Ahmedabad University

School of Engineering and Applied Science

Winter 2021 Semester

Software Engineering(SE)

System Requirements Document

Personal Finance Manager

Guided By :
Prof. Khushru Doctor

Team -14
Members :
Priyanshi Shah(AU1841009)
Varshil Shah(AU1841095)
Meet Kadiya (AU1841099)
Kahaan Patel(AU1841110)
Vidit Vaywala (AU1841128)
Sharvil Patel (AU1841134)
Hemil Shah(AU1841135)
Yash A Patel(AU1841141)

Contents

1. Introduction

- The introduction section gives a basic brief overview about the project, its scope and details about the audience it is intended towards. At the end of this section a list of acronyms and abbreviations has also been provided.

1.1 Overview/Aim

- The Software Specifications Specification (SRS) will provide a comprehensive overview of the Personal Finance management system's requirements. This SRS will provide a thorough understanding of what can be expected from the newly implemented framework that will be built. The clear understanding of the system and its features would allow for the creation of the appropriate software for the end user, which will be used for the development of the project's future stages. It is mainly intended to be proposed to a customer for approval and to serve as a basis for the development team to create the first iteration of the framework.

1.2 Scope of the Project

- Our application revolves around creating a user-friendly interface allowing the users to manage their personal finances more efficiently, tracking their incomes and expenses, getting a detailed analysis of their income and expenses and much more. This web application should be free and accessible from any browser on desktop. This software is a way through which a user can easily get access to his/her income and expense reports and also analyze his/her personal finance requirements. A user can add and track his/her income and expenditure and get various insights about it. A user will get various options to add expenses (e.g. food, entertainment, etc.) and detailed analysis and graphs. A user will get a detailed statement about his income and expenditure with various filters (e.g. daily, weekly, monthly, only food expenses, etc.). A user can set savings goals for the future (e.g. wedding, to buy a car, etc.). Through this interface, a user can decide how much he wants to save and in what timeframe he wants to achieve the goal and also track how his “goal-progress” is going. This

system should be user appropriate, easy to use, provide easy recovery of errors and have an overall end user high subjective satisfaction.

1.3 Intended Audience

This project is intended to be used by users of all age group who wants to keep track of their finances. This SRS is intended for several audiences, including the customer, as well as the project manager, designers, developer and tester.

- This SRS would be used by the customer to ensure that the developer team has produced a product that is appropriate to the customer.
- This SRS will be used by the designer to create the system's architecture. The designer will constantly refer back to this SRS to ensure that the system they are developing will meet the needs of the consumer.
- This SRS will be used by the developer to construct the system's functionality. The developer will relate the specifications specified in this SRS to the software they build to ensure that it meets all of the customer's recorded requirements.
- This SRS will be used by the tester to generate test plans and test cases for each recorded requirement. When sections of the program are finished, the tester will run his tests on it to ensure that it meets the specifications outlined in this SRS. When the system is finished, the tester will run his tests on it again to ensure that all of the specifications documented in this SRS have been met.

2. Overall Description

2.1 Product Perspective

- The personal finance management system will act as a virtual assistant for the users. The system will be able to manage the finances of the user and also provide him/her with detailed analysis about their finances. The system also contains an option to save future goals where a user can create new goals and track his goal progress.

- Moreover the system also provides an option to the user to track his/her bills. Since this is a data-centric product it will need somewhere to store the data. For that, a database will be used. The system is data-centric and hence will use a database to store, access and view information. The system uses the PostGreSQL database for meeting the database requirements.

2.2 Product Functions

- The main purpose of the project is to provide the user with a one-solution platform where he can manage his/her finances in an organized way. The user interface is made in such a way as to make sure that the user finds the design attractive and the navigation throughout the website easy.
- A user can add and track his/her income and expenditure and get various insights about it. A user will get various options to add expenses (e.g. food, entertainment, etc.) and detailed analysis and graphs. With the help of different filters, a user will receive a comprehensive statement of his income and expenditure (e.g. daily, weekly, monthly, only food expenses, etc.). A user can set future savings targets (e.g. wedding, to buy a car, etc.). A user can use this interface to determine how much he wants to save and when he wants to achieve the target, as well as monitor his "goal-progress." The user will also be provided with interactive charts about his income and expenditure to analyze his/her incomes and expenses more efficiently.

2.3 User Characteristics

- Our application will only have a single type of user. A user can be anyone, from any age group, looking for a platform to manage his/her finances more efficiently. The user will be able to add income, add expenses and calculate savings. He/she will also be able to view income and expense charts with various filters in order to visualize the finances more efficiently. A user will also be able to create future goals and track his/her goals' progress. The user can deposit money into the goal from time to time and can also edit his goal targets. The user will also get an option to add and track his bill payments so as to maintain all his bills at one place and keep a track of it.

2.4 Operating Environment

- Our system uses a client-server architecture and the system can be accessed via the internet as it is a data-centric system which uses a database to store and retrieve information.

Software

- The website is intended to run on any platform that supports web browsers such as Google Chrome, Mozilla Firefox, and others.
- The system uses the PostGreSQL database to store data.

Hardware

- Operating System Supports all known operating systems, such as Windows, Linux or Mac, Android, IOS, etc.
- The website is a responsive website and hence can be accessed on a PC, Laptop, Smartphone, Tablet, etc.

2.5 Design and Implementation constraints

- PostgreSQL can support a reasonably large database, but if the data exceeds a certain size, more cloud-based alternatives can be used.
- Only users who have knowledge of English language will be able to understand the system. The framework does not accept multiple languages.
- Since the system is data-centric and includes data storage and retrieval, an active internet connection is required for the system to operate without interruptions.

2.6 Assumptions and Dependencies

- One assumption about the product is that it will always be accessible through a web browser on a desktop computer. It is dependent on Internet-based servers in order for its web-based functionality to be displayed on the website. Users should be familiar with the basics of how to link to the Internet and navigate to a website.
- The system is using the PostGreSQL for its functioning. Hence the smooth usage of the system demands that the servers of the database work efficiently.

3.SPECIFIC REQUIREMENTS

- This section outlines the system's functional and quality specifications. It provides a concise overview of the system's capabilities.

3.1 External User Requirement

- This section contains a brief summary of the system's inputs and outputs. It also includes an overview of the device communication interfaces as well as simple user interface prototypes.

3.1.1 Hardware interfaces

The system comprises of a software architecture and hence it does not require any hardware components as such. The internet required for the system's efficient working will be taken care of by the Operating System.

3.1.2 Software interfaces

The interface should be able to connect with the database whenever a user requires to store or retrieve data from the system. The software also requires an element of responsiveness so that it can be accessed from any device without worrying about the resolution of that specific device.

3.1.3 Communication interfaces

The system requires communicating with the database in order to function properly. The operating system takes care of these basic communication needs.

3.2 FUNCTIONAL REQUIREMENTS

- This section includes the requirements that specify all the fundamental actions of the software system.

3.2.1 The User

3.2.1.1 Functional requirement 1.1

ID: PM1

TITLE: Open Web Application

DESC: A user should be able to view the web application through any browser on his laptop, smartphone, tablet or any other device.

RAT: The user should be able to view the web application.

DEP: None

3.2.1.2 Functional requirement 1.2

ID: PM2

TITLE: User Login and Sign Up

DESC: After viewing the homepage, the user shall be able to login to the system if already registered. And if the user is not registered, he should be able to sign up after providing the necessary details.

RAT: In order for a user to sign up or log in to the personal finance management system's web application, they must first create an account.

DEP: PM1

3.2.1.3 Functional requirement 1.3

ID: PM3

TITLE: User Dashboard

DESC: A user should be able to see all the important information like total income this month, total expense, line charts of income and expenses, goal-progress, etc. on the dashboard page.

RAT: In order for a user to view a summary of his account on the dashboard page.

DEP: PM1**,** PM2

3.2.1.4 Functional requirement 1.4

ID: PM4

TITLE: Add Income

DESC: A user should be able to add income to his account and also specify the details as to from where he earned that income e.g. salary, rent, etc to keep a track of it.

RAT: In order for a user to add income in the web application.

DEP: PM1, PM2, PM3

3.2.1.5 Functional requirement 1.5

ID: PM5

TITLE: Add Expense

DESC: A user should be able to add expense to his account and also specify the details as to where he incurred that expense e.g. food, rent, etc to keep a track of it.

RAT: In order for a user to add expense in the web application.

DEP: PM1, PM2, PM3

3.2.1.6 Functional requirement 1.6

ID: PM6

TITLE: View Income Statement and Charts

DESC: A user should be able to view the statement of his/her income and also view relevant charts as to from which sources he earned what proportion of his/her income.

RAT: In order to view the income statement and charts to get a better analysis of the income.

DEP: PM1, PM2, PM3

3.2.1.7 Functional requirement 1.7

ID: PM7

TITLE: View Expense Statement and Charts

DESC: A user should be able to view the statement of his/her expense and also view relevant charts as to on which sources he incurred what proportion of his/her expense.

RAT: In order to view the expense statement and charts to get a better analysis of the expense.

DEP: PM1, PM2, PM3

3.2.1.8 Functional requirement 1.8

ID: PM8

TITLE: Add Goals

DESC: A user should be able to create goals for future and add it to the database. He/she has to input various parameters like goal length, amount to save, reason (goal name), goal description etc. Hence the system will give the user an option to track his/her future goals in an efficient manner.

RAT: In order for a user to add new goals to track it in future.

DEP: PM1, PM2, PM3

3.2.1.9 Functional requirement 1.9

ID: PM9

TITLE: View , Edit and delete goals

DESC: A user should be able to view his/her current goals, as well as edit the goal if required. If the user feels that a goal has become irrelevant for him now then he/she can even delete it.

RAT: In order for a user to view, edit and delete the goals he created.

DEP: PM1, PM2, PM3, PM8

3.2.1.10 Functional requirement 1.10

ID: PM10

TITLE: Add Bills

DESC: A user should be able to add bills and add it to the database. He/she has to input various parameters like bill amount, due date, bill name, bill description etc. Hence the system will give the user an option to track his/her bills in an efficient manner.

RAT: In order for a user to add new goals to track it in future.

DEP: PM1, PM2, PM3

3.2.1.11 Functional requirement 1.11

ID: PM11

TITLE: View and delete Bills

DESC: A user should be able to view his/her current bill. If the user feels that a bill has become irrelevant for him or he has already paid it, then he/she can even delete it.

RAT: In order for a user to view and delete the bills he added.

DEP: PM1, PM2, PM3, PM10

3.3 NON- FUNCTIONAL REQUIREMENTS

3.3.1 Non- Functional requirement : Security

- **ID: PMNR 1**

TAG: User login Account

GIST: Only verified login details.

SCALE: If a user tries to enter mobile number with less than 10 digits and an email in incorrect format, he/she will not be able to login.

MUST: All the time.

3.3.2 Non- Functional requirement : Maintainability

- **ID: PMNR 2**

TITLE: Application lifespan

DESC: With regular maintenance and updates in the code, the application will have a longer lifespan.

RAT: In order to increase the application lifespan

DEP: none

3.3.3 Non- Functional requirement: Portability

- **ID: PMNR 3**

TITLE: Application maintainability
DESC: Maintainability means the time required for a solution or a component to be fixed, changed or adapted. Our application, with regular maintenance and updates, can be maintained properly.
RAT: In order to maintain the application
DEP: none

- **ID: PMNR 3**

TITLE: Application Testability
DESC: Proper test environments should be built for the proper testing of the application and errors and bugs should be solved as and when detected.
RAT: In order to test the application and keep it bug-free
DEP: none

3.3.4 Non- Functional requirement: Portability

- **ID: PMNR 4**

TITLE: Application portability
DESC: If hosted, the application should be able to run in any browser on any device. The website has been made responsive in order to run properly on any device in spite of the resolution.
RAT: In order to provide portability to the application.
DEP: none

3.2.5 Non- Functional requirement: Performance

- **ID: PMNR 5**

TAG: Response Time
GIST: The response time of the website is the ability of the website to load its pages and switch between the pages at a faster rate.
SCALE: The response time of the different output pages after performing different functionalities.
MUST: The website should response in no more than 5 seconds.
WISH: The website should response in no more than 2 seconds.

3.3.6 Non- Functional requirement : Availability

- **ID: PMNR 6**

QR8 TAG: System Dependability
GIST: The ability of the system to inform the user of the faults.
SCALE: Whenever the application gets a strange input, it will inform the user that the input is not acceptable. For e.g. if a user inputs negative income or expense then the system will not accept that input. Also the system should inform the user if the internet connectivity is lost in between.
MUST: All the time.

3.3.7 Non- Functional requirement : Availability

- **ID: PMNR 7**

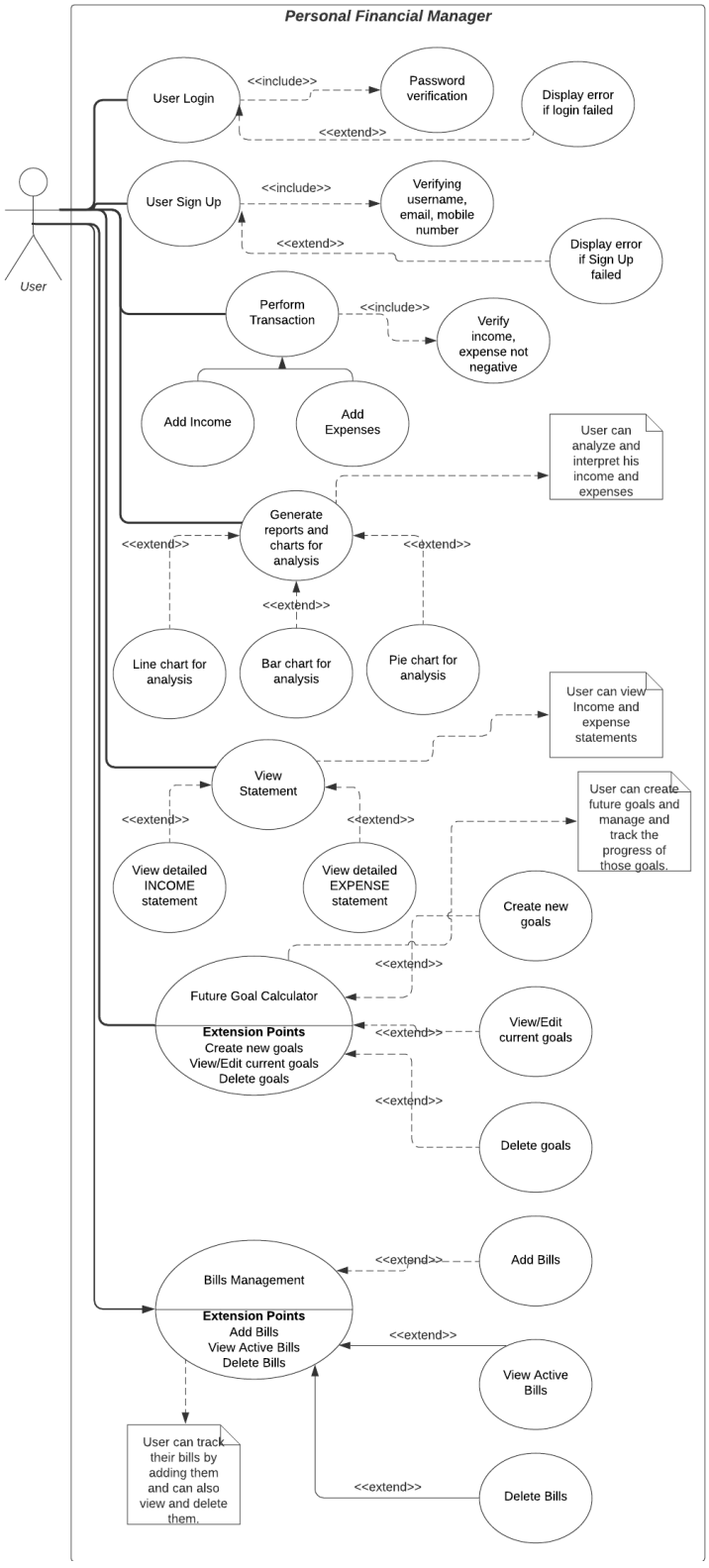
TITLE: Internet Connection Requirement

DESC: The application uses a database to store, retrieve and manipulate data and hence an internet connection at all times while using the application is a must.

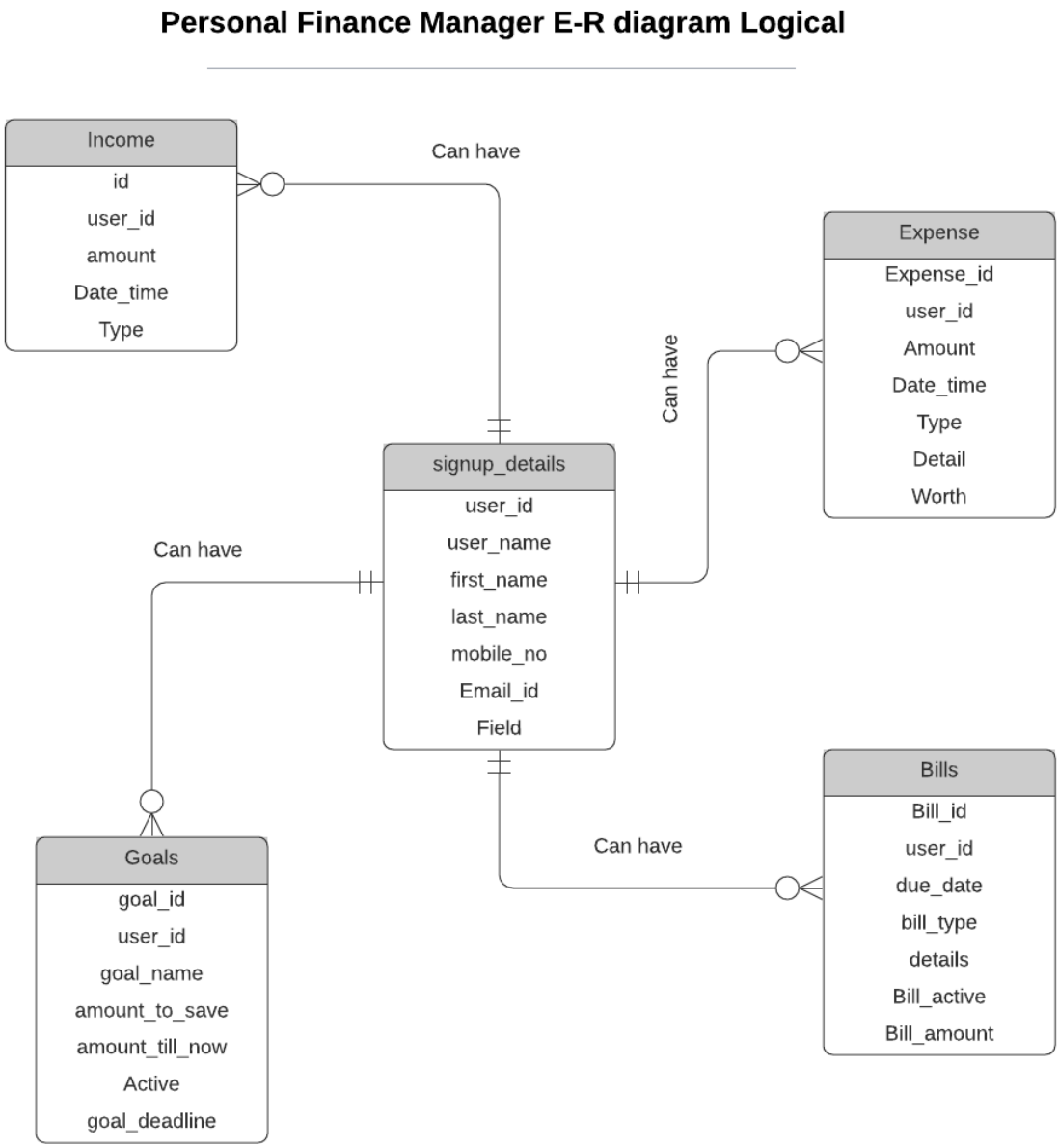
RAT: In order for the application to communicate with the database effectively.

DEP: none

4. Use Case Diagram

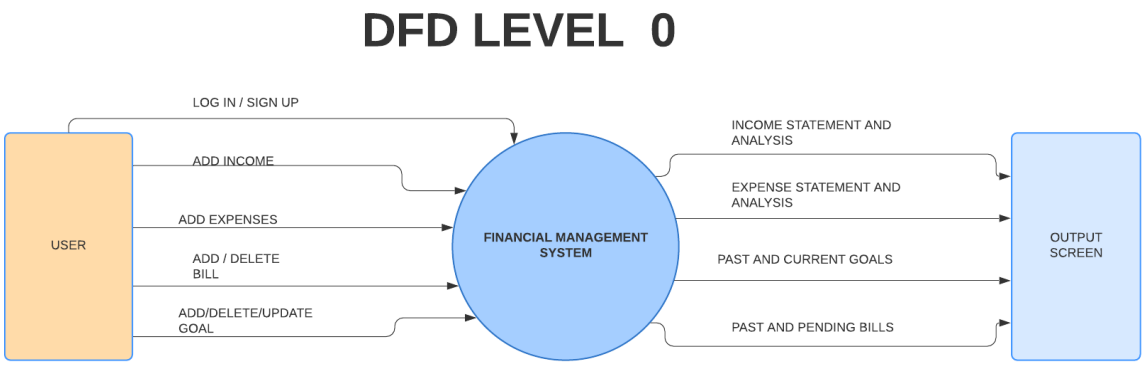


5. ER Diagram

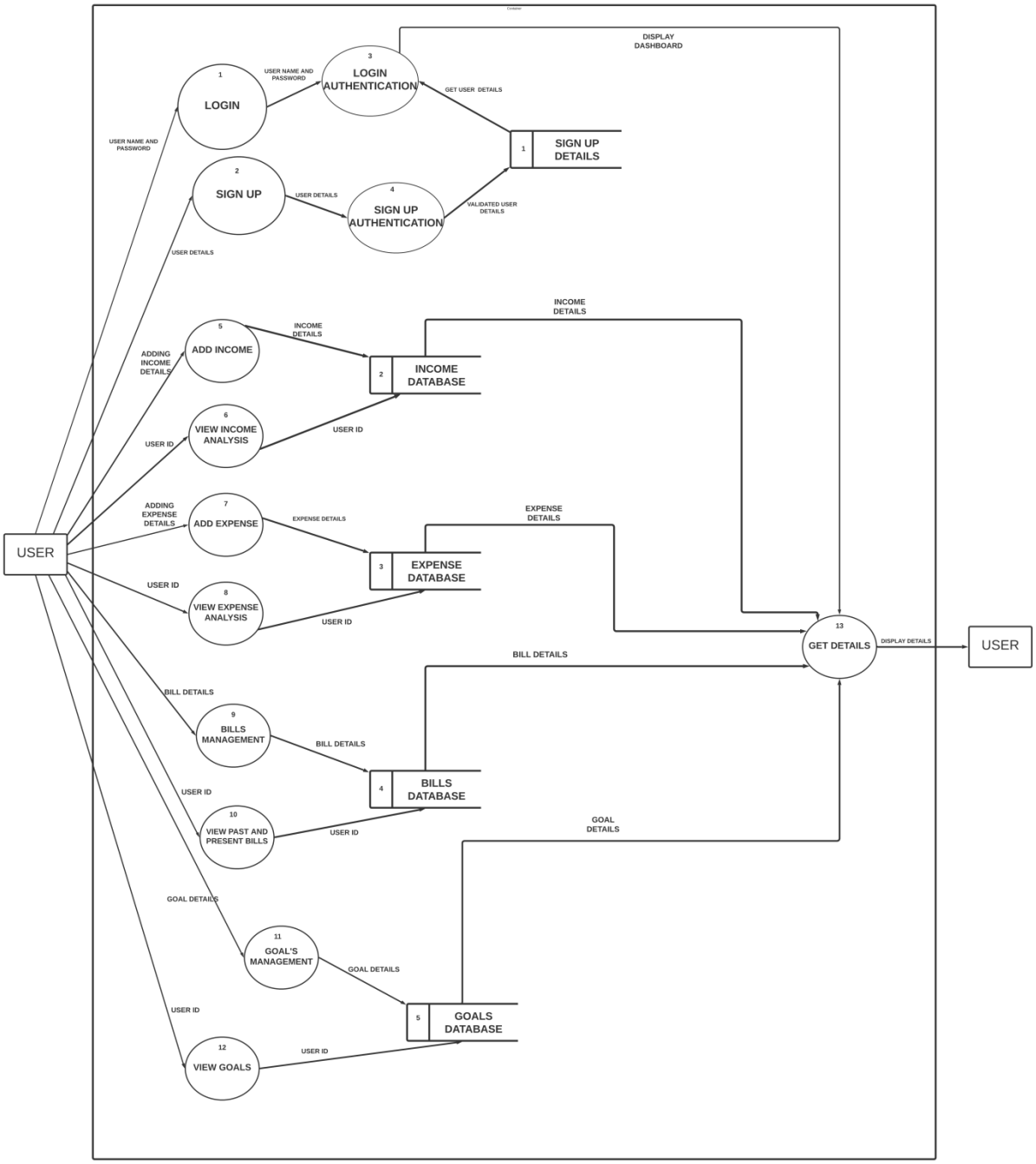

Personal Finance Manager E-R diagram Logical
Income
id
user_id
amount
Date_time
Type
Can have
Expense
Expense_id
user_id
Amount
Date_time
Type
Detail
Worth
Can have
signup_details
user_id
user_name
first_name
last_name
mobile_no
Email_id
Field
Can have
Goals
goal_id
user_id
goal_name
amount_to_save
amount_till_now
Active
goal_deadline
Can have
Bills
Bill_id
user_id
due_date
bill_type
details
Bill_active
Bill_amount

6. Data-Flow Diagram

6.1 Level 0 - Context Level Diagram

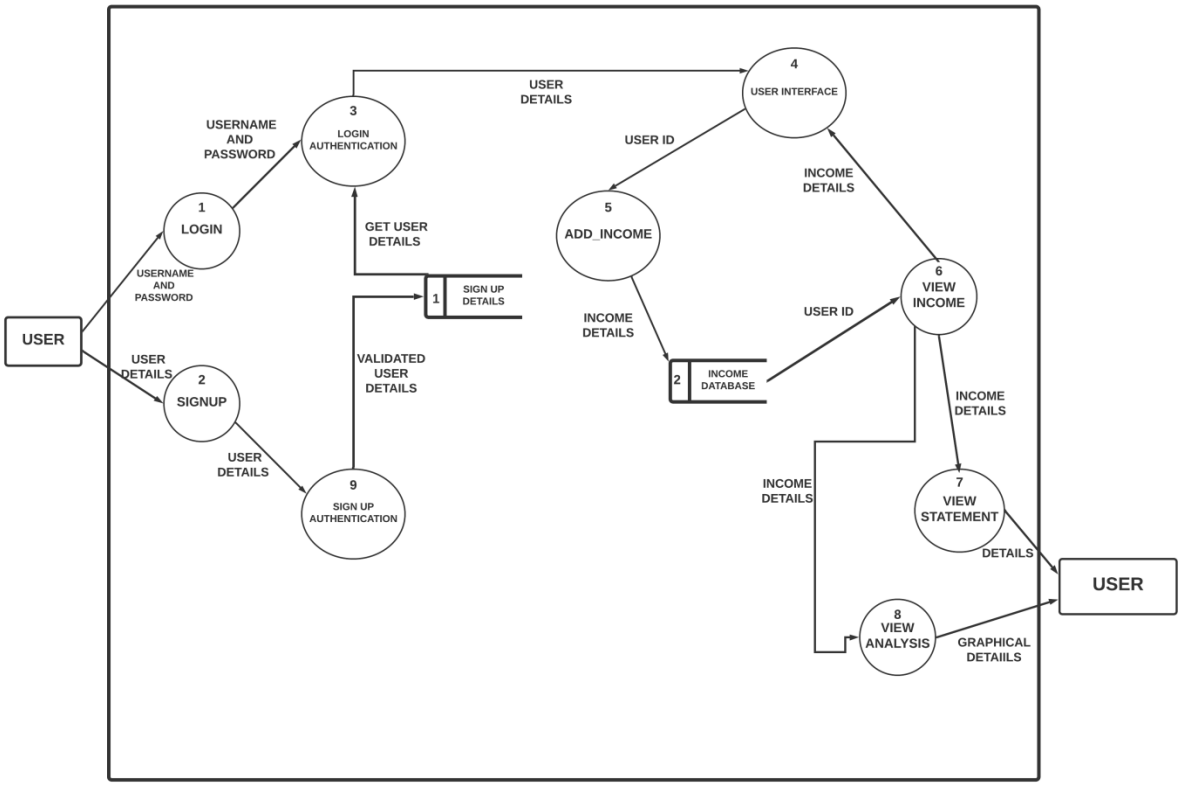


6.2 Level – 1 Data Flow Diagram

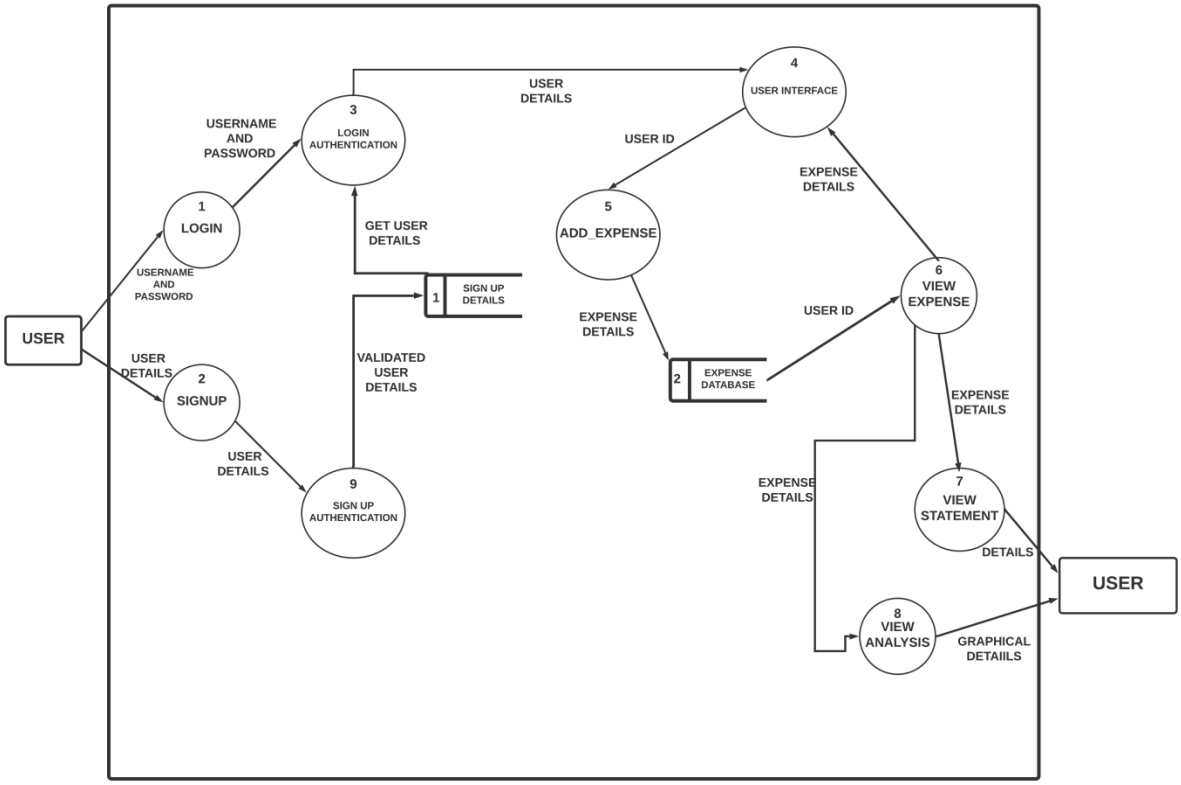


6.3 Level 2

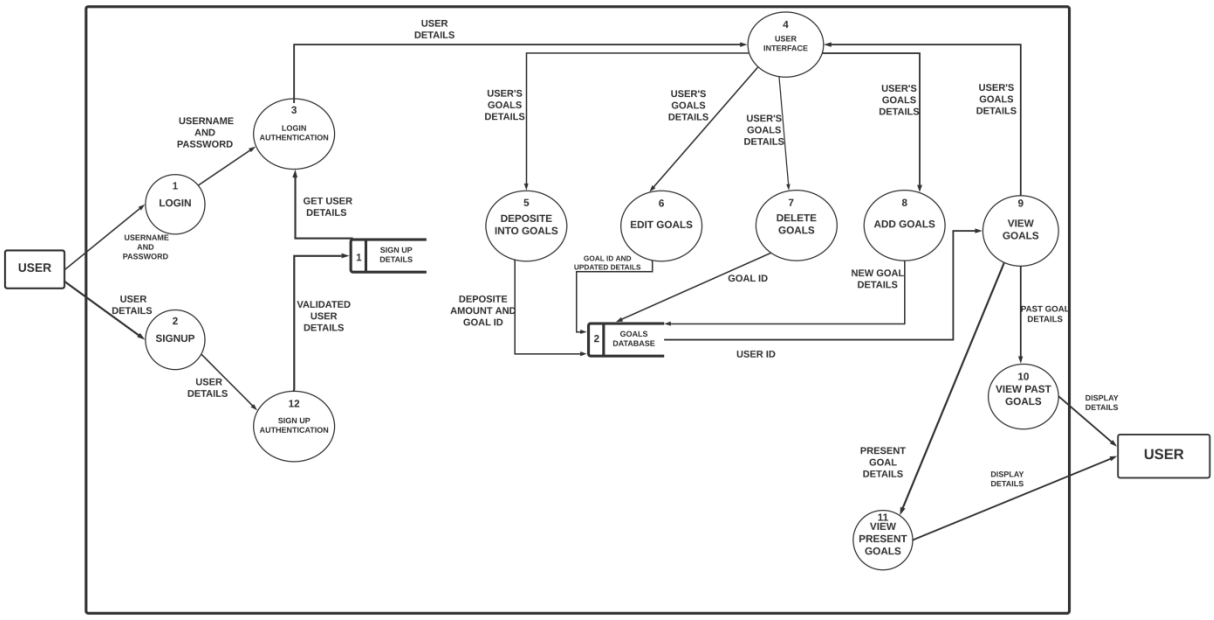
6.3.1 Level 2 - Income



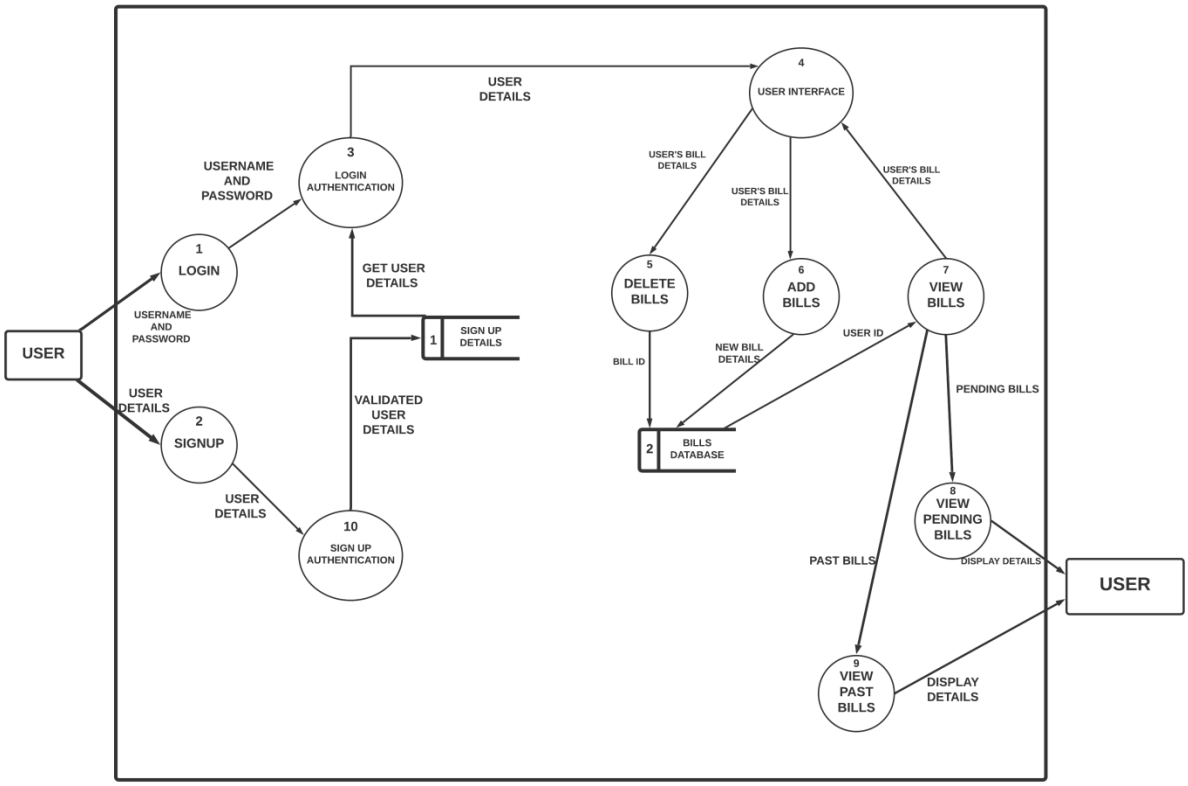
6.3.2 Level 2 – Expense process



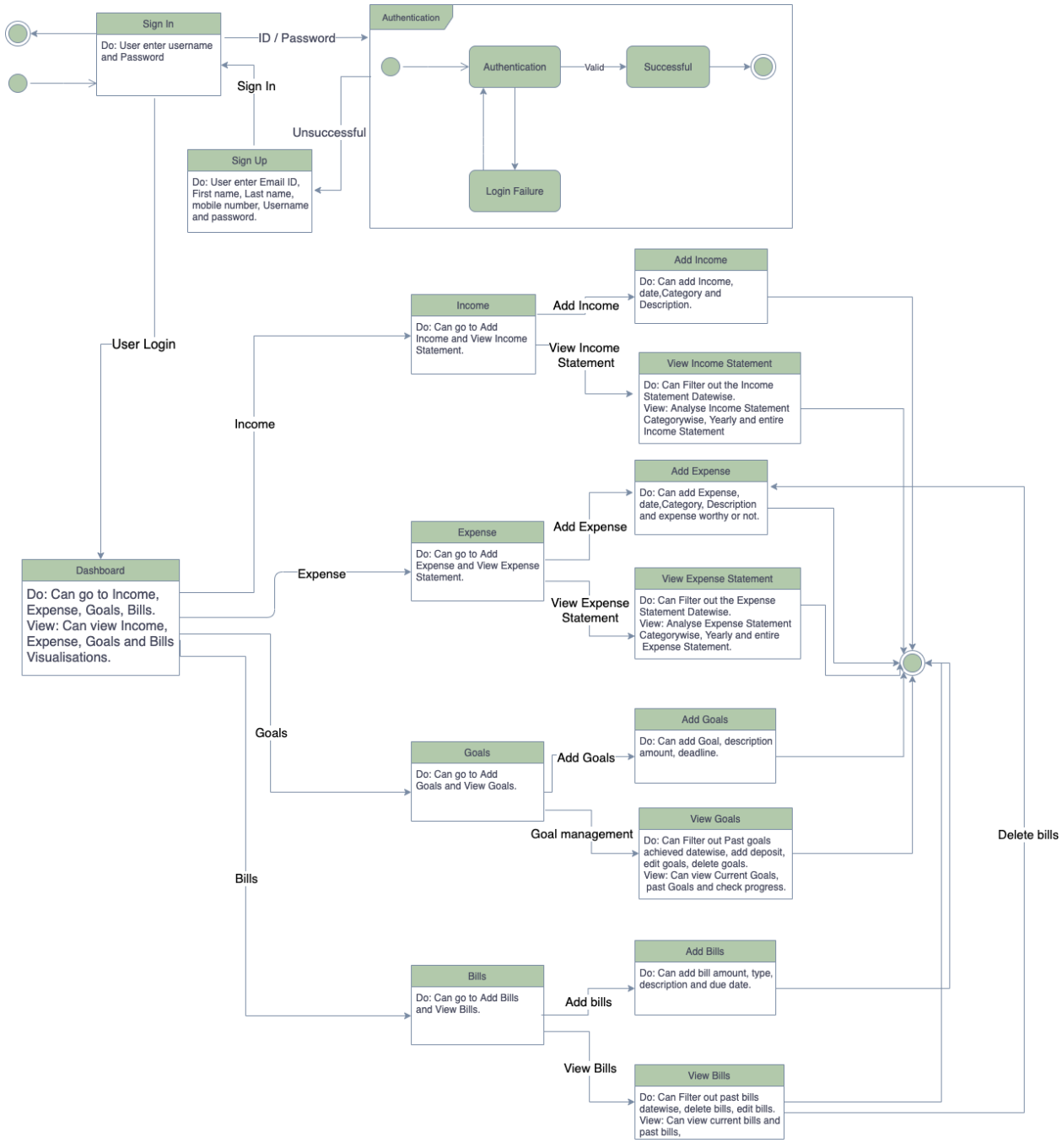
6.3.3 Level 2 - Goals process



6.3.4 Level 2 – Bills process

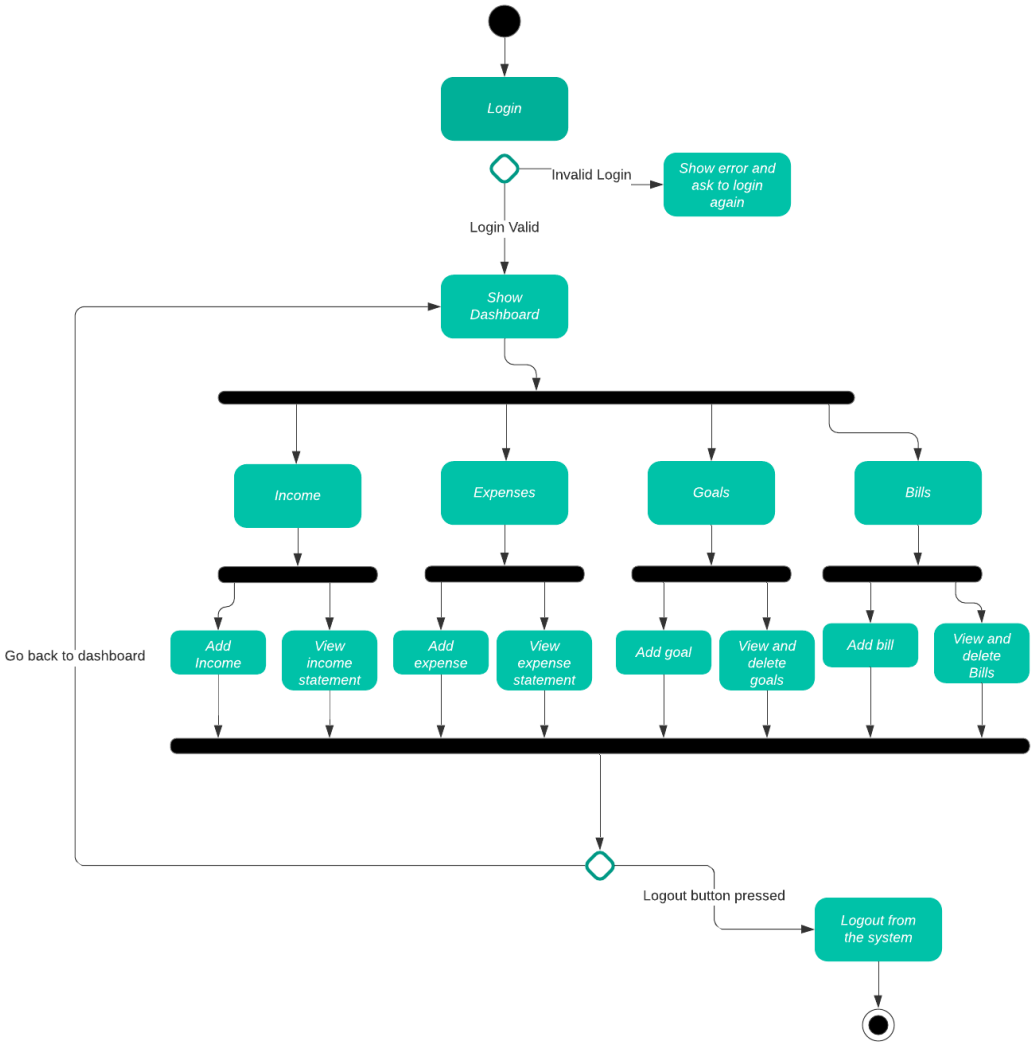


7. State Diagram

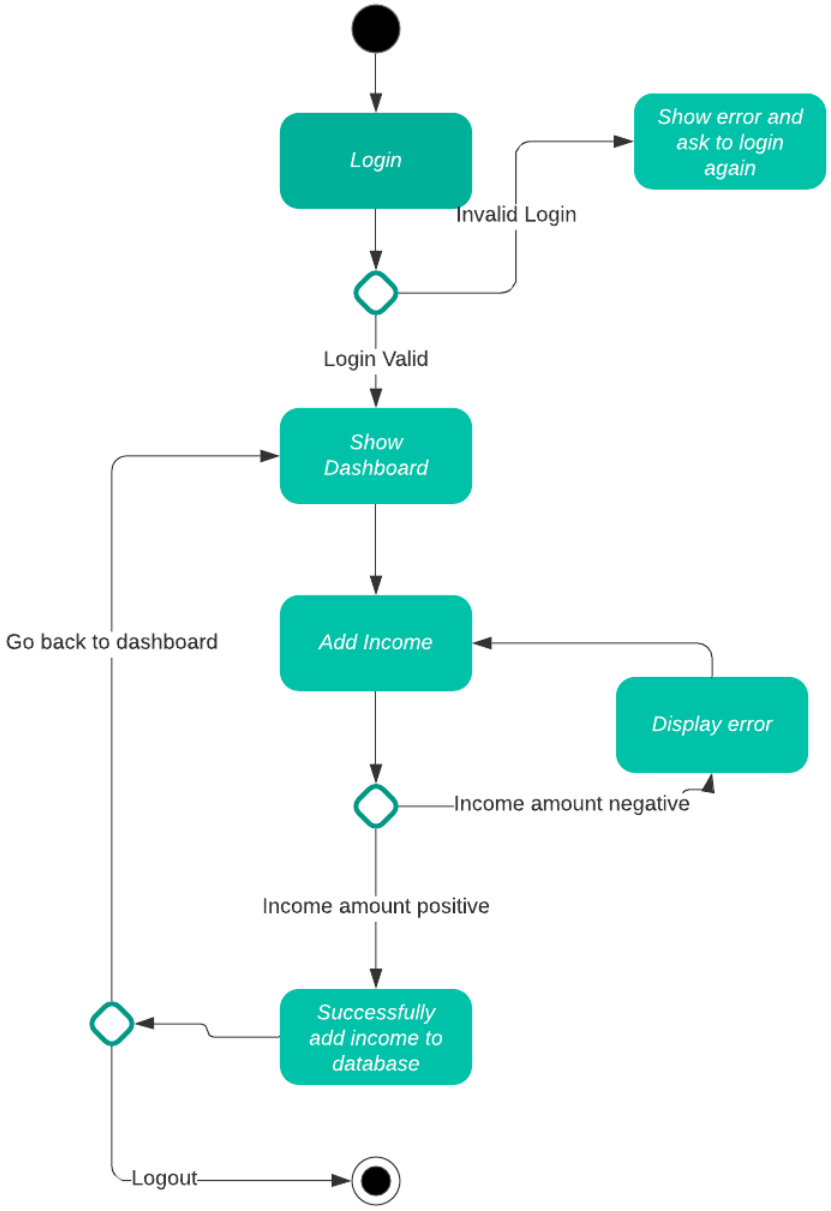


8. Activity Diagram

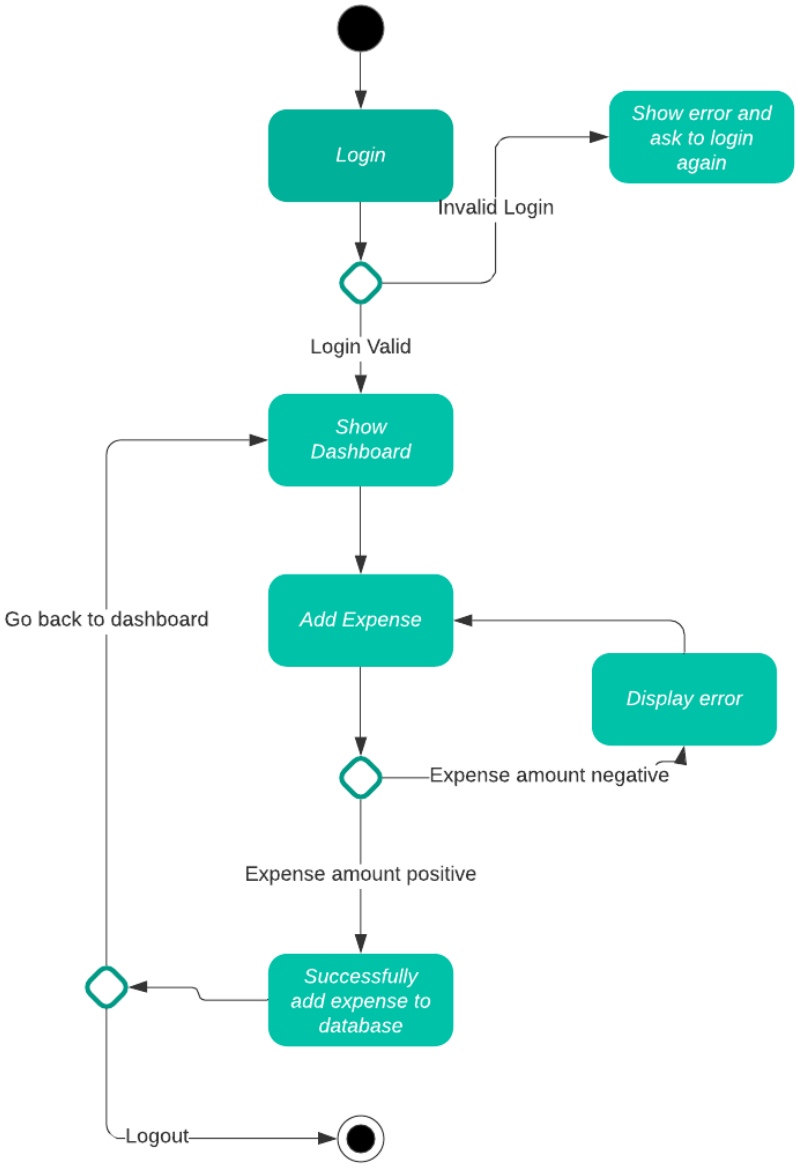
8.1 User Activity Diagram



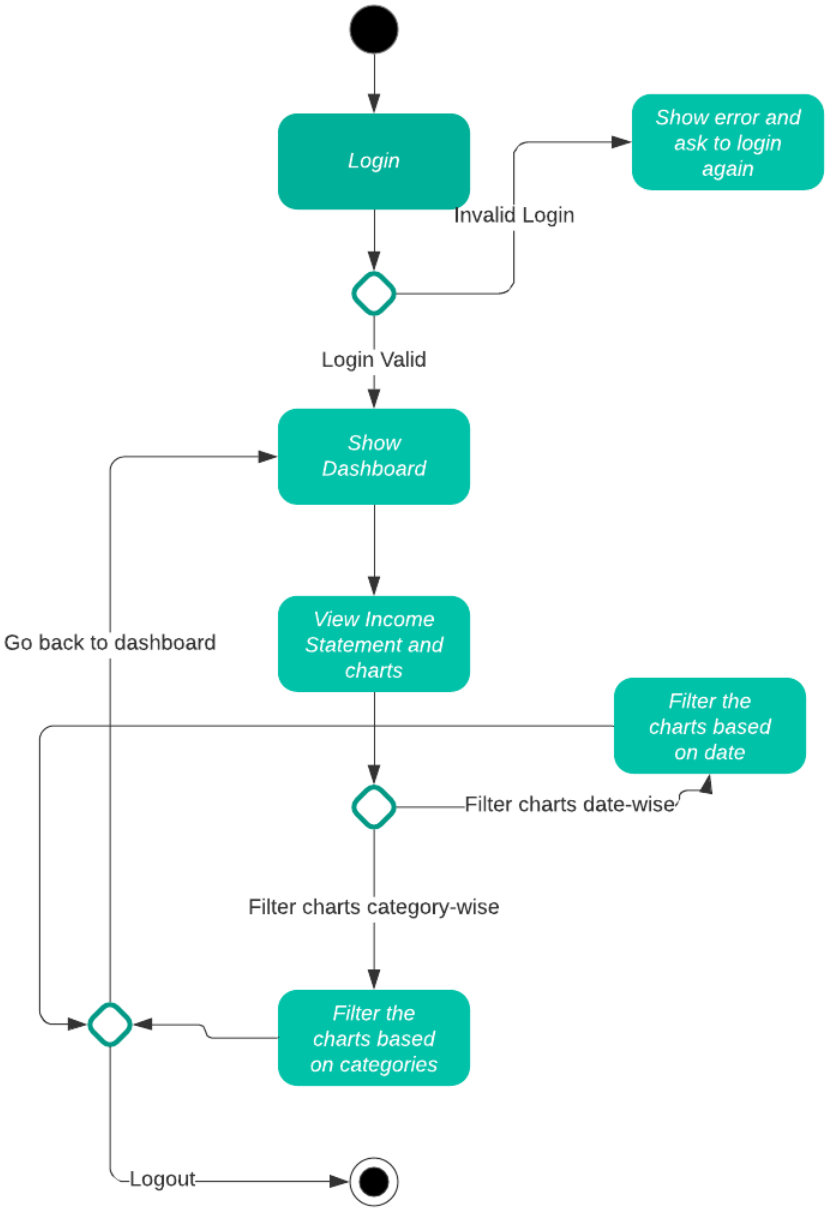
8.2 Add income Activity Diagram



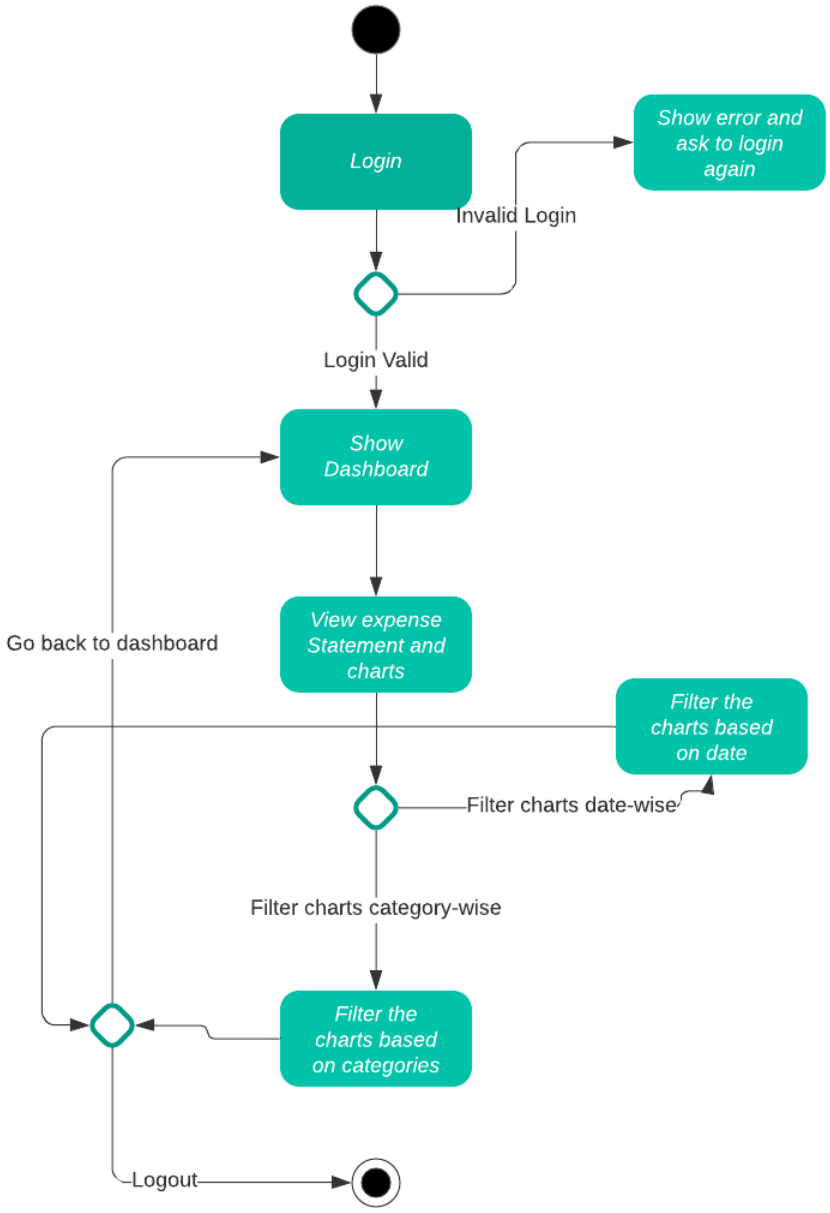
8.3 Add Expense Activity Diagram



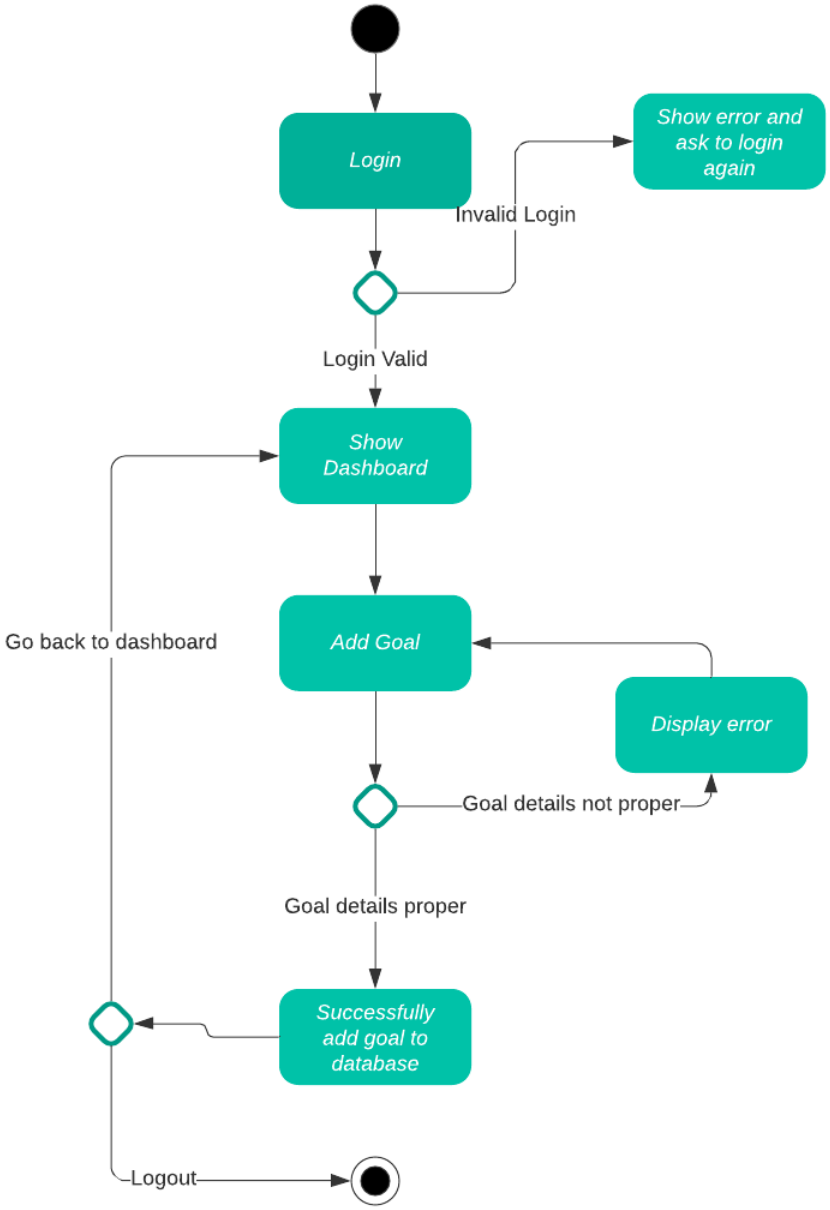
8.4 Income statement Activity Diagram



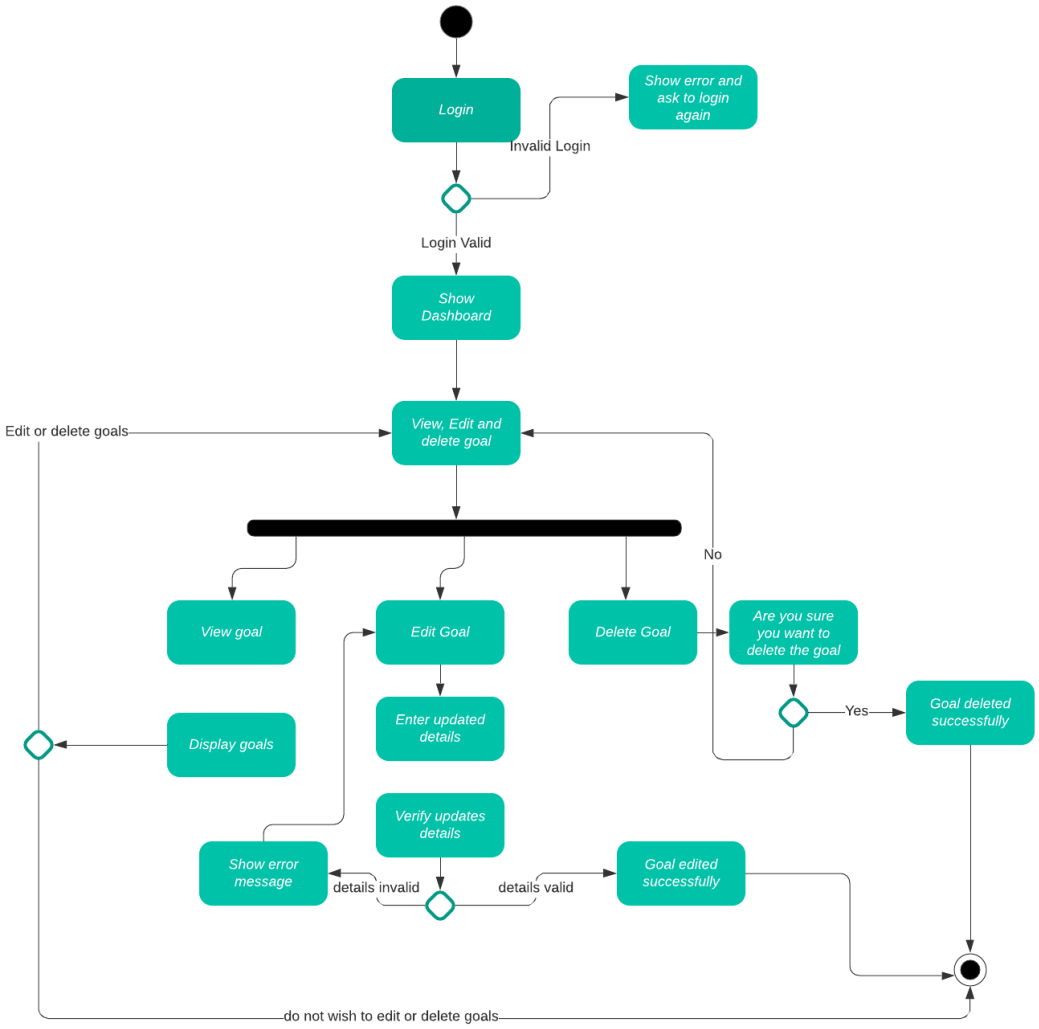
8.5 Expense Statement Activity Diagram



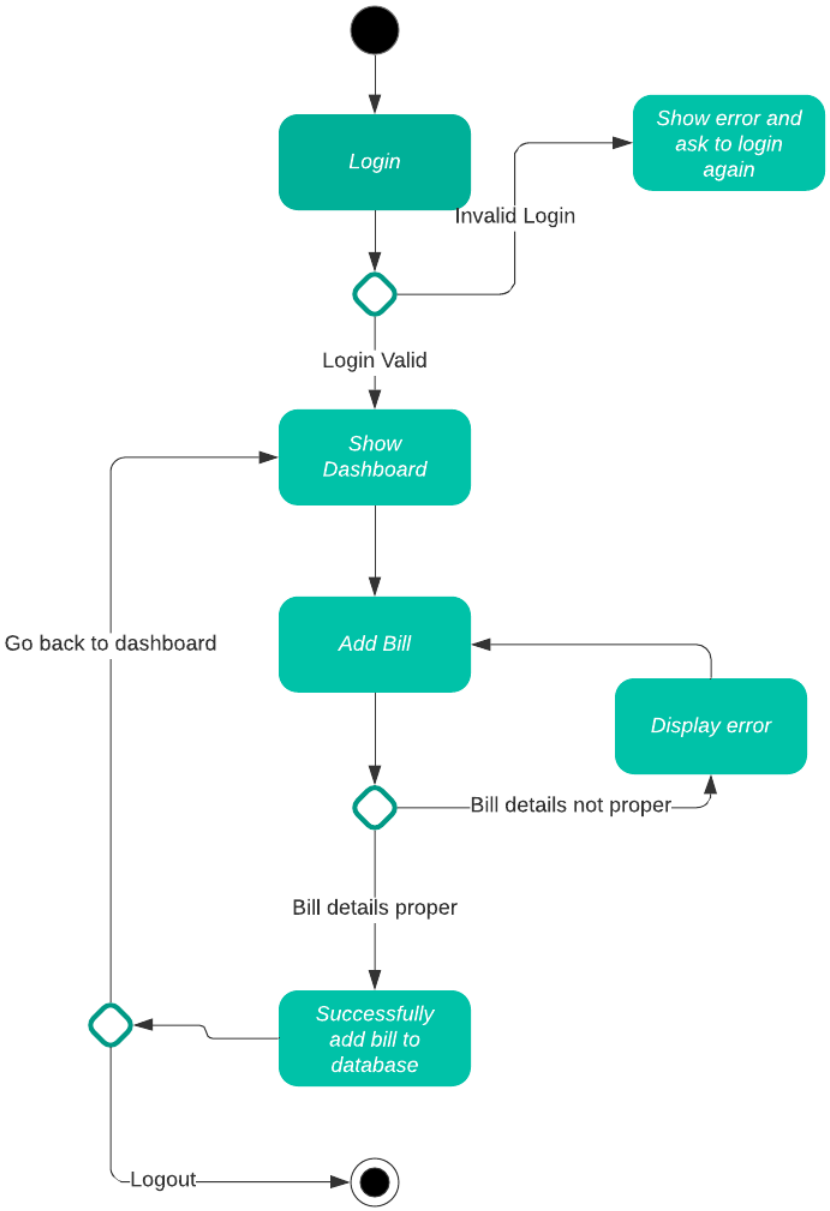
8.6 Add goal Activity Diagram



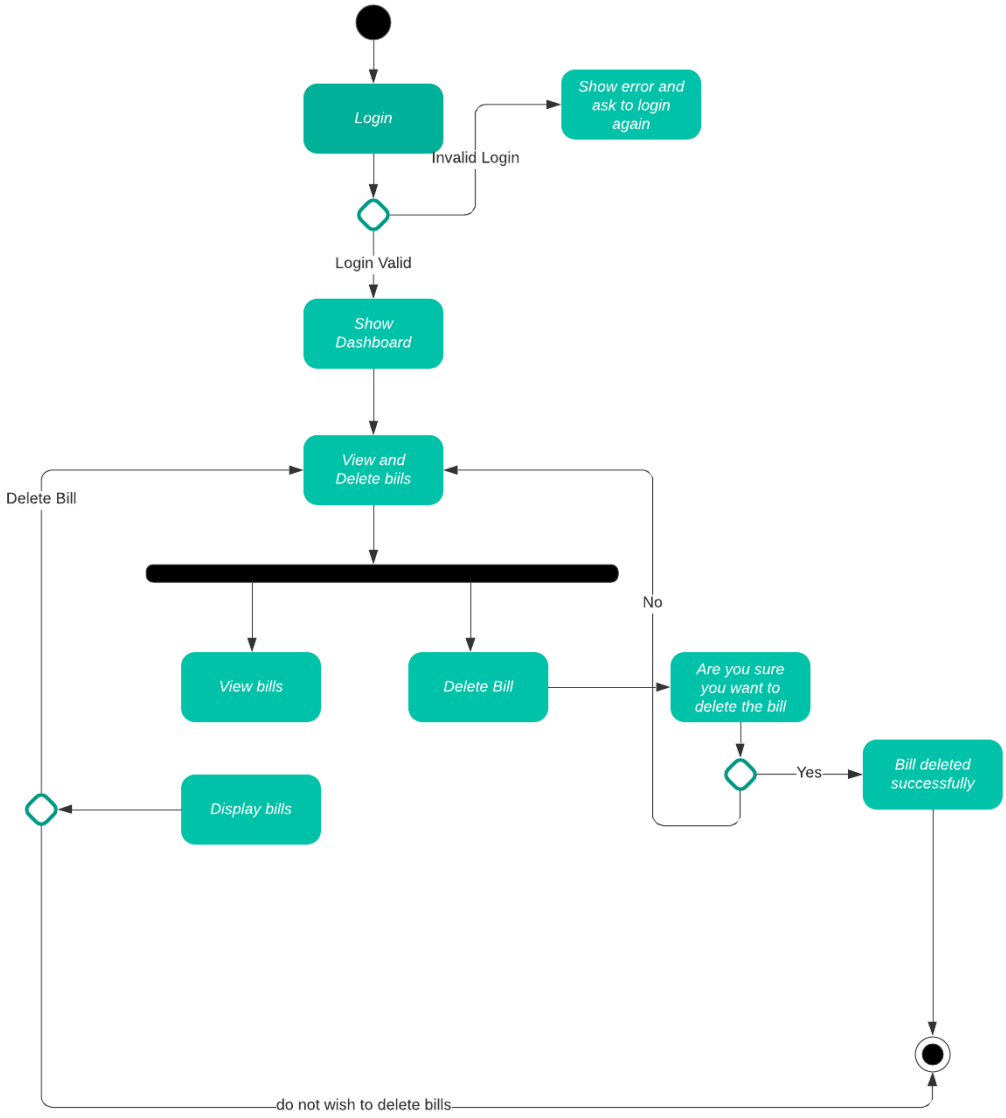
8.7 View/Edit/Delete Goals Activity Diagram



8.8 Add Bill Activity Diagram



8.9 View/Delete Bills Activity Diagram



9. Summary

- The SRS gives a summary of the system's capabilities as well as how it interacts with other systems. It also outlines the likely user interfaces so that end users and stakeholders have a clear understanding of the system's appearance. It offers high-grade definitions for the functional and non-functional specifications of the software, and can also include use cases that illustrate how a user would interact with the system upon completion. It shows how data in the system will flow with the help of a data flow diagram. It helps us see how the system would change states by providing us with the state diagram. The SRS document also shows us the relation between the various entities of the project and how they are related with each other by providing us the E-R diagram. The SRS document aids the developer in gaining a deeper understanding of the framework and serves as a solid reference point during the project's development and implementation stages. A simple set of specifications ensures that a development team produces software that meets the needs of the clients. An SRS will assist in calculating project costs and ensuring that the project scope is covered. It also helps coders schedule their work by giving them an idea of the software stack they'll need.