

# **Summer Project On Financial Tracker**

**By**

**Shantanu Kamte (2021510026)**

Under the guidance of  
**Internal Supervisor**

**Prof. Nikhita Mangaonkar**



Department of Master Of Computer Application  
Sardar Patel Institute of Technology  
Autonomous Institute Affiliated to Mumbai University  
2022-23

## **CERTIFICATE OF APPROVAL**

This is to certify that the following students

**Shantanu Kamte (2021510026)**

Have satisfactorily carried out work on the project  
entitled

**“Financial Tracker”**

Towards the fulfilment of project, as laid down  
by

Sardar Patel Institute of Technology  
during year  
2022-23.

Project Guide:  
Prof. Nikita Mangaonkar

## PROJECT APPROVAL CERTIFICATE

This is to certify that the following students

**Shantanu Kamte (2021510026)**

Have successfully completed the Project report on

**“Financial Tracker”,**

which is found to be satisfactory and is approved

at

SARDAR PATEL INSTITUTE OF TECHNOLOGY,  
ANDHERI (W), MUMBAI

INTERNAL EXAMINER

EXTERNAL EXAMINER

HEAD OF DEPARTMENT

PRINCIPAL

# Contents

|  |           |
|--|-----------|
| <b>Abstract</b>  | <b>i</b>  |
| <b>Objectives</b>  | <b>i</b>  |
| <b>List Of Figures</b>                                       | <b>ii</b> |
| <b>List Of Tables</b>  | <b>ii</b> |
| <b>1 Introduction</b>  | <b>1</b>  |
| 1.1 Problem Definition . . . . .                             | 1         |
| 1.2 Objectives and Scope . . . . .                           | 1         |
| 1.2.1 Objectives . . . . .                                   | 1         |
| 1.2.2 Scope . . . . .  | 1         |
| 1.3 Existing System . . . . .                                | 2         |
| 1.4 Proposed System . . . . .                                | 2         |
| 1.5 System Requirements . . . . .                            | 3         |
| <b>2 Software Requirement Specification (SRS) and Design</b> | <b>4</b>  |
| 2.1 Purpose . . . . .  | 4         |
| 2.2 Definition . . . . .                                     | 4         |
| 2.3 Overall Description . . . . .                            | 4         |
| 2.3.1 Product Functions . . . . .                            | 4         |
| 2.3.2 User Characteristics . . . . .                         | 4         |
| <b>3 Project Analysis and Design</b>                         | <b>5</b>  |
| 3.1 Methodologies Adapted . . . . .                          | 5         |
| 3.2 Modules . . . . .  | 7         |
| 3.2.1 Activity diagram . . . . .                             | 7         |
| 3.2.2 Deployment Diagram . . . . .                           | 8         |
| 3.2.3 Architecture Design . . . . .                          | 8         |
| 3.2.4 Gantt Chart . . . . .                                  | 9         |
| 3.2.5 Use-Case . . . . .                                     | 10        |
| <b>4 Project Implementation and Testing</b>                  | <b>14</b> |
| 4.1 Register . . . . .                                       | 14        |
| 4.2 Login . . . . .  | 15        |
| 4.3 Home . . . . .   | 16        |
| 4.4 Add Transaction . . . . .                                | 17        |
| 4.5 Edit Transaction . . . . .                               | 18        |
| 4.6 Analytics . . . . .                                      | 19        |
| 4.7 Code 1 . . . . .   | 20        |
| 4.8 Code 2 . . . . .   | 20        |
| 4.9 Code 3 . . . . .   | 21        |
| <b>5 Test Cases</b>  | <b>22</b> |

|          |                            |           |
|----------|----------------------------|-----------|
| <b>6</b> | <b>Limitations</b>         | <b>23</b> |
| <b>7</b> | <b>Future Enhancements</b> | <b>23</b> |
| <b>8</b> | <b>User Manual</b>         | <b>24</b> |
| <b>9</b> | <b>Bibliography</b>        | <b>25</b> |
| 9.1      | Web References . . . . .   | 25        |

## Abstract

Financial Tracker is a platform where we will list all our income and expenditures from various sources like salary, freelance, food, travel, medical, education etc to make an analysis about our money. So we have an idea where we are spending our money. We can analyze our expenses in graphical format and cut down unnecessary expenses if any. Using the analysis we can structure our budget in a better way.

## Objectives

The Web based Application "Financial Tracker" is used

- To take income and expenses of the users.
- To let users use filters so that the user can view all their income and expenses according to their preference.
- To use category wise analysis and statistics for both income and expense transactions.
- To analyse the transactions based on count, amount, total turnover.

## List of Figures

|   |    |
|---|----|
| 3.1.1 Iterative Model Diagram . . . . . | 6  |
| 3.2.1 Activity Diagram . . . . .        | 7  |
| 3.2.2 Deployment Diagram . . . . .      | 8  |
| 3.2.3 Architecture Design . . . . .     | 8  |
| 3.2.4 Gantt Chart . . . . .             | 9  |
| 3.2.5 Use-Case Diagram . . . . .        | 10 |
| 4.1.1 Register . . . . .                | 14 |
| 4.2.1 Login . . . . .                   | 15 |
| 4.3.1 Home . . . . .                    | 16 |
| 4.4.1 Add Transaction . . . . .         | 17 |
| 4.5.1 Edit Transaction . . . . .        | 18 |
| 4.6.1 Analytics . . . . .               | 19 |

## List of Tables

|  |    |
|--|----|
| 1.5.1 Hardware Requirements on Server Side . . . . . | 3  |
| 1.5.2 Hardware Requirements on Client Side . . . . . | 3  |
| 1.5.3 Software Requirements on Server Side . . . . . | 3  |
| 1.5.3 Software Requirements on Client Side . . . . . | 3  |
| 4.2.1 Use Case Table - Register . . . . .            | 11 |
| 4.2.2 Use Case Table - Login . . . . .               | 11 |
| 4.2.3 Use Case Table - Add Transactions . . . . .    | 11 |
| 4.2.4 Use Case Table - Update Transactions . . . . . | 12 |
| 4.2.5 Use Case Table - View Transactions . . . . .   | 12 |
| 4.2.6 Use Case Table - Delete Transactions . . . . . | 12 |
| 4.2.7 Use Case Table - View Analytics . . . . .      | 12 |
| 4.2.8 Use Case Table - Logout . . . . .              | 13 |
| 6.1 Test Case - Login and Register . . . . .         | 22 |

# 1 Introduction

## 1.1 Problem Definition

We perform multiple transactions throughout the day. It is extremely difficult to track our expenses. People use traditional software like Excel but they do not provide proper analysis. We have no idea where we are spending all our money. Most of the existing websites/applications do not provide proper graphical analysis so it becomes difficult for the user to track and analyze their expenses.

## 1.2 Objectives and Scope

### 1.2.1 Objectives

The Web based application "Financial Tracker" is

- To take income and expenses of the users.
- To let users use filters so that the user can view all their income and expenses according to their preference.
- To use category wise analysis and statistics for both income and expense transactions.
- To analyse the transactions based on count, amount, total turnover.

### 1.2.2 Scope

All the users can create their profile and login to the website. Using this website the users can add/update their transactions and analyse their transactions in a graphical format.



### 1.3 Existing System

Currently no such website exists which helps users track and analyse their transactions. They do not provide proper filter to provide accurate analysis. Some of the disadvantages of existing system are as follows :

- No Filters  
The existing websites do not provide proper filters to analyze users record.
- No Graphical Representation  
It becomes difficult to analyse data with no graphical representation.
- Portability  
The existing app do not provide portability, people usually have to download the app. Being a responsive website users can access the website from anywhere on their phone.

### 1.4 Proposed System

Financial Tracker is a platform where we will list all our income and expenditures from various sources like salary, freelance, food, travel, medical, education etc to make an analysis about our money. So we have an idea where we are spending our money. We can analyze our expenses in graphical format and cut down unnecessary expenses if any. Using the analysis we can structure our budget in a better way.

Some of the advantages of our system are as follows :

- User Friendly  
It provides attractive interface to the user for navigation through a dynamic flow of content in the app.  
The items of the application are already connected to each other.
- All transactions at one place  
Users can view all the transactions at per their requirement in a single list view and also in graphical format.

## 1.5 System Requirements

- Hardware Requirements on Server Side

Table 1.5.1: Hardware Requirements on Server Side

|           |  |
|-----------|--|
| Processor | Dual Core Processor or Above                 |
| RAM       | Minimum 4 GB RAM                             |
| Storage   | Minimum 10 GB Hard Disk Space for smooth run |

- Hardware Requirements on Client Side

Table 1.5.2: Hardware Requirements on Client Side

|           |                              |
|-----------|------------------------------|
| Device    | Mobile, Laptop or Desktop    |
| Processor | Dual Core Processor or Above |
| RAM       | Minimum 2 GB RAM             |
| Storage   | Minimum 250 MB Storage Space |

- Software Requirements on Server Side

Table 1.5.3: Software Requirements on Server Side

|                  |                |
|------------------|----------------|
| Operating System | OS Independent |
| Database         | MongoDB        |

- Software Requirements on Client Side

Table 1.5.3: Software Requirements on Client Side

|                  |                |
|------------------|----------------|
| Operating System | OS Independent |
| Server           | Not Required   |

## **2 Software Requirement Specification (SRS) and Design**

### **2.1 Purpose**

The purpose of my project is to develop an UI application that can help user (students) to easily access their past transactions and to keep all the required information handy.

This can save lots of effort of users who face difficulty in tracking their financial records. This app provides analytics feature which helps users offer a visual representation of the data so that the user can find which section they are spending money on and spend wisely in the future.

### **2.2 Definition**

To build a Financial Tracker Application so the users can track their records on the go.

### **2.3 Overall Description**

#### **2.3.1 Product Functions**

The product function includes:

The basic function of the website is that it allows the user to register to the website. After checking whether the user's email ID is verified, all the relevant data related to that user is fetched from the database. The transactions which are already added can be viewed by the user. Alternatively the user can add new transactions and update the existing transactions. The user can view the statistics as well.

#### **2.3.2 User Characteristics**

- User: The people who want to view and add/update their transactions. The website can be used by many of the users at a time. When the users visit the website, they can view the details of the transactions added and can add/update transactions.

## 3 Project Analysis and Design

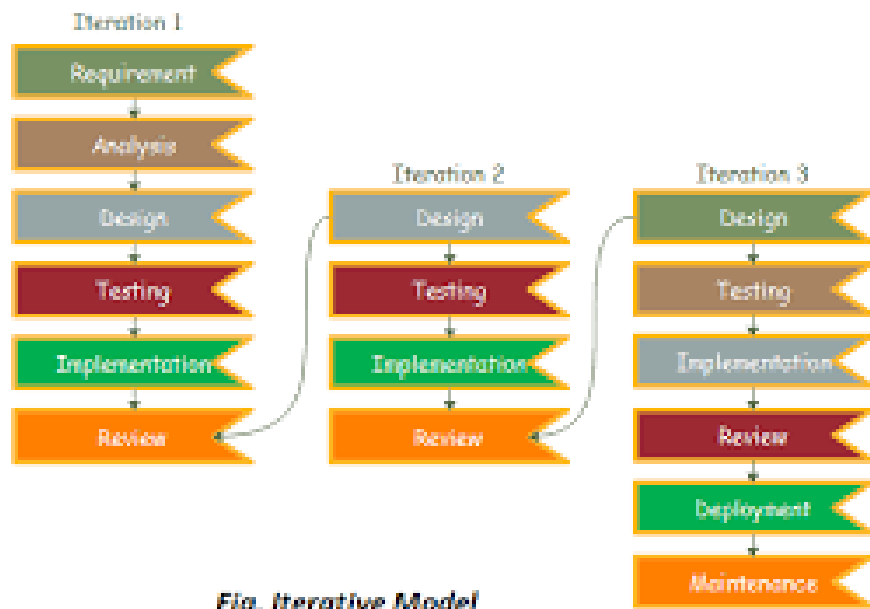
### 3.1 Methodologies Adapted

Methodology involves dividing software development work into distinct stages and coming up with tasks or activities aimed at achieving better planning and time management. It is considered a trivial part of the systems development life cycle.

**Iterative Model:** The iterative model is a particular implementation of a software development life cycle (SDLC) that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete. In this Model, you can start with some of the software specifications and develop the first version of the software. After the first version if there is a need to change the software, then a new version of the software is created with a new iteration. Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.

The Iterative Model allows accessing earlier phases, in which the variations are made respectively. The final output of the project was renewed at the end of the Software Development Life Cycle (SDLC) process.

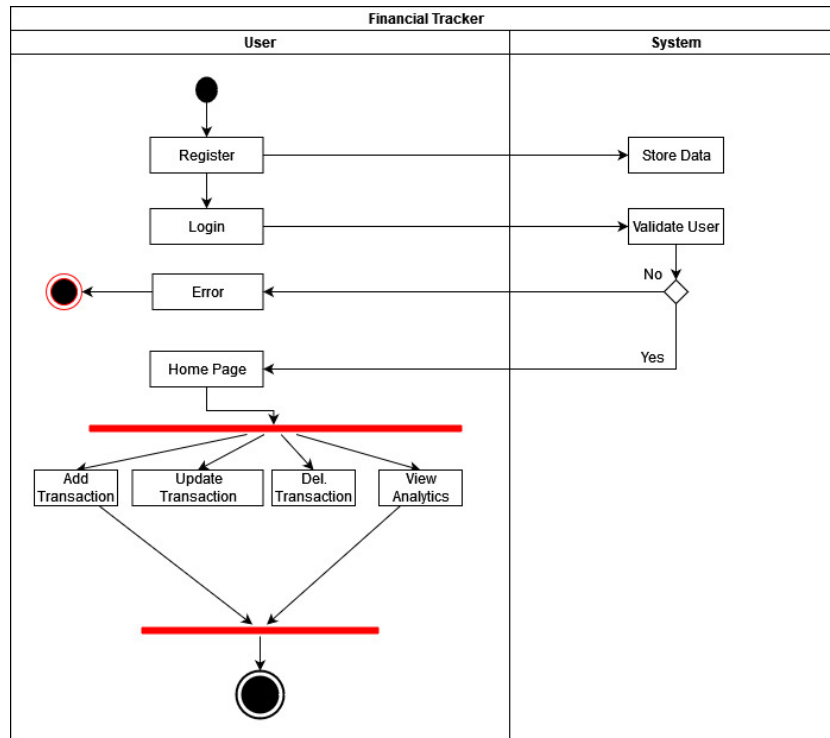
While creating this website in the first iteration we focused on creating the simple modules like static pages and other later iterations, register , login. We focused on creating the complex modules such as database connections, data fetching, displaying data from API and MongoDB in the second iteration. And then as modules were completed they were evaluated and again if any glitches were there planning was done to solve those and the process went on.



3.1.1: Iterative Model Diagram

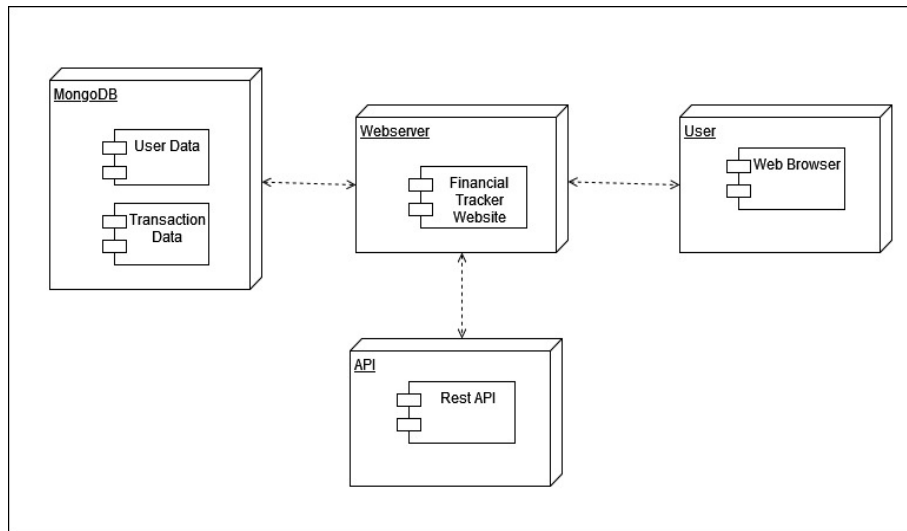
## 3.2 Modules

### 3.2.1 Activity diagram



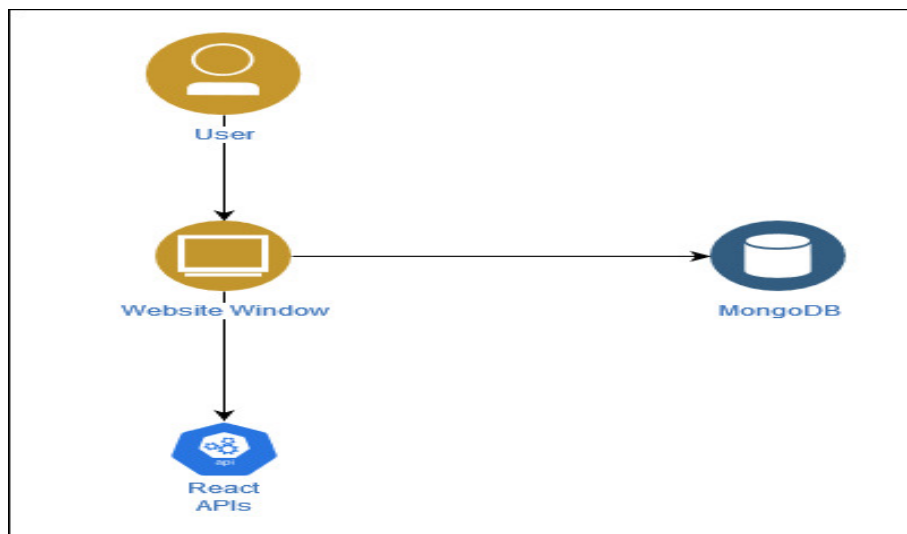
3.2.1: Activity Diagram

### 3.2.2 Deployment Diagram



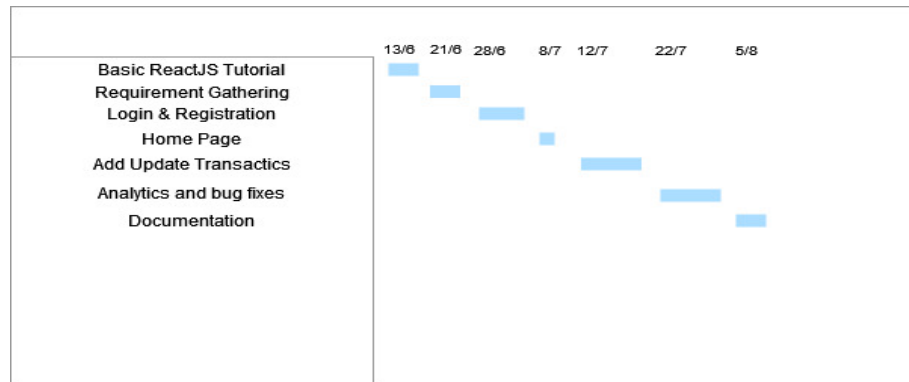
3.2.2: Deployment Diagram

### 3.2.3 Architecture Design



3.2.3: Architecture Design

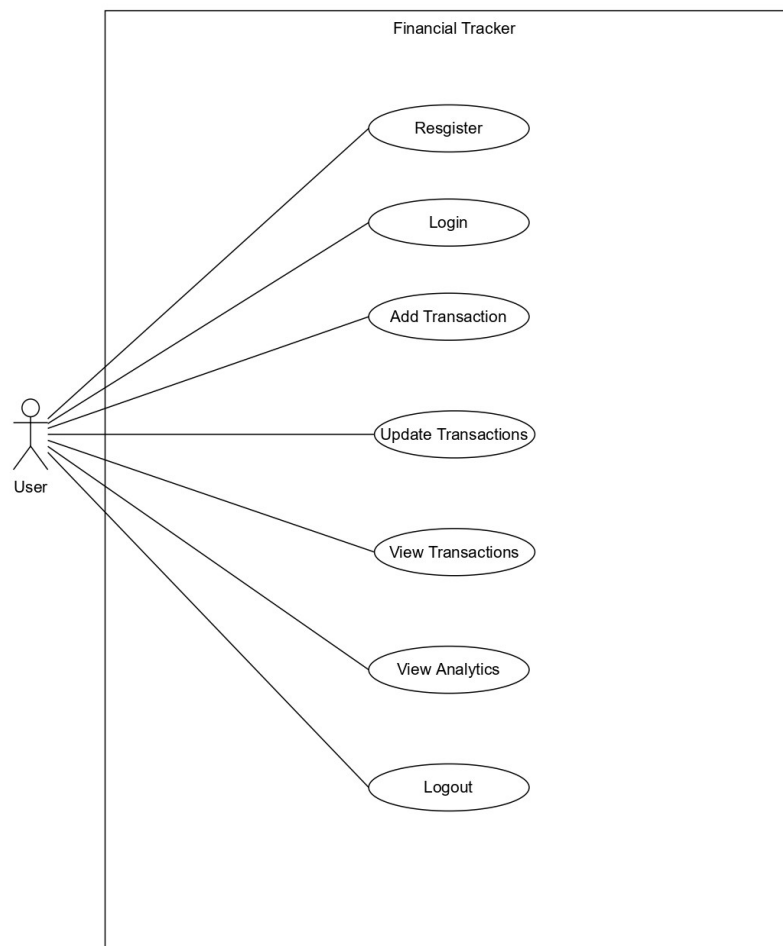
### 3.2.4 Gantt Chart



3.2.4: Gantt Chart



### 3.2.5 Use-Case



3.2.5: Use-Case Diagram

Use Cases:

1. Register
2. Login
3. Add Transactions
4. Update Transactions
5. View Transactions
6. Delete Transactions
7. View Analytics
8. Logout

Table 4.2.1: Use Case Table - Register

|                |                                     |
|----------------|-------------------------------------|
| Use Case ID    | 1                                   |
| Use Case Name  | Register                            |
| Actor          | User                                |
| Pre-Condition  | They must register themselves first |
| Post-Condition | User can login                      |

Table 4.2.2: Use Case Table - Login

|                |   |
|----------------|---|
| Use Case ID    | 2   |
| Use Case Name  | Login   |
| Actor          | User  |
| Pre-Condition  | They must register themselves first                               |
| Post-Condition | User can view its details and marks and view and upload Companies |

Table 4.2.3: Use Case Table - Add Transactions

|                |                                       |
|----------------|---------------------------------------|
| Use Case ID    | 3                                     |
| Use Case Name  | Add Transactions                      |
| Actor          | User                                  |
| Pre-Condition  | Login                                 |
| Post-Condition | User can view or add the transactions |

Table 4.2.4: Use Case Table - Update Transactions

|                |   |
|----------------|---|
| Use Case ID    | 4                                       |
| Use Case Name  | Update Transactions                     |
| Actor          | User                                    |
| Pre-Condition  | Login                                   |
| Post-Condition | User can view and edit the transactions |

Table 4.2.5: Use Case Table - View Transactions

|                |                       |
|----------------|-----------------------|
| Use Case ID    | 5                     |
| Use Case Name  | View Transactions     |
| Actor          | User                  |
| Pre-Condition  | Login                 |
| Post-Condition | Can view transactions |

Table 4.2.6: Use Case Table - Delete Transactions

|                |                     |
|----------------|---------------------|
| Use Case ID    | 6                   |
| Use Case Name  | Delete Transactions |
| Actor          | User                |
| Pre-Condition  | Login               |
| Post-Condition | Delete transactions |

Table 4.2.7: Use Case Table - View Analytics

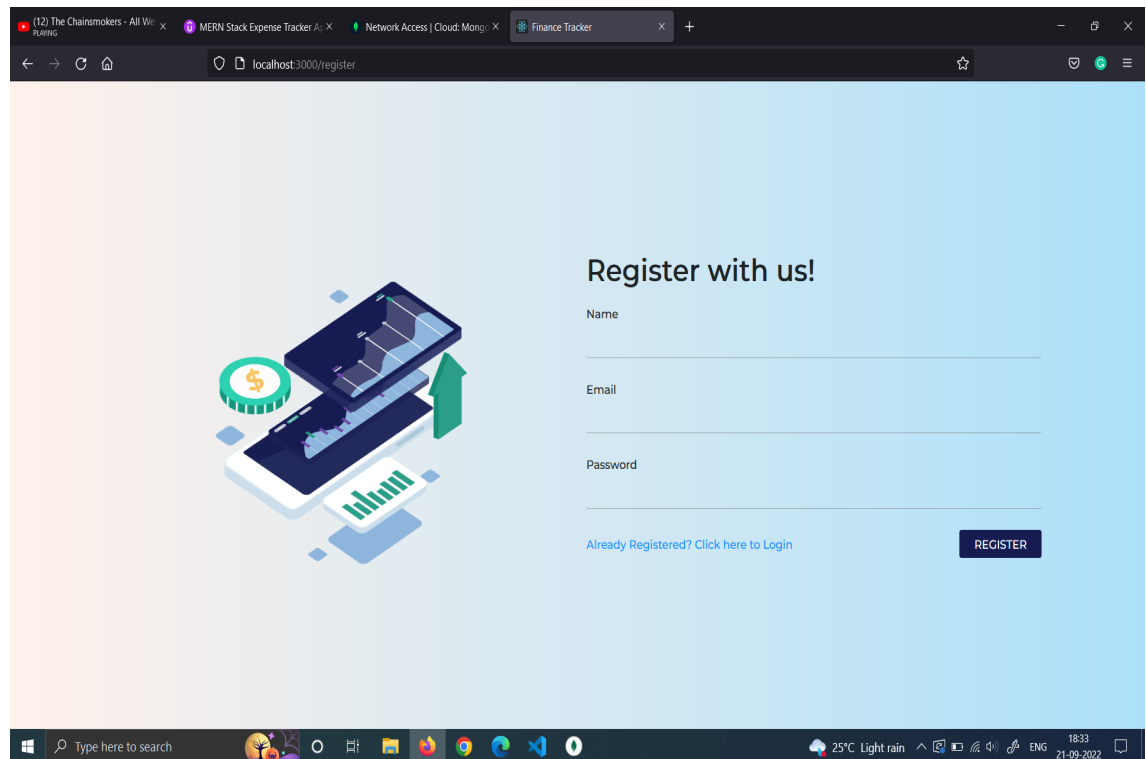
|                |   |
|----------------|---|
| Use Case ID    | 7   |
| Use Case Name  | View Analytics  |
| Actor          | User  |
| Pre-Condition  | Login   |
| Post-Condition | User can view total transactions and total transactions. Users can also view on which section they are spending more money. |

Table 4.2.8: Use Case Table - Logout

|                |   |
|----------------|---|
| Use Case ID    | 8   |
| Use Case Name  | Logout  |
| Actor          | User  |
| Pre-Condition  | Login   |
| Post-Condition | User will be logged out and redirected to the login screen. |

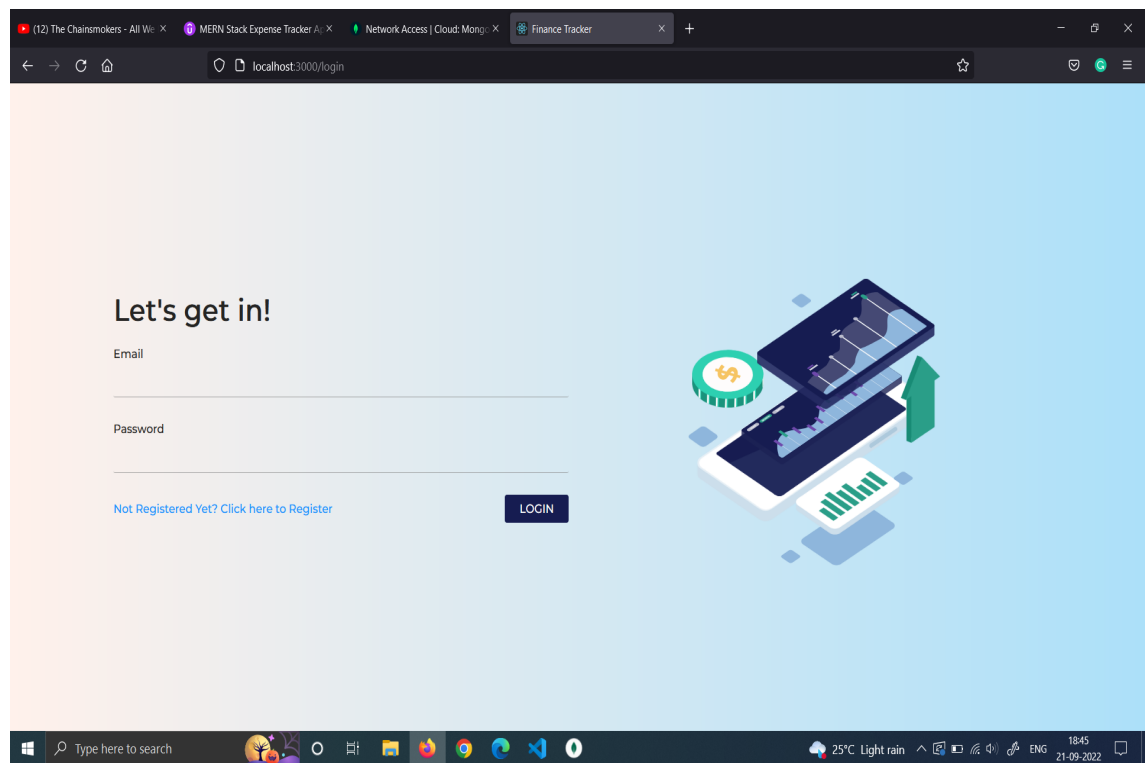
## 4 Project Implementation and Testing

### 4.1 Register



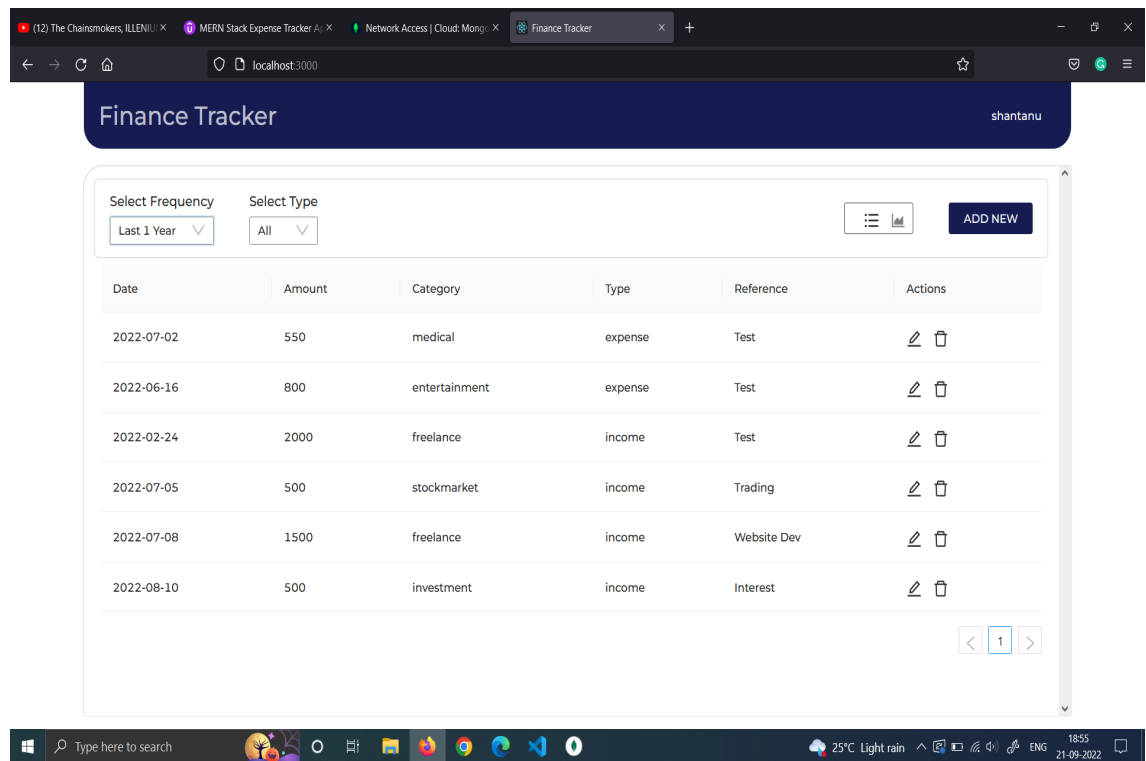
4.1.1: Register

## 4.2 Login



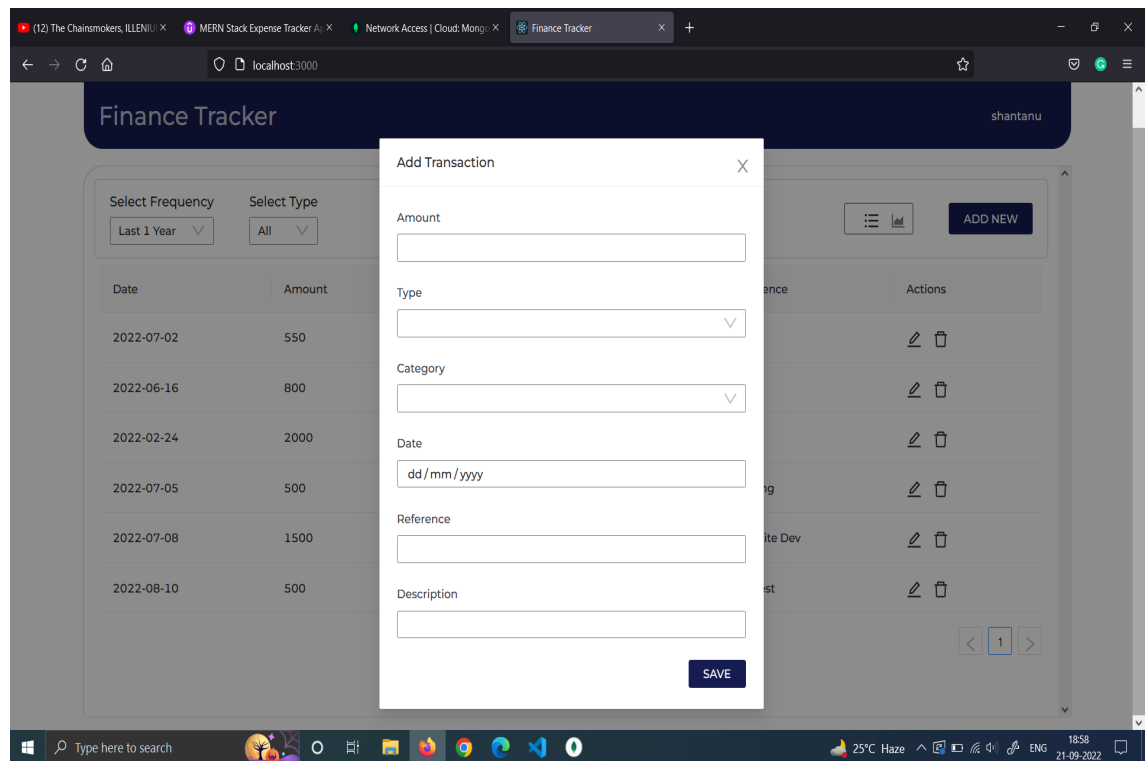
### 4.2.1: Login

### 4.3 Home



#### 4.3.1: Home

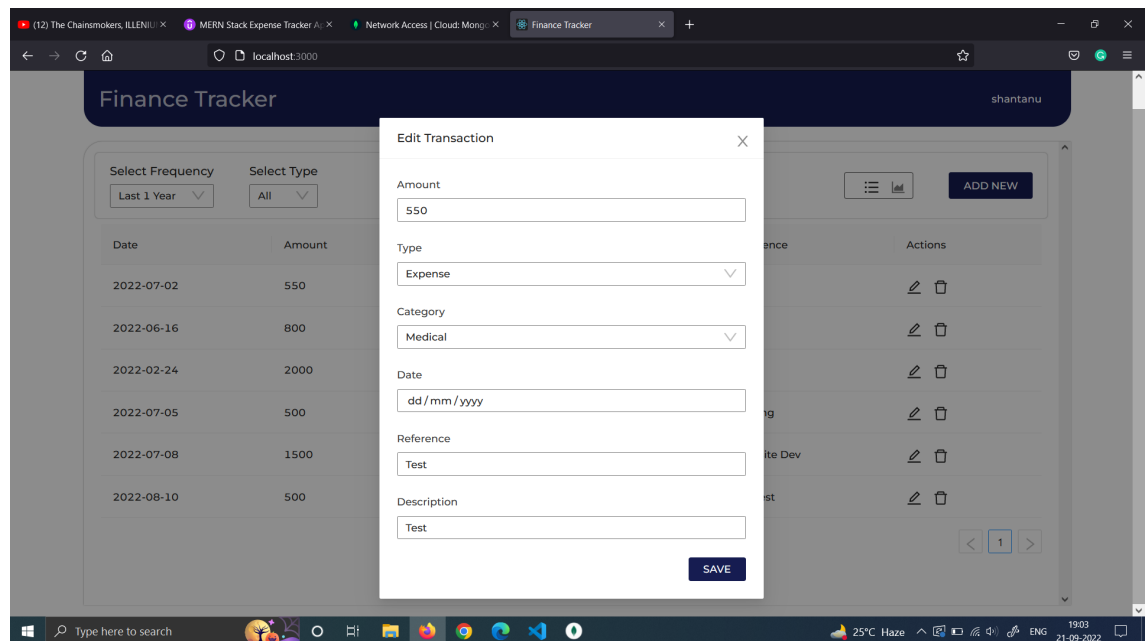
#### 4.4 Add Transaction



4.4.1: Add Transaction

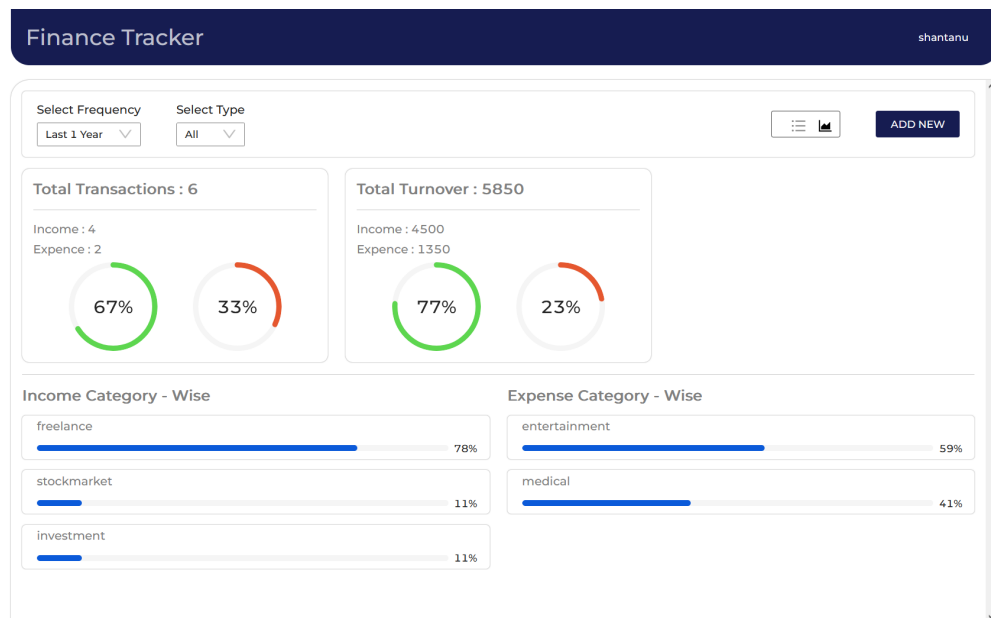


## 4.5 Edit Transaction



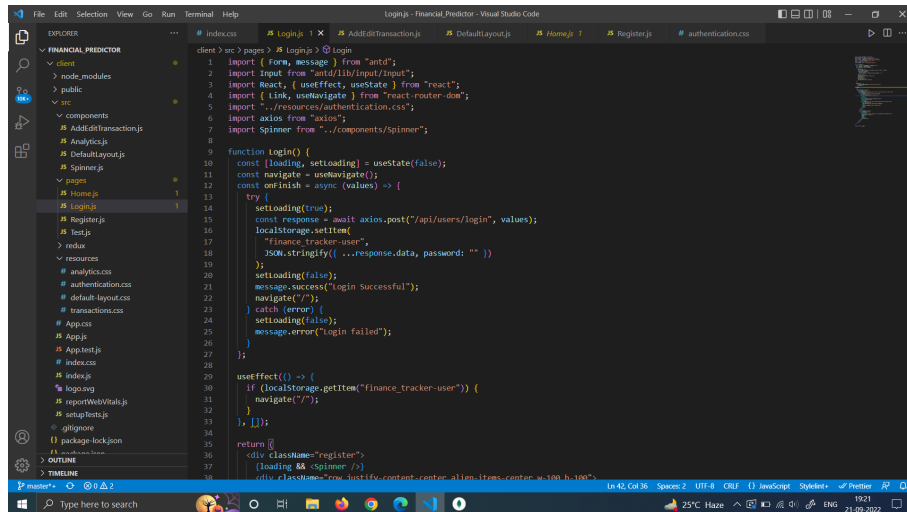
### 4.5.1: Edit Transaction

## 4.6 Analytics



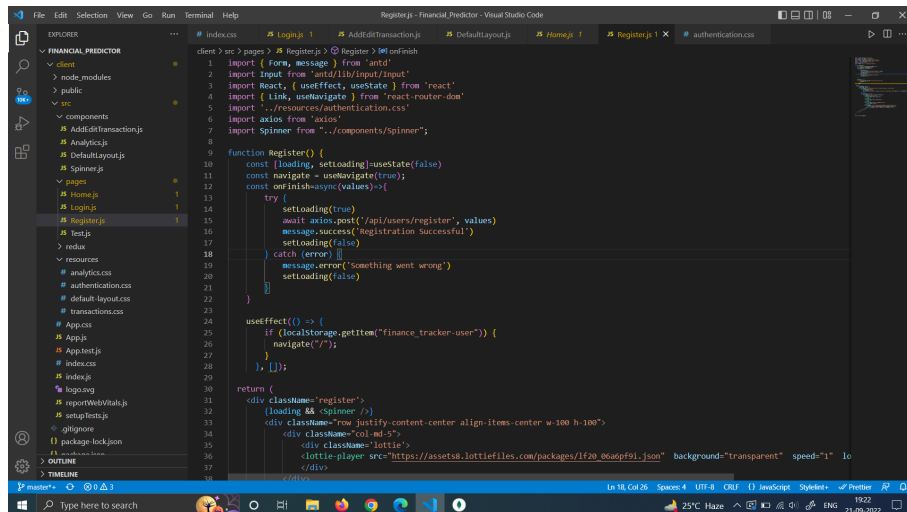
### 4.6.1: Analytics

## 4.7 Code 1



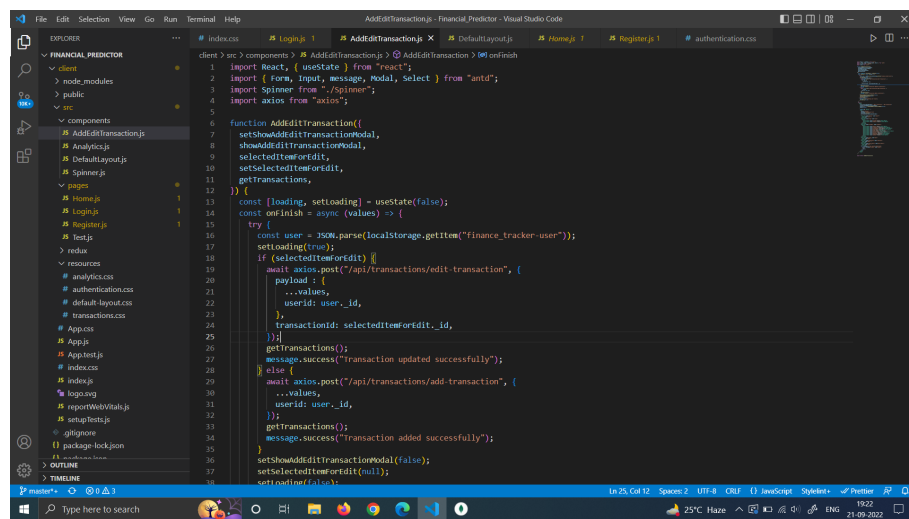
```
client > src > pages > # Login.js > Login
1 import { form, message } from "antd";
2 import Input from "antd/lib/input/input";
3 import React, { useEffect, useState } from "react";
4 import { Link, useNavigate } from "react-router-dom";
5 import "../resources/authentication.css";
6 import axios from "axios";
7 import Spinner from "../components/Spinner";
8
9 function Login() {
10   const [loading, setloading] = useState(false);
11   const navigate = useNavigate();
12   const onFinish = async (values) => {
13     try {
14       setloading(true);
15       const response = await axios.post("/api/users/login", values);
16       localStorage.setItem(
17         "finance_tracker-user",
18         JSON.stringify({ ...response.data, password: "" })
19       );
20       setloading(false);
21       message.success("Login Successful");
22       navigate("/");
23     } catch (error) {
24       setloading(false);
25       message.error("Login failed");
26     }
27   };
28
29   useEffect(() => {
30     if (localStorage.getItem("finance_tracker-user")) {
31       navigate("/");
32     }
33   }, []);
34
35   return (
36     <div className="register">
37       <loading && <Spinner />
38     </div>
39   );
40 }
```

## 4.8 Code 2



```
client > src > pages > # Register.js > Register > onFinish
1 import { form, message } from "antd";
2 import Input from "antd/lib/input/input";
3 import React, { useEffect, useState } from "react";
4 import { Link, useNavigate } from "react-router-dom";
5 import "../resources/authentication.css";
6 import axios from "axios";
7 import Spinner from "../components/Spinner";
8
9 function Register() {
10   const [loading, setloading] = useState(false);
11   const navigate = useNavigate();
12   const onFinish = async (values) => {
13     try {
14       setloading(true);
15       await axios.post("/api/users/register", values);
16       message.success("Registration Successful");
17       setloading(false);
18     } catch (error) {
19       message.error("something went wrong");
20       setloading(false);
21     }
22   };
23
24   useEffect(() => {
25     if (localStorage.getItem("finance_tracker-user")) {
26       navigate("/");
27     }
28   }, []);
29
30   return (
31     <div className="register">
32       <loading && <Spinner />
33       <div className="row justify-content-center align-items-center w-100 h-100">
34         <div className="col-md-5">
35           <div className="lottie">
36             <lottie-player src="https://assets8.lottiefiles.com/packages/lf28_0taop91.json" background="transparent" speed="1"
37             </div>
38         </div>
39       </div>
40     </div>
41   );
42 }
```

#### 4.9 Code 3



```
1  import React, { useState } from "react";
2  import { Form, Input, message, Modal, Select } from "antd";
3  import Spinner from "../Spinner";
4  import axios from "axios";
5
6  function AddEditTransaction({
7    setShowAddEditTransactionModal,
8    showAddEditTransactionModal,
9    selectedItemForEdit,
10    setSelectedItemForEdit,
11    getTransactions,
12  }) {
13    const [loading, setLoading] = useState(false);
14    const onFinish = async (values) => {
15      try {
16        const user = JSON.parse(localStorage.getItem("finance_tracker-user"));
17        setLoading(true);
18        if (selectedItemForEdit) {
19          await axios.post("/api/transactions/edit-transaction", {
20            payload: {
21              ...values,
22              userId: user._id,
23            },
24            transactionId: selectedItemForEdit._id,
25          });
26          getTransactions();
27          message.success("Transaction updated successfully");
28        } else {
29          await axios.post("/api/transactions/add-transaction", {
30            ...values,
31            userId: user._id,
32          });
33          getTransactions();
34          message.success("Transaction added successfully");
35        }
36        setShowAddEditTransactionModal(false);
37        setSelectedItemForEdit(null);
38        setLoading(false);
39      } catch (error) {
40        console.log(error);
41      }
42    };
43  }
44
45  export default AddEditTransaction;
```

## 5 Test Cases

Table 6.1: Test Case - Login and Register

| Test Case ID | Test Case Name                  | Test Data                               | Expected Output                  | Actual Output                    | Result |
|--------------|---------------------------------|---|----------------------------------|----------------------------------|--------|
| 1            | User enter user id and password | Enters the correct user id and password | Log in Successful                | Home Page                        | Pass   |
| 2            | User enter user id and password | Enters the user id and password         | Prompt error                     | Prompt error                     | Pass   |
| 3            | Add Transaction                 | User enters transaction details         | Transaction added successfully   | Transaction added successfully   | Pass   |
| 4            | Update Transaction              | User updates transaction details        | Transaction updated successfully | Transaction updated successfully | Pass   |
| 5            | Delete Transaction              | User deletes transaction details        | Transaction deleted successfully | Transaction deleted successfully | Pass   |

## **6 Limitations**

- Requires continuous internet connection.
- Requires a web browser
- Requires one working email account to register

## **7 Future Enhancements**

- Suggestion can be given to the user based on his/her transaction history, to reduce expenses.
- Financial prediction feature can be added to the website.
- Monthly reports can be generated and sent to users registered email ID.

## 8 User Manual

### Part 1 – Login

Upon opening the application, user will be greeted with the login screen. If the user has no account, user can click on register and register self.

After verification of user id, User's account will be verified from the data saved in the database.

### Part 2 – Register

User needs to enter email first and then password. The data entered by the user will be saved in the database for verification.

### Part 3 – Home

Users can view past 1 week transaction by default. They can change the frequency to 1 month, year and also use custom range. Users can add, update and delete a particular transaction.

### Part 4 – Add, Update Transaction

Users will be get a form as soon as they click on add/update transaction button. They can fill the form and the data will be saved in the database.

### Part 4 – Delete Transaction

The data of the transaction will be deleted and also erased from the database.

### Part 6 – Analytics

Users can view their total transactions and total turnover in a graphical format. Users can also view on which section they have spent more money on in a graphical format so that they can analyse and spend money wisely in the future.

## 9 Bibliography

### 9.1 Web References

- [1.] [https://youtube.com/playlist?list=PLu0W\\_9lII9agx66oZnT6IyhcmIbUMNMdt](https://youtube.com/playlist?list=PLu0W_9lII9agx66oZnT6IyhcmIbUMNMdt)
- [2.] <https://reactjs.org/docs/getting-started.html>
- [3.] <https://www.mongodb.com/docs/>
- [4.] <https://www.geeksforgeeks.org/>
- [5.] <https://stackoverflow.com/>
- [6.] <https://www.draw.io/>