visible, showing numbers, letters, and symbols.

Login Screen

Purpose

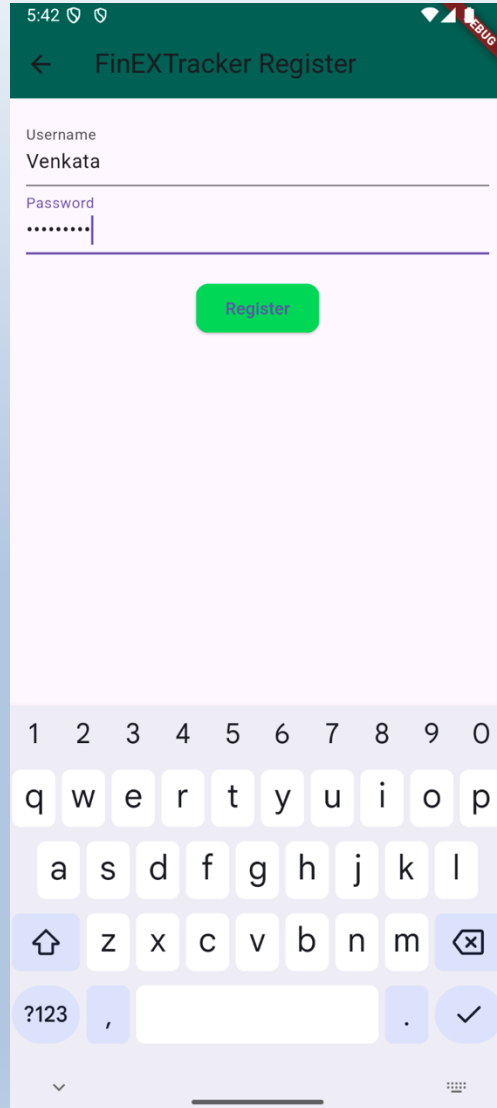
- Provide secure access to the app by authenticating users with a login.

Login Screen

- **Username Field**
- **Password Field**
- **Login Button** (for user authentication)
- **Register Button** (for new users to sign up)

Login Screen Logic:

- **onLoginButtonClick:** Checks if both fields are filled. If they are, it calls `validateUser` (where the logic to check stored credentials takes place) and either grants access or displays an error.
- **onRegisterButtonClick:** Navigates the user to the registration screen.



5:42 5G

← FinEXTracker Register

Username
Venkata

Password
.....

Register

1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
⬆ z x c v b n m ⬇
?123 , . ✓

Registration Screen

Purpose

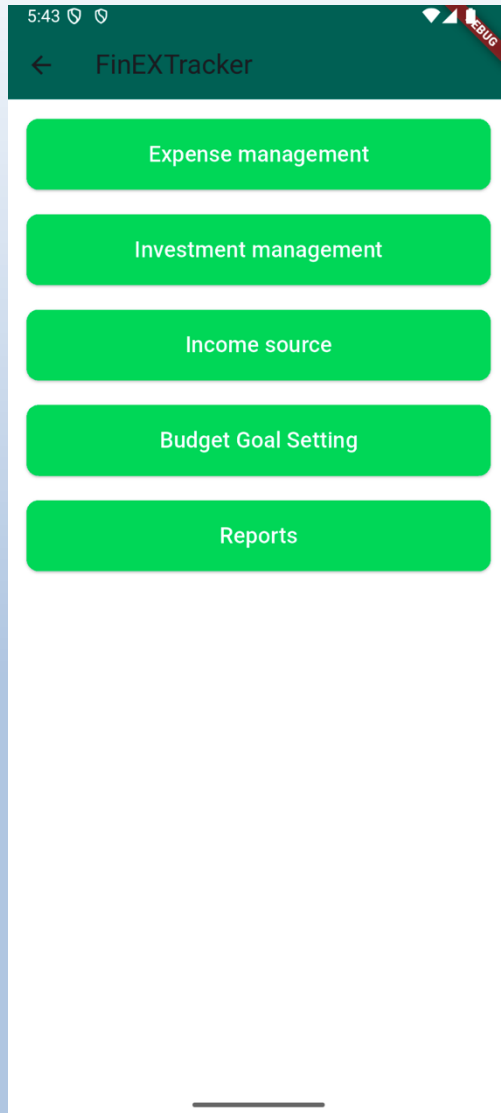
- Provide secure access to the app by providing facility to register their details.

Components

- **Username Field**
- **Email Field**
- **Password Field**
- **Confirm Password Field**
- **Register Button** (to submit new user details)

Registration Screen Logic:

- **onRegisterButtonClick:** Validates if all fields are filled, checks if the password and confirmation match, and verifies that the username and email are unique. It then securely stores the hashed password and redirects to the login screen if successful.



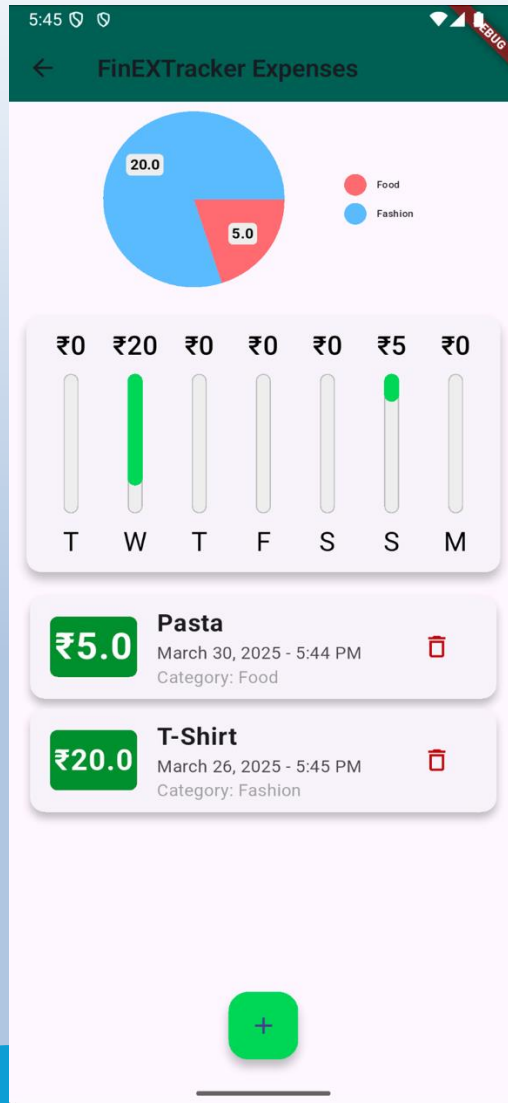
Dashboard Screen

Purpose

Provide a central hub for users to quickly access different sections within the app, including **Expense, Income, Investment, Savings Goals, and Reports.**

Components

- **Navigation Options:** Buttons or tabs to navigate to respective screens for Expense, Income, Investment, Savings Goals, and Reports.
- **Main Logic**
- **Initialize Screen Components:**
 - Load and display navigation options for each section (Expense, Income, etc.).
 - Optionally, show a quick overview of key financial metrics (e.g., total expenses, income summary) as a dashboard preview.
- **Navigation Logic:**
 - **onNavigationButtonClick:**
 - Detects which section button is clicked (e.g., Expense, Income).
 - Navigates to the respective screen based on user input.
- **Data Synchronization:**
 - Ensure all sections display up-to-date information by synchronizing data across screens.
 - When user returns to the Dashboard, update any preview or summary data for accurate reflection of user's financial status.



Expense Tracker Screen

Purpose

Allow users to efficiently track, categorize, and visualize their expenses for better financial management.

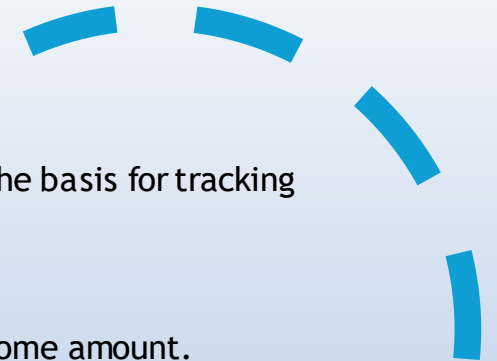
Components

- **Expense List:** Displays a list of expenses organized by category (e.g., Food, Transport, Utilities) and amount.
- **Add Expense Button:** Provides a quick way to add a new expense, prompting user input for details.
- **Weekly Expense Chart:** Graphical representation of weekly expenses, enabling users to see trends over time.

Main Logic

- **Initialize Screen Components:**
 - Load the **Expense List** by fetching stored expense data categorized by type and date.
 - Display the **Weekly Expense Chart** based on the expense data, visualizing the expenses for each day of the current week.
- **Add New Expense:**
 - **onAddExpenseButtonClick:**
 - Opens a form for the user to input details (amount, category, date, optional notes).
 - Validates that fields (like amount and category) are filled.
 - Saves the new expense entry to the database or local storage.
 - Refreshes the **Expense List** and **Weekly Expense Chart** to reflect the added expense.
- **Update Expense List and Chart:**
 - **onScreenRefresh:**
 - Updates the **Expense List** to display the latest categorized expenses, sorted by date.
 - Re-renders the **Weekly Expense Chart** to reflect any recent changes, showing a day-wise breakdown of total expenses.
- **Expense Filtering and Sorting :**
 - Add a feature to filter expenses by category, date, or amount.
 - Provide sorting options (e.g., sort by amount, recent, or category) to organize the **Expense List** based on user preference.
- **Data Synchronization:**
 - Ensure consistent data between this screen and any other sections of the app that display expense summaries or total spending.

Income Tracker Screen

A screenshot of a mobile application screen titled "FinEXTracker Income". The screen has a dark green header bar with a back arrow and the title. Below the header, it displays "Current Income: \$10000.00". There is a text input field labeled "Enter New Income" with a small vertical cursor. Below the input field is a large green button labeled "Save Income". The background of the screen is light pink. The status bar at the top shows the time 5:47 and various icons.

Purpose

Allow users to set, view, and update their monthly income, which serves as the basis for tracking expenses and financial planning.

Components

- **Income Input:** A text field where users can input or edit their monthly income amount.
- **Save Income Button:** A button to save or update the entered income value.

Main Logic

- **Initialize Screen Components:**
 - Load the current saved monthly income from the database or local storage (if available) and display it in the **Income Input** field.
 - Ensure that the **Income Input** field is editable to allow users to update their income.
- **Save or Update Income:**
 - **onSaveIncomeButtonClick:**
 - Validate the **Income Input** field to ensure that a valid numeric value is entered.
 - Save the income value to the database or local storage.
 - Display a confirmation message indicating successful save/update.
- **Data Synchronization:**
 - **onScreenRefresh:**
 - Synchronize the monthly income data with other screens (like the Dashboard or Expense Tracker) to reflect the updated income value for accurate financial calculations.

Expense Tracker Screen



Purpose

Enable users to log and track their investments, helping them monitor investment details and manage their portfolio.

Components

- **Investment List:** A list displaying each investment with relevant details (e.g., investment name, amount, date, and status).
- **Add Investment Button:** A button to add new investments by inputting details.

Main Logic

- **Initialize Screen Components:**
 - Load the **Investment List** from the database or local storage, displaying each investment with details such as name, amount, date of investment, and current status (e.g., active, completed).
 - Ensure that the list is dynamically updated to show the latest investment entries.
- **Add New Investment:**
 - **onAddInvestmentButtonClick:**
 - Opens a form for the user to input investment details (investment name, amount, date, optional notes).
 - Validates that required fields (like name and amount) are filled.
 - Saves the new investment entry to the database or local storage.
 - Refreshes the **Investment List** to reflect the newly added investment.
- **Update Investment Details:**
 - Allow users to select an existing investment and update details, such as changing the amount, status, or date.
 - After editing, save the updated information and refresh the **Investment List** to display the changes.
- **Data Synchronization:**
 - Ensure that data across screens remains consistent. For example, if the Dashboard screen provides an investment summary, synchronize it with this screen's data.

Savings Goal Screen



5:49

← FinEXTracker Savings Goal

Current Goal: \$0.00

Enter New Goal

4000

Save Goal

Purpose

Allow users to set, view, and update their savings goals, providing a target for financial planning and motivation.

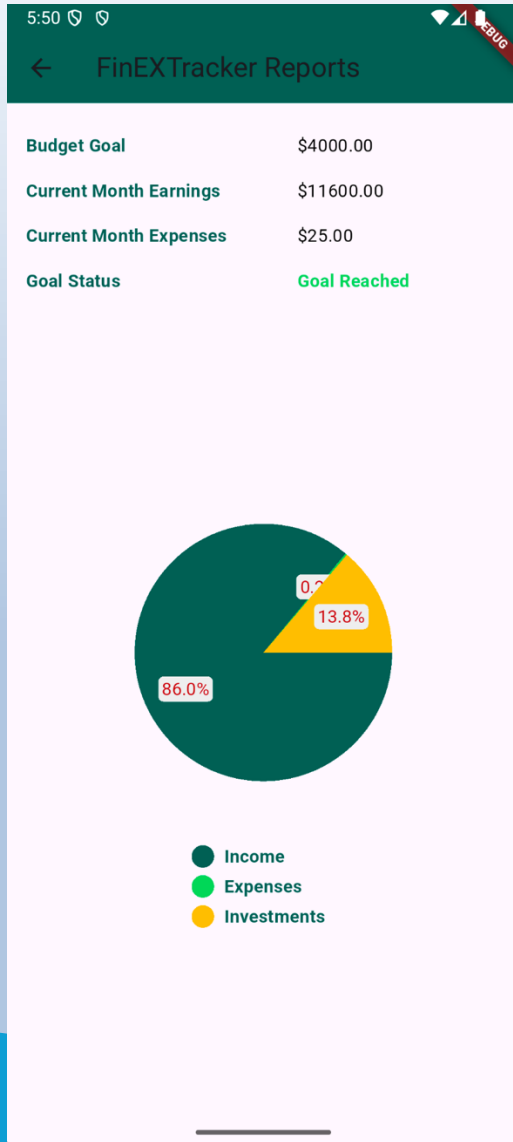
Components

- **Goal Input Field:** A text field for users to input their savings goal (target amount).
- **Save Goal Button:** A button to save or update the entered savings goal.

Main Logic

- **Initialize Screen Components:**
 - Load the current saved savings goal from the database or local storage (if available) and display it in the **Goal Input Field**.
 - Ensure the **Goal Input Field** is editable so users can update their goal.
- **Set or Update Savings Goal:**
 - **onSaveGoalButtonClick:**
 - Validate the **Goal Input Field** to ensure a valid numeric target is entered.
 - Save the goal value to the database or local storage.
 - Display a confirmation message indicating successful save or update.
- **Progress Monitoring :**
 - Track progress toward the savings goal by calculating the percentage of the target achieved based on the user's current balance or savings.
 - Display a visual progress bar or chart for motivation.
- **Data Synchronization:**
 - **onScreenRefresh:**
 - Synchronize the savings goal with other screens (like the Dashboard or Reports) to ensure accurate goal-related calculations.

Reports Screen



Purpose

Provide users with a visual and summary-based overview of their financial status, including budget goals, income, expenses, and investments.

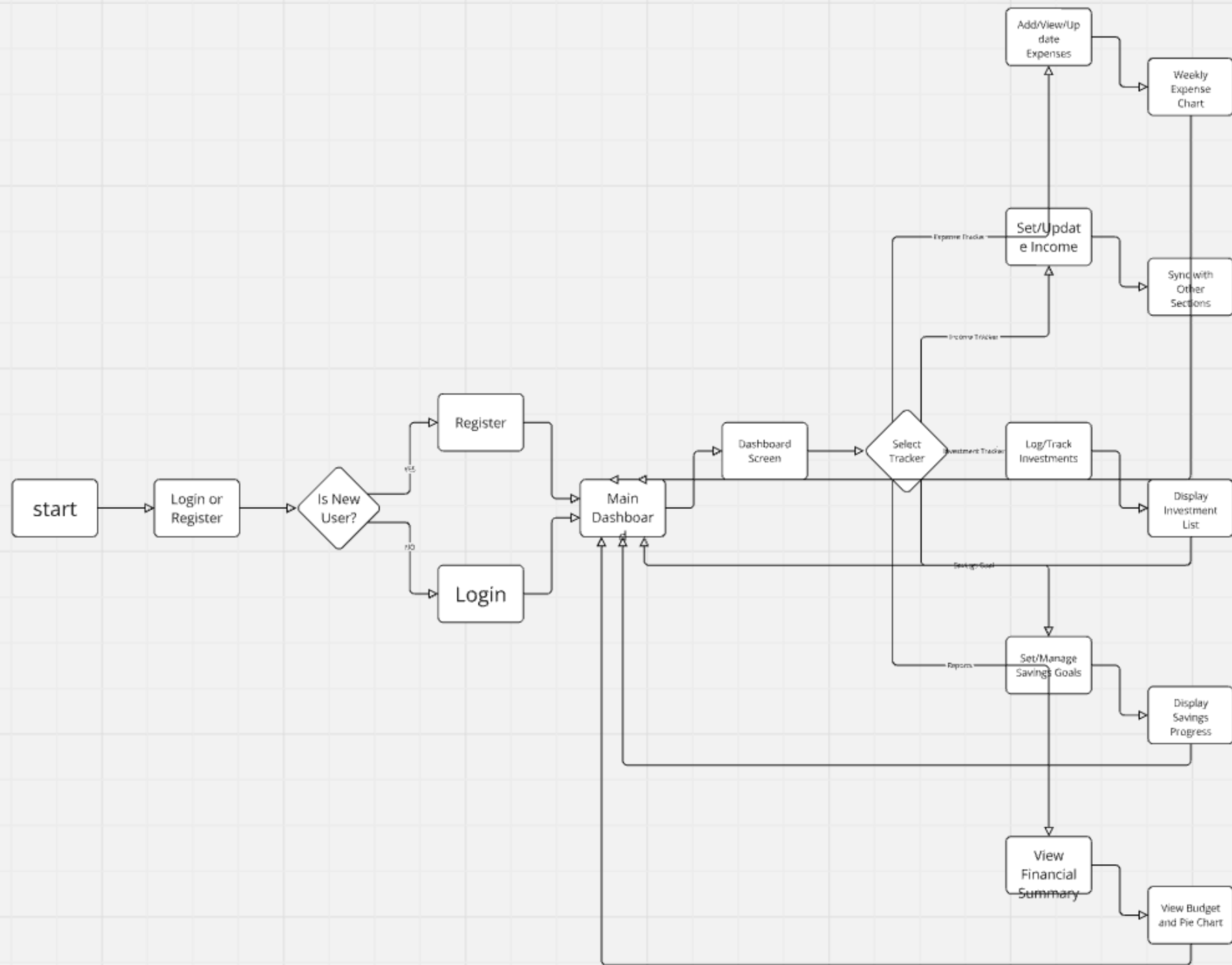
Components

- **Budget Goal Status:** Displays the progress toward any set savings or budget goals, allowing users to see how close they are to reaching their targets.
- **Pie Chart:** Visualizes the breakdown of financial categories, showing the proportion of income, expenses, and investments in an easily digestible format.

Main Logic

- **Initialize Screen Components:**
 - Load data for **Budget Goal Status** and display the percentage or amount achieved toward the current goal.
 - Gather financial data (income, expenses, investments) for the **Pie Chart** and calculate proportions for each category.
- **Display Budget Goal Status:**
 - Calculate the progress of the savings or budget goal based on the user's current balance or saved amount.
 - Display a visual indicator, such as a progress bar or percentage, to show how close the user is to achieving their goal.
- **Generate Financial Breakdown in Pie Chart:**
 - **onScreenLoad:**
 - Calculate the total income, expenses, and investments.
 - Compute the percentages for each category based on the total.
 - Render the **Pie Chart** with sections for each category, using different colors to represent income, expenses, and investments.
- **Data Synchronization:**
 - **onScreenRefresh:**
 - Ensure the data displayed in the Reports Screen is up-to-date and accurately reflects recent transactions or updates from other screens (Expense, Income, Investment).
 - Synchronize with the **Dashboard** screen if it provides a summary of the same data.

Functionality



User Authentication

- **Login and Registration Screens** provide secure access, allowing new users to register and returning users to log in with their credentials.
- Smooth transitions between the login, registration, and main dashboard screens enhance the onboarding experience.

Dashboard Screen

- Serves as the central hub, displaying navigation options to access different sections: Expense Tracker, Income Tracker, Investment Tracker, Savings Goal, and Reports.
- Displays a summary of key financial information, enabling users to access specific areas quickly and efficiently.

Expense Tracker

- Allows users to add and categorize expenses, viewing them in an organized list format.
- Includes a **Weekly Expense Chart** for a visual representation of spending patterns, helping users manage their spending habits.
- Smooth transitions between adding, viewing, and updating expenses provide an intuitive experience.

Income Tracker

- Provides a simple interface for setting and updating monthly income.
- Users can enter or update their income amount, which synchronizes with other sections for accurate financial analysis.

Investment Tracker

- Allows users to log and track their investments, displaying a list of all investments with relevant details (amount, date, and status).
- Users can add new investments, making it easier to monitor and update their portfolio as they progress toward their financial goals.

Savings Goal

- Users can set and manage their savings goals by entering target amounts and tracking their progress.
- Displays progress toward the savings goal, with smooth screen updates reflecting any changes.

Reports Screen

- Provides a financial summary through a **Budget Goal Status** indicator and a **Pie Chart** showing the breakdown of income, expenses, and investments.
- Allows users to visualize their finances, empowering them to make informed decisions about their financial habits and progress toward their goals.

Data Synchronization and Consistency

- All screens synchronize data so that changes in one section reflect across the app.
- The Dashboard and Reports screens display up-to-date information, allowing users to see an accurate picture of their financial status at any time.

User Experience and Smooth Transitions

- The app features smooth transitions between screens, creating a seamless experience as users move between sections.
- Transition animations and page navigation are optimized to improve responsiveness and make the app feel intuitive and easy to navigate.

Backend

- 1. User Authentication (Login and Registration)
- **Registration:**
 - Check if the username or email already exists.
 - Hash the password for security.
 - Insert user details (username, email, hashed password) into the users table.
- **Login:**
 - Verify that the username exists.
 - Compare the entered password with the stored hashed password.
 - If valid, generate a session token (or JWT) for the user.
 - Pseudocode

```
FUNCTION registerUser(username, email, password)
```

```
  IF username OR email EXISTS in users THEN
```

```
    RETURN "User already exists"
```

```
  END IF
```

```
  hashedPassword = hash(password)
```

```
  INSERT INTO users (username, email, password) VALUES (username, email, hashedPassword)
```

```
  RETURN "Registration successful"
```

```
END FUNCTION
```

```
FUNCTION loginUser(username, password)
```

```
  user = SELECT * FROM users WHERE username = username
```

```
  IF user IS NULL THEN
```

```
    RETURN "User not found"
```

```
  END IF
```

```
  IF password MATCHES user.password THEN
```

```
    sessionToken = generateToken(user.id)
```

```
    RETURN sessionToken
```

```
  ELSE
```

```
    RETURN "Invalid credentials"
```

```
  END IF
```

```
END FUNCTION
```


Backend

2. Dashboard

- The Dashboard screen aggregates data from other sections, so no specific table is needed. It will display a summary of information from the expenses, income, investments, and savings_goals tables.

Backend Logic

- **Fetch Summary Data:** Gather total income, total expenses, investment value, and savings goals.
- **Calculate Progress:** Calculate percentages of income spent, goals achieved, etc.
- Pseudocode
- ```
FUNCTION getDashboardSummary(userId)
 totalIncome = SELECT SUM(amount) FROM income WHERE user_id = userId
 totalExpenses = SELECT SUM(amount) FROM expenses WHERE user_id = userId
 totalInvestments = SELECT SUM(amount) FROM investments WHERE user_id = userId
 totalSavingsGoal = SELECT goal_amount FROM savings_goals WHERE user_id = userId

 RETURN {
 income: totalIncome,
 expenses: totalExpenses,
 investments: totalInvestments,
 savingsGoal: totalSavingsGoal
 }
END FUNCTION
```

# Backend

## 3. Expense Tracker

### Backend Logic

- **Add Expense:** Insert a new expense record.
- **Fetch Expenses:** Retrieve all expenses for a user.
- **Update Expense:** Modify an existing expense entry.
- **Delete Expense:** Remove an expense entry.

### Pseudocode

```
FUNCTION addExpense(userId, category, amount, date)
```

```
 INSERT INTO expenses (user_id, category, amount, date) VALUES (userId, category, amount, date)
```

```
 RETURN "Expense added successfully"
```

```
END FUNCTION
```

```
FUNCTION getExpenses(userId)
```

```
 expenses = SELECT * FROM expenses WHERE user_id = userId
```

```
 RETURN expenses
```

```
END FUNCTION
```

```
FUNCTION updateExpense(expenseId, category, amount, date)
```

```
 UPDATE expenses SET category = category, amount = amount, date = date WHERE id = expenseId
```

```
 RETURN "Expense updated successfully"
```

```
END FUNCTION
```

```
FUNCTION deleteExpense(expenseId)
```

```
 DELETE FROM expenses WHERE id = expenseId
```

```
 RETURN "Expense deleted successfully"
```

```
END FUNCTION
```

# Backend

## 4. Income Tracker

### Backend Logic

- **Set Monthly Income:** Insert or update monthly income for a user.
- **Fetch Income:** Retrieve income data by user and month.

### Pseudocode

```
FUNCTION setIncome(userId, amount, month)
 REPLACE INTO income (user_id, amount, month) VALUES (userId, amount, month)
 RETURN "Income saved successfully"
END FUNCTION
```

```
FUNCTION getIncome(userId)
 income = SELECT * FROM income WHERE user_id = userId
 RETURN income
END FUNCTION
```

# Backend

## 5. Income Tracker

### Backend Logic

- **Add Investment:** Insert a new investment.
- **Fetch Investments:** Retrieve all investments for a user.
- **Update Investment:** Modify an existing investment record.
- **Delete Investment:** Remove an investment record.

### Pseudocode

```
FUNCTION addInvestment(userId, investment_name, amount, date, status)
 INSERT INTO investments (user_id, investment_name, amount, date, status) VALUES (userId, investment_name, amount, date, status)
 RETURN "Investment added successfully"
END FUNCTION
```

```
FUNCTION getInvestments(userId)
 investments = SELECT * FROM investments WHERE user_id = userId
 RETURN investments
END FUNCTION
```

```
FUNCTION updateInvestment(investmentId, investment_name, amount, date, status)
 UPDATE investments SET investment_name = investment_name, amount = amount, date = date,
 status = status WHERE id = investmentId
 RETURN "Investment updated successfully"
END FUNCTION
```

```
FUNCTION deleteInvestment(investmentId)
 DELETE FROM investment
 RETURN "Investment deleted successfully"
END FUNCTION
```

# Backend

## 5. Savings Goal Tracker

### Backend Logic

- **Set Savings Goal:** Insert or update a savings goal.
- **Fetch Savings Goal:** Retrieve a user's savings goals.
- **Update Progress:** Update current savings toward the goal.
- **Delete Savings Goal:** Remove a savings goal.

### Pseudocode

```
FUNCTION setSavingsGoal(userId, goal_name, goal_amount, current_savings, target_date)
```

```
 INSERT INTO savings_goals (user_id, goal_name, goal_amount, current_savings, target_date) VALUES (userId, goal_name, goal_am
current_savings, target_date)
```

```
 RETURN "Savings goal set successfully"
```

```
END FUNCTION
```

```
FUNCTION getSavingsGoals(userId)
```

```
 savingsGoals = SELECT * FROM savings_goals WHERE user_id = userId
```

```
 RETURN savingsGoals
```

```
END FUNCTION
```

```
FUNCTION updateSavingsProgress(goalId, current_savings)
```

```
 UPDATE savings_goals SET current_savings = current_savings WHERE id = goalId
```

```
 RETURN "Savings goal progress updated"
```

```
END FUNCTION
```

```
FUNCTION deleteSavingsGoal(goalId)
```

```
 DELETE FROM savings_goals WHERE id = goalId
```

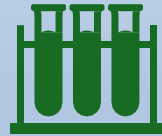
```
 RETURN "Savings goal deleted successfully"
```

```
END FUNCTION
```

# Iterative Design and Testing Approach



To ensure the app meets user needs effectively, we will employ an iterative design process.



Throughout the design process, the wireframe will be continuously refined based on user feedback. We will conduct usability tests with low-fidelity prototypes to gather insights, focusing on:



Navigation: Ensuring users can intuitively move between screens.



Element Placement: Assessing the effectiveness of UI components in facilitating user tasks.



Visual Clarity: Making sure that each screen communicates its purpose clearly.

# Potential Benefits to Users -

**Improved Financial Awareness:** Users gain insights into their spending patterns, enabling better financial planning.

**User-Friendly Interface:** Simple navigation and input methods make it accessible to all users, regardless of tech-savviness.

**Personalized Experience:** Users can customize settings to fit their financial habits, enhancing engagement and satisfaction.

# Conclusion



The Personal Expenses Tracker app presents a valuable tool for users seeking to manage their finances effectively. Through a collaborative effort, we aim to deliver a high-quality application that not only meets user needs but also provides an enjoyable and intuitive experience.