School of Engineering and Applied Science

Winter 2021 Semester

Software Engineering(SE)

Design 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

1.1 Purpose of this document

- In this design document we are presenting a proper description of our Finance management system design to get intuition of how our software will work and how it will work.
- This document will provide information about what tools have been used and what are the system dependencies, assumptions and the constraints.

1.2 Identification

- For backend framework we are using Django which is connected to Postgre database for our database management and there we can easily follow up on our required queries.

1.3 Document Scope

- The scope for this document is to provide necessary information in respect to the design and architecture of our project. Basic need of this document is to get insight of the major components of the quality attribute. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.4 Intended Audience

- This project will target audiences who are interested in making best use of their income and are trying to trace his/her expenses and how fast they can achieve their goals or those who are interested in managing their goals with respect to their needs.
- Also the audiences can manage their expenses according to the worthy or unworthy percentage. They can give review to their expense based on their experience and can label it as a worthy expenditure or unworthy expenditure.
- Also the user can add his bills that he has paid or he needs to pay in nearby time and they can also trace which bills need to be paid in particular time.

1.5 Key Stakeholder

- Employees working in any company: All the employees who are getting paid as monthly salary or annual salary and want to manage that income such that important expenditures are given priority.
- For startup purpose: Startups generally have very few members and they can easily manage their project expenses and incomes in a worthy manner.
- Students who are earning in their school/colleges: In premature stage they can easily manage their income.

1.6 Language and Tools

- **LANGUAGE**
  - Html
  - Css/Scss
  - Javascript
  - Django
  - PostgreSQL
- **TOOLS**
  - Lucidchart
  - draw.io
  - Bootstrap
  - pgAdmin
  - Canvas
  - Creatly

2 GENERAL OVERVIEW AND DESIGN CONSIDERATION

2.1 General Overview

- For an individual, in daily life, it is very difficult to maintain a track of his/her income and expenses. Hence, our aim is to provide a rich user-friendly interface software to a customer through which he might be able to maintain records of his/her income, expenses, view graphs, plan a strategy to save for future, and many such exciting things. Financial management is an important part of financial planning for any particular user. When managing personal finances, having a clear objective and knowledge about one's finances is an essential part of creating a workable plan that is right for you.

2.2 Current

- The statement of a need for a financial management system is when we explain why this system should exist or what is the need of this system. Many times dealing with the income or the salary becomes a headache or one gets confused how to deal with this and what will be the right choice. This question arises mostly because one is not able to choose between need of expense and desire of expense.

2.2.1 Proposed Solution - Statement of Need

- The solution for the above statement of need can be proper management of their desires and needs and the preferences among them based on their monthly/annual income. Expenses one makes should be given a label of whether the expense is worthy or not. Also to manage their goals they can save money from their income and how long they have to wait should also be known. While managing they should know which bills they have to pay in the

upcoming days as that would also be part of their expenses.

2.3 System Assumptions

- The main assumption is that the user should not haphazardly enter the details, they should be meaningful in each and every terms he enters. There is also a feature where user can find weather his.her expense is worthy or not so he/she can make

2.4 System constraints

- The income and expenses given by the user are not verified and it will be only for their management purpose and we are not responsible for their wrong inputs. Also payment gateway for bills and expenses is not covered by our project.

3 DESIGN CONSIDERATIONS

3.1 Goals and Guidelines

3.1.1 Architecture

The proposed solution to the issue would meet all of the user's practical needs while still taking into account all of the non-functional conditions. It is expected to respond to future developments, such as the inclusion of new functions, functionalities, and use cases.

3.1.2 Development Environment

The development of the web application is expected to be consistent. Taking advantage of technical advancements, this project seeks to develop the framework using the most recent iterations of technology while maintaining backward compatibility for improved performance.

3.1.3 Ease of Use

For a successful user interface, the web application's features must be user-friendly. Since the online application is tailored to the University, it should be simple and quick to use.

3.1.4 Extensibility

The programme is designed to be extensible, allowing for the addition of additional functionality while maintaining backward compatibility.

3.2 Operational Environment

- PostgreSQL
- Git version control
- GitHub repository
- Javascript
- Django
- Html/css/Bootstrap
- Material-UI

3.3 Development Methods Contingencies

- **Scalability**
  Determine that the architecture can be horizontally scaled across many servers and territories. This ensures that as the traffic grows, you should be able to introduce and delete new servers as required by the solution.
- **Availability**
  A high-availability ecosystem should be supported by the architecture. It is important to provide continuity in the infrastructure. If several servers or an entire data centre crash, the solution would still be open.
- **Security**
  The solution design should show as little code as possible. The majority of the back-end components should be hidden. Furthermore, each system's protection can be multi-layered.
- **Extensibility**
  Without needing to think about the underlying data contracts in place, architecture must be able to switch out components, alter layers, and apply parts to the programme.
- **Separation of Responsibility**
  The system should be flexible so that each piece of code only has a limited number of responsibilities. Front-end code should not be generated by the back-end, and front-end code should not contain business logic.
- **Functionality**
  When used under defined conditions, the programme is capable of providing functions that satisfy identified and implied needs (what the software does to fulfil needs)
- **Reliability**
  Under specified conditions, the programme is capable of maintaining its level of output for a specified period of time.
- **Usability**
  When used under defined circumstances, the programme is capable of being

understood, trained, used, and appealing to the customer (the effort needed for use)

- **Efficient**
  Under defined conditions, the programme is capable of providing sufficient output in regards to the quantity of resources used.
- **Maintainability**
  The software is capable of being modified. Modifications may include corrections, improvements or adaptations of the software to changes in the environment and in the requirements and functional specifications (the effort needed to be modified)
- **Portability**
  The software is capable of being transferred from one environment to another. The environment may include organizational, hardware or software environment

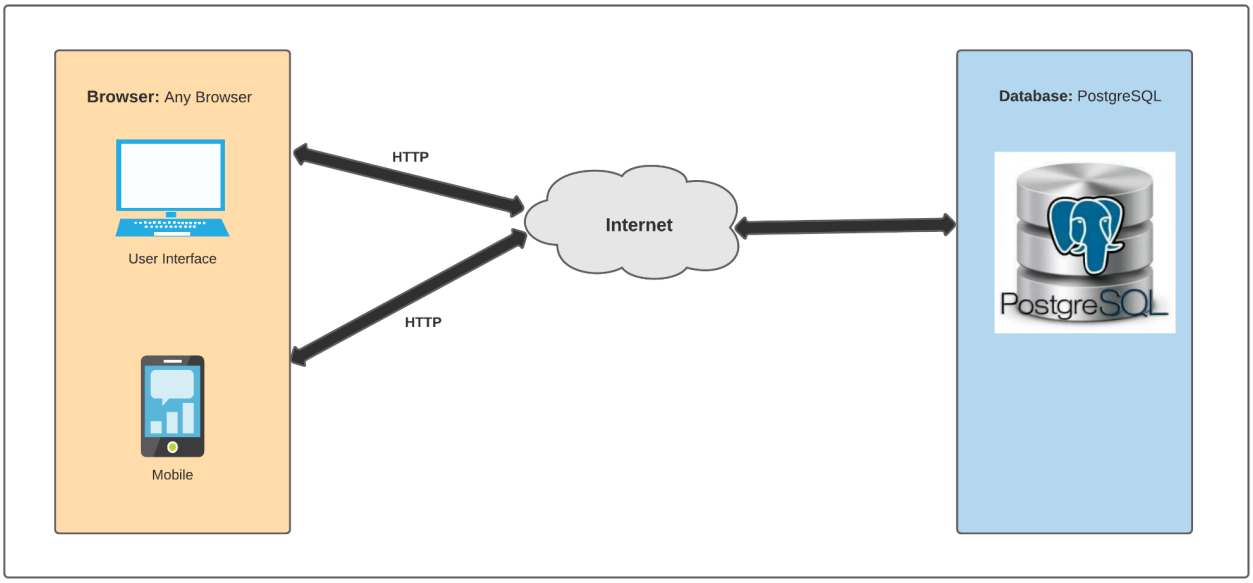
3.4 Development Environment

Software	Description
PostgreSQL	Database
Github	Version control repository
Javascript	Programing Language at server
Html/javascript/Bootstrap	Programming Framework for Web
Django	Backend Framework
Material-UI	UI Framework / Theme

4 SYSTEM ARCHITECTURE AND ARCHITECTURE DESIGN

4.1 Topology Diagram

A network topology diagram depicts how a computer network's components are linked. It helps you to see how various nodes are linked and interact with one another.



4.2 System Architecture Diagrams

This section provides the conceptual view of the system and its functionality

The Major components are :

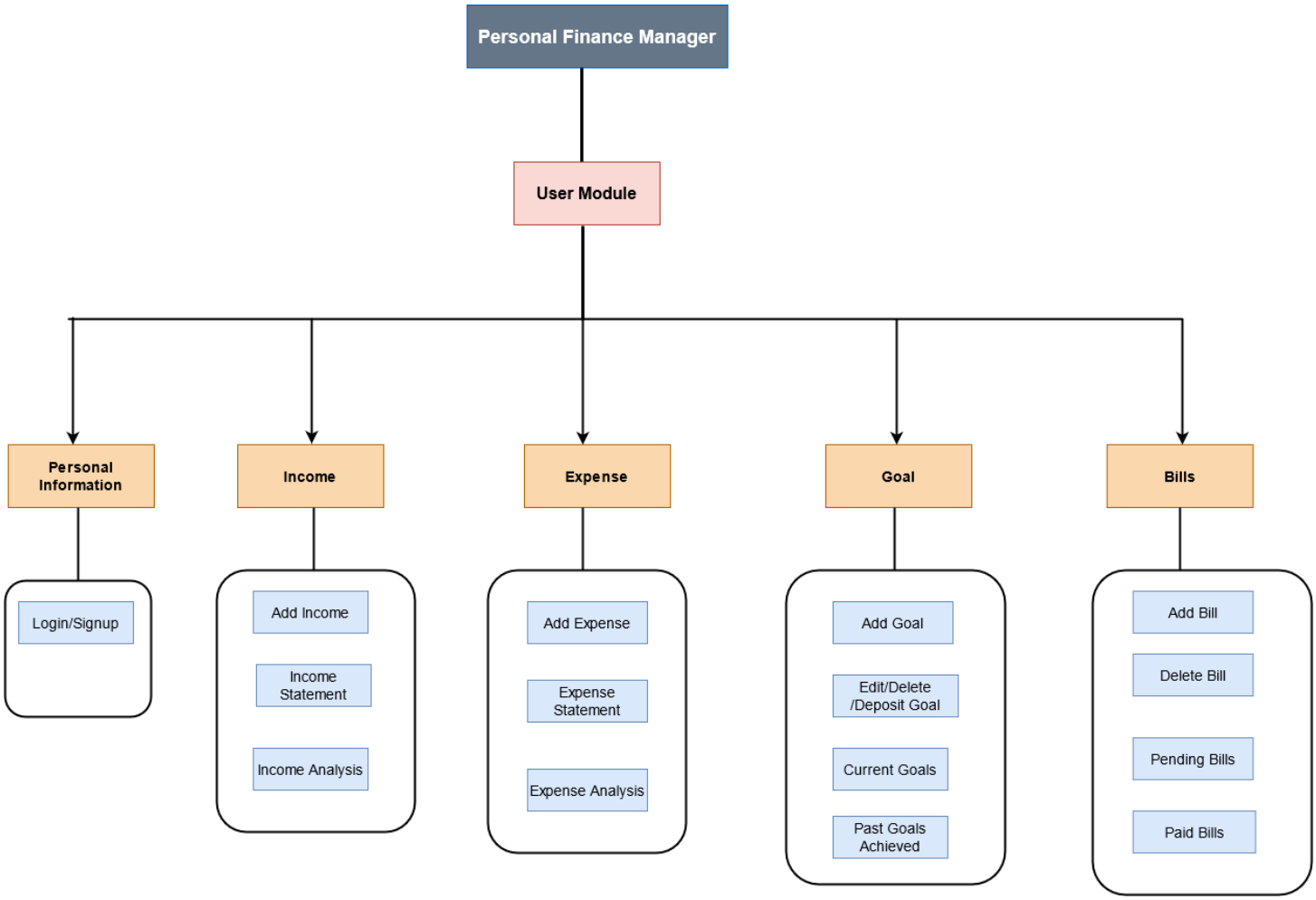
- view income
- add income
- view expense
- add expense
- view goals
- add goals
- delete goals
- modify goals
- view bills
- add bills
- delete bills

4.2.1 External Systems Diagram

The system context diagram's goal is to draw attention to external factors and activities that should be taken into account while creating a comprehensive collection of system parameters and constraints. Early in a project, system context diagrams are used to gain agreement on the scope of the project.

4.2.2 Website Architecture Diagram

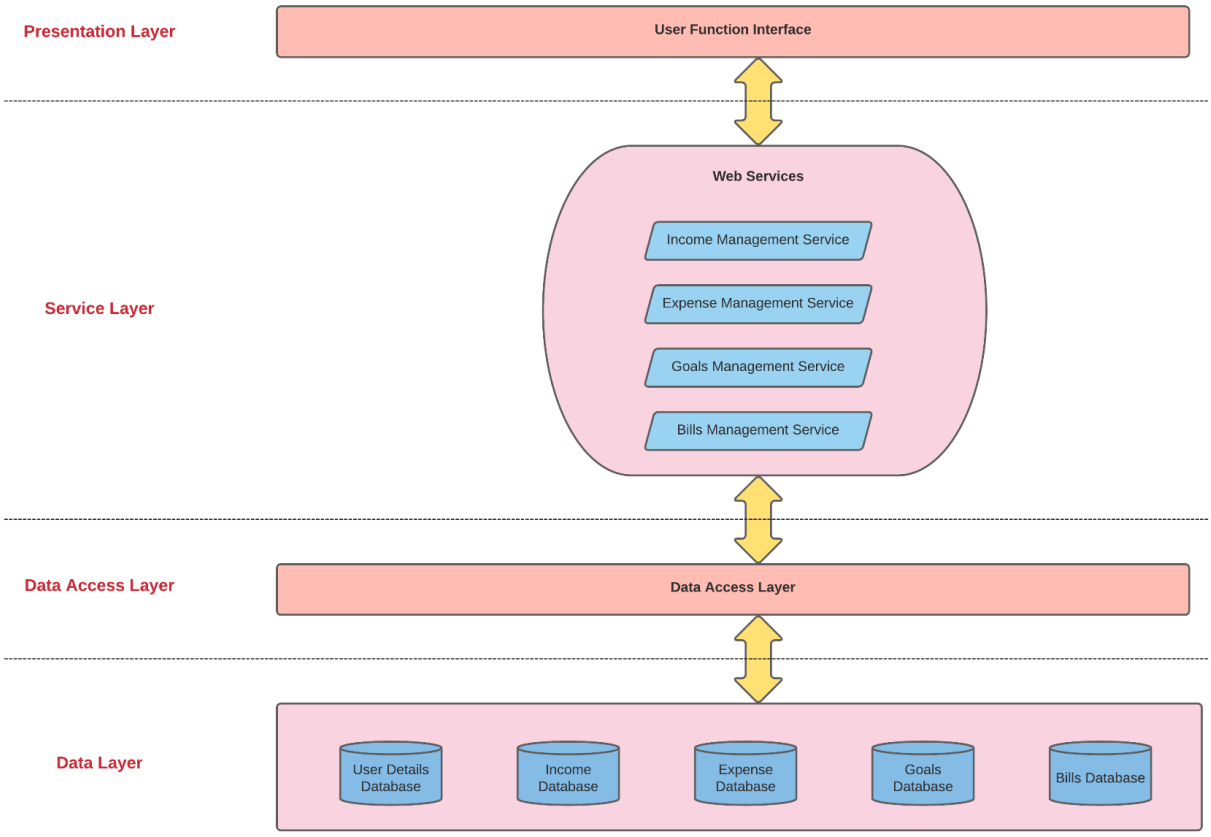
- Website information architecture uses a hierarchy structure to visualize a website's overall framework, from which you can see the directory structure of the web pages and organization of website content.



4.3 Software Architecture

4.3.1 Software Architecture Diagram

- Software architecture diagrams will be helpful to visualize high level structure for developers and to understand the overall structure. This will help to get whether the developed project meets the user's needs or not.



4.3.2 Software Element

	SYSTEM	NOTE
Programming Language	JavaScript	JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.
Frontend Framework	HTML/CSS/ BOOTSTRAP	HTML is the standard markup language for creating Web pages. HTML describes the structure of a Web page. HTML consists of a series of elements. HTML elements tell the browser how to display the content.CSS and Bootstrap are free, front-end development frameworks designed to help developers build websites faster and easier
Backend Framework	DJANGO	Django is an open-source python web framework used for rapid development, pragmatic, maintainable, clean design, and secures websites. A web application framework is a toolkit of all components need for application development.
DATABASE	POSTGRESQL	PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads.It is a highly stable database management system

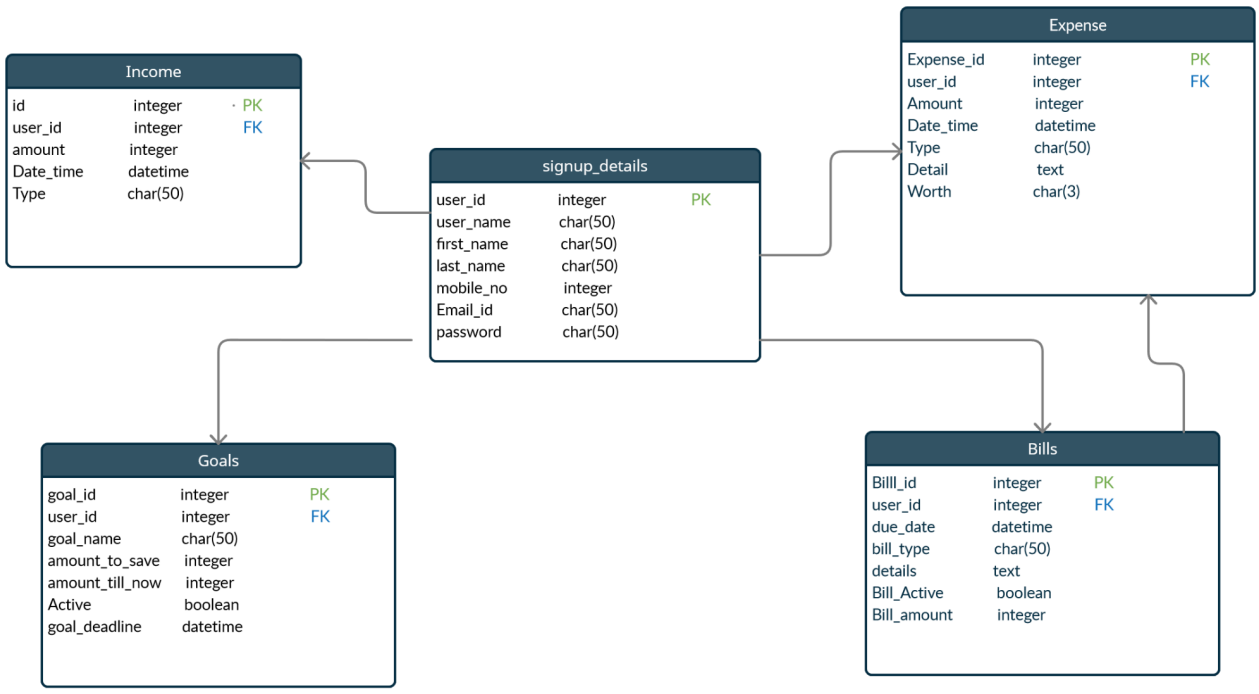
5 SYSTEM DESIGN

5.1 Business Requirements

For the proposed system the main requirements would be

- **Day to Day Expenses:** expenses would be a necessary call to use this software to keep track of earnings.
- **Monthly/Annual Income:** User should input his income as soon as he/she signs up.
- **Honest reviews towards expenses:** reviews towards each expense should be honest or else it would not make any sense to use this software.

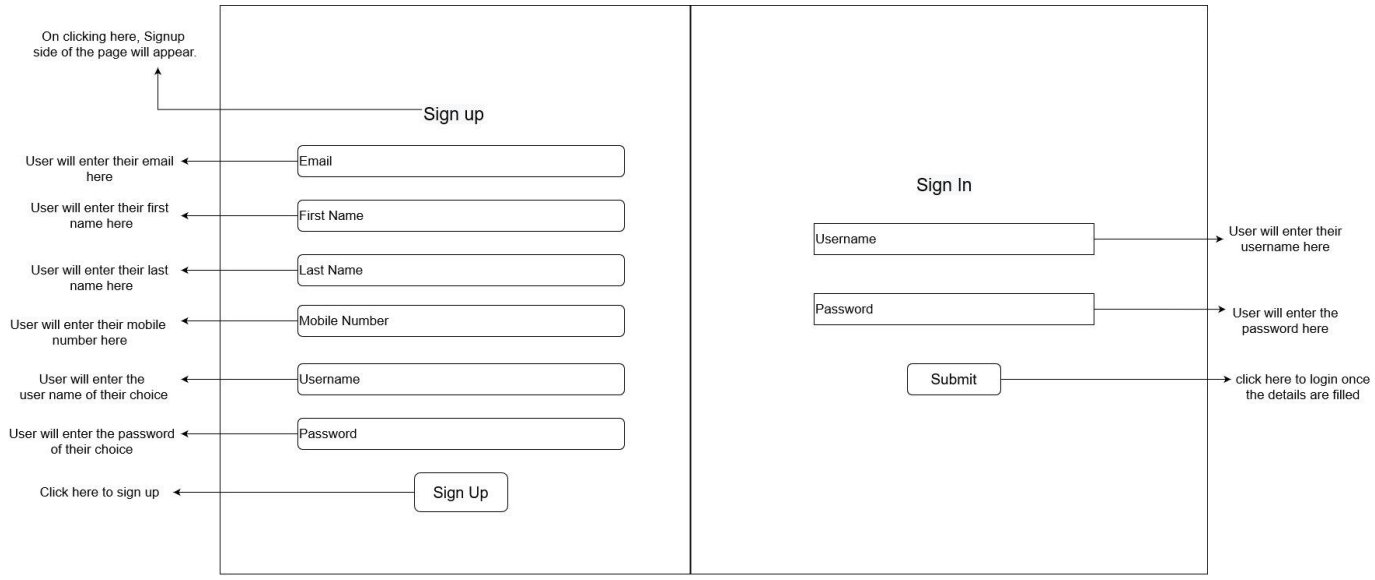
5.2 Database Design



5.3 User Interface Design

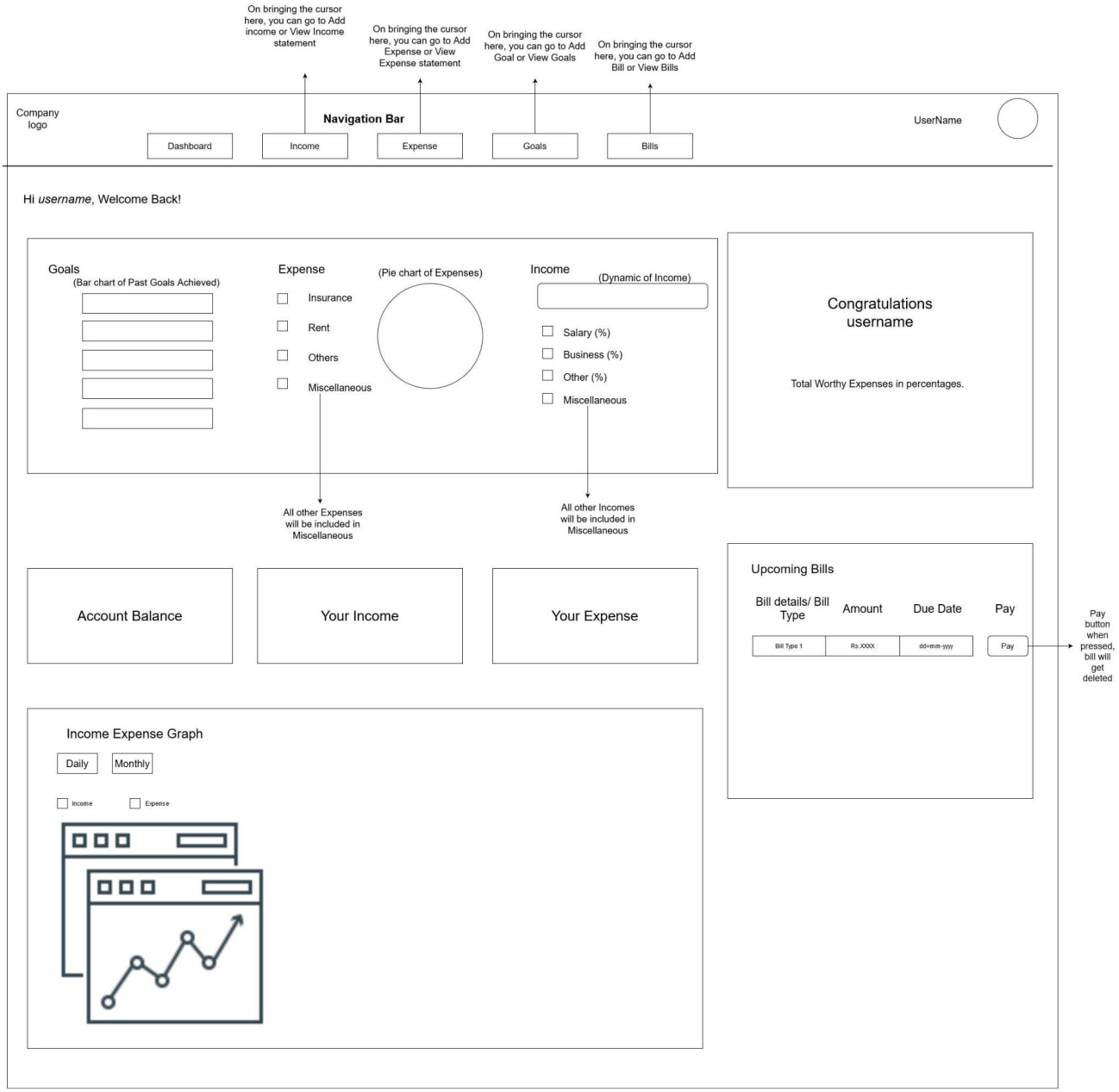
5.3.1 User Interface

➔ Sign in /Sign up page



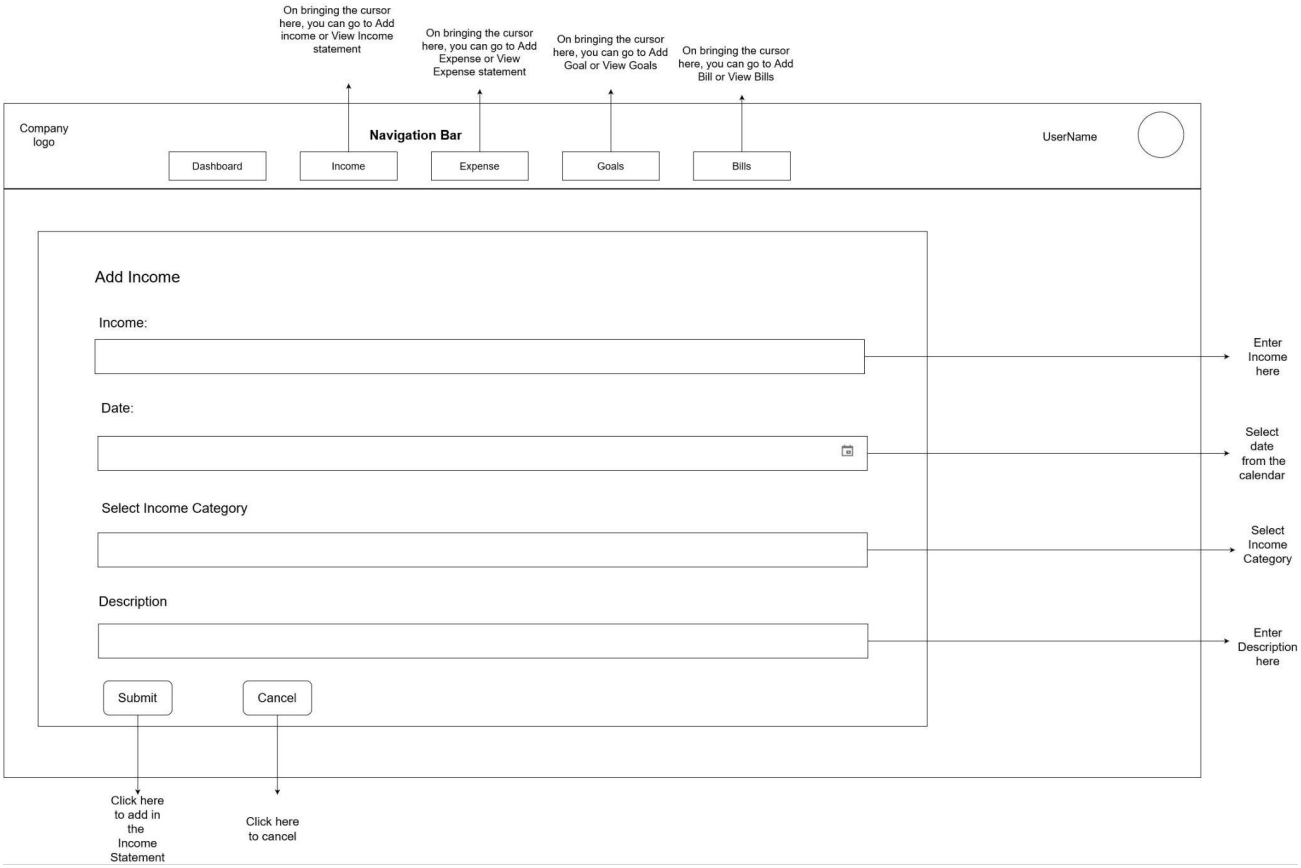
- The above image shows the signin/signup page , where any new user wants to register can register by filling the details like mail,firstname,lastname,mobile number username and password.And if any existing user wants to login can enter his registered email and password.

➔ Dashboard



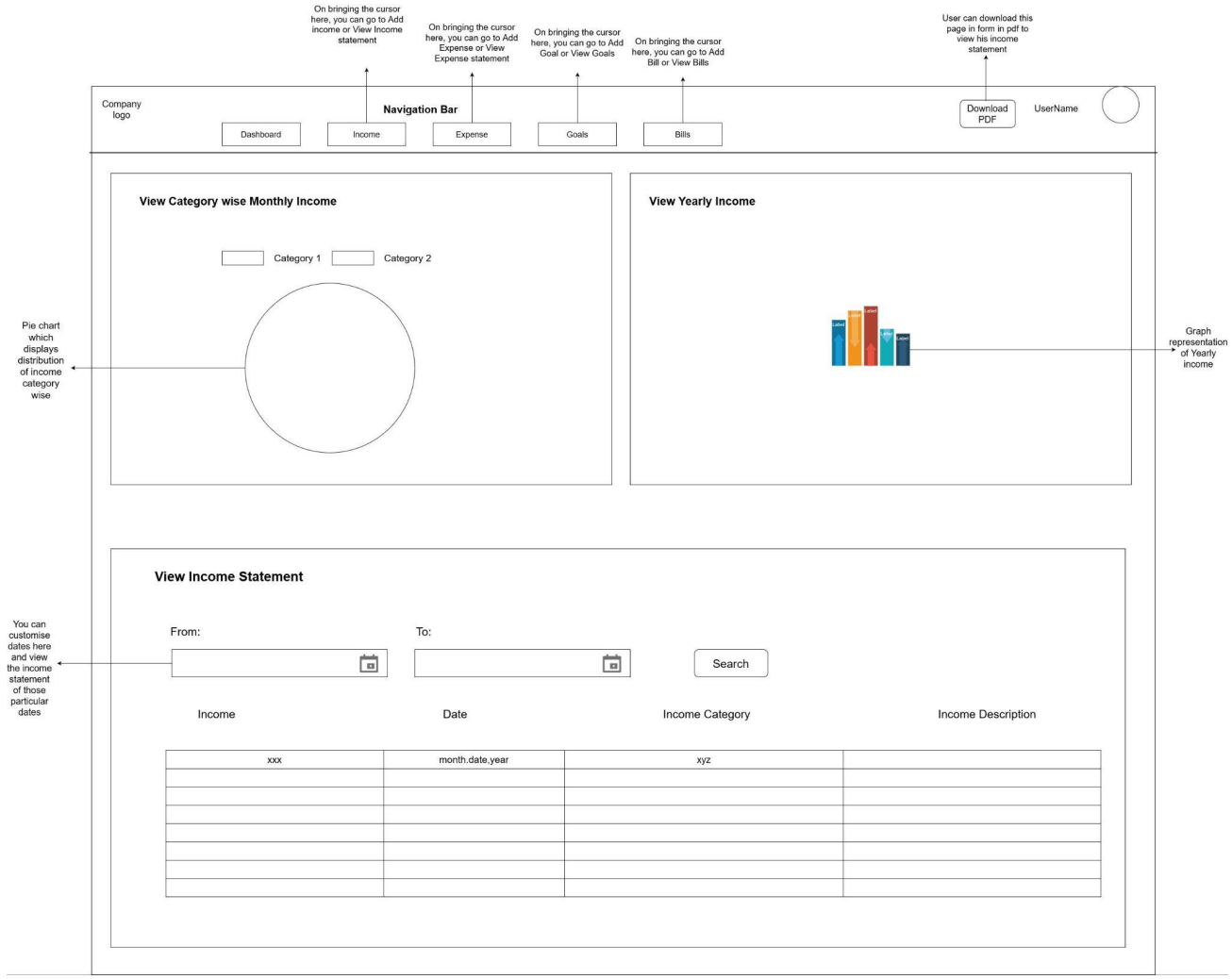
- When the user login to his account, his dashboard screen will appear.So basically the dashboard is the home page for the user.In dashboard there are various options in navbar, income,expense,bills,goals.
- At dashboard user's current account balance, income and expense amount will be shown as per his activities.
- There is also a table in which his upcoming bills will be displayed,so that he can have a glance at them and can pay directly.
- There is an income expense graph in a dashboard,where there is an option to view both income and expenses weekly and monthly.

➔ Add Income



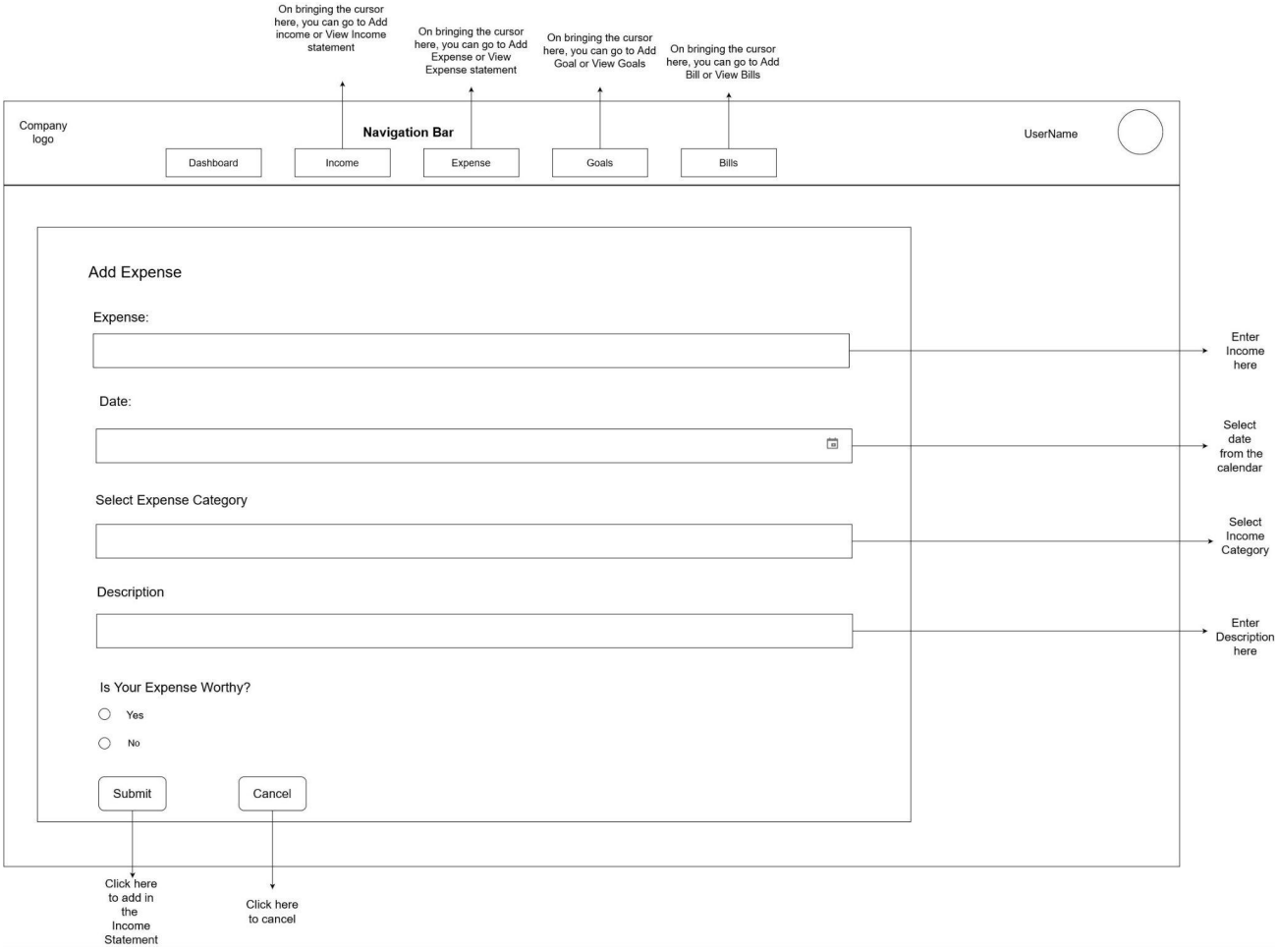
- When a user wants to add his/her income ,an income page is there so that the user can directly enter income, by filling the required fields like income amount, date,income category like salary,business income,rental income and others.. and entering the description about income, as shown in the add income interface figure above.

➔ View Income



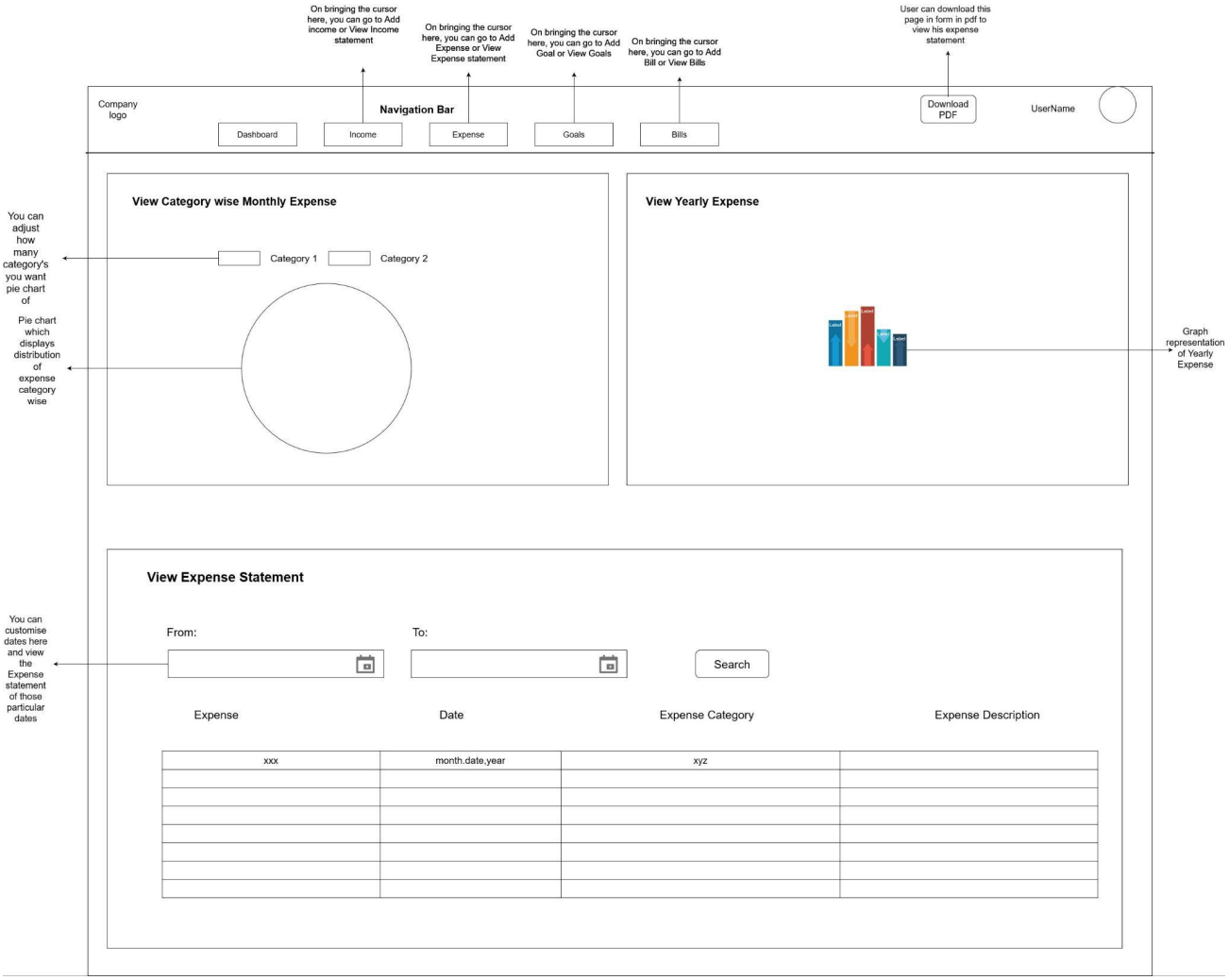
- In the view income page there , the user can examine his/her income in the form of a statement. There is a pie chart and bar chart where he can analyze it monthly and yearly,so that the user can get to know.In income statement there is an option where the user can filter from month to month to view his income.There is also an option to download the statement as pdf.

➔ Add Expense



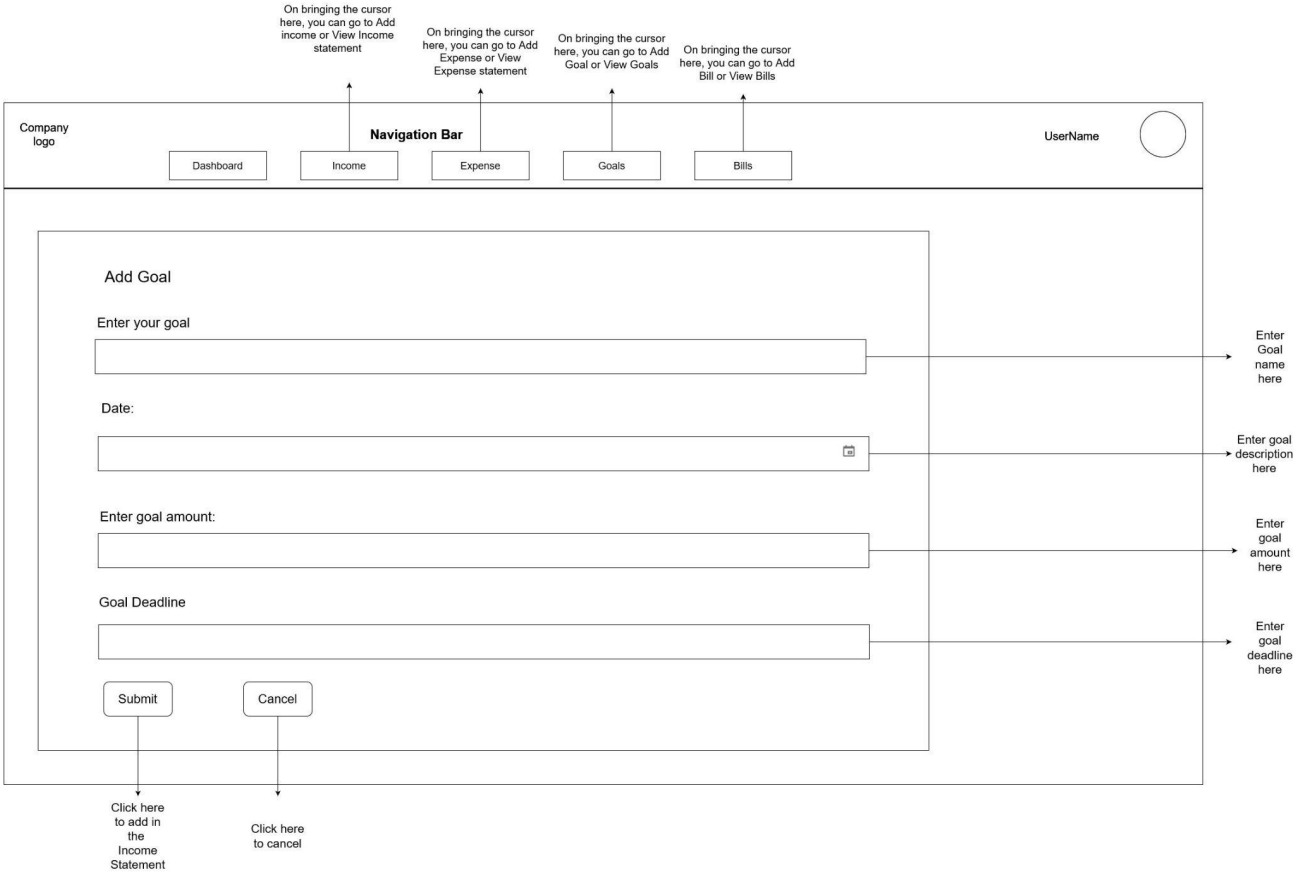
- When a user wants to add his/her expense ,an expense page is there so that the user can directly enter expenses , by filling the required fields like expense amount, date,expense category like household,travelling ,entertainment,bill payments and others.. and entering the description about expense and there is one detail in which user has to mention whether his/her expense are worthy or not , as shown in the add expense interface figure above.

➔ View Expense



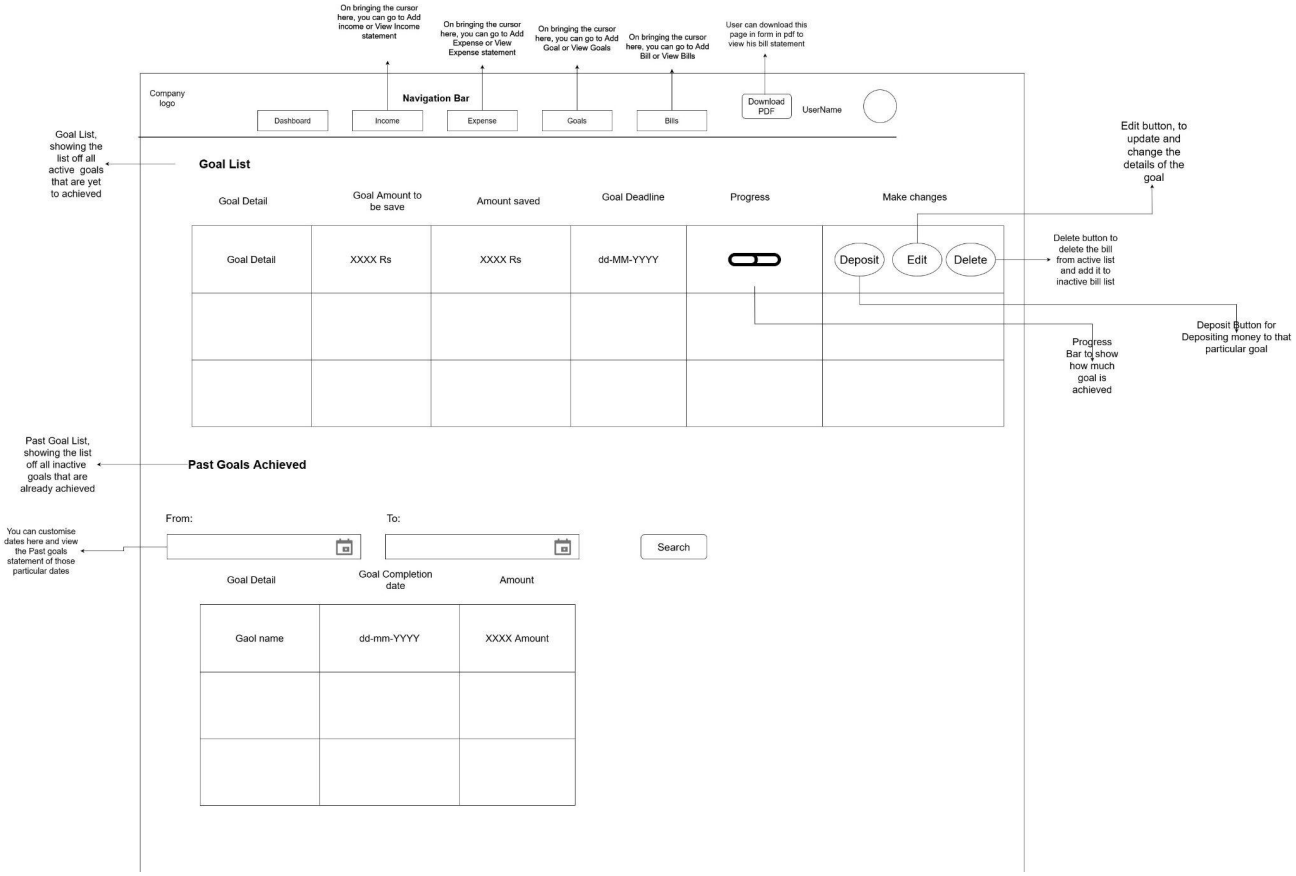
- In the view expense page , the user can examine his/her expense in the form of a statement. There is a pie chart and bar chart where he can analyze it monthly and yearly,so that the user can get to know.In income statement there is an option where the user can filter from month to month to view his expense.There is also an option to download the statement as pdf.

➔ Add Goal



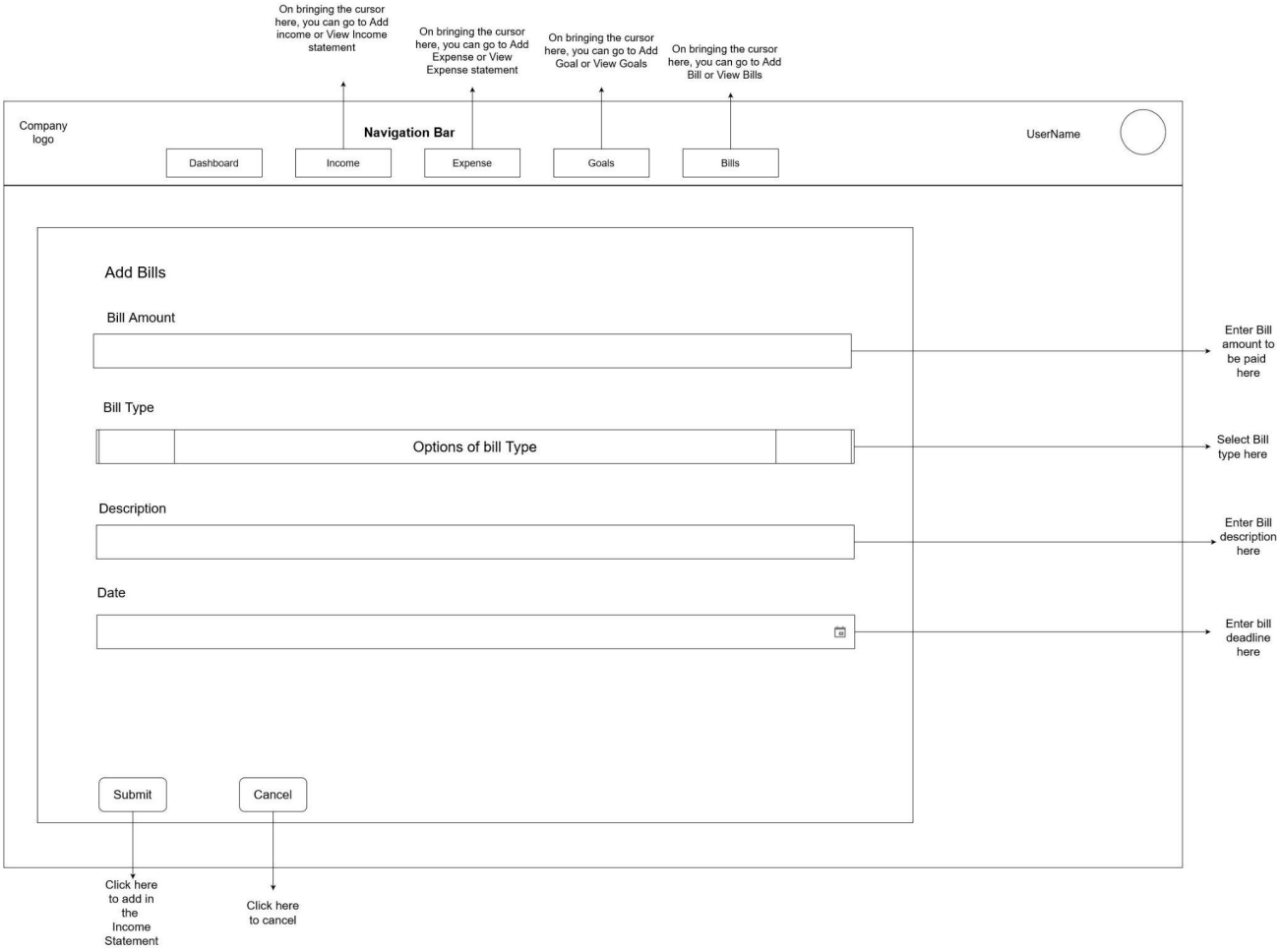
- When a user wants to add his/her expense ,an goal page is there so that the user can directly enter expenses , by filling the required fields like goal detail,goal description,goal amount,goal deadline , as shown in the add goal interface figure above.

➔ View Goal



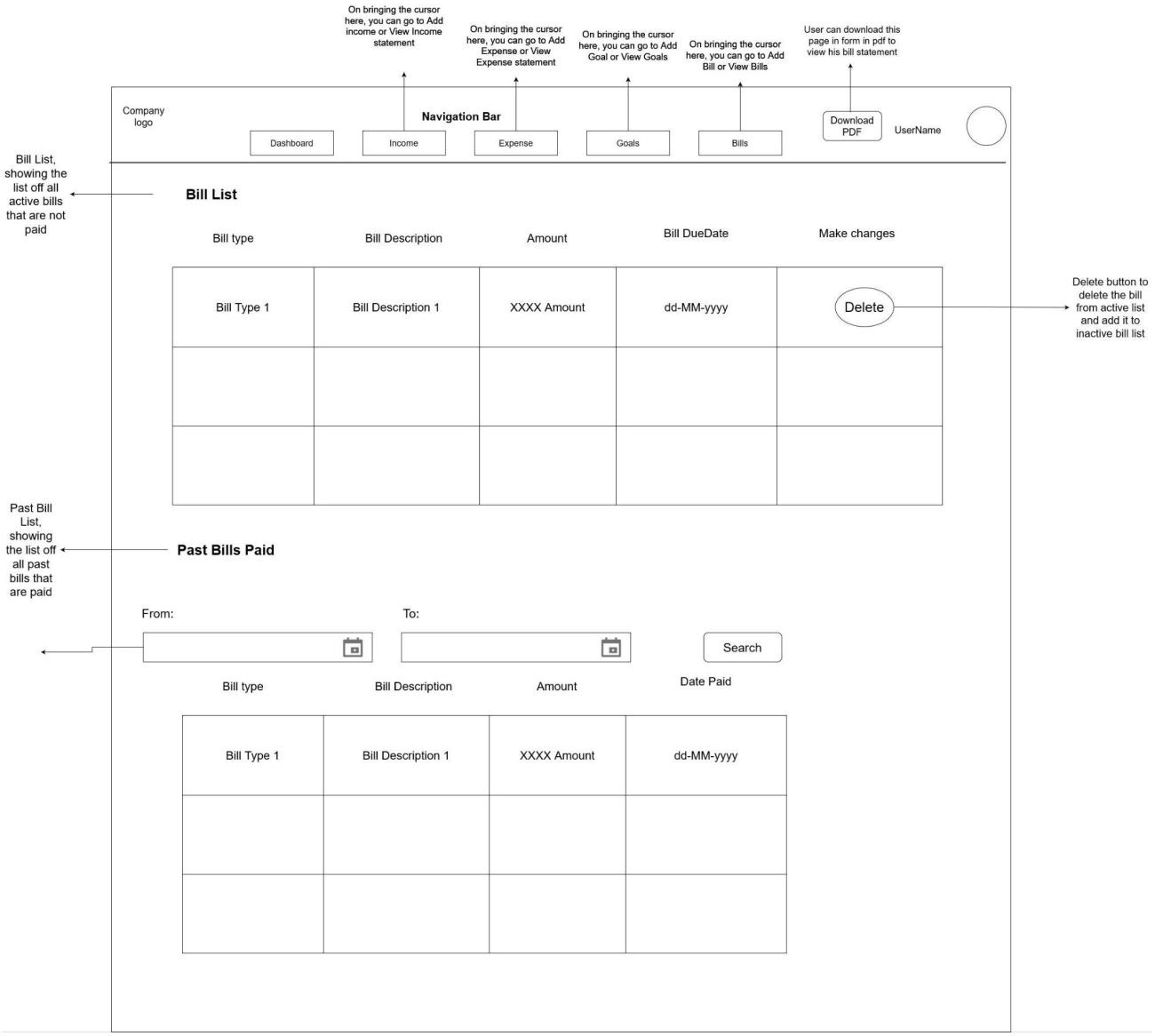
- In the view goals page ,the user can examine the goals list where there is an option if he.she wants to deposit some money edit some details and delete the particular goal, There is a table where there is a list of past goals which are achieved by the user.There is also an option to download the statement as pdf.

➔ Add Bill



- When a user wants to add his/her bill ,an add bill page is there so that the user can directly enter bills , by filling the required fields like bil, amount,bill type, description, due date as shown in add bill interface figure above.

➔ View Bill



- In the view bill page ,the user can examine the bill list where there is an option of deleting any bill of his/her choice. There is a table where there is a list of past bills which are paid by the user. and user can filter from month to month to see monthly bills.There is also an option to download the statement as pdf.