INTERNS ROLES AND RESPONSIBILITY

Software developer responsible as intern is for assisting the software engineers with the design implementation and shadowing the duties is to understand the processes more efficiently. Software developer interns use their knowledge on practical applications, suggesting recommendations on how to make the project successful. They also respond to clients' inquiries and concerns, attend meetings, and help with maintaining complex systems and networks. A software intern will communicate with their officials about their ongoing project.

SOFTWARE ENGINEER INTERN DUTIES AND RESPONSIBILITIES

- Developing applications (coding, programming)
- Debugging and testing code
- Documenting and testing new software applications
- Researching, investigating and fixing a wide range of technical issues
- Collaborating with senior leaders
- Approaching problems and creating solutions
- Proactively learning about new technologies

SOFTWARE DEVELOPER INTERNSHIP WORK PROFILE:

- Update web app remotely to be more responsive and user facing with HTML, REACT js, Material UI and CSS.
- Implement test client in Java, J2EE and run test code in windows
- Establish and update databases system using Java, J2EE, HTML, MongoDB.
- Code review and test for teammates, ensuring JavaScript code to be testable and CSS to be accessible and reusable.
- Utilize git for maintaining code base for back-end and front-end.

DATAFLOW DIAGRAM

What is a data flow diagram?

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled.

DATA FLOW

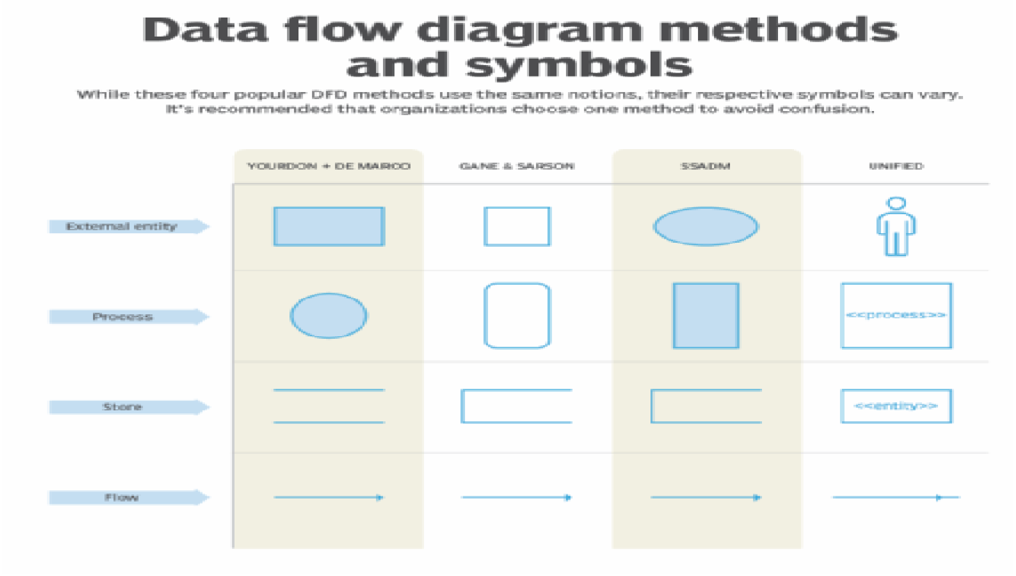
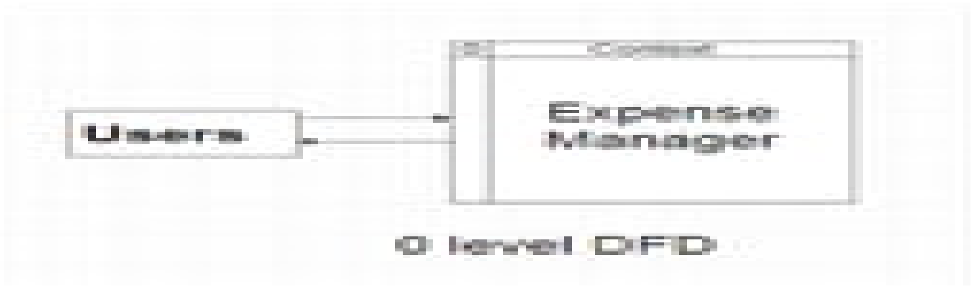

Data flow diagram methods and symbols
While these four popular DFD methods use the same notions, their respective symbols can vary.
It's recommended that organizations choose one method to avoid confusion.
YOURDON + DE MARCO
GANE & SARSON
SSADM
UNIFIED
External entity
Process
Store
Flow
<<process>>
<<entity>>

DIAGRAM OF EXPENSES TRACKER:

1. **Level 0 DFD:** This is the highest-level DFD, which provides an overview of the entire system. It shows the major processes, data flows, and data stores in the system, without providing any details about the internal workings of these processes.

Level 0 DFD



DATABASE

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS).In This project we have used the MONGO Database as back end.

MONGO DATABASE:

MongoDB is a non-relational document database that provides support for JSON-like storage. The MongoDB database has a flexible data model that enables you to store unstructured data, and it provides full indexing support, and replication with rich and intuitive APIs.

MONGODB DATABASE FEATURES

MongoDB has become popular with developers in part due to the its intuitive API, flexible data model, and features that include:

- **Ad-hoc queries**

  MongoDB supports field, range, and regular-expression queries which can return entire documents, specific fields of documents, or random samples of results.
- **Indexing**

  Fields in a MongoDB document can be indexed with primary and secondary indices. MongoDB supports a number of different index types, including single field, compound (multiple fields), multikey (array), geospatial, text, and hashed.
- **Replication**

  MongoDB provides high availability with replica sets including two or more copies of the data. Writes are handled by the primary replica, while any replica is capable of serving read requests. If the primary replica fails, a secondary replica is promoted to become the primary replica.d

TABLES IN MONGODB

Two tables have been created in this Vyapar system one to create the invoice and the one to keep the user details. The database and the table information is given below

DATA DICTIONARY:

USERS DETAIL TABLE

FIELD NAME	DATA TYPE	DESCRIPTION	CONSTRAINTS
_id	Number	Id, Auto generated	Primary key
name	varchar	Name of the user	Not null
email	varchar	Email of the user	Not null
password	varchar	Login password	Not null

TRANSACTIONS TABLE

FIELD NAME	DATA TYPE	DESCRIPTION	CONSTRAINTS
id	Number	Id Auto generated	Primary key
userid	varchar	Userid Auto generated	
amount	Number	User add amount	Not null
type	varchar	User select income or expense	Not null
category	varchar	User select category	Not null
reference	varchar	User reference	Not null
description	varchar	User description	Not null
date	date	User add date	Not null

Data base name :**expanseApp.**

```
{

 "title": "users",
  "properties": {
  "_id": {"$oid": "ObjectId" },
  "name{“bsonType”:"String"},
  "email": {“bsonType”:"String"},
  "password": {“bsonType”:"String"},
  "createdAt": { "$date": "Date" },
  "updatedAt": {"$date": "Date"},
  "__v": Int32
}
```

```
{
  "title": "transections",
  "properties":{
  "_id": {"$oid": "ObjectId" },
  "userid": {“bsonType”:"String"},
  "amount": Int32,
  "type": {“bsonType”:"String"},
  "category": {“bsonType”:"String"},
  "refrence": {“bsonType”:"String"},
  "description": {“bsonType”:"String"},

  "date": {"$date": "Date"},
  "createdAt": {"$date": "Date"},
  "updatedAt": {"$date": "Date"},
  "__v": Int32,
  "reference": {“bsonType”:"String"},
}
```

STRUCTURE OF THE PROJECT:

a) MODULES:

A module is a collection of source files and build settings that let you divide your project into discrete units of functionality. Your project can have one or many modules, and one module can use another module as a dependency. You can independently build, test, and debug each module. The Modules used in this project is as follows:

- Login Module
- Register Module
- Home Page (Introduction to project and features of Project)
- View Expenses (table and analytic)
- Add Expenses
- Logout Module

- **LOGIN MODULE:** This Module is used to enter into the homepage with the authenticated of registered email and password.

- **REGISTER MODULE:** This Module is used to do a new registration for an new user into the application by their name, email, password.
- **HOME PAGE:** This module is used to display the detailed information about the Expense tracker.
- **VIEWEXPENSES:** This module is used to display the list of the Expenses by table and analytic.
- **ADD EXPENSES:** This module is used to add the new expense.
- **LOGOUT:** This module is used to logout for the application.

b) <u>REPORTS</u>

- User information
- Transaction information

c.<u>TESTING FUNDAMENTALS:</u>

Software testing is an important element of S/W quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of S/W as a system element and the costs associated with a S/W failure are motivating forces for well planned, through testing.

Though the test phase is often thought of as separate and distinct from the development effort--first develop, and then test--testing is a concurrent process that provides valuable information for the development team.

TESTING OBJECTIVES

There are several rules that can serve as testing objectives. They are

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is one that has a high probability of finding an undiscovered error.

A successful test is one that uncovers an undiscovered error. If testing is conducted successfully according to the objectives stated above, it will uncover errors in the software. Also, testing demonstrates that software functions appear to the working according to specification, that performance requirements appear to have been met.

UNIT TESTING

Unit testing is the process of taking a module and running it in isolation from the rest of the software product by using prepared test cases and comparing the actual results with the results predicated by the specification and design of the module. One purpose of testing is to find and remove as many errors in the software as practical. There are number of reasons in support of unit testing then testing the entire product.

1. The size of a single module is small enough that we can locate an error fairly easily.
2. The module is small enough that we can attempt to test it in some demonstrable exhaustive fashion.
3. Confusing interaction s of multiple error in widely different parts of the software are eliminated.

Unit testing focuses the verification effort on the smallest unit of S/W design i.e., the module. The unit testing is always white-box oriented and the step can be conducted in parallel for modules.

INTEGRATION TESTING

Integration testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated with interfacing. The objective is to take unit-tested modules and build a program structure that has been dictated by design.

d. IMPLEMENTATION

Once the system has been designed, the next step is to convert the designed one in to actual code, so as to satisfy the user requirements as excepted. If the system is approved to be error free it can be implemented.

When the initial design was done for the system, the department was consulted for acceptance of the design so that further proceedings of the system development can be carried on. After the development of the system a demonstration was given to them about working of the system. The aim of the system illustration was to identify any malfunctioning of the system.

Implementation includes proper training to end-users. The implemented software should be maintained for prolonged running of the software.

Initially the system was run parallel with manual system. The system has been tested with data and has proved to be error-free and user-friendly. Training was given to end -user about the software and its features.

CODING

Frontend

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"
```

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;
```

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 />
</>
  );
```

```
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-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbc
nCeuOxjzrPF/et3URy9Bv1WTRi"
crossorigin="anonymous"
    />
<title>Expense Managment System</title>
```

Contd..

<u>client\src\components\Layout\Header.js</u>

```
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);
    }
  }, []);
constlogoutHandler = () => {
```

```
};
export default Layout;
```

<u>client\src\components\Analytics.js</u>

```
import React from "react";
import { Progress } from "antd";
const Analytics = ({ allTransaction }) => {
  // category
const categories = [
    "salary",
    "tip",
    "bill",
    "food",
    "tax",
    "project",
    "medicine",
    "fees",
  ];
  // totaltransaction
consttotalTransaction =
allTransaction.length;
consttotalIncomeTransactions =
allTransaction.filter(
    (transaction) =>transaction.type ===
"income"
  );
```

```
localStorage.removeItem("user");
message.success("Logout Successfully");
    navigate("/login");
  };
  return (
<nav className="navbar navbar-expand-lgbg-
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>
<ulclassName="navbar-nav ms-auto mb-2 mb-
lg-0">
```

Contd..

client\src\components\Spinner.js

```
import React from "react";
const Spinner = () => {
```

```
consttotalExpenseTransactions =
allTransaction.filter(
    (transaction) =>transaction.type ===
"expense"
  );
consttotalIncomePercent =
    (totalIncomeTransactions.length /
totalTransaction) * 100;
consttotalExpensePercent =
    (totalExpenseTransactions.length /
totalTransaction) * 100;

  // totalturnover
consttotalTurnover =
allTransaction.reduce(
    (acc, transaction) =>acc +
transaction.amount,
    0
  );
```

Contd..

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,
  return (
<>
<div className="d-flex justify-content-center">
<div className="spinner-border" role="status">
<span className="visually-
hidden">Loading...</span>
</div>
</div>
</>
  );
};
export default Spinner;
```

<u>client\src\index.js</u>

```
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.getElementBy
Id("root"));
root.render(
<React.StrictMode>
<BrowserRouter>
<App />
```

```
} 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;
constHomePage = () => {
const [showModal, setShowModal] =
useState(false);
const [loading, setLoading] =
useState(false);
const [allTransaction, setAllTransaction] =
useState([]);
const [frequency, setFrequency] =
useState("7");
const [selectedDate, setSelectedate] =
useState([]);
const [type, setType] = useState("all");
const [viewData, setViewData] =
useState("table");
const [editable,setEditable]=useState(null)
  //   table data
const columns = [
    {
      title: "Date",
dataIndex: "date",
</BrowserRouter>
</React.StrictMode>
);
reportWebVitals();
```

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
constsubmitHandler = 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) {
```

```
      render: (text) =>
```

Contd..

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
constsubmitHandler = 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");
    }
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"
```

Contd..

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 (
```

```
  };
  // Prevent for logged-in user
useEffect(() => {
    if (localStorage.getItem("user")) {
navigate("/");
    }
  }, [navigate]);
  return (
<div className="register-page">
      {loading &&<Spinner />}
<Form
        layout="vertical"
onFinish={submitHandler}
className="register-form"
>
```

Contd..

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;
<>
<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;
```

```
  background-image:
url("https://tse4.mm.bing.net/th?id=OIP.Nf
FhTSxEndiClX3FbMakxQHaE8&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;
}
```

Contd..

client\package.json

```
{
  "proxy": "http://localhost:8080/api/v1",
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@ant-design/icons": "^5.1.2",
    "@testing-library/jest-dom": "^5.14.1",
    "@testing-library/react": "^13.0.0",
    "@testing-library/user-event": "^13.2.1",
    "antd": "^5.5.1",
    "axios": "^1.4.0",
    "moment": "^2.29.4",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-redux": "^8.0.5",
    "react-router": "^6.11.2",
    "react-router-dom": "^6.11.2",
    "react-scripts": "5.0.1",
    "redux": "^4.2.1",
    "web-vitals": "^2.1.0"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
```

Backend

config\connectDb.js

```
const mongoose = require("mongoose");
constcolors = require("colors");
constconnectDb = async () => {
  try {
    await
mongoose.connect(process.env.MONGO_URL);
console.log(`Server Running On
${mongoose.connection.host}`.bgCyan.white);
  } catch (error) {
    console.log(`${error}`.bgRed);
  }
};
module.exports = connectDb;
```

controllers\transectionCtrl.js

```
consttransectionModel =
require("../models/transectionModel");
const moment = require('moment');
constgetAllTransection = async (req, res)
=> {
  try {
const{ frequency, selectedDate, type } =
req.body;
    // const{ userid } = req.body; // Extract
the userid from req.body
const transections = await
transectionModel.find({
    //   userid: userid, // Use the extracted
```

```
   "extends": [
     "react-app",
     "react-app/jest"
   ]
 },
 "browserslist": {
   "production": [
     ">0.2%",
     "not dead",
     "notop_mini all"
   ],
   "development": [
     "last 1 chrome version",
     "last 1 firefox version",
     "last 1 safari version"
   ]
 }
}
```

Contd of config\connectDb.js

```
userid: req.body.userid,
...(type !== "all" && { type }),
   });
res.status(200).json(transections);
 } catch (error) {
   console.log(error);
res.status(500).json(error);
 }
};
constdeleteTransection =async(req,res)=>{
 try {
```

```
userid
...(frequency !== "custom"
        ? {
          date: {
            $gt:
moment().subtract(Number(frequency),
"d").toDate(),
          },
        }
        : {
          date: {
            $gte: selectedDate[0],
            $lte: selectedDate[1],
          },
        }),
```

controllers\userController.js

```
constuserModel =
require("../models/userModel");
// login callback
constloginController = async (req, res) =>
{
  try {
const{ email, password } = req.body;
const user = await userModel.findOne({
email, password });
    if (!user) {
      return res.status(404).send("User Not
Found");
    }
res.status(200).json({
    await transectionModel.findByIdAndDelete({
      _id:req.body.transectionId});
res.status(200).send('transaction deleted')
  } catch (error) {
    console.log(erroe);
res.status(500).json(error)
  }
};
consteditTransection=async(req,res)=>{
  try {
    await
transectionModel.findOneAndUpdate({_id:req.b
ody.transectionId},req.body.payload);
res.status(200).send('Edit Successfully')
  } catch (error) {
    console.log(error);
res.status(500).json(error)
  }
};
constaddTransection = async (req, res) => {
  try {
constnewTransection = new
transectionModel(req.body);
    await newTransection.save();
res.status(201).send("Transection Created");
  } catch (error) {
    console.log(error);
res.status(500).json(error);
  }
};
module.exports = { getAllTransection,
```

```
      success: true,
      user,
    });
  } catch (error) {
res.status(400).json({
      success: false,
      error,
    });
  }
};
//Register Callback
constregisterController = async (req, res)
=> {
  try {
constnewUser = new userModel(req.body);
    await newUser.save();
res.status(201).json({
      success: true,
newUser,
    });
  } catch (error) {
res.status(400).json({
      success: false,
      error,
    });
  }
};
module.exports = { loginController,
registerController };
```

models\userModel.js

```
const mongoose = require("mongoose");
```

```
addTransection,editTransection,deleteTransectio
n };
```

models\transectionModel.js

```
const mongoose =require('mongoose')
consttransectionSchema=new
mongoose.Schema({
userid: {
          type: String,
          required: true,
        },
    amount: {
        type: Number,
        required: [true, "amount is required"],
      },
      type: {
        type: String,
        required: [true, "type is required"],
      },
      category: {
        type: String,
        requires: [true, "cat is required"],
      },
      reference: {
        type: String,
      },
      description: {
        type: String,
        required: [true, "desc is required"],
      },
      date: {
```

```
//schema design
constuserSchema = new
mongoose.Schema(
  {
    name: {
      type: String,
      required: [true, "name is required"],
    },
    email: {
      type: String,
      required: [true, "email is required and
should be unique"],
      unique: true,
    },
    password: {
      type: String,
      required: [true, "password is
required"],
    },
  },
{ timestamps: true }
);
//export
constuserModel =
mongoose.model("users", userSchema);
module.exports = userModel;
```

routes\userRoute.js

```
const express = require("express");
const {
loginController,
      type: Date,
      required: [true, "data is required"],
    },
  },
{ timestamps: true }
 );
consttransectionModel =
mongoose.model('transections',transectionSchem
a)
module.exports = transectionModel;
```

.env

```
PORT=8080
MONGO_URL=mongodb+srv://nishmi:nishmi
@cluster.nvmxlgt.mongodb.net/expanseApp
```

routes\transectionRoutes.js

```
const express =require('express');
const {addTransection,
getAllTransection,editTransection,deleteTransec
tion } = require('../controllers/transectionCtrl');
//router object
const router=express.Router();
//routes
//add transection POST MEthod
router.post("/add-transection", addTransection);
//edit transection POST MEthod
router.post("/edit-transection", editTransection);
//add transection POST MEthod
```

```
registerController,
} = require("../controllers/userController");
//router object
const router = express.Router();
//routers
// POST || LOGIN USER
router.post("/login", loginController);
//POST || REGISTER USER
router.post("/register", registerController);
module.exports = router;
```

server.js

```
const express = require("express");
constcors = require("cors");
const morgan = require("morgan");
constdotenv = require("dotenv");
constcolors = require("colors");
constconnectDb =
require("./config/connectDb");
// config dot env file
dotenv.config();
//databse call
connectDb();
//rest object
const app = express();
//middlewares
app.use(morgan("dev"));
app.use(express.json());
app.use(cors());
```

```
router.post("/delete-transection",
deleteTransection);
//get transactions
router.post('/get-transection', getAllTransection);
module.exports=router;
```

package.json

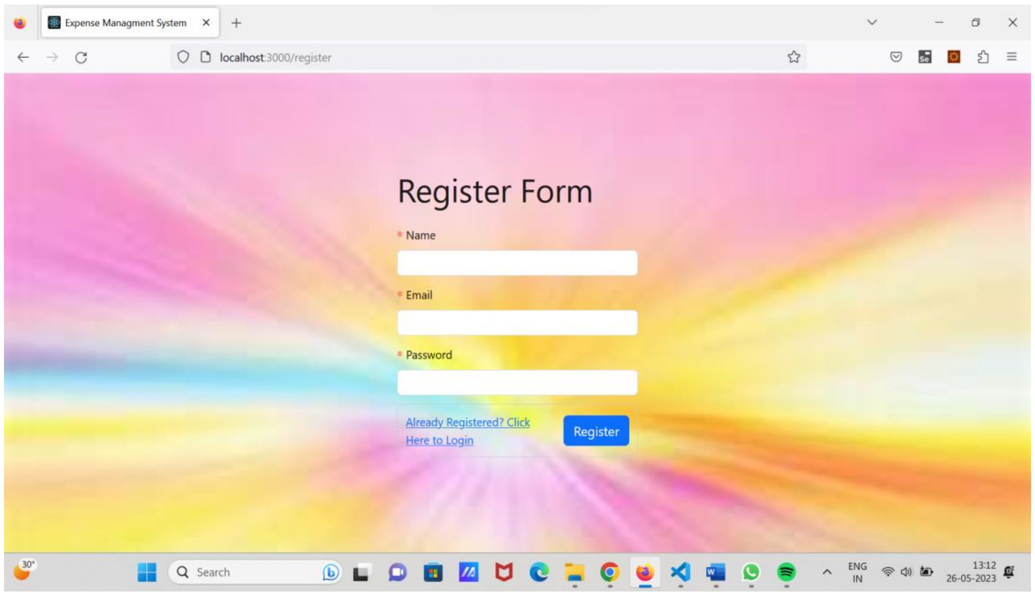
```
{
  "name": "et",
  "version": "1.0.0",
  "description": "expense tracker",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" &&
exit 1",
    "start": "node server.js",
    "server": "nodemon server.js",
    "client": "yarn start --prefix client",
    "dev": "concurrently \"npm run server\"\"yarn
run client\""
  },
  "keywords": [],
  "author": "",
  "license": "MIT",
  "dependencies": {
    "colors": "^1.4.0",
    "concurrently": "^8.0.1",
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2",
    "moment": "^2.29.4",
```

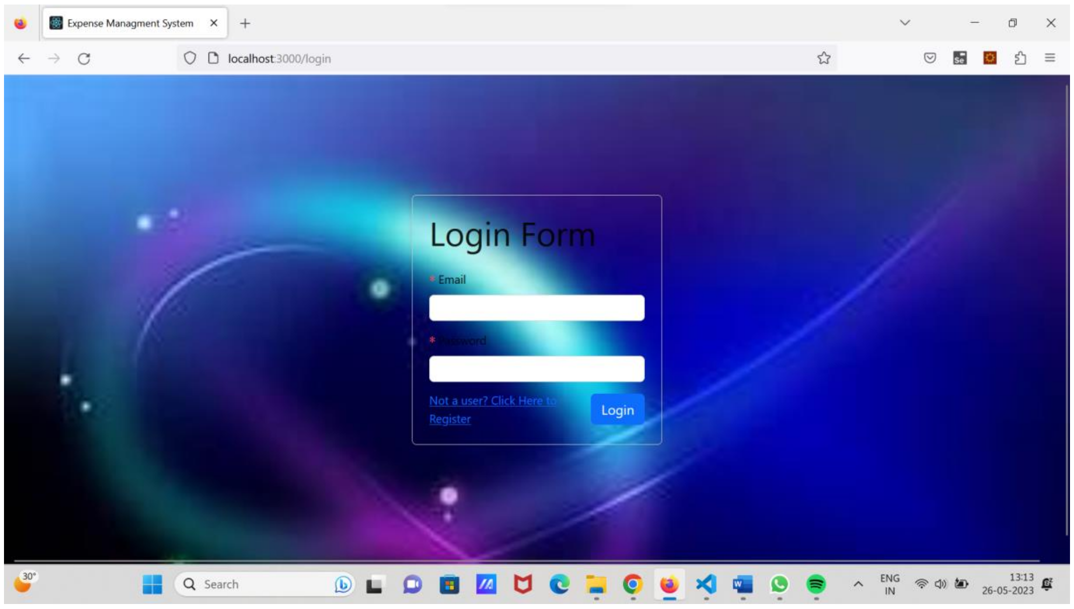
```
//routes
app.use("/api/v1/users",
require("./routes/userRoute"));
//transections routes
app.use("/api/v1/transections",
require("./routes/transectionRoutes"));
//port
const PORT = 8080 || process.env.PORT;
//listen server
app.listen(PORT, () => {
console.log(`Server running on port
${PORT}`);
});
```

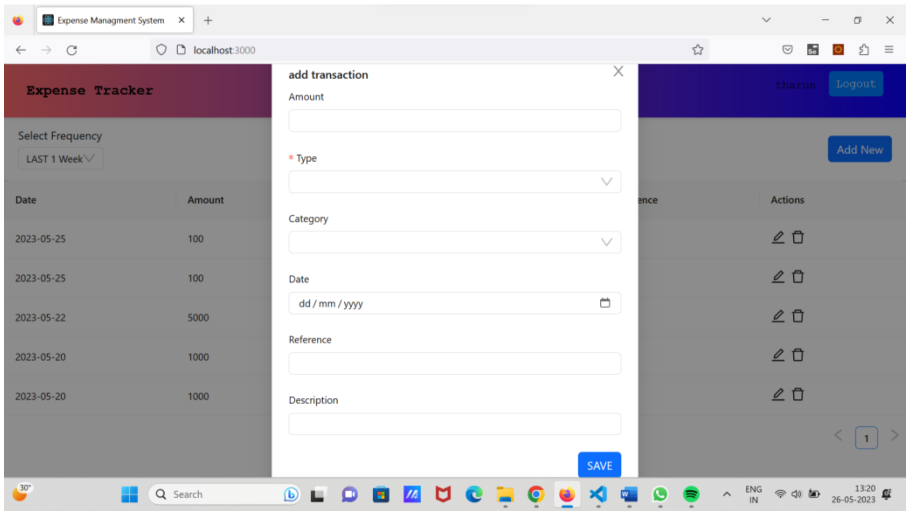
Contd of package.json

```
    "mongoose": "^7.2.0",
    "morgan": "^1.10.0",
    "nodemon": "^2.0.22"
  }
}
```

SCREEN SHOOTS






Expense Managment System

localhost:3000

Expense Tracker

add transaction

Amount

* Type

Category

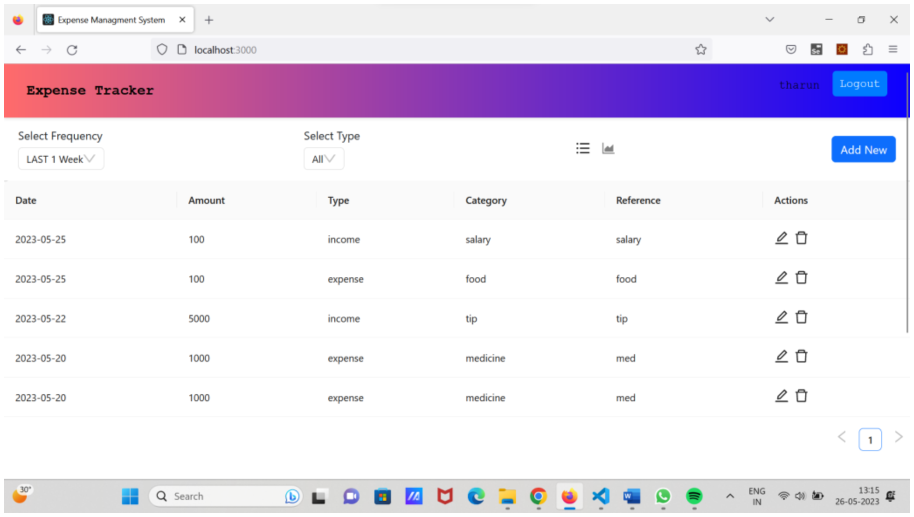
Date

dd / mm / yyyy

Reference

Description

SAVE


Expense Managment System

localhost:3000

Expense Tracker

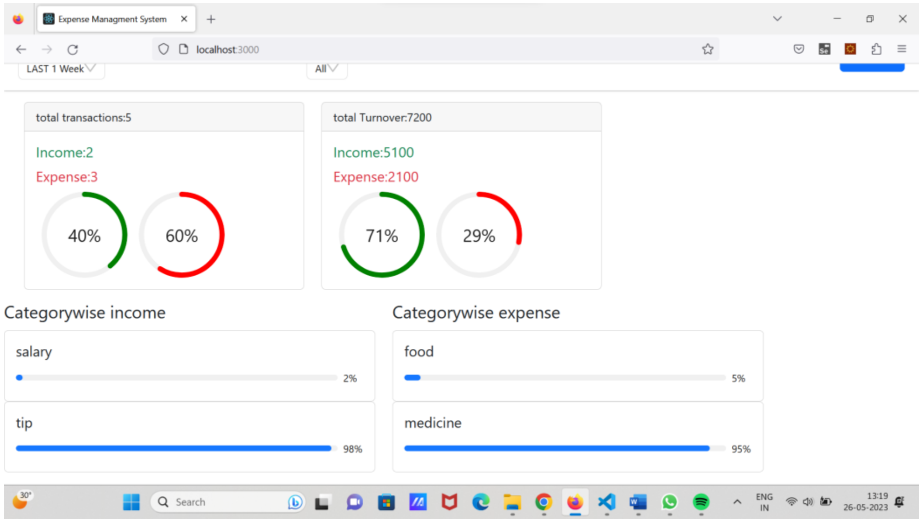
tharun Logout

Select Frequency: LAST 1 Week

Select Type: All

Add New

Date	Amount	Type	Category	Reference	Actions
2023-05-25	100	income	salary	salary	
2023-05-25	100	expense	food	food	
2023-05-22	5000	income	tip	tip	
2023-05-20	1000	expense	medicine	med	
2023-05-20	1000	expense	medicine	med	


Expense Managment System

localhost:3000

LAST 1 Week All

total transactions:5

Income:2

Expense:3

40% 60%

total Turnover:7200

Income:5100

Expense:2100

71% 29%

Categorywise income

salary 2%

tip 98%

Categorywise expense

food 5%

medicine 95%

CONCLUSION

The Expense Tracker app tracks all the expenses and helps the user to manage his/her expenses so that the user is the path of financial stability.Tracking your expenses daily can save your amount, but it can also help you set financial goals for the future. If you know exactly where your amount is going every month, you can easily see where some cutbacks and compromises can be made. The project what we have developed is work more efficient than the other income and expense tracker. The project successfully avoids the manual calculation for avoiding calculating the income and expense per month. The modules are developed with efficient and also in an attractive manner.

IMPLEMENTATION OF SECURITY MECHANISMS

The application requires a central server, similar to the one provided by the ISP. A well connected network of clients that connect to the server using the http protocol and a URL is required. Although the OS is not a dependent factor (JVM), a web server such as apache has to be installed, configured and available throughout. A dedicated port number (8080) to which the incoming request and outgoing response has to be communicated should be assigned. The design of the application is addressed as follows,

- Expense tracker – Windows Forms (Web forms)
- Online / Server – Web forms

Once the system has been designed, the next step is to convert the designed one in to actual code, so as to satisfy the user requirements as excepted. If the system is approved to be error free it can be implemented. When the initial design was done for the system, the department was consulted for acceptance of the design so that further proceedings of the system development can be carried on. After the development of the system a demonstration was given to them about working of the system. The aim of the system illustration was to identify any malfunctioning of the system. Implementation includes proper training to end-users. The implemented software should be maintained for prolonged running of the software. Initially the system was run parallel with manual system. The system has been tested with data and has proved to be error-free and user-friendly. Training was given to end -user about the software and its features.

FURTHER ENHANCEMENT

1. User can directly get a mail after an adding the expenses or changes in the transaction
2. Can track all the expenses of a bank account thorough the online transaction platform.

BIBLIOGRAPHY

1. https://ant.design/
2. https://youtu.be/KPPeaDiCwOg
3. https://www.slideshare.net/RashnaMaharjan2/daily-expense-tracker-153160282
4. https://www.slideshare.net/RashnaMaharjan2/daily-expense-tracker
5. https://getbootstrap.com/docs/5.3/getting-started/introduction/
6. https://github.com/techinfo-youtube/prouction-Expense-app-mern/tree/main
7. https://github.com/techinfo-youtube/Expense-Management-System-MERN-STACK-Project/tree/main
8. https://www.mongodb.com/languages/mern-stack-tutorial
9. https://www.w3schools.com/react/
10. https://www.w3schools.com/mongodb/
11. https://www.w3schools.com/nodejs/

APPENDICES

DFD	-	Data Flow Diagram A graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).
JSON	-	(JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. JSON is perfect for storing temporary data.
MONGODB	-	MongoDB is a document database used to build highly available and scalable internet applications.
API	-	API stands for application programming interface—a set of definitions and protocols to build and integrate application software.
NODEjs	-	Node. js to create server-side web applications, and it is perfect for data-intensive applications since it uses an asynchronous, event-driven model.
IEEE	-	Institute of Electrical and Electronics Engineers IEEE goals include scientific and educational pursuit directed toward the advancement of the theory and practice of electrical, electronics, communications and computer engineering, as well as computer science, the allied branches of engineering and the related arts and sciences.