***BUDGET BUDDY***

***Introduction :***

An Expense Manager Application is a digital tool designed to help individuals or businesses track, manage, and analyze their expenses efficiently. It provides a convenient platform to record, categorize, and monitor expenditures, thereby aiding in budgeting, financial planning, and expense control.

***Scope :***

The scope of an Expense Manager Application encompasses various features such as expense tracking, categorization, budget setting, reporting, and analysis. It should offer user-friendly interfaces across multiple devices, ensuring accessibility and convenience for users.

***Definition :***

An Expense Manager Application is a software solution that enables users to record, categorize, analyze, and manage their expenses effectively. It typically includes features like expense tracking, budgeting tools, customizable categories, reporting, and sometimes integration with banking or financial platforms.

***User stories :***

***1. Secure Onboarding and Access***

● As a new user, I want to sign up with my mobile number and easily verify my identity through OTP-based authentication to securely access the app.

● As an existing user, I want to log in quickly using my mobile number and OTP, without having to remember a complex password.

● As a user, I want to be able to enable two-factor authentication for an extra layer of security.

***2. Comprehensive Transaction Management***

● As a user, I want to easily add, update, and delete my income and expense transactions, categorized by date, amount, and description.

● As a user, I want to attach receipts or notes to my transactions for richer context.

● As a user, I want to receive daily reminders to log my transactions, preventing gaps in my financial records.

***3. Customizable Categorization***

● As a user, I want to create custom income and expense categories beyond the defaults, aligning with my specific needs and spending habits.

● As a user, I want to personalize categories with icons or colors for better organization and recognition.

***4. Goal-Setting and Tracking***

● As a user, I want to set financial goals (e.g., saving for a vacation, paying off debt) and track my progress over time, visualized with clear and actionable insights.

● As a user, I want to be able to challenge myself with milestones or deadlines for my goals, staying focused and accountable.

***5. Actionable Insights and Analytics***

● As a financially curious user, I want to see insightful graphs and charts that visually represent my income, expenses, and spending patterns by category, time period, and other relevant factors.

● As a user, I want to filter and drill down into my data to identify trends, areas for improvement, and make informed financial decisions.

***6. Powerful Search and Export***

● As a user, I want to quickly find specific transactions using a powerful search function based on keywords, dates, categories, or amounts.

● As a user, I want to export my transaction data to CSV or Excel files for further analysis, sharing, or backup purposes.

***7. Streamlined Debt and Credit Management***

● As a responsible user, I want to create and manage a list of debtors and creditors, tracking amounts owed or owed to me, with due dates and reminders.

● As a collaborative user, I want to be able to mark debts or credits as settled with ease, facilitating clear communication and record-keeping.

***8. Personalization and Accessibility***

● As a user, I want to personalize the app's interface by changing the accent color theme to suit my preferences.

● As a user with accessibility needs, I want to navigate the app easily with clear language, design, and support for technologies.

***9. Secure Sharing and Communication***

● As a social user, I want to share details of specific transactions with friends and family on social media platforms, while maintaining security and privacy.

● As a collaborative user, I want to split bills or track joint expenses with others seamlessly and securely within the app.

***10. Logout and Account Management***

● As a user, I want to securely log out of the app when I'm done, ensuring my data remains protected.

● As a responsible user, I want to manage my account settings, including updating my information, changing my password, or deactivating my account when needed

***Use cases :***

* ***Use Case: Manage Transactions***

**Actor:** Users

**Precondition:** The user has successfully logged in to the personal finance management system.

**Goal:** The user wants to add, edit, or delete a financial transaction.

**Basic Flow:**

1. The user selects the "Transactions" tab or menu option.
2. The system displays a list of existing transactions.
3. The user can:
   * Add a new transaction by entering the date, amount, payee/payer, category, and optional memo.
   * Select an existing transaction to view its details.
   * Edit the details of a selected transaction.
   * Delete a selected transaction.

**Alternative Flows:**

1. If the user tries to add a transaction with missing information, the system prompts the user to enter the required details.
2. If the user tries to delete a transaction, the system prompts the user for confirmation to prevent accidental deletion.

**Postcondition:**

* The system successfully adds, edits, or deletes the transaction.
* The updated list of transactions is displayed.
* ***Use Case: Manage Categories***

**Actor:** User

**Goal:** The user wants to create, edit, or delete spending or income categories.

**Basic Flow:**

1. The user selects the "Categories" tab or menu option.
2. The system displays a list of existing categories.
3. The user can:
   * Add a new category by entering a name and optionally setting it as an expense or income category.
   * Select an existing category to view its details.
   * Edit the name or type (expense/income) of a selected category.
   * Delete a selected category.

**Alternative Flows:**

1. If the user tries to add a category with a duplicate name, the system prompts the user to enter a unique name.
2. If the user tries to delete a category that is assigned to existing transactions, the system prompts the user to select a different category for those transactions before deleting the selected category.

**Postcondition:**

* The system successfully creates, edits, or deletes the category.
* The updated list of categories is displayed.
* ***Use Case : Manage Budgets***

**Actor:**  User

**Precondition:** The user has successfully logged in to the system and has created some spending categories.

**Goal:** The user wants to create, edit, or delete budgets for different spending categories.

**Basic Flow:**

1. The user selects the "Budgets" tab or menu option.
2. The system displays a list of existing budgets or a prompt to create one if none exist.
3. The user can:
   * Create a new budget by selecting a spending category, entering a budgeted amount for a specific time period (e.g., month, year), and optionally setting aside funds for recurring expenses.
   * Select an existing budget to view details like the allocated amount, remaining balance, and recently assigned transactions.
   * Edit the details of a selected budget, such as the budgeted amount or time period.
   * Delete a selected budget.

**Alternative Flows:**

1. If the user tries to create a budget with an insufficient remaining balance for recurring expenses, the system prompts the user to adjust the budget amount or recurring expense allocation.
2. If the user tries to delete a budget that has transactions assigned to it, the system prompts the user to reassign those transactions to a different budget category before deletion.

**Postcondition:**

* The system successfully creates, edits, or deletes the budget.
* The updated list of budgets is displayed.
* ***Use Case: Create Profile***

**Actor:** New User

**Goal:** The user wants to create a new account and profile to start using the system.

**Basic Flow:**

1. The user selects the "Sign Up" or "Create Account" option.
2. The system prompts the user to enter required information such as name, email address, and a secure password.
3. The user may optionally provide additional information such as phone number or preferred currency.
4. The system prompts the user to agree to terms and conditions (if applicable).
5. The user submits the signup form.

**Alternative Flows:**

1. If the user enters an existing email address or a password that doesn't meet security requirements, the system prompts the user to enter a unique email or a stronger password.
2. If the user forgets to enter required information, the system prompts the user to complete the missing fields.

**Postcondition:**

* The system successfully creates a new user account and profile.
* The user is logged in to the system and can start using its features.
* ***Use Case: Edit Profile***

**Actor:**  User

**Goal:** The user wants to modify their existing account information.

**Basic Flow:**

1. The user selects the "Profile" or "Settings" tab or menu option.
2. The system displays the user's current profile information.
3. The user can edit fields such as name, email address, phone number, or preferred currency.
4. The user may also be able to change their password.
5. The user submits the changes.

**Alternative Flows:**

1. If the user tries to change their email address to an existing one, the system prompts them to enter a unique email address.
2. If the user attempts to update their password with one that doesn't meet security requirements, the system prompts them to create a stronger password.

**Postcondition:**

* The system successfully updates the user's profile information.
* The updated profile details are displayed.
* ***Use Case: Change Password***

**Actor:** User

**Goal:** The user wants to update their login password for security purposes.

**Basic Flow:**

1. The user selects the "Profile," "Settings," or "Security" tab or menu option.
2. The system may require the user to enter their current password for verification.
3. The user then enters their new desired password, followed by re-entering it for confirmation.
4. The user submits the password change request.p

**Alternative Flows:**

1. If the user enters an incorrect current password, the system prompts them to re-enter the correct password.
2. If the user's new password doesn't meet minimum security requirements, the system prompts them to create a stronger password.

**Postcondition:**

* The system successfully updates the user's password.
* The system prompts the user to log in again using the new password.
* ***Use Case : Logout***

**Actor:**  User

**Goal:** The user wants to end their current session and securely log out of the system.

**Basic Flow:**

1. The user selects the "Logout" or "Sign Out" option, typically found in the account menu or profile settings.
2. The system confirms the logout request and terminates the user's session.

**Postcondition:**

* The user is successfully logged out of the system.
* The system returns to the login screen.

***Functional requirements :***

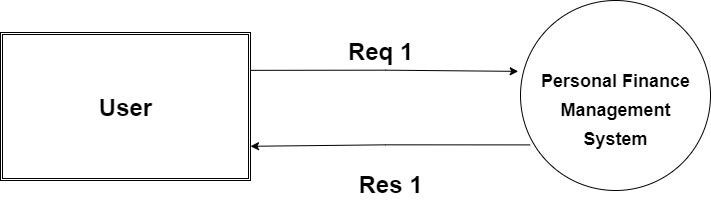
* User’s Registration and Login By Mobile Number (OTP based authentication). User can logout.
* System should allow user to create his/her profile if he is not registered earlier. If he is registered already he should be redirected to homescreen. User must be allowed edit his profile (change name, picture).
* User can easily add transaction (income or expense)
* User should be allowed to add custom income/expense category.
* User can add add/update/delete/duplicate transaction( with bill image).
* User can see transactions of any month of any year. Can see total income, expense, balance of any month.
* User can see graphs by category (Income/Expense).
* User can search any transaction by amount, note, name.
* User can add/update/delete debtors/creditors.
* User can export data of debtors/creditors.(Excel file).
* User can export data of transactions of any specific period of time (Excel file).
* User will manage goals. (create, add amount, see progress, transfer to reached module, edit, delete).
* User can change app theme.
* User can create group, add members, add group expenses. Share group expenses details as image to anyone to any social media platform.
* Daily reminder for storing your transactions. (can enable/disable)(local notifications).

***Non-functional requirements :***

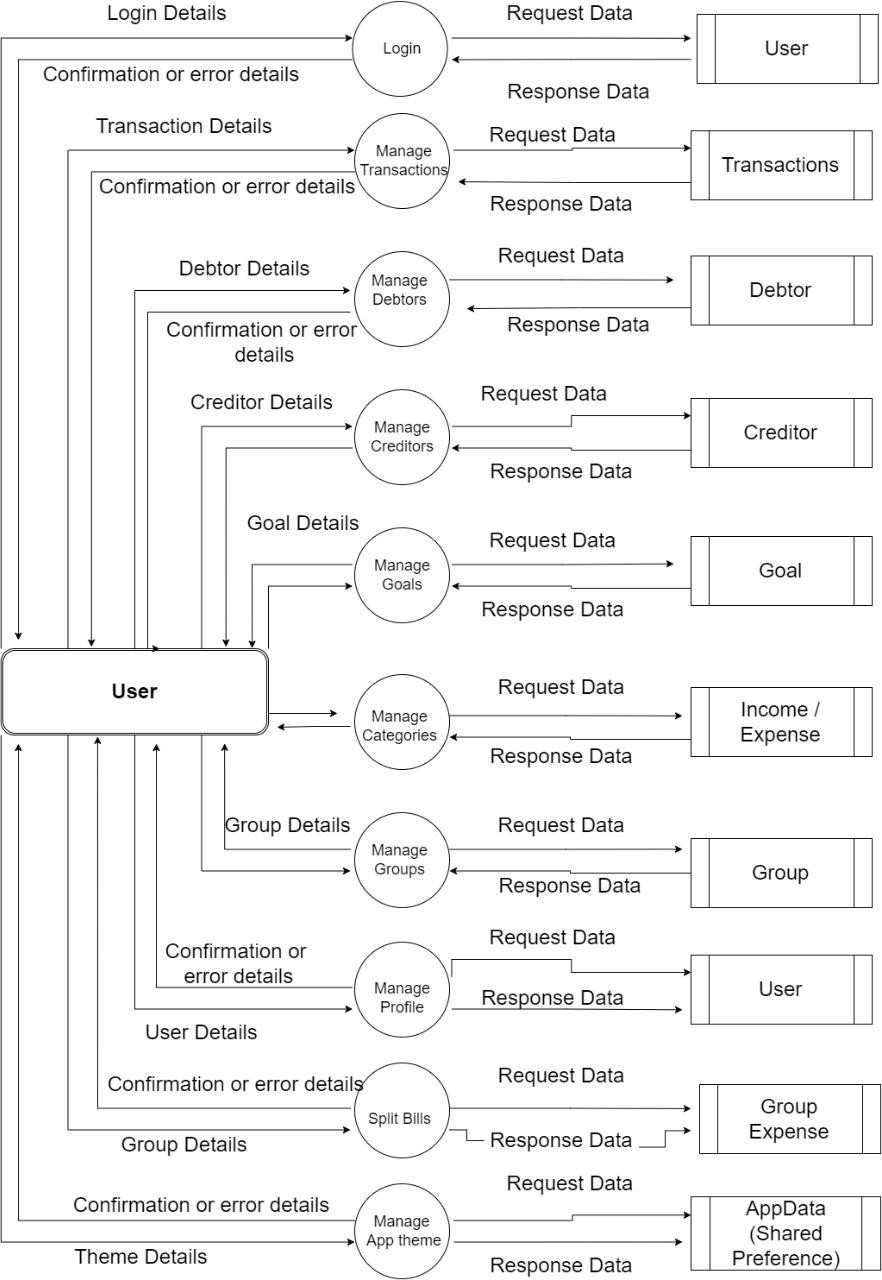
* **Security** : User has to login with mobile number and valid OTP to access data and in backend JWT( JSON Web Token) based authentication should be used so to access any data user has to provide valid token.
* **Operating System / Device** : Software should run on all operating systems like windows, Android , IOS (Because of Flutter )
* **Maintainability** : Application should build up in such a way that classifications of errors and maintenance of mechanism become easy.
* **Flexibility** : Application should be flexible so that it can easily accept all changes at low cost, time and experience.
* **Reliability** : The System will support 24 X 7 operations. User should be able to access application at any time.
* **Performance and response** time : Application should be developed using best state management techniques so performance and response timeincreases.
* **Error handling** : Application should handle all the errors effectively. Application should display valid messages if any error occurs.
* **Ease of use** : Application should be easy to use, User interface and user experience should be best.

**DFD :**

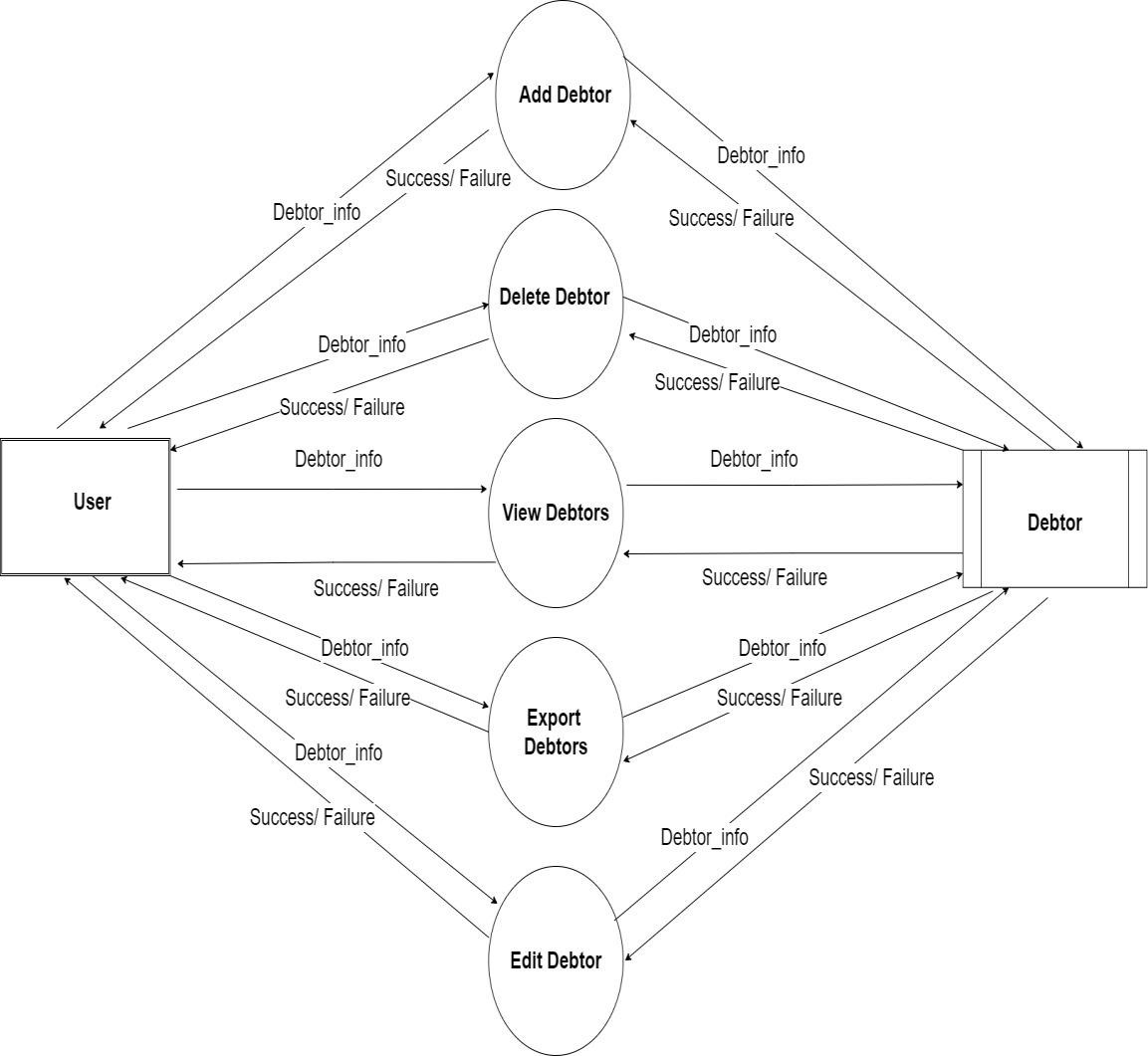
**LEVEL 0 :**



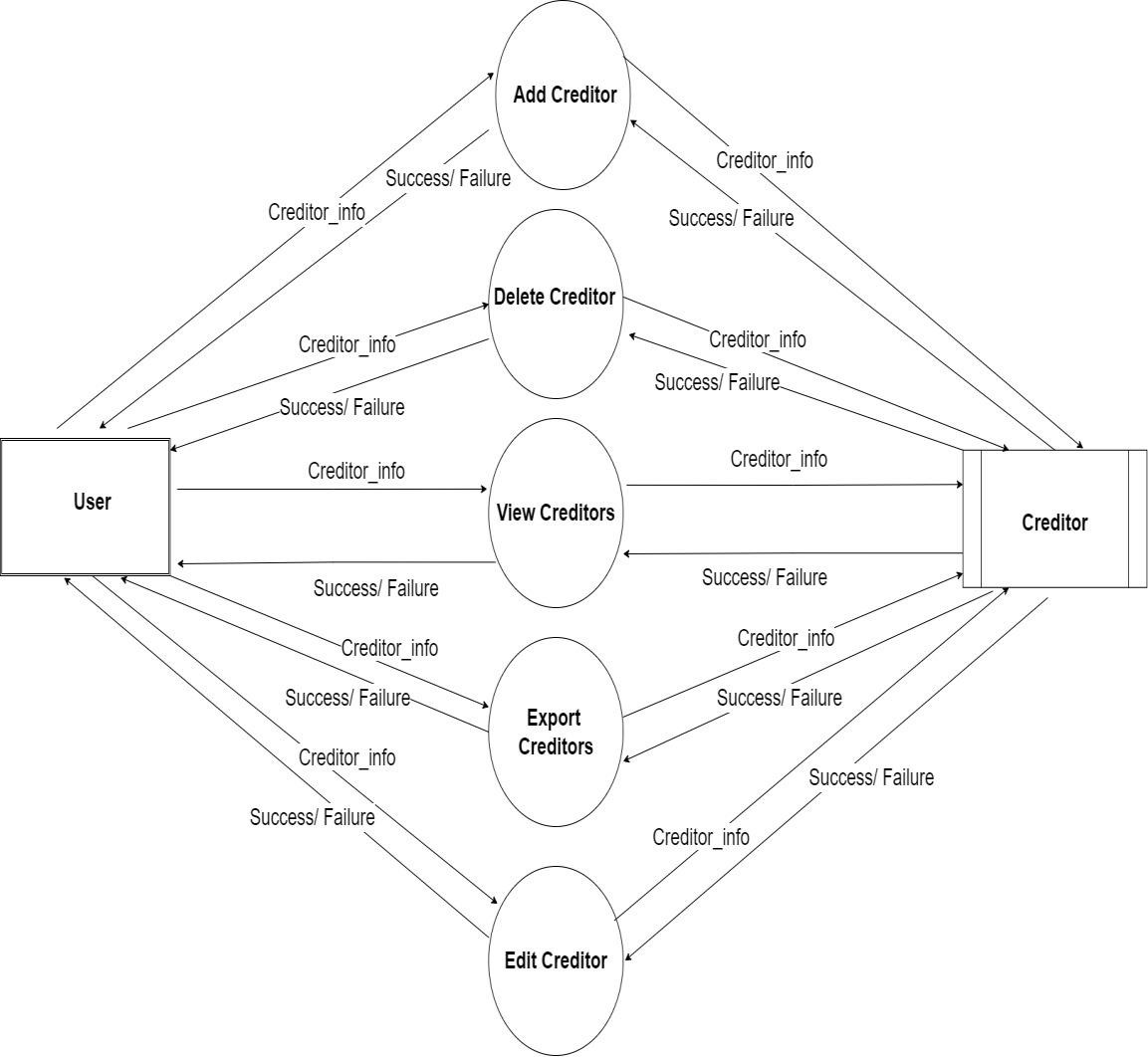
**LEVEL 1 :**



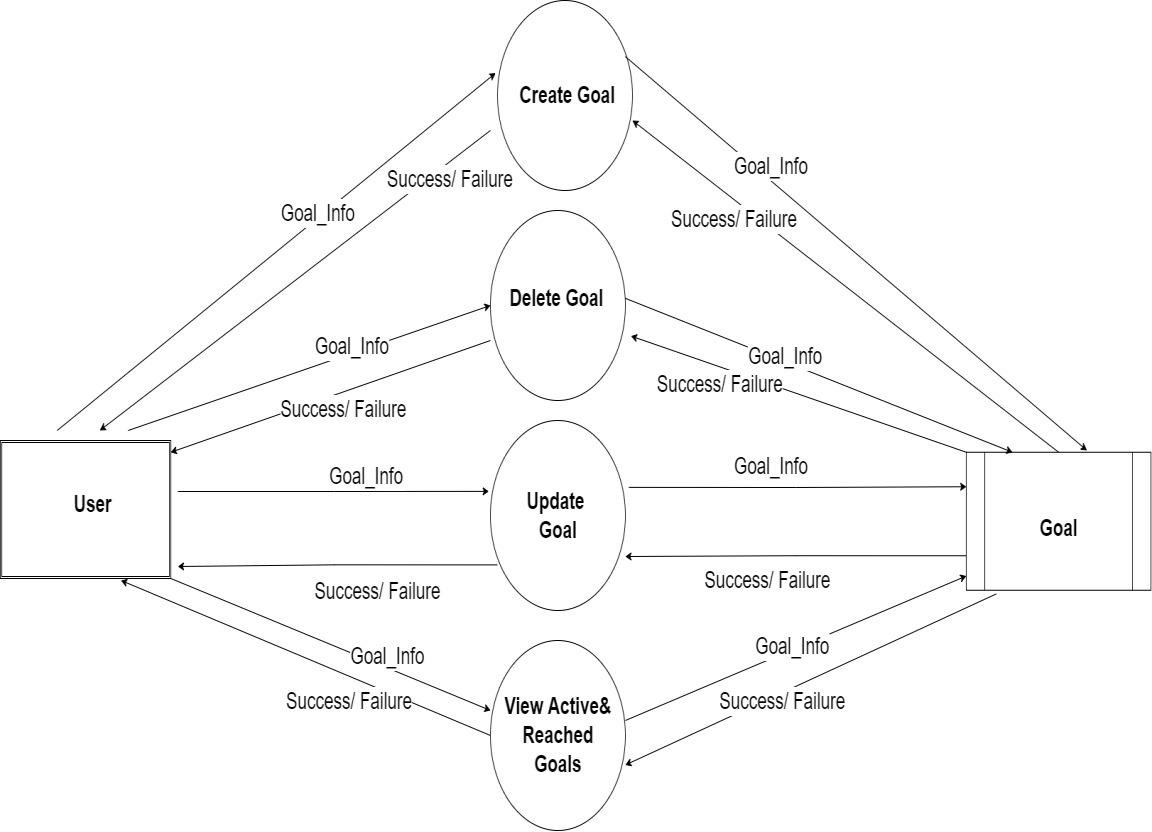
* **LEVEL 2**
* **Transaction**
* Debtor



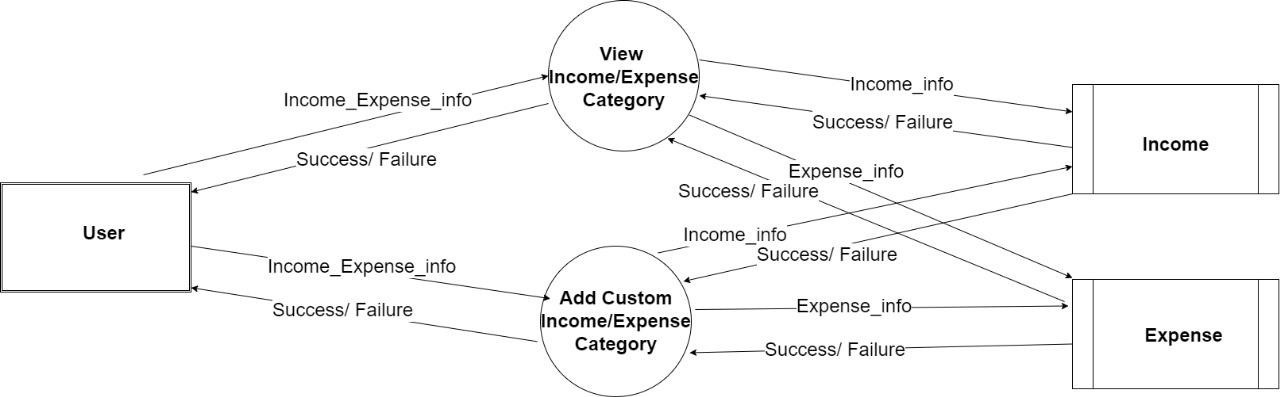
* Creditor



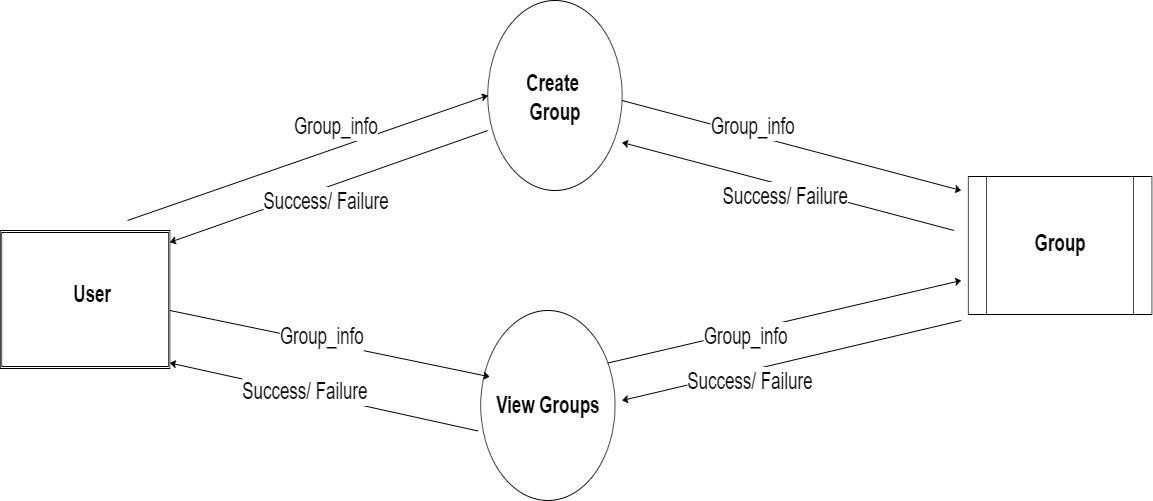
* **Goal :**



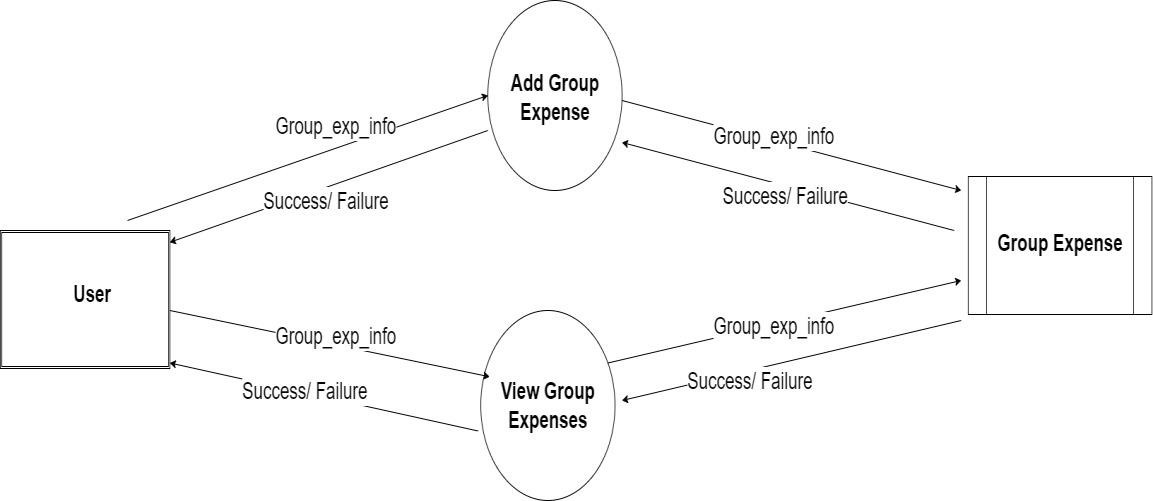
* Categories



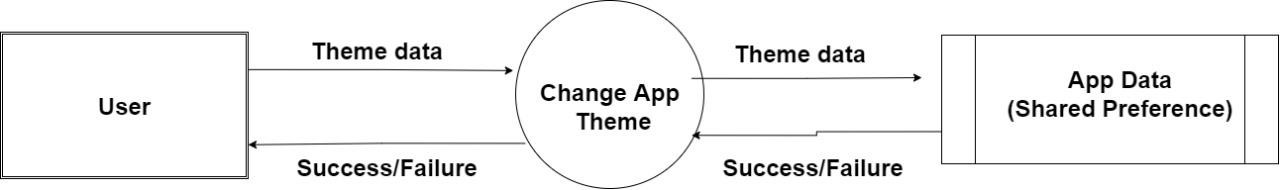
* **Group :**



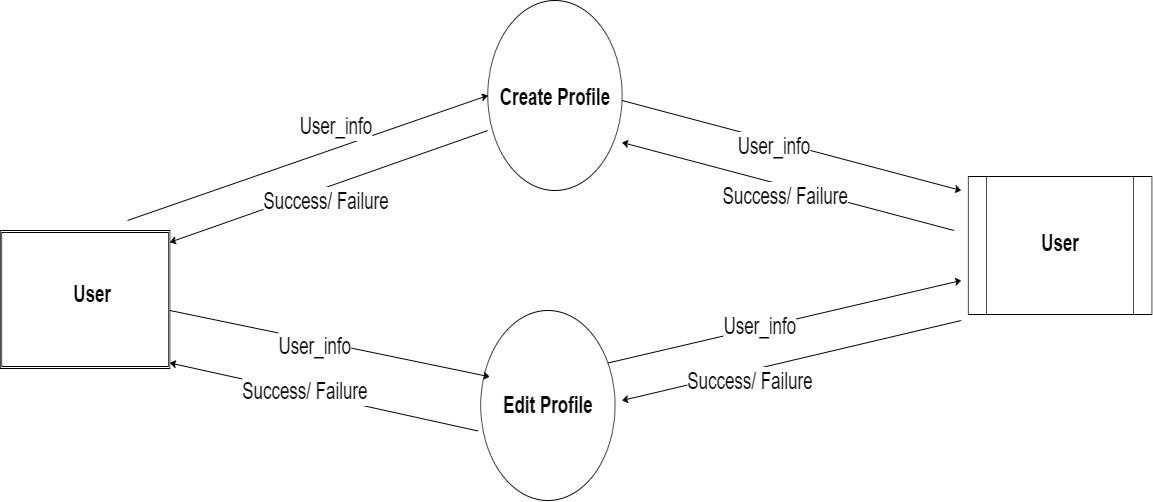
* Group Expense



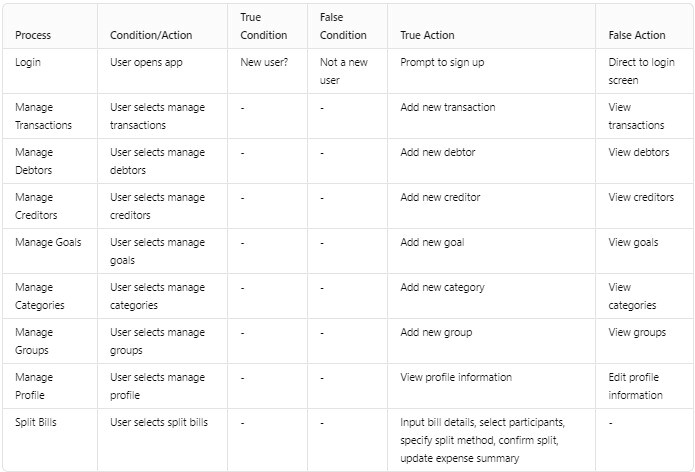
* App Theme



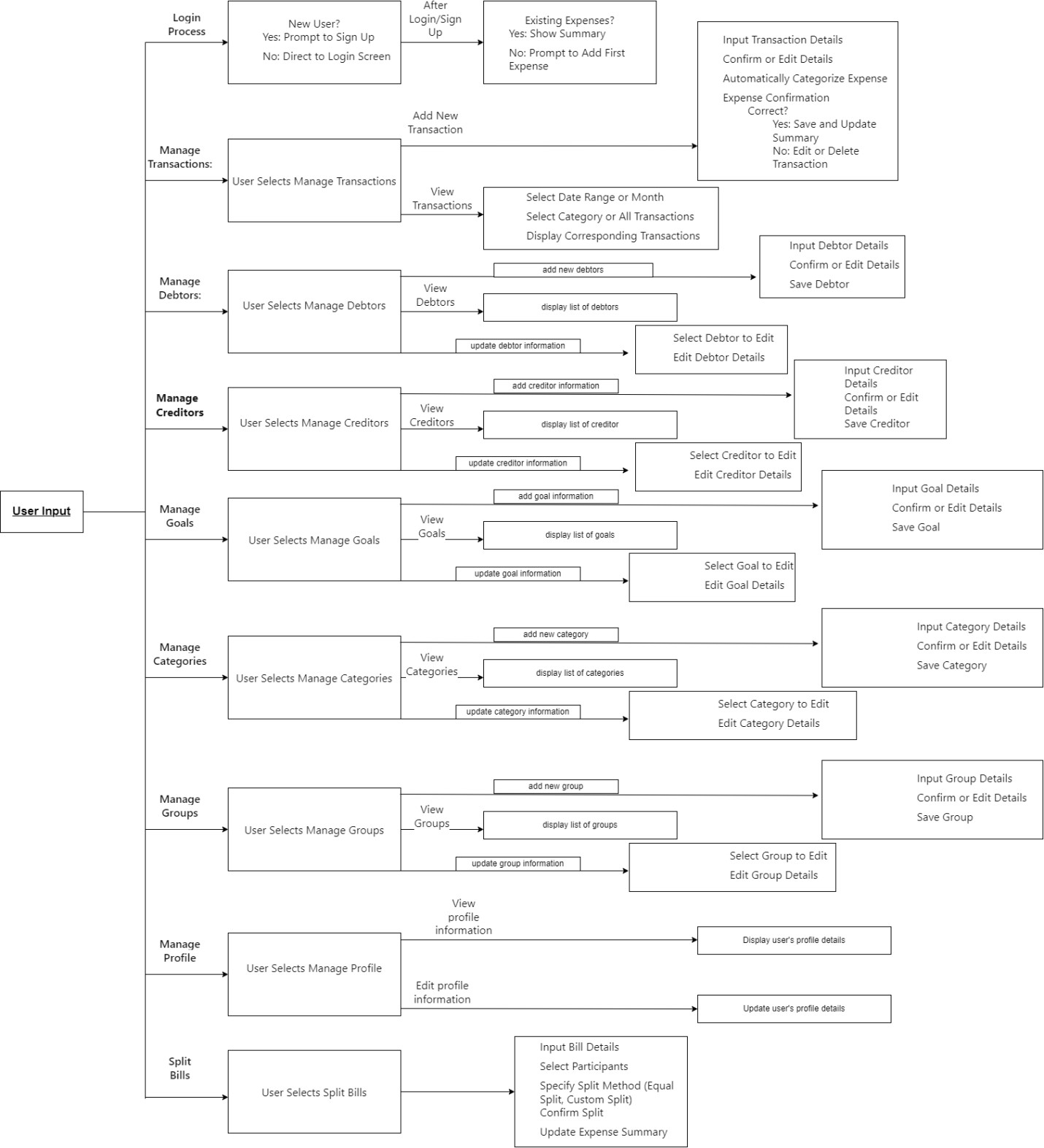
* Profile



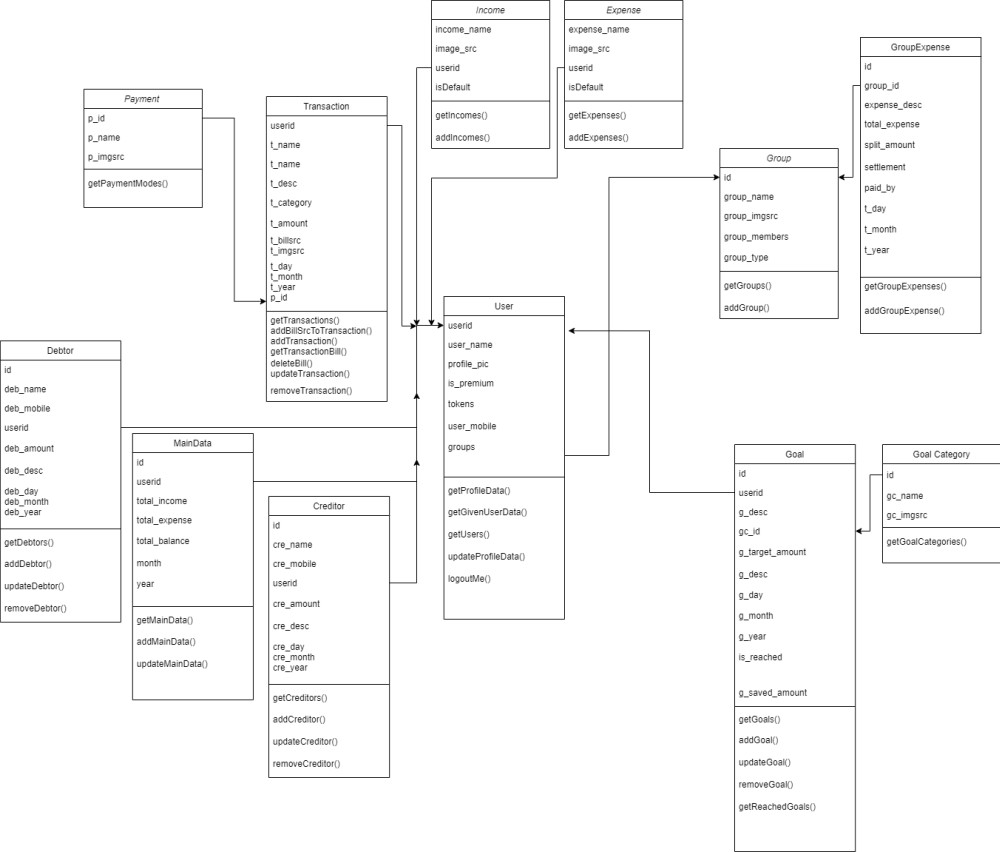
Decision Table :



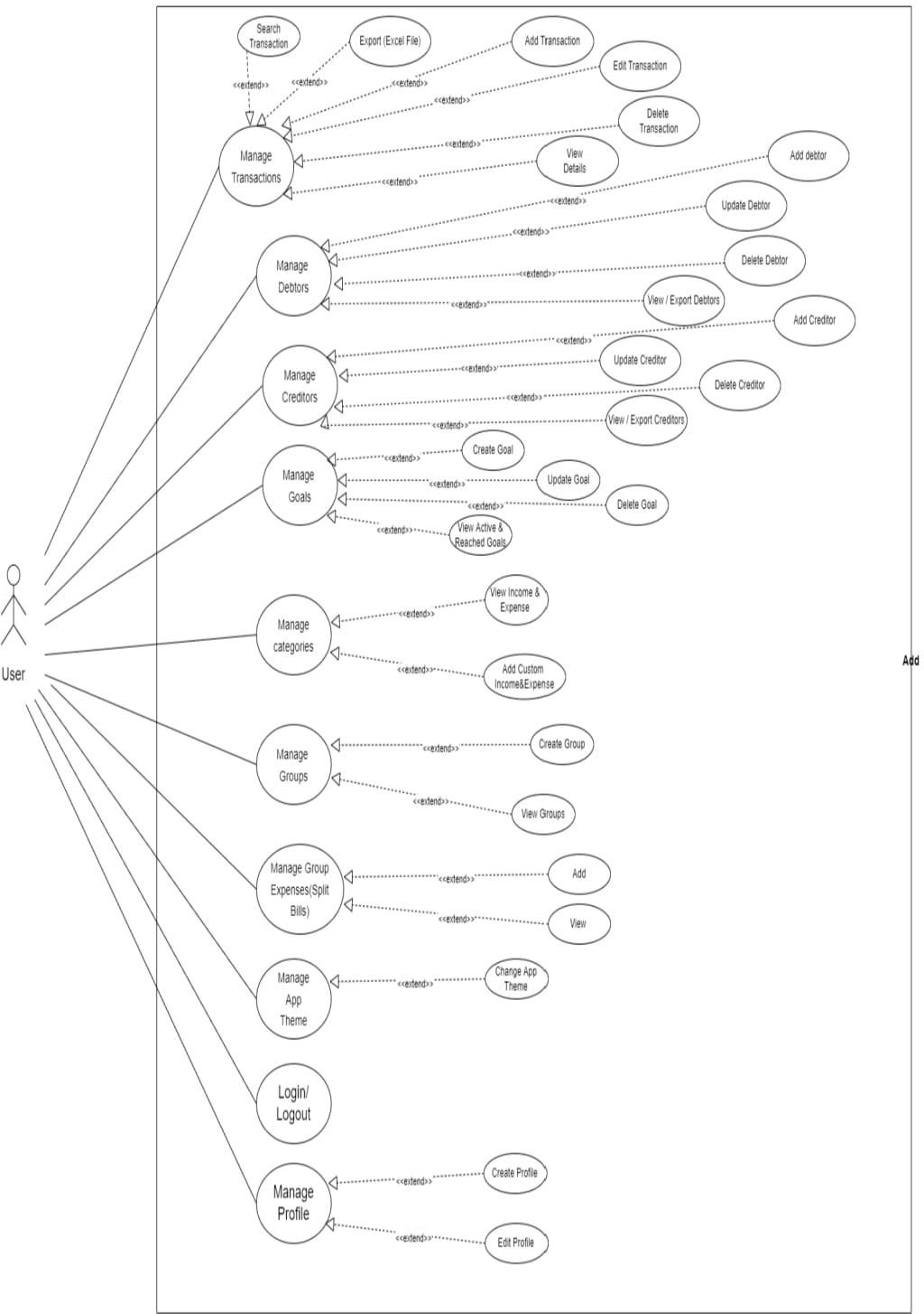
Decision Tree :



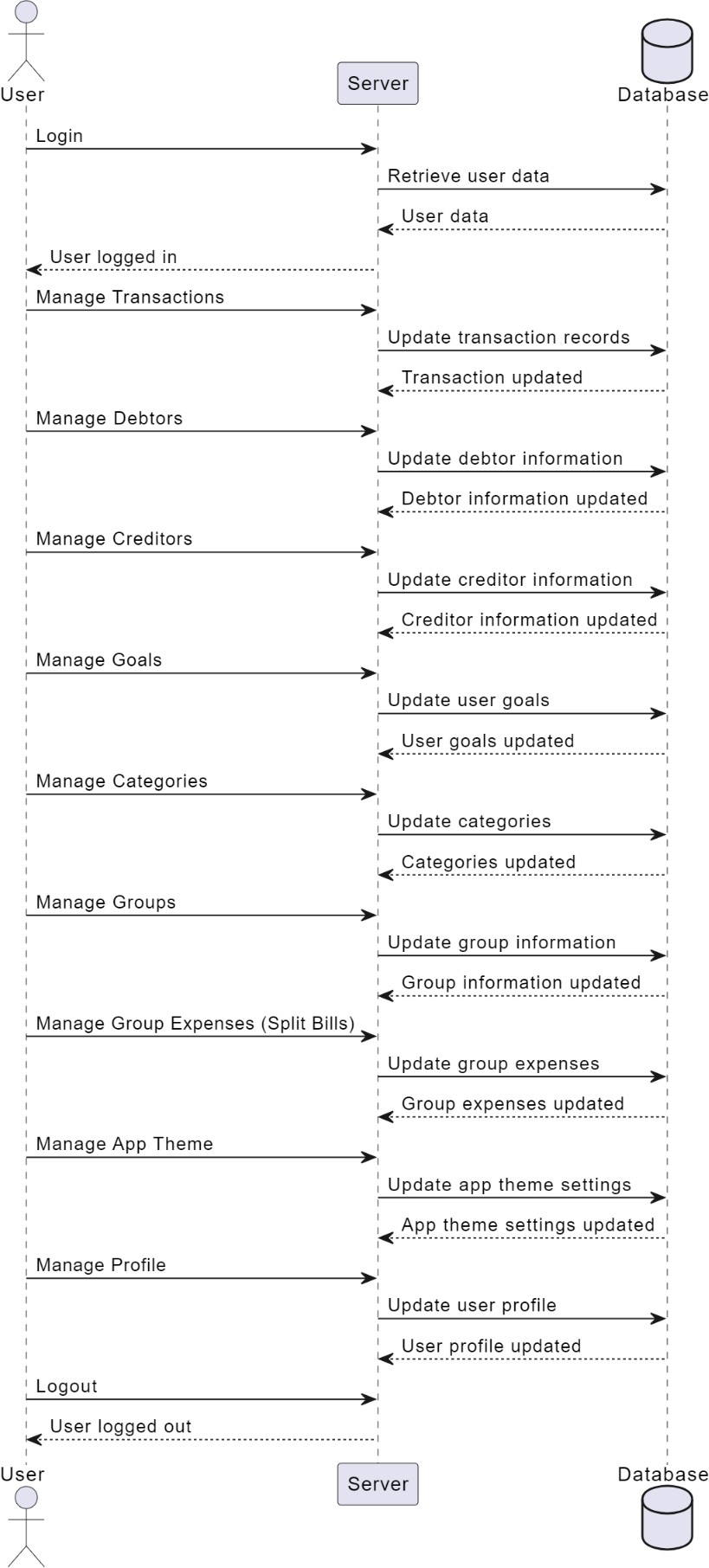
Class Diagram

****

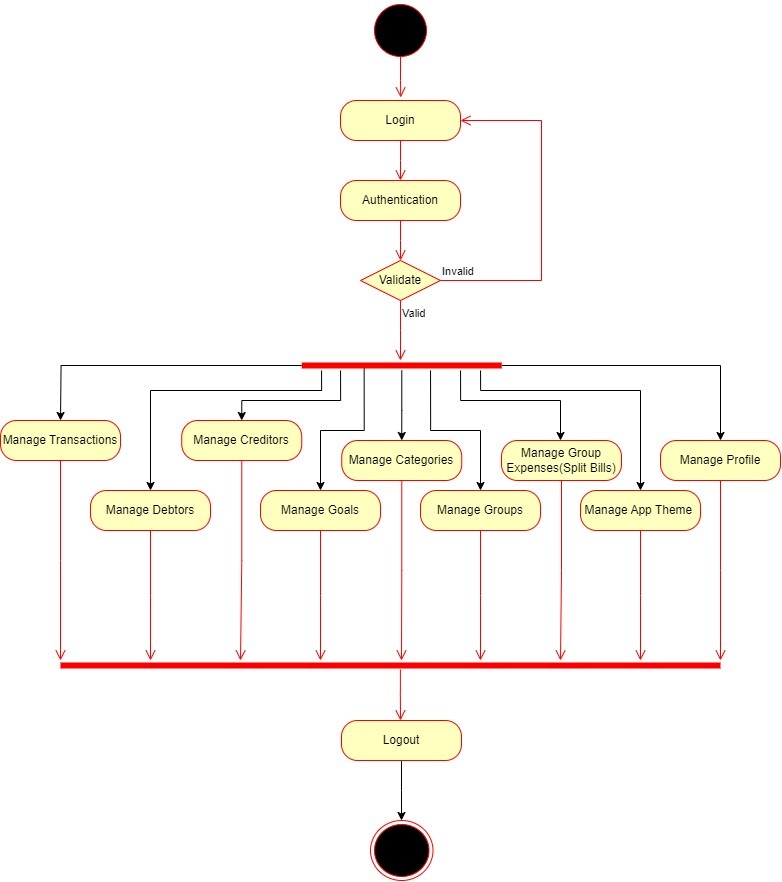
Use-case Diagram

****

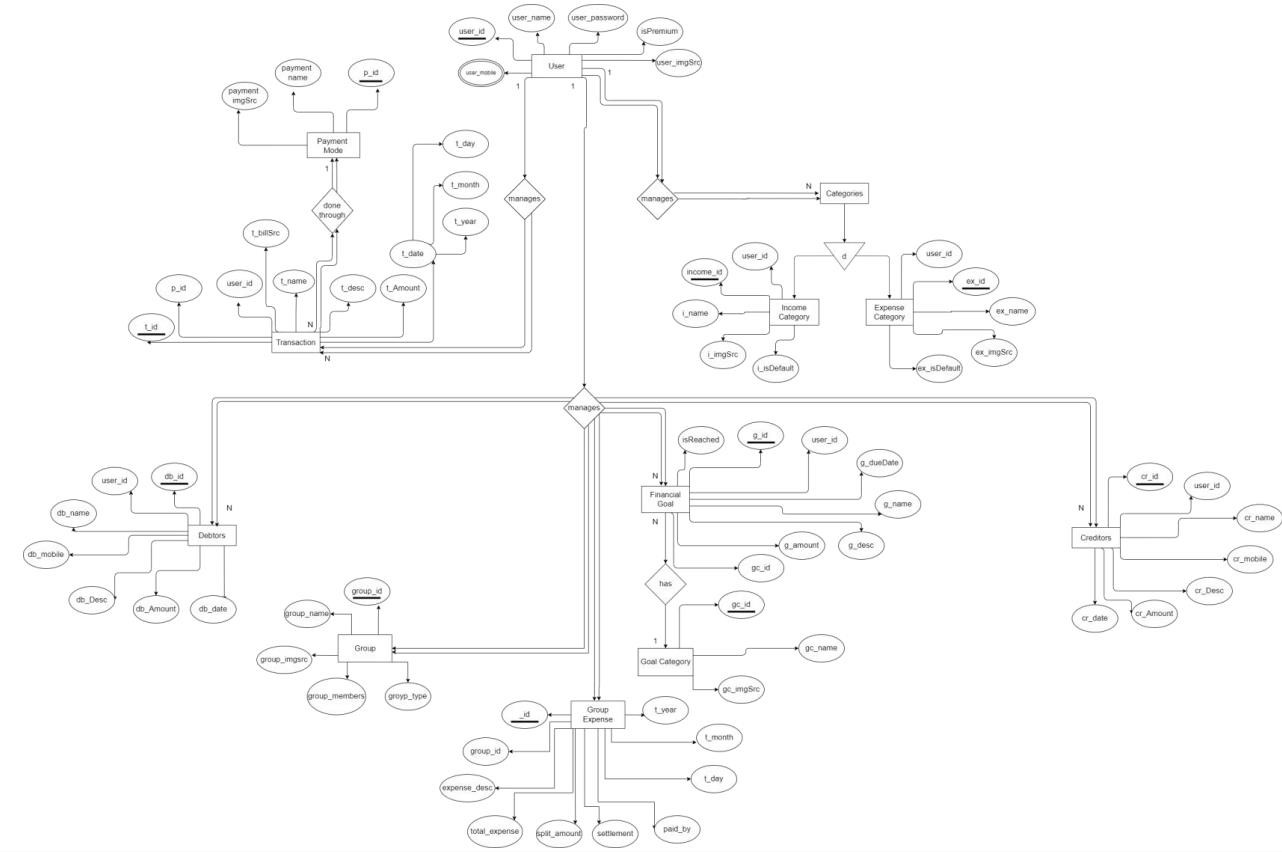
Sequence Diagram

****

Activity Diagram

****

ER Diagram :

****