

# SDM POLYTECHNIC

## UJIRE -574240



### PROJECT TITLE

### EXPENSE TRACKER



By

Sl No.	NAME OF THE STUDENT	REGISTER NO.
1.	Nishmitha K	497CS20007
2.	Tharun TU	497CS20011

A dissertation submitted to the Board of Technical Education, Bengaluru.  
In fulfilment of the requirement of the award of Internship programme in Diploma in  
computer Science for the year 2022-2023

**Under the guidance of**  
**Mr. Sudhakar Gajula**  
**Technical head**  
**Integra micro software services (IMSS), Bangalore**

## CONTENTS

SLNO.	PARTICULAS	PAGENO.
1	INTRODUCTION AND OBJECTIVES OF THE PROJECT	03
2	TOOLS PLATFORMS, HARDWARE AND SOFTWARE USED IN THE PROJECT	04
3	SYSTEM ANALYSIS a) EXISTING SYSTEM b) ADVANTAGES AND DISADVANTAGES OF EXISTING SYSTEM c) PROPOSED SYSTEM d) ADVANTAGES AND DISADVANTAGES OF PROPOSED SYSTEM	05
4	SOFTWARE USED IN THE PROJECT	07
5	CODING	14
6	SCREEN SHOTS	46
7	USE CASE DIAGRAM	48

# **1. INTRODUCTION AND OBJECTIVES OF THE PROJECT**

## **INTRODUCTION:**

Expense tracker is one kind of digital diary that helps to keep an eye on all of our money related transactions and also provides all financial activities report Delhi weekly monthly and early users get notification to record expense and income that are helpful to the tracking system of the application all the information is saved in offline mode so user can easily access anytime and any places user interface of the daily expense tracker is very simple and attractive so it is easy to understand and the best way to record our financial data

Many organizations have their own system to record their income and expenses, which they feel is the main key point of their business progress. It is a good habit for a person to record daily expenses and earning but due to unawareness and lack of proper applications to suit their privacy, lacking decision making capacity people are using traditional note keeping methods to do so. Due to lack of a complete tracking system, there is a constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.

## **OBJECTIVES**

Our goal is to create an expense tracking system where user can be tracking all financial activities and view previous income and expense report.

- ❖ Users can easily review the reports daily, weekly, monthly or yearly.
- ❖ Users can update or delete records.
- ❖ Add Expense and Income
- ❖ Minimize manual effort with daily record of expenditures and incomes.
- ❖ Immediate and easy retrieval of report.
- ❖ Secured and transparent data.
- ❖ Graphical overview of transactions.
- ❖ Help in decision making with related results.
- ❖ Help in preparing wish list for pre planning your expenses.

## **2. TOOLS/PLATFORM, HARDWARE AND SOFTWARE SPECIFICATION**

### **HARDWARE REQUIREMENTS:**

Processor : I3 processor  
Ram : 8 GB  
Hard Disk : 500 GB Space  
Monitor : VGA Color (256)

### **SOFTWARE REQUIREMENT:**

Operating System : Windows 10 or higher  
Front End : React  
Scripting : Expressjs, Node js  
Platform : JAVA  
Backend : Mongodb  
Web Server : NODE js webserver or apache  
IDE interface used : VS Code

### **3. SYSTEM ANALYSIS**

#### **3a. EXISTING SYSTEM (Manual Expense Tracking):**

- In the existing system, users manually record their expenses using pen and paper or spreadsheets.
- They need to keep receipts or invoices and manually enter the details like date, amount, category, and description.
- Users have to calculate and maintain running totals for each category or expense type.
- Generating reports or analyzing spending patterns requires manual data manipulation and calculations.
- There is a lack of real-time tracking and collaboration, as users need to update their expenses individually.

#### **3b. DISADVANTAGE OF THE EXISTING SYSTEM**

- Time-consuming: Manual expense tracking requires users to manually enter and calculate expenses, which can be time-consuming, especially for individuals with a large number of transactions.
- Prone to errors: Manual data entry increases the risk of errors and inaccuracies, such as incorrect amounts, missing receipts, or miscalculations.
- Limited analysis capabilities: Generating reports and analyzing spending patterns can be challenging and time-consuming due to the lack of automated data manipulation and calculation features.
- Lack of real-time tracking: The existing system does not provide real-time expense tracking, making it difficult for users to have an up-to-date overview of their spending.

#### **3c. PROPOSED SYSTEM(Automated Expense Tracker):**

- The proposed system is an automated expense tracker that streamlines the expense tracking process.
- Users can input their expenses through a user-friendly mobile or web application.

- The system automatically captures key details like date, amount, and category using OCR (Optical Character Recognition) technology or manual selection from pre-defined options.
- Users can upload or take pictures of receipts, and the system can automatically extract relevant information.
- The system provides real-time expense tracking and categorization, allowing users to view their spending patterns and budgets at any time.
- It generates customizable reports and visualizations to help users analyze their expenses and make informed financial decisions.
- The system can integrate with bank accounts and credit cards to import transaction data, further automating the expense tracking process.
- Users can set spending limits, receive alerts for exceeding budgets, and receive personalized recommendations for saving money.
- The system supports collaboration, allowing users to share expenses with family members or colleagues, track shared expenses, and settle debts easily.
- By implementing the proposed system, users can significantly reduce manual effort, improve accuracy, gain real-time insights into their expenses, and make informed financial decisions.

### **3d. ADVANTANGES OF PROPOSED SYSTEM**

- **Time-saving:** The proposed system automates the expense tracking process, reducing the time required for manual data entry and calculation.
- **Accuracy:** Automated data capture and integration with receipts or bank transactions minimize the risk of errors, ensuring accurate expense tracking.
- **Real-time tracking:** The system provides real-time updates on expenses, allowing users to have instant visibility into their spending habits and budgets.
- **Enhanced analysis:** Automated expense tracking systems offer advanced reporting and analysis features, enabling users to generate comprehensive reports, visualize spending patterns, and gain valuable insights into their finances.
- **Integration capabilities:** The proposed system can integrate with bank accounts, credit cards, and other financial systems, automatically importing transactions and simplifying the expense tracking process.

## 4. SOFTWAREUSEDINTHEPROJECT

Software is a set of instructions, data or programs used to operate computers and execute specific tasks. It is the opposite of hardware, which describes the physical aspects of a computer. Software is a generic term used to refer to applications, scripts and programs that run on a device. It can be thought of as the variable part of a computer, while hardware is the invariable part. Some of the main software which is used in the project is as follows:

- Java
- React js
- Node js
- Express js

The Editor used in the project is VS code. VS code is a platform which integrates the front end, server side and the back end to complete the project.

### **VSCODE:**

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET).

Visual Studio Code proper is built using the Electron shell, Node.js, TypeScript, and the Language Server Protocol, and is updated on a monthly basis. The many extensions are updated as often as needed. The richness of support varies across the different programming languages and their extensions, ranging from simple syntax highlighting and bracket matching to debugging and refactoring.

The code in the Visual Studio Code repository is open source under the MIT License. The Visual Studio Code product itself ships under a standard Microsoft product license, as it has a small percentage of Microsoft-specific customizations. It's free despite the commercial license.

Developers like Visual Studio Code's lightweight feel as an editor combined with its ability to check syntax, complete code, refactor code, debug, and check into a repository. Cloud and container developers like VS Code's remote capabilities and its explicit support

for major clouds, [Docker](#), and [Kubernetes](#). Developers who work in teams like VS Code's [Git](#) integration.

### **React js:**

React is a JavaScript library for building user interfaces. React is used to build single-page applications. React allows us to create reusable UI components. React creates a VIRTUAL DOM in memory. Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.

React is a front-end JavaScript library. React is capable of making API calls (sending the request to the backend), which deal with the data. React cannot process the database or the data source itself.

### **React Directly in HTML**

The quickest way start learning React is to write React directly in your HTML files.

```
<!DOCTYPE html>
<html>
<head>
<script src="https://unpkg.com/react@18/umd/react.development.js" cross
origin></script>
<script src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"crossorigin></script>
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
</head>
<body>
<div id="mydiv"></div>
<script type="text/babel">
function Hello() {
return <h1>Hello World!</h1>;
}
const container = document.getElementById('mydiv');
const root = ReactDOM.createRoot(container);
root.render(<Hello />)
</script>
```



</body>

</html>

## **Express JS:**

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy.

Express provides a thin layer of fundamental web application features, without obscuring Node.js features that you know and love. Many popular frameworks are based on Express.

Express is a node js web application framework that provides broad features for building web and mobile applications. It is used to build a single page, multipage, and hybrid web application. It's a layer built on the top of the Node js that helps manage servers and routes.

Some of the apps using Express js are as follows:

- Yummly
- Accuweather
- Accenture
- Myspace

## **NODE JS:**

*Node JS is open-source, cross-platform JavaScript code that runs on servers. It uses an asynchronous event-driven model and is designed to build scalable network applications. It's built on Google Chrome's V8 engine, which makes it fast and efficient to develop apps.*

Some of the apps using Node JS are as follows:

- Paypal
- Godaddy
- Netflix
- LinkedIn

## **Express vs Node: Performance Comparison**

### **Express JS**

Express is a minimal and flexible NodeJS framework that provides a robust set of features for web applications like routing, sessions, caching, etc. Comparing Node JS vs Node JS

Express, the latter manages all the clutter that comes with setting up a server and gives you a simple interface to build your routes and then use them for handling requests made by the users.

## **Node.JS**

Node.JS uses an event-driven, non-blocking I/O model which makes it lightweight and efficient for the server environment that has to handle multiple concurrent connections with low latency. Using NodeJS will allow you to create such an application because it has high performance and speed.

js web application framework that provides a robust set of features for web and mobile applications. In other words, Node JS is the package, which provides the JavaScript run-time environment, whereas Express is a framework that sits on top of NodeJS and helps us to handle requests and responses.

## **VSCODE:**

Visual Studio for Mac is a .NET integrated development environment on the Mac that can be used to edit, debug, and build code and then publish an app. In addition to a code editor and debugger, Visual Studio for Mac includes compilers, code completion tools, graphical designers, and source control features to ease the software development process.

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

## **Features**

VS Code supports a wide array of programming languages from Java, C++, and Python to CSS, Go, and Dockerfile. Moreover, VS Code allows you to add on and even creating new extensions including code linters, debuggers, and cloud and web development support.

The VS Code user interface allows for a lot of interaction compared to other text editors. To simplify user experience, VS Code is divided into five main regions:

- The activity bar
- The side bar
- Editor groups
- The panel

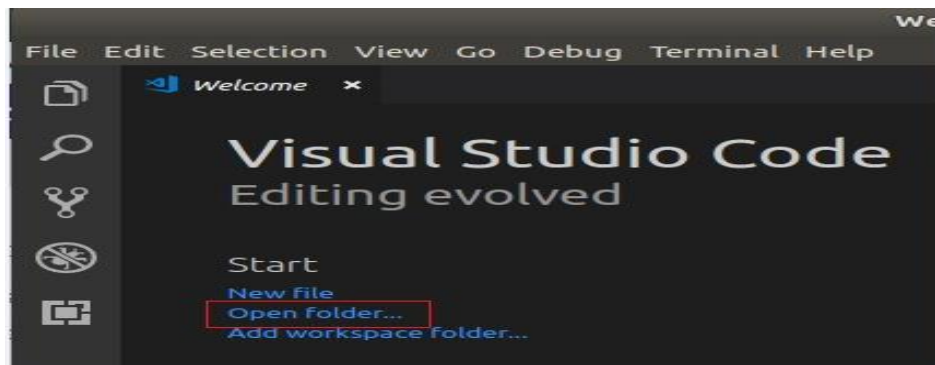
- The status bar

### Installation steps of VSCode:

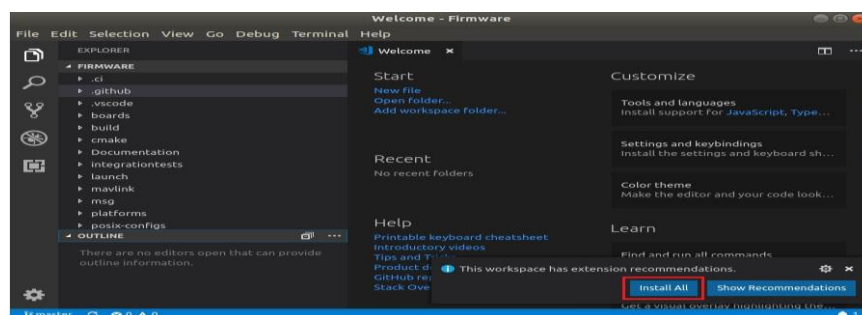
Download and install VSCode (opens new window)(you will be offered the correct version for your OS).

Open VSCode and add the PX4 source code:

- Select *Open folder ...* option on the welcome page (or using the menu: **File > Open Folder**):

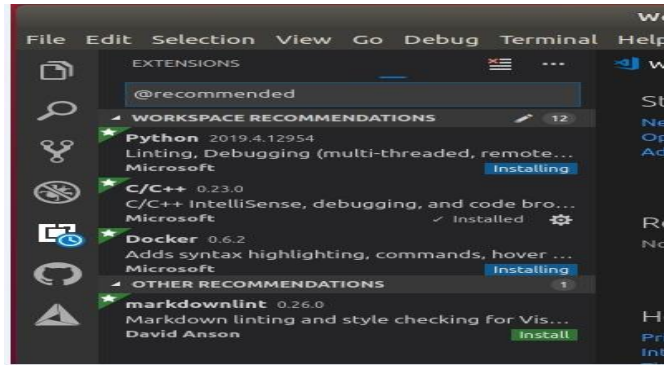


- A file selection dialog will appear. Select the PX4-Autopilot directory and then press OK.
- The project files and configuration will then load into VSCode.
- Press Install All on the This workspace has extension recommendations prompt (this will appear on the bottom right of the IDE).



- VSCode will open the *Extensions* panel on the left hand side so you can watch the progress of installation.

- A



number of  
notifications/prompts

may appear in the bottom right corner

#### TIP

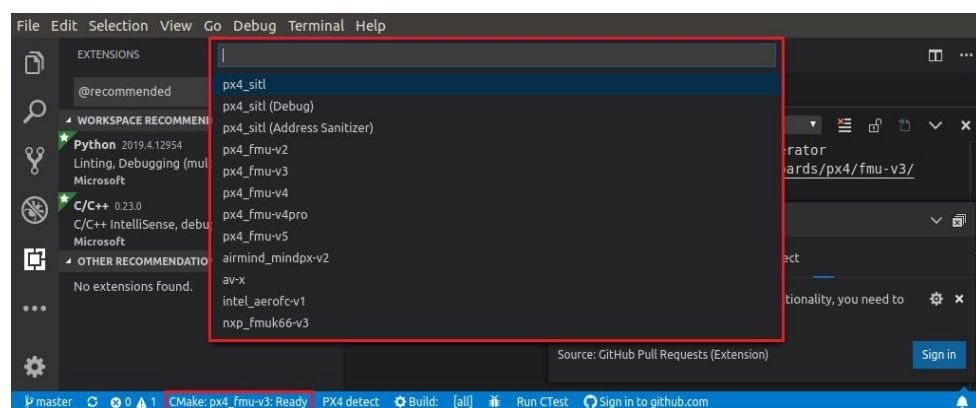
If the prompts disappear, click the little "alarm" icon on the right of the bottom blue bar.

- If prompted to install a new version of *cmake*:
  - Say **No** (the right version is installed with the PX4 developer environment).
- If prompted to sign into *github.com* and add your credentials:
  - This is up to you! It provides a deep integration between Github and the IDE, which may simplify your workflow.
  - Other prompts are optional, and may be installed if they seem useful.

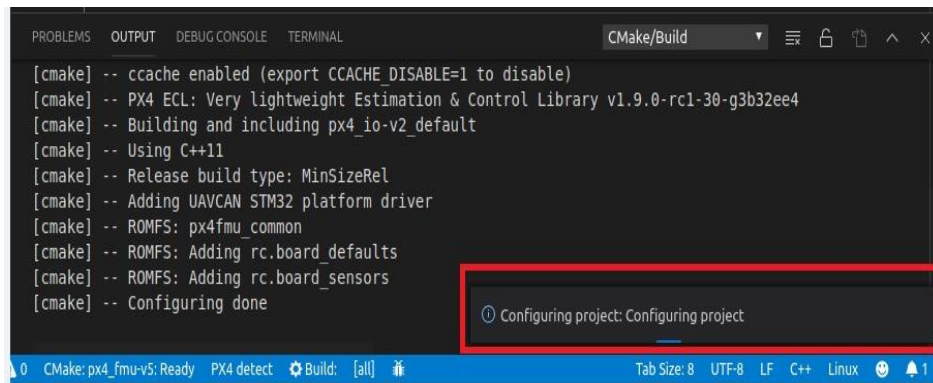
#### #Building PX4

To build: Select your build target ("cmake build config"):

The current cmake build target is shown on the blue config bar at the bottom (if this is already your desired target, skip to next step).



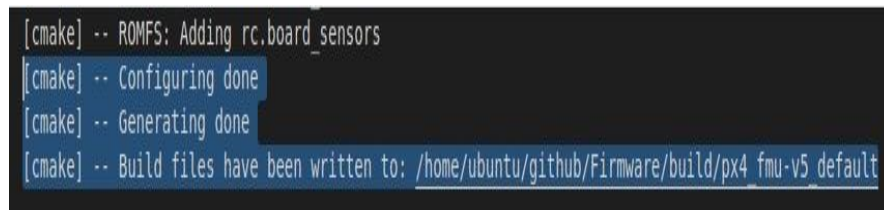
- Click the target on the config bar to display other options, and select the one you want (this will replace any selected target).
- *Cmake* will then configure your project (see notification in bottom right).



```
[cmake] -- ccache enabled (export CCache DISABLE=1 to disable)
[cmake] -- PX4 ECL: Very lightweight Estimation & Control Library v1.9.0-rc1-30-g3b32ee4
[cmake] -- Building and including px4_io-v2_default
[cmake] -- Using C++11
[cmake] -- Release build type: MinSizeRel
[cmake] -- Adding UAVCAN STM32 platform driver
[cmake] -- ROMFS: px4fmu_common
[cmake] -- ROMFS: Adding rc.board_defaults
[cmake] -- ROMFS: Adding rc.board_sensors
[cmake] -- Configuring done
```

Configuring project: Configuring project

Wait until configuration completes. When this is done the notification will disappear and you'll be shown the build location:



```
[cmake] -- ROMFS: Adding rc.board_sensors
[cmake] -- Configuring done
[cmake] -- Generating done
[cmake] -- Build files have been written to: /home/ubuntu/github/Firmware/build/px4_fmu-v5_default
```

You can then kick off a build from the config bar (select either **Build** or **Debug**).

After building at least once you can now use [code completion] (#code completion) and other *VSCode* features.

## 5. CODING

### Desktop\6thsem\ExpenseTracker\ET\client\public\index.html

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta
      name="description"
      content="Web site created using create-react-app"
    />

    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />

    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See
      https://developers.google.com/web/fundamentals/web-app-manifest/
    -->

    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-
      Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
      crossorigin="anonymous"
    />

    <title>Expense Managment System</title>

  </head>
```

```
<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

<div id="root"></div>

<!--

  This HTML file is a template.

  If you open it directly in the browser, you will see an empty page.

  You can add webfonts, meta tags, or analytics to this file.
  The build step will place the bundled scripts into the <body> tag.

  To begin the development, run `npm start` or `yarn start`.
  To create a production bundle, use `npm run build` or `yarn build`.

-->

</body>
</html>
```

#### **Desktop\6thsem\ExpenseTracker\ET\client\src\components\Layout\Footer.js**

```
import React from "react";

const Footer = () => {
  return (
    <div className="bg-dark text-light p-4">
      <h6 className="text-center">All rights reserved &copy; ET</h6>
    </div>
  );
};

export default Footer;
```

#### **Desktop\6thsem\ExpenseTracker\ET\client\src\components\Layout\Header.js**

```
import React, { useState, useEffect } from "react";
import { Link, useNavigate } from "react-router-dom";
import { message } from "antd";

const Header = () => {
  const [loginUser, setLoginUser] = useState("");
  const navigate = useNavigate();

  useEffect(() => {
    const user = JSON.parse(localStorage.getItem("user"));
    if (user) {
      setLoginUser(user);
    }
  }, []);

  const logoutHandler = () => {
    localStorage.removeItem("user");
    message.success("Logout Successfully");
    navigate("/login");
  };

  return (
    <nav className="navbar navbar-expand-lg bg-light">
      <div className="container-fluid">
        <button
          className="navbar-toggler"
          type="button"
          data-bs-toggle="collapse"
          data-bs-target="#navbarTogglerDemo01"
          aria-controls="navbarTogglerDemo01"

```



```
        aria-expanded="false"
        aria-label="Toggle navigation"
    >
    <span className="navbar-toggler-icon" />
</button>
<div className="collapse navbar-collapse" id="navbarTogglerDemo01">
    <Link className="navbar-brand" to="/">
        Expense Tracker
    </Link>
    <ul className="navbar-nav ms-auto mb-2 mb-lg-0">
        <li className="nav-item">
            {loginUser && <p className="nav-link">{loginUser.name}</p>}
        </li>
        <li className="nav-item">
            <button className="logout-button" onClick={logoutHandler}>
                Logout
            </button>
        </li>
    </ul>
</div>
</div>
</nav>
);
};
export default Header;
```

**Desktop\6thsem\ExpenseTracker\ET\client\src\components\Layout\Layout.js**

```
import React from "react";
```

```
import Footer from "./Footer";
import Header from "./Header";

const Layout = ({ children }) => {
  return (
    <
      <Header />
      <div className="content">{children}</div>
      <Footer />
    </>
  );
};

export default Layout;
```

### **Desktop\6thsem\ExpenseTracker\ET\client\src\components\Analytics.js**

```
import React from "react";
import { Progress } from "antd";

const Analytics = ({ allTransaction }) => {
  // category
  const categories = [
    "salary",
    "tip",
    "bill",
    "food",
    "tax",
    "project",
    "medicine",
  ]
```

```
    "fees",
  ];
  // totaltransaction
  const totalTransaction = allTransaction.length;
  const totalIncomeTransactions = allTransaction.filter(
    (transaction) => transaction.type === "income"
  );
  const totalExpenseTransactions = allTransaction.filter(
    (transaction) => transaction.type === "expense"
  );
  const totalIncomePercent =
    (totalIncomeTransactions.length / totalTransaction) * 100;
  const totalExpensePercent =
    (totalExpenseTransactions.length / totalTransaction) * 100;

  // totalturnover
  const totalTurnover = allTransaction.reduce(
    (acc, transaction) => acc + transaction.amount,
    0
  );
  const totalIncomeTurnover = allTransaction
    .filter((transaction) => transaction.type === "income")
    .reduce((acc, transaction) => acc + transaction.amount, 0);
  const totalExpenseTurnover = allTransaction
    .filter((transaction) => transaction.type === "expense")
    .reduce((acc, transaction) => acc + transaction.amount, 0);

  const totalIncomeTurnoverPercent =
    (totalIncomeTurnover / totalTurnover) * 100;
  const totalExpenseTurnoverPercent =
```

```
(totalExpenseTurnover / totalTurnover) * 100;

return (
  <
    <div className="row m-3">
      <div className="col-md-4">
        <div className="card">
          <div className="card-header">
            total transactions: {totalTransaction}
          </div>
          <div className="card-body">
            <h5 className="text-success">
              Income: {totalIncomeTransactions.length}
            </h5>
            <h5 className="text-danger">
              Expense: {totalExpenseTransactions.length}
            </h5>
            <div>
              <Progress
                type="circle"
                strokeColor={"green"}
                className="mx-2"
                percent={totalIncomePercent.toFixed(0)}
              />
              <Progress
                type="circle"
                strokeColor={"red"}
                className="mx-2"
                percent={totalExpensePercent.toFixed(0)}
              />
            </div>
          </div>
        </div>
      </div>
    </div>
  </

```

```

        </div>
    </div>
</div>
</div>
<div className="col-md-4">
    <div className="card">
        <div className="card-header">total Turnover: {totalTurnover}</div>
        <div className="card-body">
            <h5 className="text-success">Income: {totalIncomeTurnover}</h5>
            <h5 className="text-danger">Expense: {totalExpenseTurnover}</h5>
            <div>
                <Progress
                    type="circle"
                    strokeColor={"green"}
                    className="mx-2"
                    percent={totalIncomeTurnoverPercent.toFixed(0)}
                />
                <Progress
                    type="circle"
                    strokeColor={"red"}
                    className="mx-2"
                    percent={totalExpenseTurnoverPercent.toFixed(0)}
                />
            </div>
        </div>
    </div>
</div>
</div>
</div>
</div>
<div className="row mt-3">
    <div className="col-md-5">

```

```

<h4>Categorywise income</h4>
{categories.map((category) => {
  const amount = allTransaction
    .filter(
      (transaction) =>
        transaction.type === "income" &&
        transaction.category === category
    )
    .reduce((acc, transaction) => acc + transaction.amount, 0);
  return(
    amount > 0 && (
      <div className="card">
        <div className="card-body">
          <h5>{category}</h5>
          <Progress
percent={(((amount/totalIncomeTurnover)*100).toFixed(0))}/>
        </div>
      </div>
    )
  );
}})
</div>

<div className="col-md-5">
<h4>Categorywise expense</h4>
{categories.map((category) => {
  const amount = allTransaction
    .filter(
      (transaction) =>
        transaction.type === "expense" &&
        transaction.category === category
    )

```

```

        .reduce((acc, transaction) => acc + transaction.amount,0);
    return(
        amount>0&&(
            <div className="card">
                <div className="card-body">
                    <h5>{category}</h5>
                    <Progress
percent={(((amount/totalExpenseTurnover)*100).toFixed(0))}/>
                </div>
            </div>
        )
    );
    }
}
</div>
</div>
</>
);
};

```

```
export default Analytics;
```

### **Desktop\6thsem\ExpenseTracker\ET\client\src\components\Spinner.js**

```

import React from "react";

const Spinner = () => {
    return (
        <div className="d-flex justify-content-center">
            <div className="spinner-border" role="status">
                <span className="visually-hidden">Loading...</span>
            </div>
        </div>
    )
}

```

```
        </div>

    </>

    );

};

export default Spinner;
```

### **Desktop\6thsem\ExpenseTracker\ET\client\src\pages\HomePage.js**

```
import React, { useState, useEffect } from "react";
import { Modal, Form, Input, Select, message, Table, DatePicker } from "antd";
import {
    UnorderedListOutlined,
    AreaChartOutlined,
    EditOutlined,
    DeleteOutlined,
} from "@ant-design/icons";
import Layout from "../components/Layout/Layout";
import axios from "axios";
import Spinner from "../components/Spinner";
import moment from "moment";
import "../index.css";
import Analytics from "../components/Analytics";
const { RangePicker } = DatePicker;

const HomePage = () => {
    const [showModal, setShowModal] = useState(false);
    const [loading, setLoading] = useState(false);
    const [allTransaction, setAllTransaction] = useState([]);
    const [frequency, setFrequency] = useState("7");
```



```
const [selectedDate, setSelecteddate] = useState([]);
const [type, setType] = useState("all");
const [viewData, setViewData] = useState("table");
const [editable,setEditable]=useState(null)

// table data
const columns = [
  {
    title: "Date",
    dataIndex: "date",
    render: (text) => <span>{moment(text).format("YYYY-MM-DD")}</span>,
  },
  {
    title: "Amount",
    dataIndex: "amount",
  },
  {
    title: "Type",
    dataIndex: "type",
  },
  {
    title: "Category",
    dataIndex: "category",
  },
  {
    title: "Reference",
    dataIndex: "reference",
  },
  {
    title: "Actions",
```

```
render:(text,record)=>(  
  <div>  
    <EditOutlined onClick={()=>{  
      setEditable(record)  
      setShowModal(true)  
    }}/>  
    <DeleteOutlined className="mx-2" onClick={()=>{handleDelete(record)}}/>  
  </div>  
)  
,  
];  
  
// useEffect Hook  
useEffect(() => {  
  // Get all transactions  
  const getAllTransaction = async () => {  
    try {  
      const user = JSON.parse(localStorage.getItem("user"));  
      setLoading(true);  
      const res = await axios.post("/transections/get-transection", {  
        userid: user._id,  
        frequency,  
        selectedDate,  
        type,  
      });  
      setLoading(false);  
      setAllTransaction(res.data);  
      console.log(res.data);  
    } catch (error) {  
      console.log(error);  
      message.error(" fetch Issue with transactions");  
    }  
  }  
});
```

```
    }  
  };  
  
  getAllTransaction();  
}, [frequency, selectedDate, type]);  
  
// delete handler  
const handleDelete=async(record)=>{  
  try {  
    setLoading(true)  
    await axios.post("/transections/delete-transection",{transectionId:record._id})  
    setLoading(false)  
    message.success("transaction deleted");  
  } catch (error) {  
    setLoading(false)  
    console.log(error);  
    message.error('unable to delete');  
  }  
};  
  
// Form handling  
const handleSubmit = async (values) => {  
  try {  
    const user = JSON.parse(localStorage.getItem("user"));  
    setLoading(true);  
    // await axios.post("/transections/add-transection", {  
    //   ...values,  
    //   userid: user._id,  
    //   type: values.type,  
    // });
```

```
if(editable){
  await axios.post("/transections/edit-transection",{
    payload:{
      ...values,
      userId:user._id,

    },
    transectionId:editable._id
  });
  setLoading(false);
  message.success("Transaction updated successfully");

} else{
  await axios.post("/transections/add-transection",{
    ...values,
    userid:user._id,
  });
}
setLoading(false);
message.success("Transaction added successfully");
setShowModal(false);
setEditable(null);
} catch (error) {
  setLoading(false);
  message.error("Failed to add transaction");
}
};

return (
  <Layout>
```

```

{loading && <Spinner />}
<div className="filters">
  <div>
    <h6>Select Frequency</h6>
    <Select value={frequency} onChange={(values) => setFrequency(values)}>
      <Select.Option value="7">LAST 1 Week</Select.Option>
      <Select.Option value="31">LAST 1 Month</Select.Option>
      <Select.Option value="365">LAST 1 Year</Select.Option>
      <Select.Option value="custom">Custom</Select.Option>
    </Select>
    {frequency === "custom" && (
      <RangePicker
        value={selectedDate}
        onChange={(values) => setSelectedDate(values)}
      />
    )}
  </div>
  <div>
    <h6>Select Type</h6>
    <Select value={type} onChange={(values) => setType(values)}>
      <Select.Option value="all">All</Select.Option>
      <Select.Option value="income">Income</Select.Option>
      <Select.Option value="expense">Expense</Select.Option>
    </Select>
    {frequency === "custom" && (
      <RangePicker
        value={selectedDate}
        onChange={(values) => setSelectedDate(values)}
      />
    )}
  </div>
</div>

```

```

</div>

<div className="switch-icons">
  <UnorderedListOutlined
    className={`mx-2 ${
      viewData === "table" ? "active-icon" : "inactive-icon"
    }}
    onClick={() => setViewData("table")}
  />
  <AreaChartOutlined
    className={`mx-2 ${
      viewData === "analytics" ? "active-icon" : "inactive-icon"
    }}
    onClick={() => setViewData("analytics")}
  />
</div>
<div>
  <button
    className="btn btn-primary"
    onClick={() => setShowModal(true)}
  >
    Add New
  </button>
</div>
</div>
<div className="content">
  {viewData === "table" ? (
    <Table columns={columns} dataSource={allTransaction} />
  ) : (
    <Analytics allTransaction={allTransaction} />
  )}

```

```
    )}
  </div>

  <Modal
    title={editable?"edit transaction":"add transaction"}
    open={showModal}
    onCancel={() => setShowModal(false)}
    footer={null}
  >
    <Form layout="vertical" onFinish={handleSubmit} initialValues={editable}>
      <Form.Item label="Amount" name="amount">
        <Input type="text" />
      </Form.Item>
      <Form.Item
        label="Type"
        name="type"
        rules={[{ required: true, message: "Please select a type" }]}
      >
        <Select>
          <Select.Option value="income">Income</Select.Option>
          <Select.Option value="expense">Expense</Select.Option>
        </Select>
      </Form.Item>
      <Form.Item label="Category" name="category">
        <Select>
          <Select.Option value="salary">Salary</Select.Option>
          <Select.Option value="tip">Tip</Select.Option>
          <Select.Option value="bill">Bill</Select.Option>
          <Select.Option value="food">Food</Select.Option>
          <Select.Option value="tax">Tax</Select.Option>
          <Select.Option value="project">Project</Select.Option>
        </Select>
      </Form.Item>
    </Form>
  </Modal>
</div>
```

```

        <Select.Option value="medicine">Medicine</Select.Option>
        <Select.Option value="fees">Fees</Select.Option>
    </Select>
</Form.Item>
<Form.Item label="Date" name="date">
    <Input type="date" />
</Form.Item>
<Form.Item label="Reference" name="reference">
    <Input type="text" />
</Form.Item>
<Form.Item label="Description" name="description">
    <Input type="text" />
</Form.Item>
<div className="d-flex justify-content-end">
    <button type="submit" className="btn btn-primary">
        SAVE
    </button>
</div>
</Form>
</Modal>
</Layout>
);
};

```

```
export default HomePage;
```

**Desktop\6thsem\ExpenseTracker\ET\client\src\pages\Login.js**

```

import React, { useState, useEffect } from "react";
import { Form, Input, message } from "antd";
import { Link, useNavigate } from "react-router-dom";

```



```
import axios from "axios";
import Spinner from "../components/Spinner";

const Login = () => {
  const [loading, setLoading] = useState(false);
  const navigate = useNavigate();

  // Form submit
  const submitHandler = async (values) => {
    try {
      setLoading(true);
      const { data } = await axios.post("/users/login", values);
      setLoading(false);
      message.success("Login success");
      localStorage.setItem("user", JSON.stringify({ ...data.user, password: "" }));
      navigate("/");
    } catch (error) {
      setLoading(false);
      message.error("Something went wrong");
    }
  };

  // Prevent for logged-in user
  useEffect(() => {
    if (localStorage.getItem("user")) {
      navigate("/");
    }
  }, [navigate]);

  return (
```

```
<div className="login-page">
  {loading && <Spinner />}
  <Form layout="vertical" onFinish={submitHandler} className="login-form">
    <h1>Login Form</h1>
    <Form.Item
      label="Email"
      name="email"
      rules={[
        { required: true, message: "Please enter your email" },
        { type: "email", message: "Please enter a valid email" },
      ]}
    >
      <Input type="email" />
    </Form.Item>
    <Form.Item
      label="Password"
      name="password"
      rules={{ { required: true, message: "Please enter your password" } }}
    >
      <Input type="password" />
    </Form.Item>
    <div className="d-flex">
      <div className="register-link">
        <Link to="/register">Not a user? Click Here to Register</Link>
      </div>
      <div className="login-button">
        <button className="btn btn-primary">Login</button>
      </div>
    </div>
  </Form>
</div>
```

```
    </div>

  );
};

export default Login;
```

### **Desktop\6thsem\ExpenseTracker\ET\client\src\pages\Register.js**

```
import React, { useState, useEffect } from "react";
import { Form, Input, message } from "antd";
import { Link, useNavigate } from "react-router-dom";
import axios from "axios";
import Spinner from "../components/Spinner";

const Register = () => {
  const navigate = useNavigate();
  const [loading, setLoading] = useState(false);

  // Form submit
  const submitHandler = async (values) => {
    try {
      setLoading(true);
      await axios.post("/users/register", values);
      message.success("Registration Successful");
      setLoading(false);
      navigate("/login");
    } catch (error) {
      setLoading(false);
      message.error("Something went wrong");
    }
  };
};
```

```
    }  
  };  
  
  // Prevent for logged-in user  
  useEffect(() => {  
    if (localStorage.getItem("user")) {  
      navigate("/");  
    }  
  }, [navigate]);  
  
  return (  
    <div className="register-page">  
      {loading && <Spinner />}  
      <Form  
        layout="vertical"  
        onFinish={handleSubmit}  
        className="register-form"  
      >  
        <h1>Register Form</h1>  
        <Form.Item  
          label="Name"  
          name="name"  
          rules={[  
            { required: true, message: "Please enter your name" },  
            { min: 3, message: "Name must be at least 3 characters long" },  
          ]}  
        >  
          <Input />  
        </Form.Item>  
        <Form.Item
```

```
        label="Email"
        name="email"
        rules={
          { required: true, message: "Please enter your email" },
          { type: "email", message: "Please enter a valid email" },
        }
      >
      <Input type="email" />
    </Form.Item>
    <Form.Item
      label="Password"
      name="password"
      rules={
        { required: true, message: "Please enter your password" },
        { min: 6, message: "Password must be at least 6 characters long" },
      }
    >
      <Input type="password" />
    </Form.Item>
    <div className="register-buttons">
      <div className="register-link">
        <Link to="/login">Already Registered? Click Here to Login</Link>
      </div>
      <div className="register-button">
        <button className="btn btn-primary">Register</button>
      </div>
    </div>
  </Form>
</div>
);
```

```
};
```

```
export default Register;
```

### **Desktop\6thsem\ExpenseTracker\ET\client\src\App.js**

```
import { Routes, Route, Navigate } from "react-router-dom";
```

```
import HomePage from "../pages/HomePage";
```

```
import Login from "../pages/Login";
```

```
import Register from "../pages/Register";
```

```
function App() {
```

```
  return (
```

```
    <>
```

```
    <Routes>
```

```
      <Route
```

```
        path="/"
```

```
        element={
```

```
          <ProtectedRoutes>
```

```
            <HomePage />
```

```
          </ProtectedRoutes>
```

```
        }
```

```
      />
```

```
      <Route path="/register" element={<Register />} />
```

```
      <Route path="/login" element={<Login />} />
```

```
    </Routes>
```

```
  </>
```

```
);
```

```
}
```

```
export function ProtectedRoutes(props) {  
  if (localStorage.getItem("user")) {  
    return props.children;  
  } else {  
    return <Navigate to="/login" />;  
  }  
}
```

```
export default App;
```

### **Desktop\6thsem\ExpenseTracker\ET\client\src\index.css**

```
* {  
  padding: 0;  
  margin: 0;  
  box-sizing: border-box;  
}  
  
.content {  
  height: 120vh;  
}  
  
.register-page {  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
  justify-content: center;  
  height: 100vh;  
  background-image:  
url("https://tse4.mm.bing.net/th?id=OIP.NfFhTSxEndiClX3FbMakxQHAE8&pid=Api  
&P=0&h=180");  
  background-size: cover;
```

```
}

.register-form {
  width: 300px;
}

.register-form .ant-form-item {
  margin-bottom: 10px;
}

.register-buttons {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-top: 10px;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.register-buttons .register-link {
  flex: 1;
  margin-right: 10px;
}

.register-buttons .register-button {
  margin-left: 10px;
}

/* .resgister-page {
  display: flex;
```



```
    align-items: center;
    justify-content: center;
    height: 100vh;
} */
/* CSS for Login page */
.login-page {
    position: relative;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
    background-image:
url("https://tse2.mm.bing.net/th?id=OIP.FLeRI6metHwujMau6_MAEwHaEK&pid=A
pi&P=0&h=180");
    background-size: cover;
}

.login-page::before {
    content: "";
    position: absolute;
    top: -10px;
    right: -10px;
    bottom: -10px;
    left: -10px;
    border: 10px ;
    border-radius: 10px;
    z-index: -1;
}

.login-form {
```

```
width: 300px;
padding: 20px;
border: 1px solid #ccc;
border-radius: 5px;
box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);
}

.login-form .ant-form-item {
  margin-bottom: 10px;
}

.login-form .d-flex {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-top: 10px;
}

.login-form .d-flex .register-link {
  flex: 1;
  text-align: left;
}

.login-form .d-flex .login-button {
  margin-left: 10px;
}

/* CSS for Header */
.navbar {
  background: linear-gradient(to right, #ff6b6b, #0c00ff);
  color: #f4eded;
```

```
padding: 10px 20px;  
box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
font-family: 'Courier New', Courier, monospace;  
}
```

```
.navbar-brand {  
  font-size: 20px;  
  font-weight: bold;  
  color: #000;  
  text-decoration: none;  
}
```

```
.navbar-nav .nav-item {  
  margin-left: 10px;  
}
```

```
.nav-link {  
  color: #000;  
  text-decoration: none;  
}
```

```
.logout-button {  
  padding: 6px 10px;  
  background-color: #007bff;  
  color: #fff;  
  border: none;  
  border-radius: 5px;  
  cursor: pointer;  
}
```

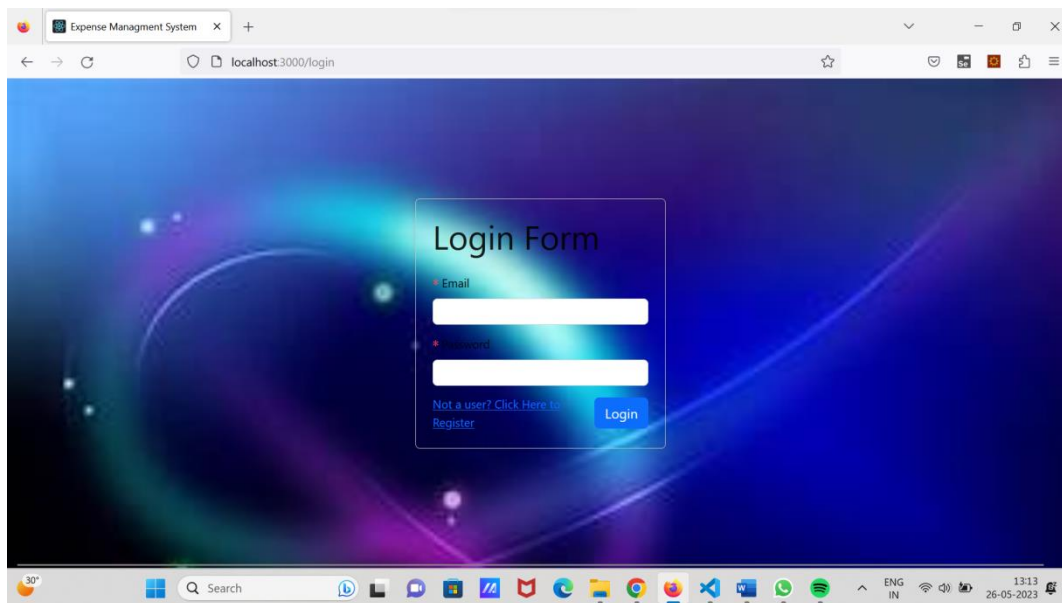
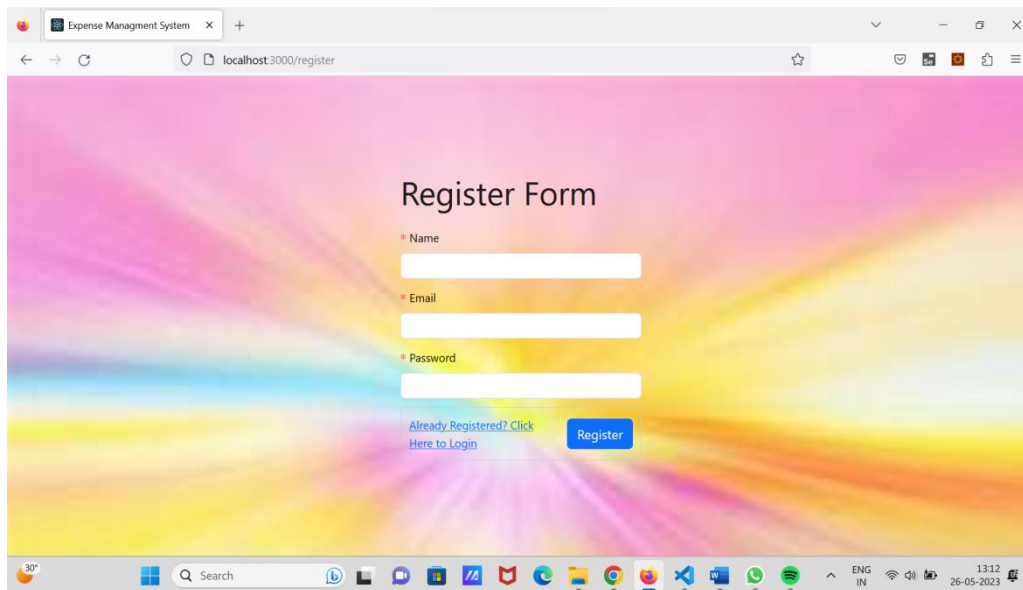
```
.logout-button:hover {  
  background-color: #0056b3;  
}  
  
.navbar-toggler {  
  border: none;  
  padding: 0;  
  background-color: transparent;  
}  
  
.navbar-toggler .navbar-toggler-icon {  
  background-color: #000;  
  width: 20px;  
  height: 2px;  
}  
  
/* transaction page*/  
.filters{  
  display: flex;  
  align-items: center;  
  justify-content: space-between;  
  padding: 15px 20px;  
  box-shadow: 0 0 3px gray;  
}  
  
/*atnd icons*/  
.switch-icons {  
  border: 1px solid black rgba(0, 0, 0, 0.677);  
  border-radius: 5px;  
  padding: 10px 15px;  
}
```

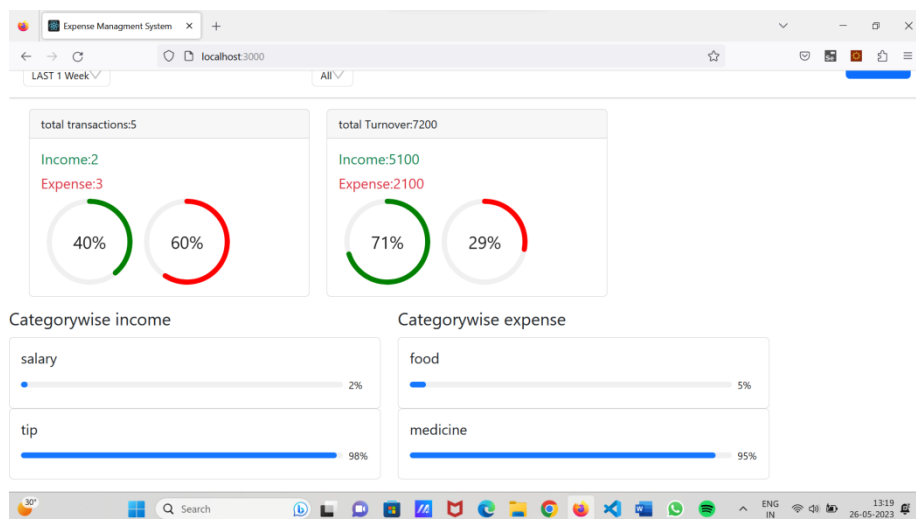
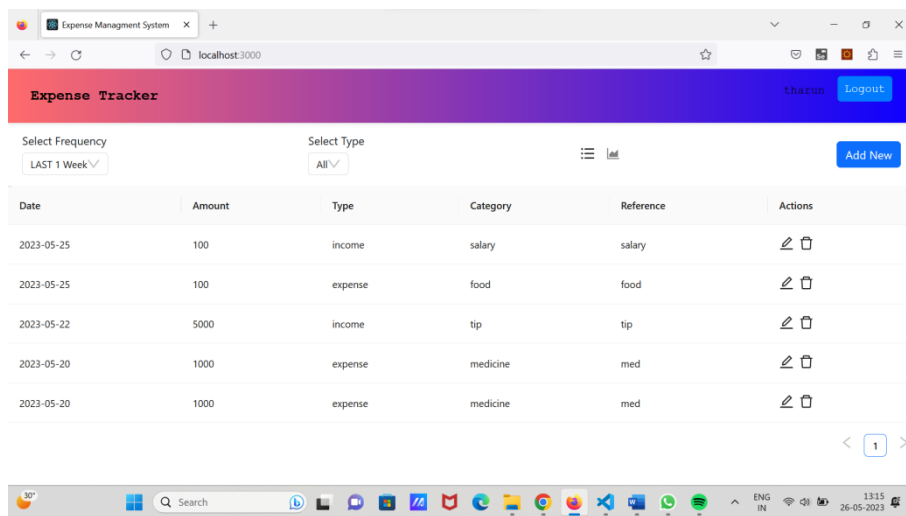
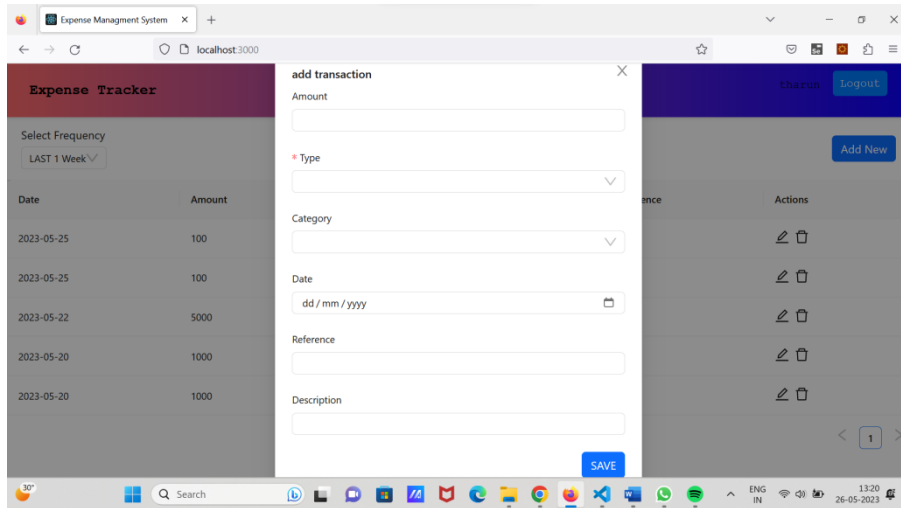
```
.anticon svg {  
  font-size: 20px;  
  cursor: pointer;  
}  
  
.active-icon {  
  color: black;  
}  
  
.inactive-icon {  
  color: gray;  
}
```

**Desktop\6thsem\ExpenseTracker\ET\client\src\index.js**

```
import React from "react";  
import ReactDOM from "react-dom/client";  
import "antd/dist/reset.css";  
import "./index.css";  
import App from "./App";  
import { BrowserRouter } from "react-router-dom";  
import reportWebVitals from "./reportWebVitals";  
  
const root = ReactDOM.createRoot(document.getElementById("root"));  
root.render(  
  <React.StrictMode>  
    <BrowserRouter>  
      <App />  
    </BrowserRouter>  
  </React.StrictMode>  
>);  
reportWebVitals();
```

## 6. SCREEN SHOTS





## 7. USECASE DIAGRAM

### USE CASE DIAGRAM:

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

Use-case diagrams are helpful in the following situations:

- Before starting a project, you can create use-case diagrams to model a business so that all participants in the project share an understanding of the workers, customers, and activities of the business.
- While gathering requirements, you can create use-case diagrams to capture the system requirements and to present to others what the system should do.
- During the analysis and design phases, you can use the use cases and actors from your use-case diagrams to identify the classes that the system requires.
- During the testing phase, you can use use-case diagrams to identify tests for the system.

A use case diagram should visualize a reason (use case) why an individual (actor) would interact with your organization (system)

