A

Seminar Report

On

"EXPENSE TRACKER SYSTEM"

Submitted In partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

In

Computer Science & Engineering

(Session 2023-2024)

Submitted to:
Ms. Madhu Choudhary

Submitted by:
Arin Sharma – 22EJCCS048

Department of Computer Science & Engineering
Jaipur Engineering College & Research Centre, Jaipur
Rajasthan Technical University, Kota

CANDIDATE'S DECLARATION

I hereby declare that the report entitled **"EXPENSE TRACKER SYSTEM"** has been carried out and submitted by the undersigned to the **Jaipur Engineering College & Research Centre, Jaipur (Raj.)** in an original work, conducted under the guidance and supervision of **Ms. Madhu Choudhary.**

The empirical findings in this report are based on the data, which has been collected by me. I have not reproduced from any report of the University neither of this year nor of any previous year.

I understand that any such reproducing from an original work by another is liable to be punished in a way the College authorities deed fit.

Date: 2023 **ARIN SHARMA – 22EJCCS048**
Place: Jaipur

VISION OF CSE DEPARTMENT

To become renowned Centre of excellence in computer science and engineering and make competent engineers & professionals with high ethical values prepared for lifelong learning.

MISSION OF CSE DEPARTMENT

1. To impart outcome-based education for emerging technologies in the field of computer science and engineering.
2. To provide opportunities for interaction between academia and industry.
3. To provide platform for lifelong learning by accepting the change in technologies
4. To develop aptitude of fulfilling social responsibilities.

PROGRAM EDUCATIONAL OUTCOMES

1. To provide students with the fundamentals of Engineering Sciences with more emphasis in Computer Science &Engineering by way of analyzing and exploiting engineering challenges.
2. To train students with good scientific and engineering knowledge so as to comprehend, analyze, design, and create novel products and solutions for the real-life problems.
3. To inculcate professional and ethical attitude, effective communication skills, teamwork skills, multidisciplinary approach, entrepreneurial thinking and an ability to relate engineering issues with social issues.
4. To provide students with an academic environment aware of excellence, leadership, written ethical codes and guidelines, and the self-motivated life-long learning needed for a successful professional career.
5. To prepare students to excel in Industry and Higher education by Educating Students along with High moral values and Knowledge.

PROGRAM OUTCOMES

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and Computer Science & Engineering specialization to the solution of complex Computer Science & Engineering problems.
2. Problem analysis: Identify, formulate, research literature, and analyze complex computer Science & Engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for complex Computer Science& Engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of Computer Science & Engineering experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern Computer Science& Engineering and IT tools including prediction and modeling to complex computer science engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional Computer Science & Engineering practice.
7. Environment and sustainability: Understand the impact of the professional Computer Science & Engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the Computer Science & Engineering practice.
9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings in Computer Science & Engineering.

10. Communication: Communicate effectively on complex Computer Science & Engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the Computer Science & Engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of Computer Science& Engineering change.

COURSE OUTCOMES

Graduates would be able:

1. To understand Software Requirement Analysis, CASE Tools, Software Testing, and other configuration tools.
2. To understand Functional Modeling (DFD), Data Modeling (DFD) - Use work products – data dictionary.
3. An ability to understand the Structural and Behavioral UML Diagrams with the use of Project Management Tool – Project Libre.

Contents

1.	**INTRODUCTION**	**3**
1.1	Purpose of the document	3
1.2	Scope of the Development Project	3
1.3	Intended Audience & Document Overview	5
1.4	Definitions, Acronyms and Abbreviations	5
1.5	References and Acknowledgments	5
2.	**Overall Description**	**6**
2.1	System Description	6
2.2	Project Perspective	6
2.3	Project Functions	7
2.4	Project Features	7
2.5	Design and Implementation Constraints	8
2.6	Assumptions and Dependencies	8
3.	**Specific Requirements**	**9**
3.1	External Interface Requirements	9
3.2	Functional Requirements	10
3.3	Use Case Diagram	11
4.	**Other Non-Functional Requirements**	**12**
5.	**Diagrams**	**13**
5.1	Activity Diagram	13
6.	**Data Flow Diagram**	**14**
6.1	DFD Level 0	14
6.2	DFD Level 1	14
6.3	Class Diagram	15
6.4	Collaboration Diagram	15
6.5	Sequence Diagram	16
6.6	State Chart Diagram	17
6.7	Component Diagram	17
6.8	Deployment Diagram	18
7.	**About Project and Codes**	**19**

7.1 Codes 19
7.2 Main Page 23
7.3 User Details Added 24
7.4 Expenses Shown 25
References **26**

1. Introduction

1.1 Purpose of this document

The purpose of this Software Requirements Specification (SRS) is to provide a detailed description of the requirements for the development of a Budget Tracker website. This document outlines the features, functionalities, and constraints of the system to be developed.

1.2 Scope of the Development Project

The Budget Tracker website is designed to help users manage their finances by providing a platform for tracking income, expenses, and creating budgets. The system will support user registration, secure authentication, and intuitive interfaces for managing financial data.

Developing a budget tracker can be a valuable project with a wide scope, as it addresses a common need for individuals and businesses. The scope of a budget tracker development project can vary based on factors such as the target audience, features, and platforms. Here are key components you might consider when scoping a budget tracker development project:

1. **User Authentication and Authorization:**
    - User registration and login functionality to allow users to create accounts.
    - Security measures to protect user data and ensure privacy.
2. **Dashboard and Overview:**
    - A user-friendly dashboard providing a summary of financial information.
    - Visual representations of income, expenses, and savings.
3. **Income and Expense Tracking:**
    - Input forms for users to add and categorize their income sources.
    - Input forms for users to add and categorize their expenses.

- Automation features for recurring transactions

4. **Multi-Platform Access:**

    - Web-based application and mobile apps for iOS and Android platforms.

    - Responsive design for seamless user experience across devices.

5. **Feedback and Support:**

    - User feedback mechanisms for continuous improvement.

    - Customer support channels for assistance.

6. **Accessibility and Inclusivity:**

    - Design considerations for users with disabilities.

    - Multilingual support.

1.3 Intended Audience and Document Overview

The intended audience for the Expense Tracker System includes individuals, businesses, and organizations seeking an efficient tool for monitoring, categorizing, and analyzing expenses. The system is designed for users with varying levels of financial knowledge, providing a user-friendly platform for managing expenses and promoting financial awareness. This document aims to provide a clear and detailed understanding of the requirements and expectations for the Expense Tracker System. It serves as a reference for the development team and stakeholders, fostering effective communication and ensuring alignment throughout the development process.

1.4 Definitions, Acronyms and Abbreviations

I. ETS - Expense Tracker System.

II. UI - User Interface.

III. UML – Unified Modelling Language

IV. DFD – Data Flow Diagram

1.5 References and Acknowledgment

I. Software Engineering by Ian Sommerville
II. SRS Format

2. Overall Description

2.1 System Description

The Budget Tracker website will allow users to create accounts, log in securely, and input financial transactions. The system will provide tools for categorizing transactions, setting budget goals, and generating reports to analyze spending patterns.

2.2 Project Perspective

The Expense Tracker System will function as a standalone application with the ability to integrate with external financial platforms and APIs. It will be accessible through web browsers and mobile applications. The Expense Tracker System operates within the context of providing users with a versatile and user-friendly tool for managing their financial expenditures. In terms of product perspective, the system serves as a standalone application with the capability to integrate seamlessly with external financial platforms and APIs. This integration enables users to synchronize their financial data, including transactions from bank accounts, fostering real-time updates and accuracy. The system is accessible through web browsers and mobile applications, catering to the diverse needs of individual users, small businesses, and financial advisors. By combining essential features such as user registration, expense tracking, category management, reporting, and analysis, the Expense Tracker System offers a holistic approach to financial management. The integration with external platforms enhances the system's functionality, allowing users to import data from diverse sources, thereby ensuring a comprehensive and tailored solution for efficient expense tracking and financial decision-making.

2.3 Project Functions

The Expense Tracker System is designed to streamline financial management by offering a range of essential functions tailored to meet the diverse needs of individuals, businesses, and financial advisors. Its core functions include user registration and authentication, providing a secure environment for users to create accounts and safeguard their financial data. The system facilitates meticulous expense tracking, allowing users to input, categorize, and monitor their expenditures seamlessly. With support for recurring expenses, users can efficiently manage regular bills, ensuring accuracy and timely tracking. Category management is a key feature, enabling users to create, edit, and delete predefined or custom expense categories, providing a structured organization for expenses. The reporting and analysis functionalities generate detailed reports on spending patterns, offering insights through visually appealing graphs and charts. Integration with external platforms, including APIs and bank account synchronization, enhances the system's capabilities, allowing users to import data from external sources and ensuring real-time updates. Through these comprehensive functions, the Expense Tracker System aims to empower users in making informed financial decisions and achieving effective financial management.

2.4 Product Features

User Registration and Authentication:

- Allow users to register securely and authenticate their identity.

Expense Tracking:

- Enable users to input, categorize, and track expenses.
- Support recurring expenses for user convenience.

Category Management:

- Allow users to create, edit, and delete predefined or custom expense categories.

2.5 Design and Implementation Constraints

The design and implementation of the Expense Tracker System are subject to certain constraints that influence the development process and functionality. One significant constraint is the adherence to a predefined budget and timeline. Given the resources allocated for development, the system must be designed and implemented within the specified financial constraints to ensure a cost-effective solution. Time constraints are also crucial, with a predetermined schedule dictating the development phases, testing, and deployment.

Another constraint involves the integration with external platforms and APIs. The system's functionality is dependent on the availability and compatibility of external financial tools and platforms. Changes or limitations in these external systems may impact the seamless integration and synchronization of data, requiring constant monitoring and potential adjustments.

2.6 Assumptions and Dependencies

The following list prevents the assumptions, dependencies or guidelines that are imposed upon implementation of System:

- The product must have a user friendly interface that is simple enough for all types of users to understand.
- Response time should not be longer than 5 seconds.
- A general knowledge of basic computer skills and internet is required to use the product.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interfaces of the Expense Tracker System are thoughtfully designed to provide an intuitive and seamless experience for individuals, businesses, and financial advisors. The system's user registration interface ensures a secure onboarding process, guiding users through the creation of accounts with clear and straightforward input fields for essential information. The expense tracking interface is user-friendly, allowing for the easy input and categorization of expenditures. Users can effortlessly navigate through features such as adding notes, dates, and amounts, as well as setting up recurring expenses for regular bills. The category management interface simplifies the organization of expenses, offering users the flexibility to create, edit, and delete predefined or custom expense categories. The reporting and analysis interface presents detailed reports on spending patterns, with visually appealing graphs and charts that make it easy for users to comprehend their financial data. The integration interface seamlessly connects with external platforms, including APIs and bank account synchronization, enhancing the system's capabilities and providing users with a comprehensive view of their financial landscape. In summary, the Expense Tracker System's user interfaces prioritize clarity, ease of use, and accessibility, aiming to empower users in efficiently managing their expenses.

3.1.2 Hardware Interfaces

- Computer: A computer will be required to open the website and use the software.
- Smartphone: A smartphone can also be required in case there is no availability of computer.
- Internet: A good internet connection is required to access the website.

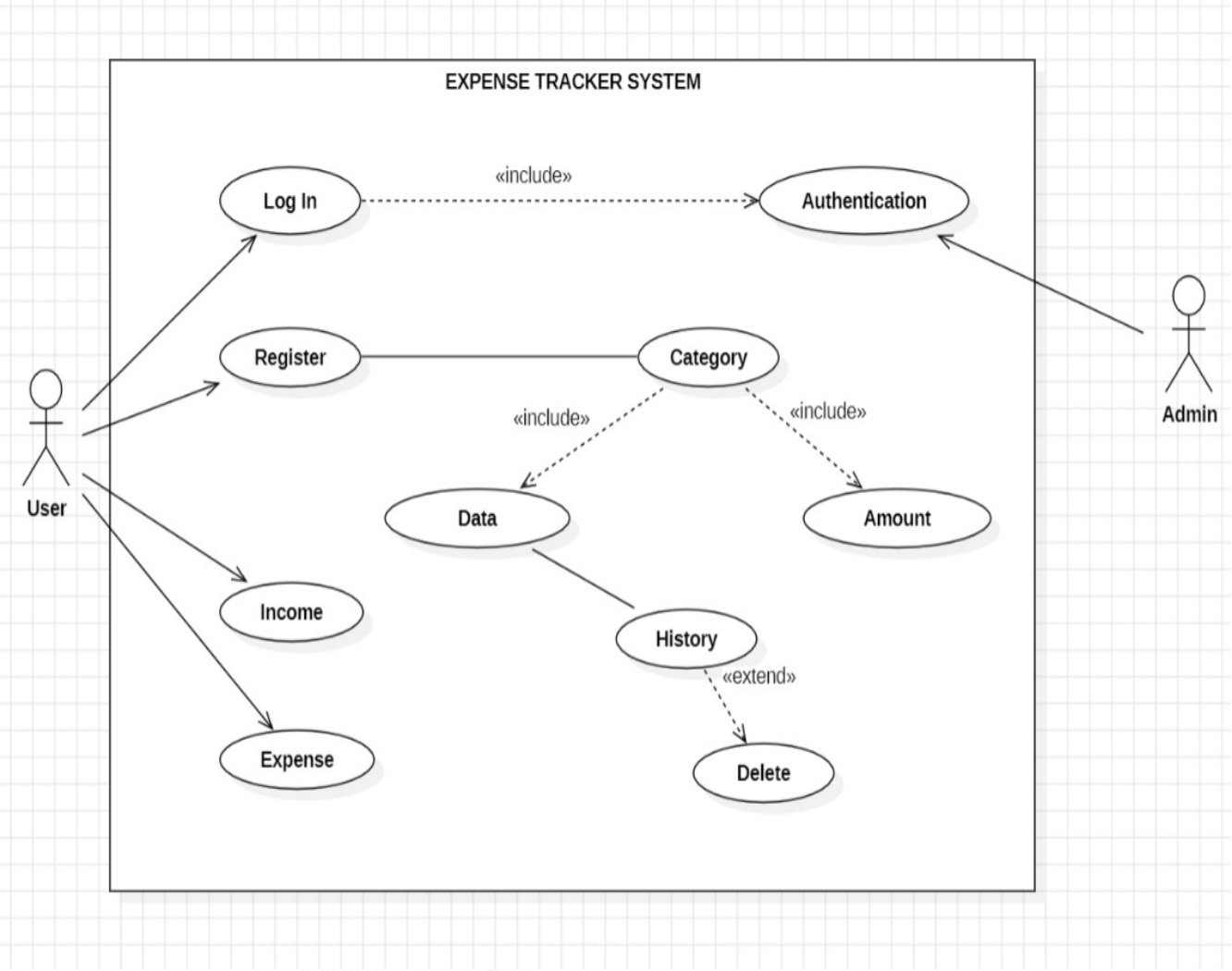
3.1.3 Software Interfaces

- A web browser will be required to open the website.

3.2 Functional Requirements

The Expense Tracker System's functional requirements encompass a suite of features designed to offer users a comprehensive and user-friendly financial management experience. The system allows for secure user registration and authentication, ensuring that users can create accounts and access their financial data with confidence. Expense tracking is a fundamental functionality, enabling users to input, categorize, and monitor their expenditures, with additional support for recurring expenses to streamline the management of regular bills. The category management feature empowers users to customize and organize expense categories, adapting the system to their specific financial needs. Detailed reporting and analysis capabilities generate comprehensive insights into spending patterns, providing a clear understanding of financial habits. Visual representation through graphs and charts enhances data comprehension. The system's integration with external platforms, facilitated by APIs and bank account synchronization, expands its utility by allowing users to import data seamlessly and ensuring real-time updates. By addressing these functional requirements, the Expense Tracker System aims to be a versatile and powerful tool for users seeking effective and intuitive financial tracking and management.

3.3 UML Use Case Diagram



4. Other Non-Functional Requirements

- **Usability**

  There is a consistency in all the modules and webpages. To ease the navigation there is a back tab to provide access to previous page. There is proper instruction on each page.

- **Reliability**

  Each data record is stored on a well-built efficient database schema. There is no risk of data loss. The internal evaluation of data is well coded.

- **Supportability**

  The system is well built to support any machine. Maintainability of the system is easy.

- **Performance**

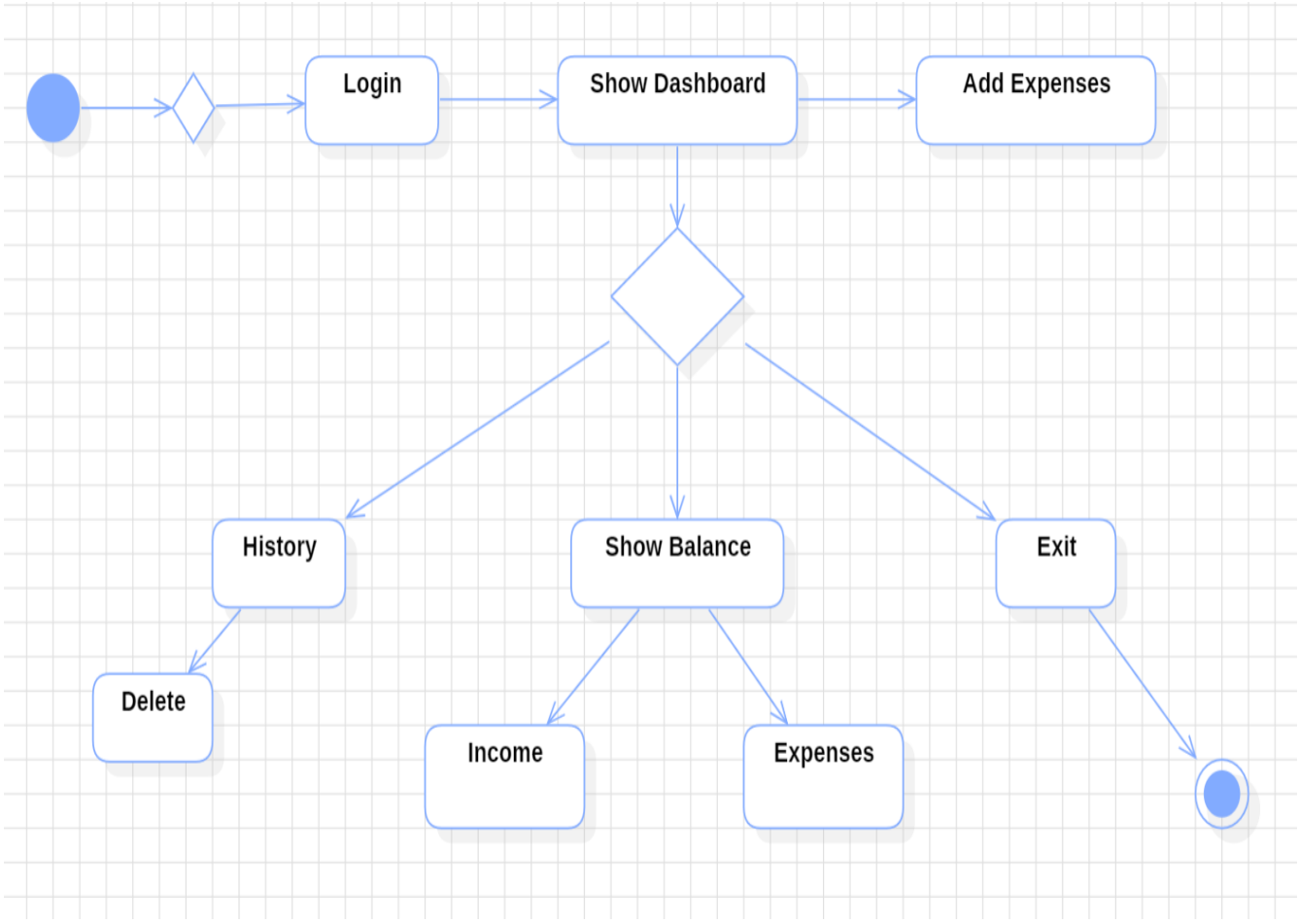
  In order to ease the accessibility, the types of expenses are categorized along with an option to name on the own. Throughput of the system is increased due to light weight database support.

- **Availability**

  The system is available all the time, no time constraint.

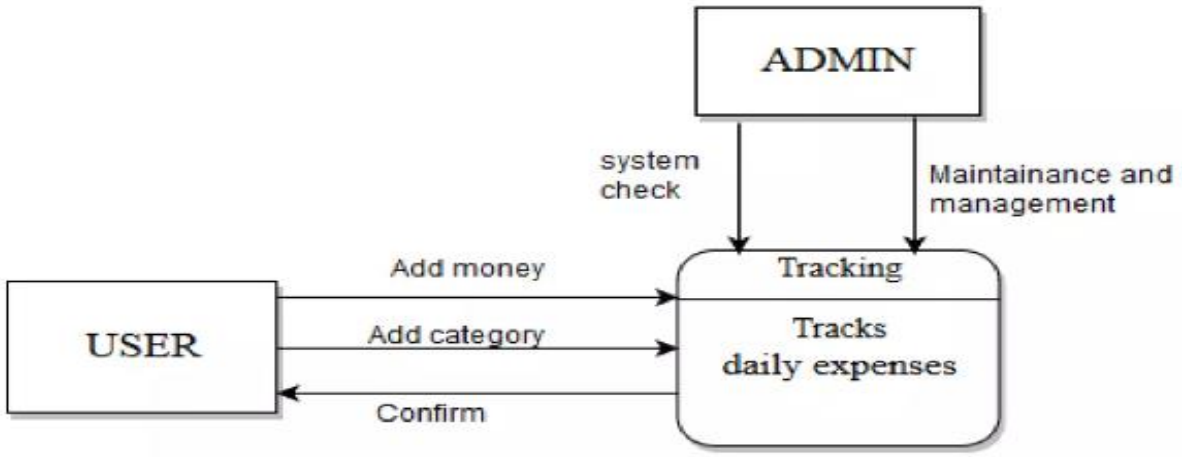
5. Diagrams

Activity Diagram

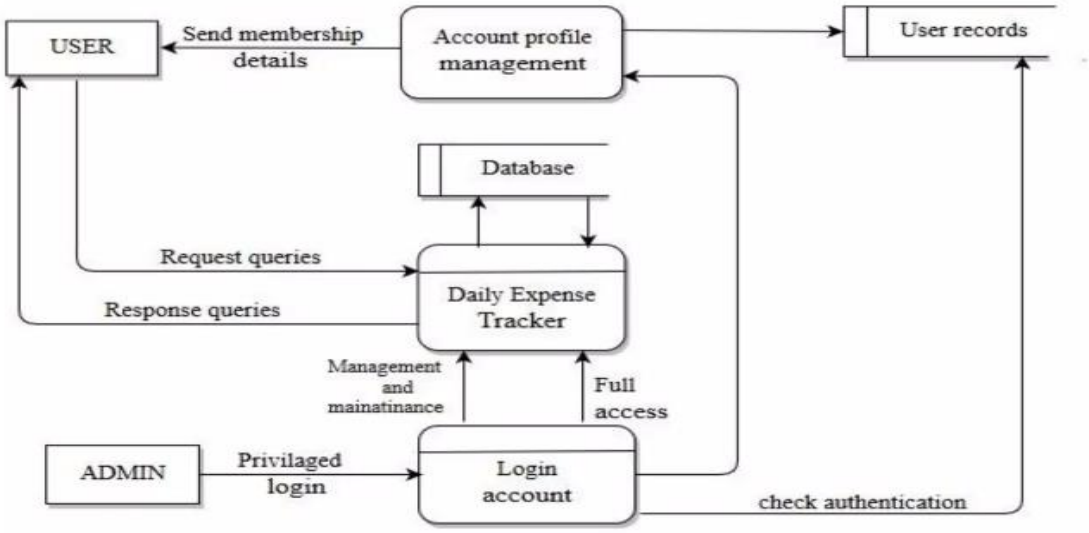


6. Data Flow Diagram

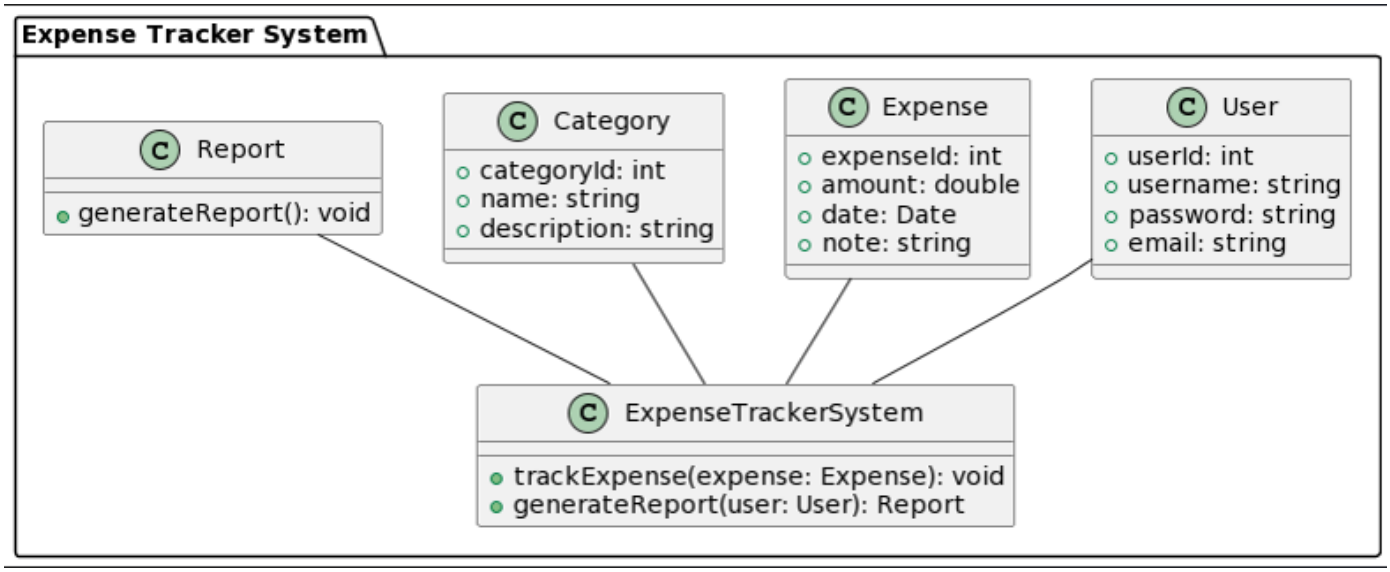
6.1 DFD Level 0



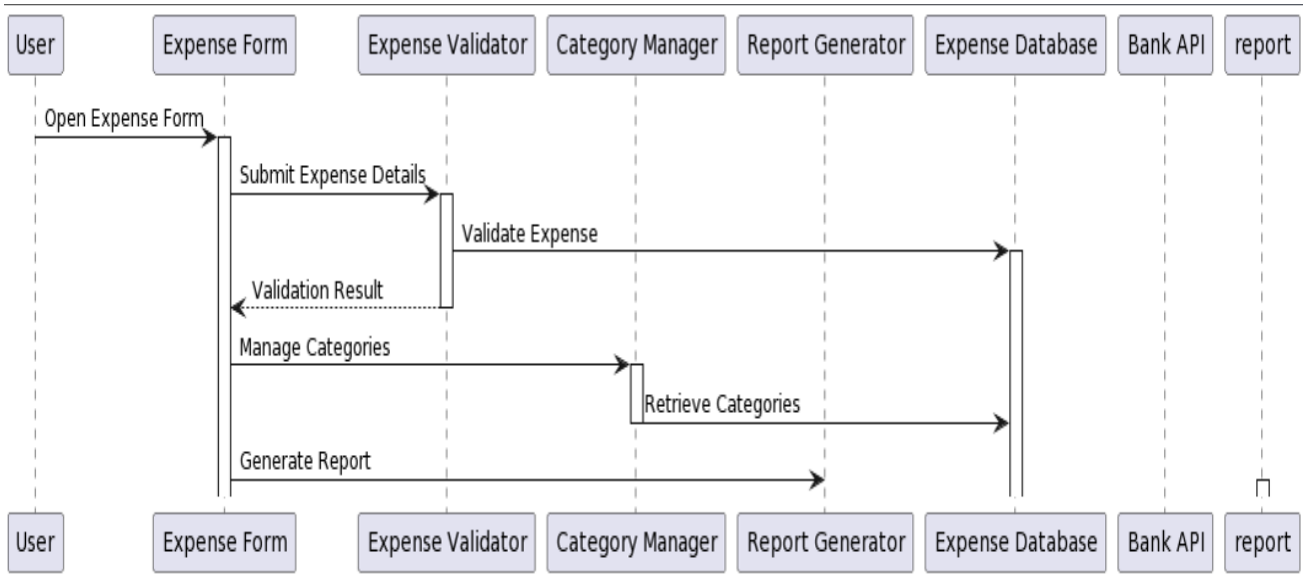
6.2 DFD Level 1



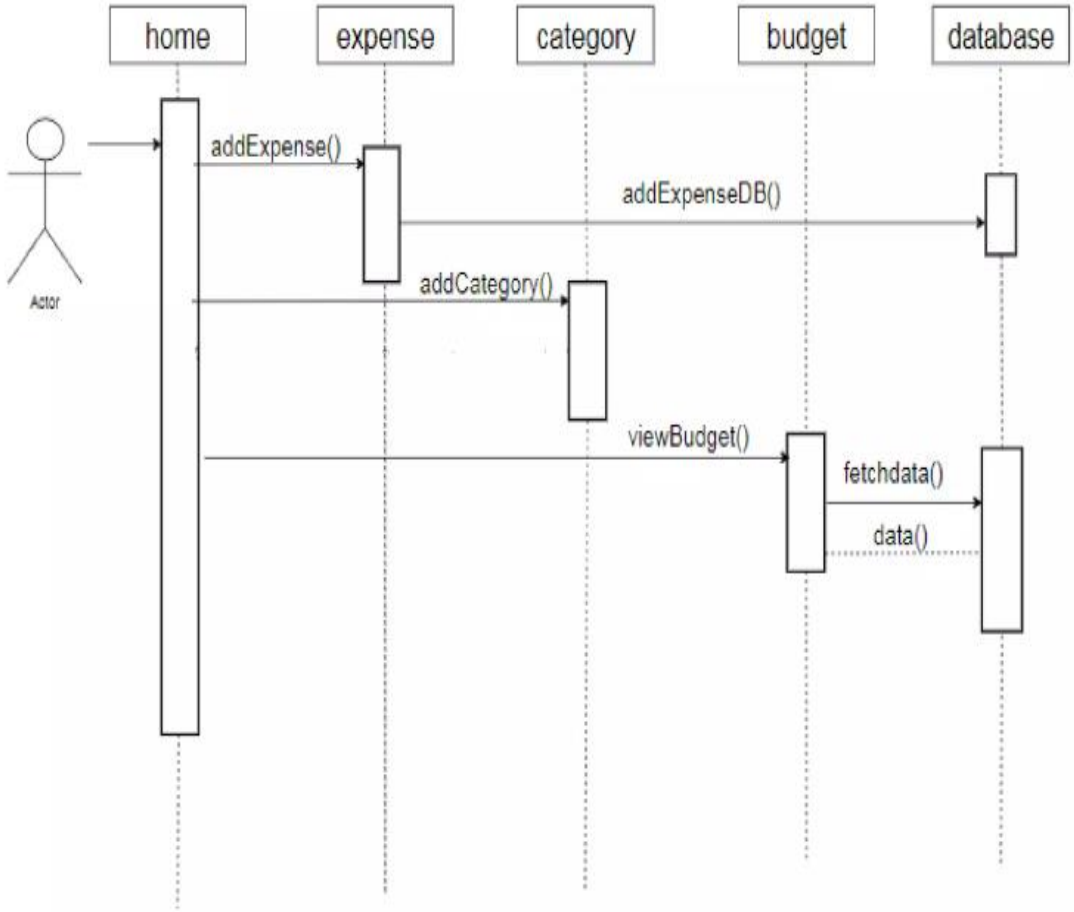
6.3 Class Diagram



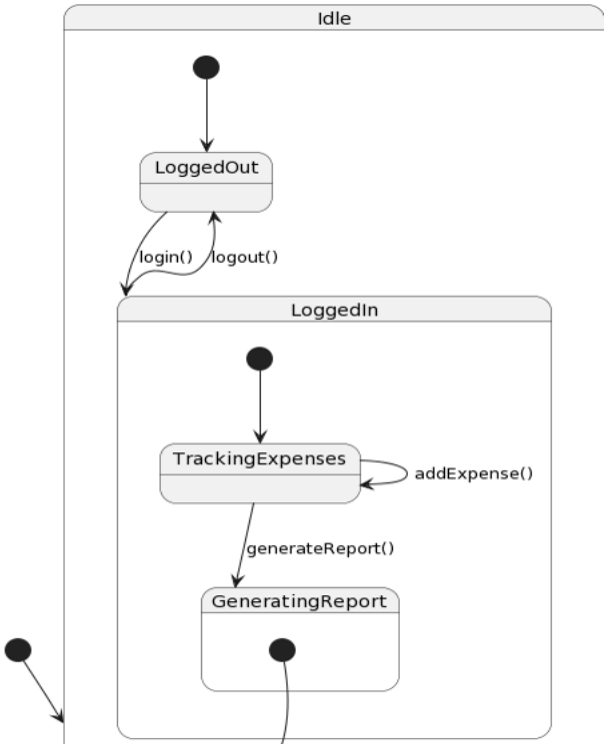
6.4 Collaboration Diagram



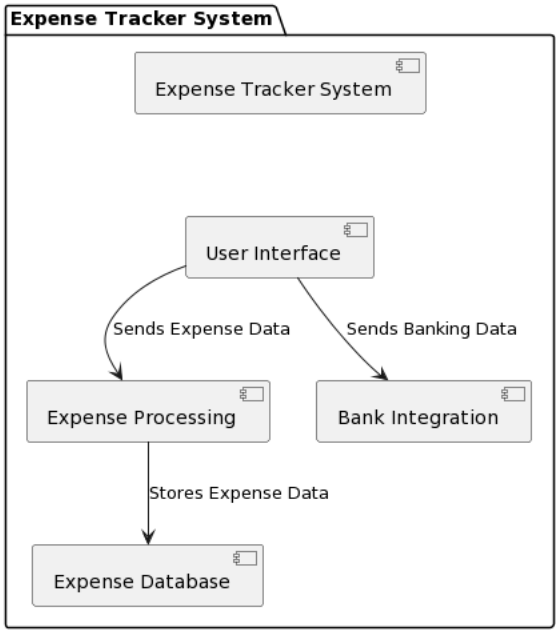
6.5 Sequence Diagram



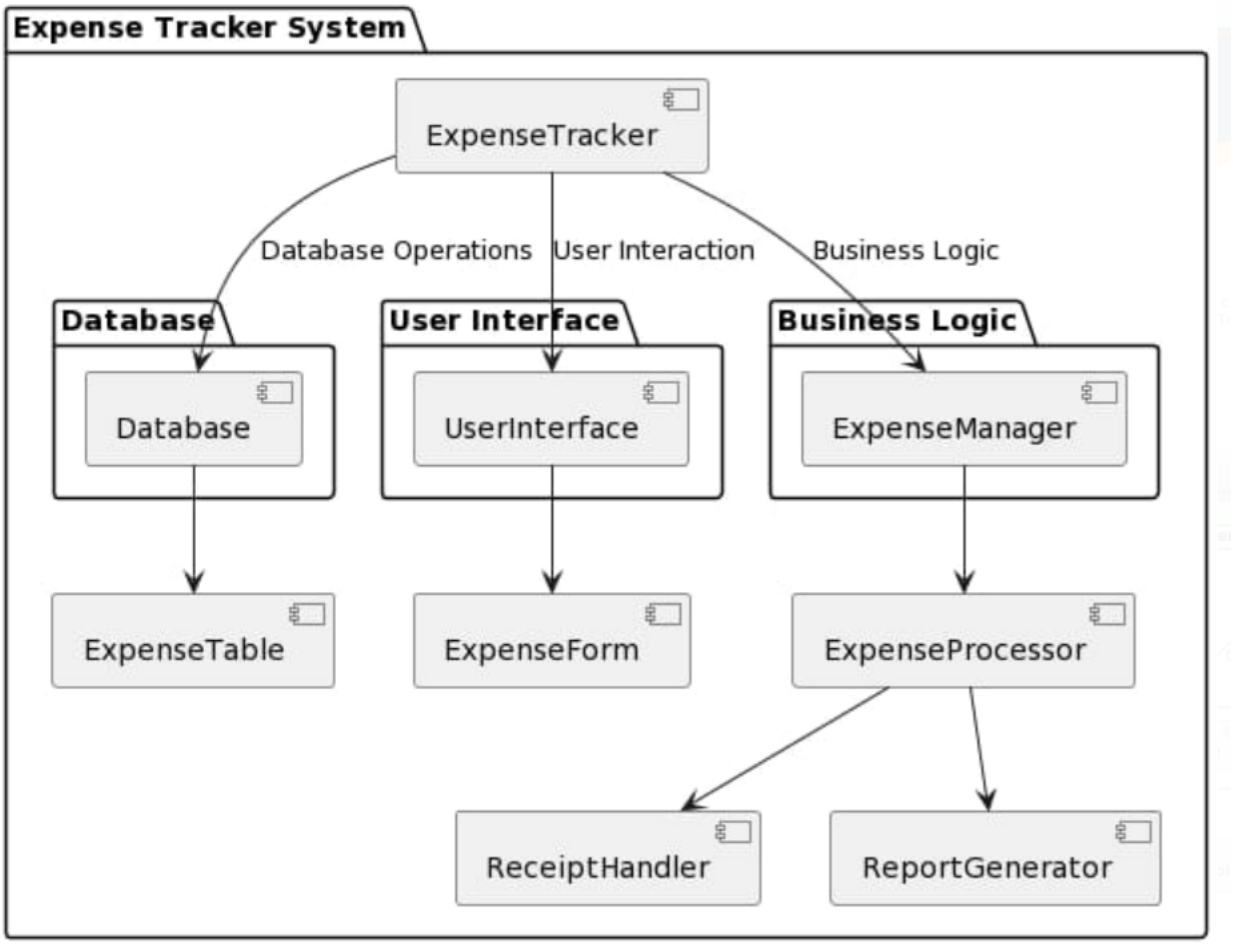
6.6 State Chart Diagram



6.7 Component Diagram

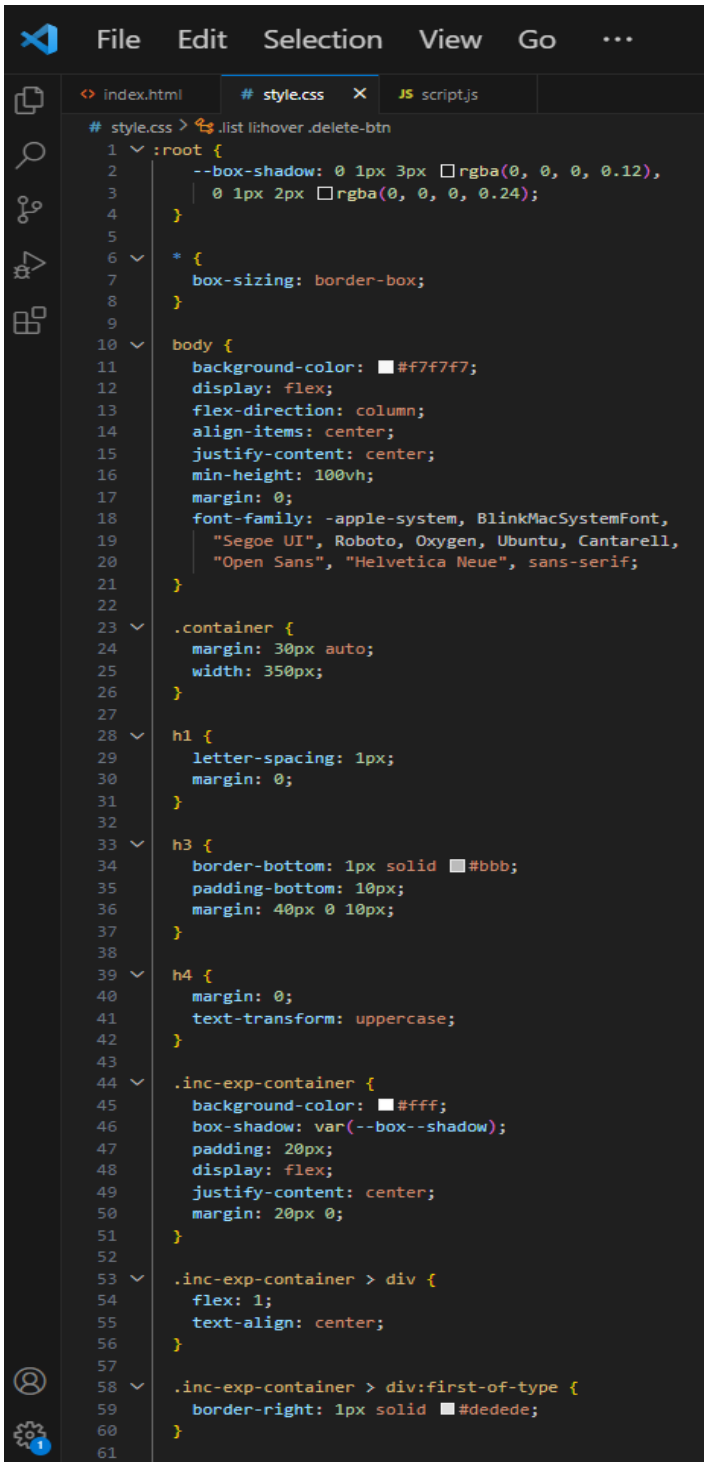


6.8 Deployment Diagram

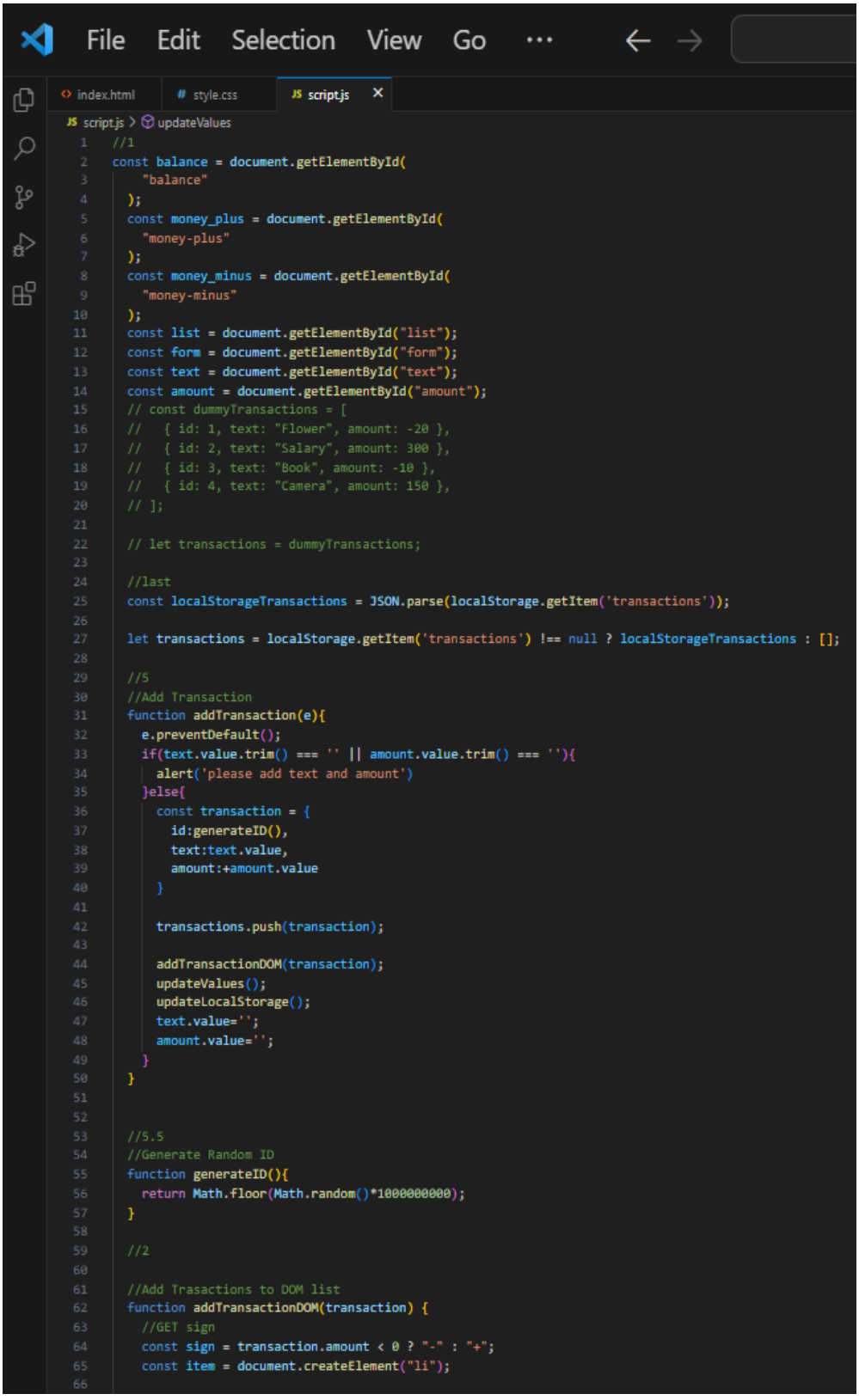


7. About Project and Codes

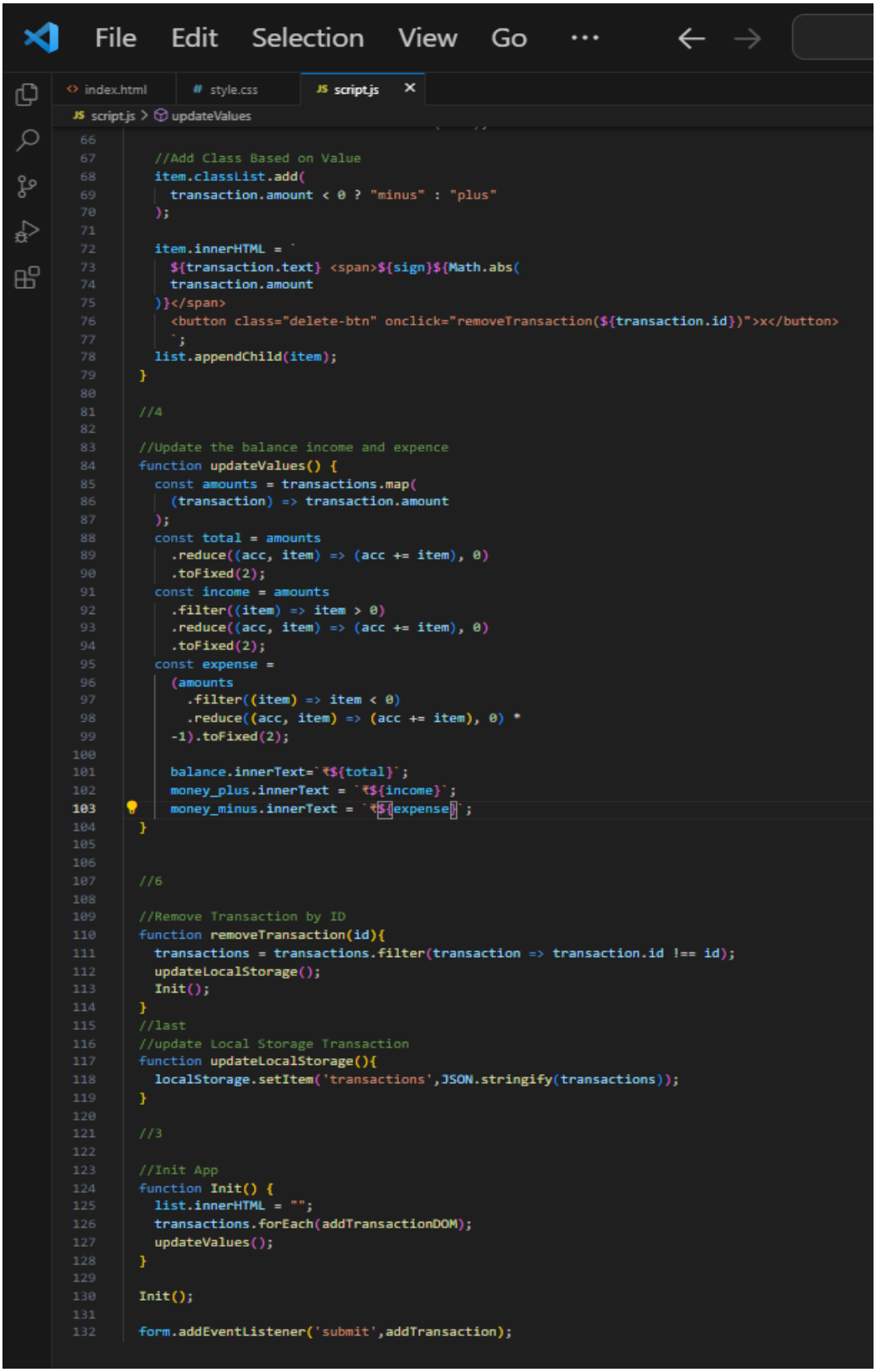
7.1 Codes


File Edit Selection View Go ...
index.html
style.css
script.js
style.css > .list li:hover .delete-btn
:root {
--box-shadow: 0 1px 3px rgba(0, 0, 0, 0.12),
0 1px 2px rgba(0, 0, 0, 0.24);
}
* {
box-sizing: border-box;
}
body {
background-color: #f7f7f7;
display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
min-height: 100vh;
margin: 0;
font-family: -apple-system, BlinkMacSystemFont,
"Segoe UI", Roboto, Oxygen, Ubuntu, Cantarell,
"Open Sans", "Helvetica Neue", sans-serif;
}
.container {
margin: 30px auto;
width: 350px;
}
h1 {
letter-spacing: 1px;
margin: 0;
}
h3 {
border-bottom: 1px solid #bbb;
padding-bottom: 10px;
margin: 40px 0 10px;
}
h4 {
margin: 0;
text-transform: uppercase;
}
.inc-exp-container {
background-color: #fff;
box-shadow: var(--box--shadow);
padding: 20px;
display: flex;
justify-content: center;
margin: 20px 0;
}
.inc-exp-container > div {
flex: 1;
text-align: center;
}
.inc-exp-container > div:first-of-type {
border-right: 1px solid #dedede;
}

File Edit Selection View Go
index.html
style.css
script.js
style.css > .list li:hover .delete-btn
.money {
font-size: 2;
letter-spacing: 1px;
margin: 5px 0;
}
.money-plus {
color: #2ecc71;
}
.money-minus {
color: #c0392b;
}
label {
display: inline-block;
margin: 10px 0;
}
input[type="text"],
input[type="number"] {
border: 1px solid #dedede;
border-radius: 2px;
display: block;
font-size: 16px;
padding: 10px;
width: 100%;
}
.btn {
cursor: pointer;
background-color: #9c88ff;
box-shadow: var(--box-shadow);
color: #fff;
border: 0;
display: block;
font-size: 16px;
margin: 10px 0 30px;
padding: 10px;
width: 100%;
}
.btn:focus,
.delete-btn:focus {
outline: 0;
}
.list {
list-style-type: none;
padding: 0;
margin-bottom: 40px;
}
.list li {
background-color: #fff;
box-shadow: var(--box--shadow);
color: #333;
display: flex;
justify-content: space-between;
position: relative;
padding: 10px;
margin: 10px 0;
}
.list li.plus {
border-right: 5px solid #2ecc71;
}
.list li.minus {
border-right: 5px solid #c0392b;
}
.delete-btn {
cursor: pointer;
background-color: #e74c3c;
border: 0;
color: #fff;
font-size: 20px;
line-height: 20px;
padding: 2px 5px;
position: absolute;
top: 50%;
left: 0;
transform: translate(-100%, -50%);
opacity: 0;
transition: opacity 0.3s ease;
}
.list li:hover .delete-btn {
opacity: 1;

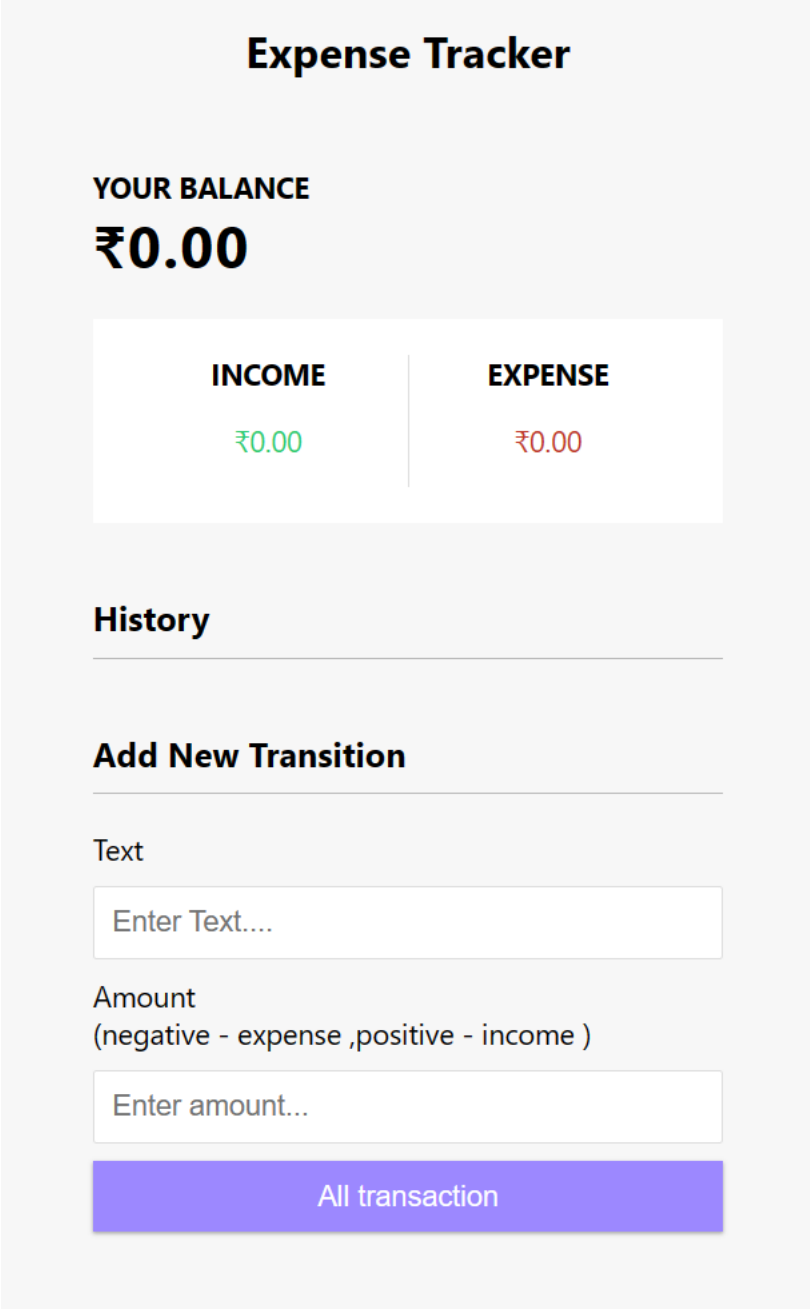


```
File  Edit  Selection  View  Go  ···

index.html   style.css   script.js

script.js > updateValues

//1
const balance = document.getElementById(
    "balance"
);
const money_plus = document.getElementById(
  "money-plus"
);
const money_minus = document.getElementById(
  "money-minus"
);
const list = document.getElementById("list");
const form = document.getElementById("form");
const text = document.getElementById("text");
const amount = document.getElementById("amount");
// const dummyTransactions = [
//   { id: 1, text: "Flower", amount: -20 },
//   { id: 2, text: "Salary", amount: 300 },
//   { id: 3, text: "Book", amount: -10 },
//   { id: 4, text: "Camera", amount: 150 },
// ];

// let transactions = dummyTransactions;

//last
const localStorageTransactions = JSON.parse(localStorage.getItem('transactions'));

let transactions = localStorage.getItem('transactions') !== null ? localStorageTransactions : [];

//5
//Add Transaction
function addTransaction(e){
  e.preventDefault();
  if(text.value.trim() === '' || amount.value.trim() === ''){
    alert('please add text and amount')
  }else{
    const transaction = {
      id:generateID(),
      text:text.value,
      amount:+amount.value
    }

    transactions.push(transaction);

    addTransactionDOM(transaction);
    updateValues();
    updateLocalStorage();
    text.value='';
    amount.value='';
  }
}


//5.5
//Generate Random ID
function generateID(){
  return Math.floor(Math.random()*100000000);
}

//2

//Add Trasactions to DOM list
function addTransactionDOM(transaction) {
  //GET sign
  const sign = transaction.amount < 0 ? "-" : "+";
  const item = document.createElement("li");
```

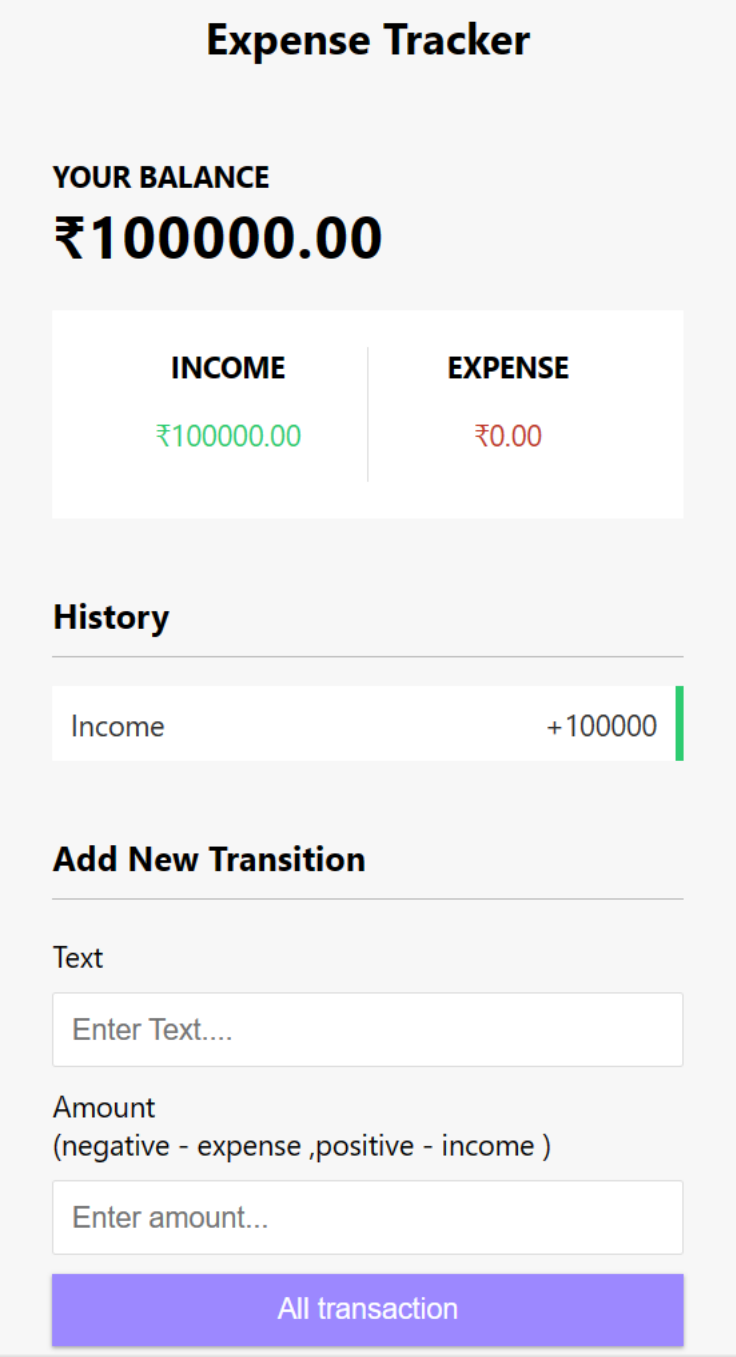



```
66
67      //Add Class Based on Value
68      item.classList.add(
69        transaction.amount < 0 ? "minus" : "plus"
70      );
71
72      item.innerHTML = `
73        ${transaction.text} <span>${sign}${Math.abs(
74        transaction.amount
75      )}</span>
76        <button class="delete-btn" onclick="removeTransaction(${transaction.id})">x</button>
77        `;
78      list.appendChild(item);
79    }
80
81    //4
82
83    //Update the balance income and expence
84    function updateValues() {
85      const amounts = transactions.map(
86        (transaction) => transaction.amount
87      );
88      const total = amounts
89        .reduce((acc, item) => (acc += item), 0)
90        .toFixed(2);
91      const income = amounts
92        .filter((item) => item > 0)
93        .reduce((acc, item) => (acc += item), 0)
94        .toFixed(2);
95      const expense =
96        (amounts
97          .filter((item) => item < 0)
98          .reduce((acc, item) => (acc += item), 0) *
99        -1).toFixed(2);
100
101       balance.innerText=`₹${total}`;
102       money_plus.innerText = `₹${income}`;
103       money_minus.innerText = `₹${expense}`;
104   }
105
106
107   //6
108
109   //Remove Transaction by ID
110   function removeTransaction(id){
111     transactions = transactions.filter(transaction => transaction.id !== id);
112     updateLocalStorage();
113     Init();
114   }
115   //last
116   //update Local Storage Transaction
117   function updateLocalStorage(){
118     localStorage.setItem('transactions',JSON.stringify(transactions));
119   }
120
121   //3
122
123   //Init App
124   function Init() {
125     list.innerHTML = "";
126     transactions.forEach(addTransactionDOM);
127     updateValues();
128   }
129
130   Init();
131
132   form.addEventListener('submit',addTransaction);
```

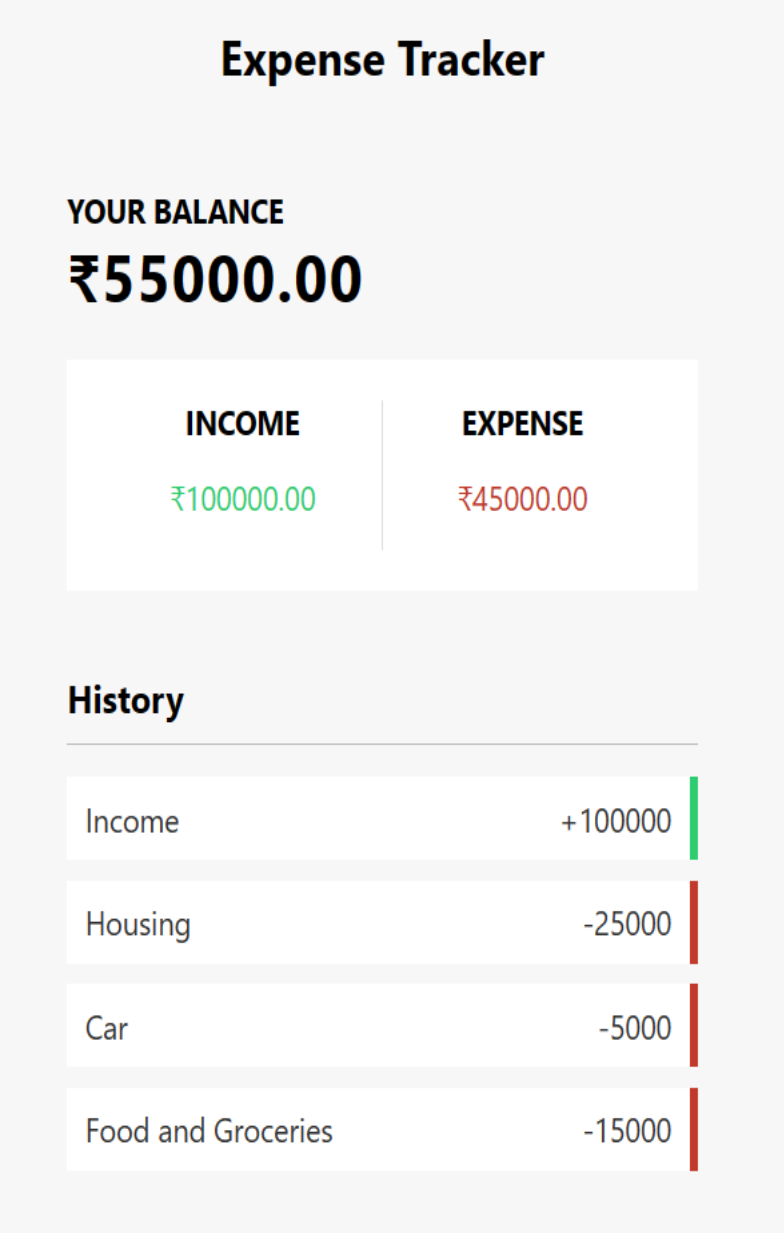
7.2 Main Page


Expense Tracker
YOUR BALANCE
₹0.00
INCOME
₹0.00
EXPENSE
₹0.00
History
Add New Transition
Text
Enter Text....
Amount
(negative - expense ,positive - income)
Enter amount...
All transaction

7.3 User Details added



7.4 Expenses shown



References

https://ijarsct.co.in/Paper391.pdf

https://www.slideshare.net/RashnaMaharjan2/daily-expense-tracker-153160282

https://www.studocu.com/in/document/dr-apj-abdul-kalam-technical-university/btech/mini-project-report/29935835

https://www.irejournals.com/formatedpaper/1702687.pdf

- Software Requirement Specifications, Expense Tracker System