****                                                                                                                                    

**RMK ENGINEERING COLLEGE**

**(An Autonomous Institution)**

**R.S.M. Nagar, Kavaraipettai, Gummidipoondi Taluk, Thiruvallur District 601 206.**

**PROJECT**

**PERSONAL EXPENSE TRACKER APPLICATION**

**DONE BY**

**TEAM ID: PNT2022TMID15866**

**SUSHANTH KUMAR NK (111719104156)**

**TARUN V (111719104160)**

**VISHNU HAASAN T (111719104179)**

**TELAGANENI CHAITANYA (111719104162)**

TABLE OF CONTENTS

1. INTRODUCTION
   1. Project Overview
   2. Purpose
2. LITERATURE SURVEY
   1. Existing problem
   2. References
   3. Problem Statement Definition
3. IDEATION & PROPOSED SOLUTION
   1. Empathy Map Canvas
   2. Ideation & Brainstorming
   3. Proposed Solution
   4. Problem Solution fit
4. REQUIREMENT ANALYSIS
   1. Functional requirement
   2. Non-Functional requirements
5. PROJECT DESIGN
   1. Data Flow Diagrams
   2. Solution & Technical Architecture
   3. User Stories
6. PROJECT PLANNING & SCHEDULING
   1. Sprint Planning & Estimation
   2. Sprint Delivery Schedule
   3. Reports from JIRA
7. CODING & SOLUTIONING (Explain the features added in the project along with code)
   1. Feature 1
   2. Feature 2
   3. Database Schema (if Applicable)
8. TESTING
   1. Test Cases
   2. User Acceptance Testing
9. RESULTS
   1. Performance Metrics
10. ADVANTAGES & DISADVANTAGES
11. CONCLUSION
12. FUTURE SCOPE
13. APPENDIX

Source Code

GitHub & Project Demo Link

**INTRODUCTON**

**Project Overview:**

ABSTRACT

This project is based on an expense and income tracking system. This project aims to

create an easy, faster and smooth tracking system between the expense and the income.

This project also offers some opportunities that will help the user to sustain all financial

activities like digital automated diary. So, for the better expense tracking system, we

developed our project that will help the users a lot.

ABSTRACT

This project is based on an expense and income tracking system. This project aims to

create an easy, faster and smooth tracking system between the expense and the income.

This project also offers some opportunities that will help the user to sustain all financial

activities like digital automated diary. So, for the better expense tracking system, we

developed our project that will help the users a lot.

ABSTRACT

This project is based on an expense and income tracking system. This project aims to

create an easy, faster and smooth tracking system between the expense and the income.

This project also offers some opportunities that will help the user to sustain all financial

activities like digital automated diary. So, for the better expense tracking system, we

developed our project that will help the users a lot.

ABSTRACT

This project is based on an expense and income tracking system. This project aims to

create an easy, faster and smooth tracking system between the expense and the income.

This project also offers some opportunities that will help the user to sustain all financial

activities like digital automated diary. So, for the better expense tracking system, we

developed our project that will help the users a lot.

ABSTRACT

This project is based on an expense and income tracking system. This project aims to

create an easy, faster and smooth tracking system between the expense and the income.

This project also offers some opportunities that will help the user to sustain all financial

activities like digital automated diary. So, for the better expense tracking system, we

developed our project that will help the users a lot.

ABSTRACT

This project is based on an expense and income tracking system. This project aims to

create an easy, faster and smooth tracking system between the expense and the income.

This project also offers some opportunities that will help the user to sustain all financial

activities like digital automated diary. So, for the better expense tracking system, we

developed our project that will help the users a lot.

This project is based on an expense and income tracking system. This project aims to create an easy, faster and smooth tracking system between the expense and the income .This project also offers some opportunities that will help the user to how to manage the expenses in efficient way and also set have an option to set a limit for the amount to be used for that particular month. So, for the better expense tracking system, we developed our project that will help the users a lot.

**Purpose :**

An expense tracking app is an exclusive suite of services for people who seek to handle their earnings and plan their expenses and savings. When you track your spending, you know where your money goes and you can ensure that your money is used wisely. Tracking your expenditures also allows you to understand why you're in debt and how you got there. This will then help you design a be fitting strategy of getting out of debt. Many people in India live on a fixed income, and they find that towards the end of the month they don't have sufficient money to meet their needs

**Literature Survey**

**Existing Problem :**

In existing, we need to maintain the Excel sheets, CSV files for the user daily, weekly and monthly expenses and there is no as such complete solution to keep a track of its daily expenses easily. To do so a person as to keep a log in a diary or in a computer system, also all the calculations need to be done by the user which may sometimes results in mistakes leading to losses. The existing system is not user friendly because data is not maintained perfectly. A writing audit is a study of insightful sources on a particular research. We found various similar products that have already been developed in the market. Unlike all those products, Personal Expense tracker (PET) provides security and graphical results. We provide the user to enter their wish-list before any purchase. It generates notifications to notify user about their money management and put an limit to weekly, monthly, expenses.

**Problem Statement Definition :**

Every earning people are mostly obsessed at the end of the month as they cannot remember where all of their money have gone when they have spent and ultimately have to sustain in little money minimizing their essential needs. There is no as such complete solution present easily to keep track of its daily expenditure easily and notify them if they are going to have money shortage. Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user.  Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an alert. the main purpose of our application is to track the user's expenses.

the main

purpose of our application is to track the user's expenses

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

A writing audit is a study of insightful sources on a particular research. We found various

similar products that have already been developed in the market. Unlike all those products,

Personal Expense Tracker (PET) provides security and graphical results. We provide the

users to enter their wish-list before any purchase. It generates notifications to notify users

about their timely entry

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

A writing audit is a study of insightful sources on a particular research. We found various

similar products that have already been developed in the market. Unlike all those products,

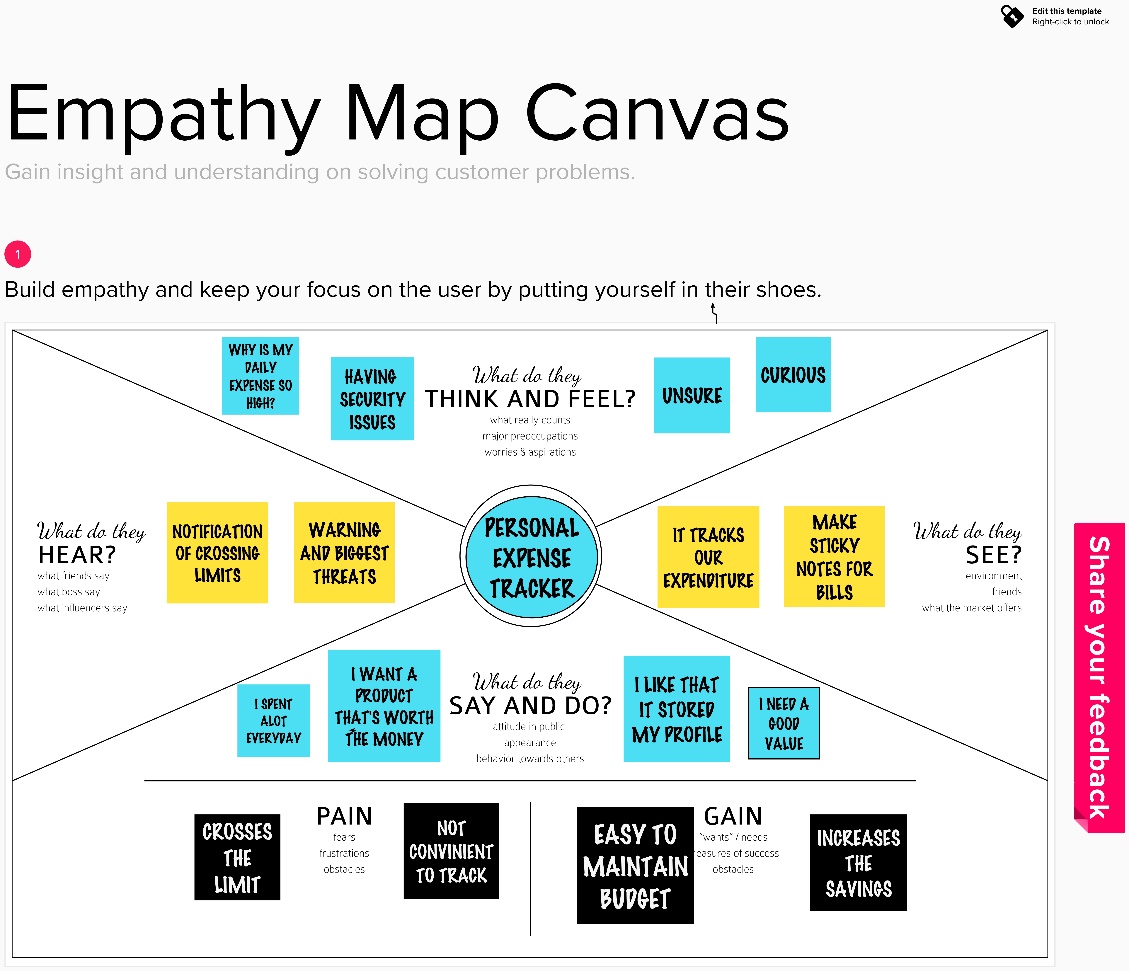
Personal Expense Tracker (PET) provides security and graphical results. We provide the

users to enter their wish-list before any purchase. It generates notifications to notify users

about their timely entry

**Ideation and Proposed Solution**

**Empathy Map Canvas :**

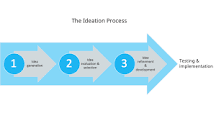
****

**Ideation and Proposed Solution**

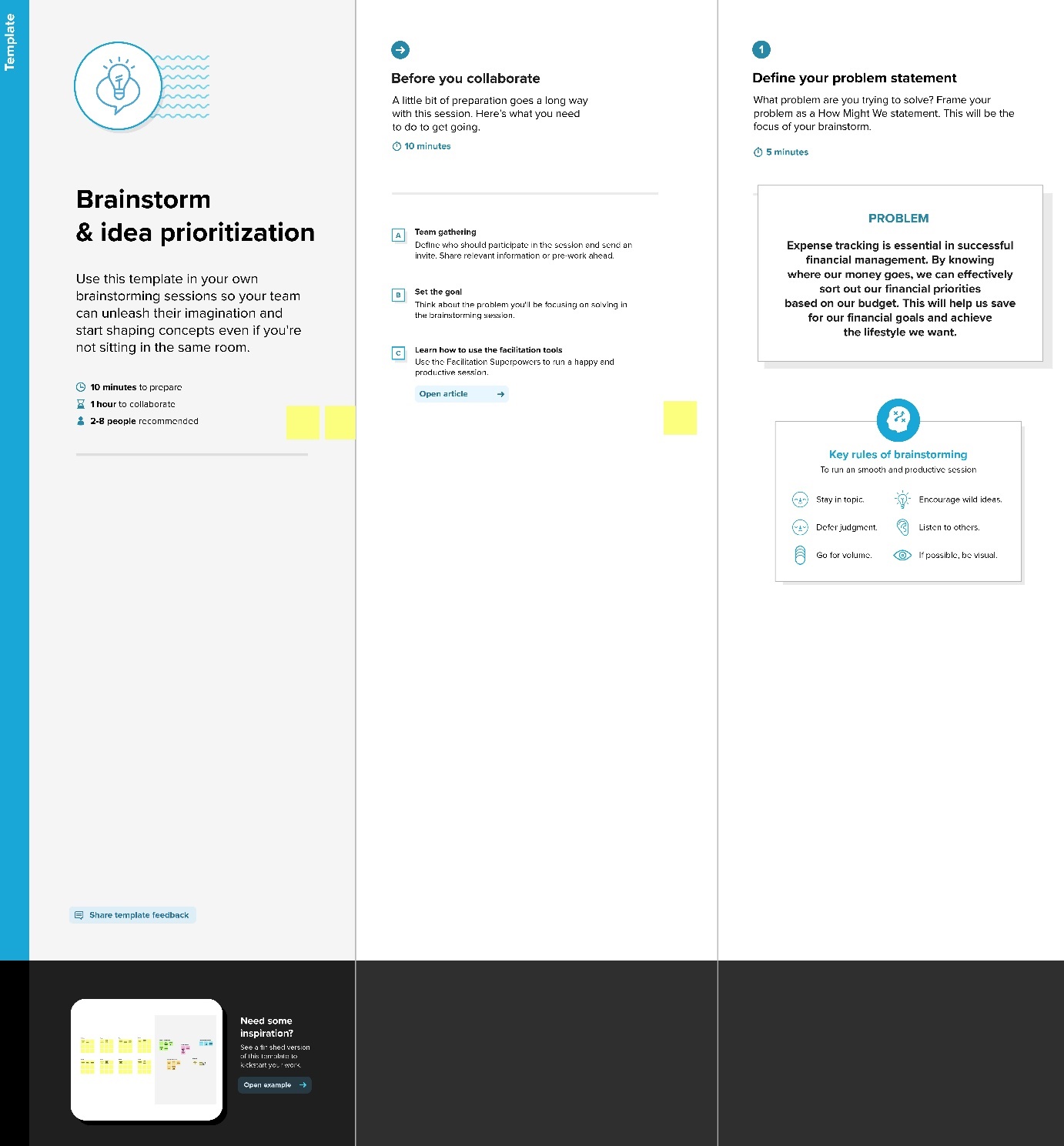
**Ideation and Brainstorming :**

Brainstorming is agroup activity where everyone comes together to discuss strategies for growth and improvement. You can exchange ideas, share important information and use these meetings as informal catch-up sessions with your co-workers. Brainstorming combines a relaxed, informal approach to problem solving with lateral thinking. It encourages people to come up with thoughts and ideas that can, at first, seem a bit crazy. Some of these ideas can be crafted into original, creative solutions to a problem, while others can spark even more ideas.

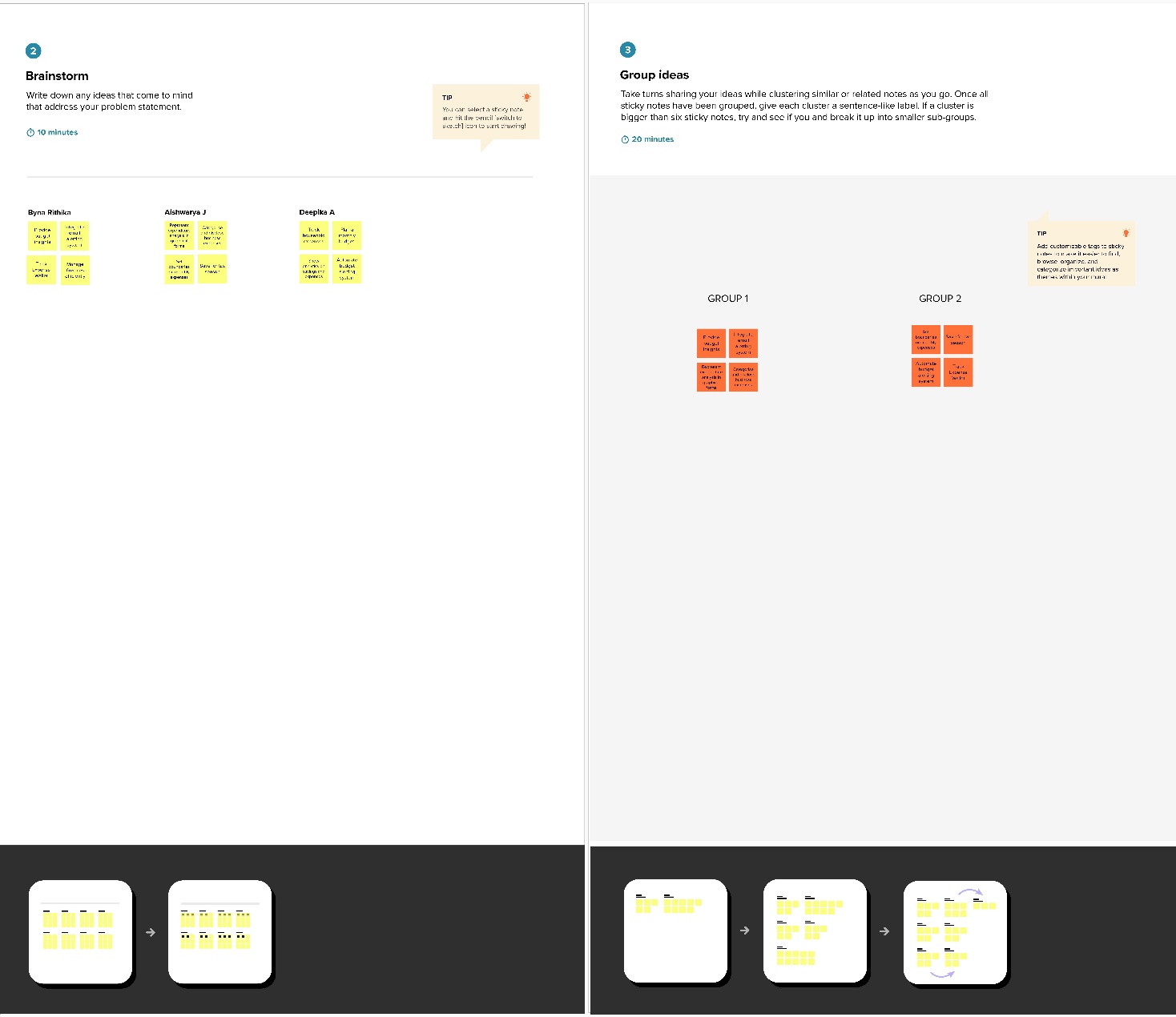
Ideation is the process where you generate ideas and solutions through sessions such as Sketching, Prototyping, Brainstorming, Brainwriting, Worst Possible Idea, and a wealth of other ideation techniques. Ideation is also the third stage in the Design Thinking process.



As you can see, ideation is not just a one-time idea generation or a brainstorming session. In fact, we can divide ideation in these three stages: generation, selection, and development.

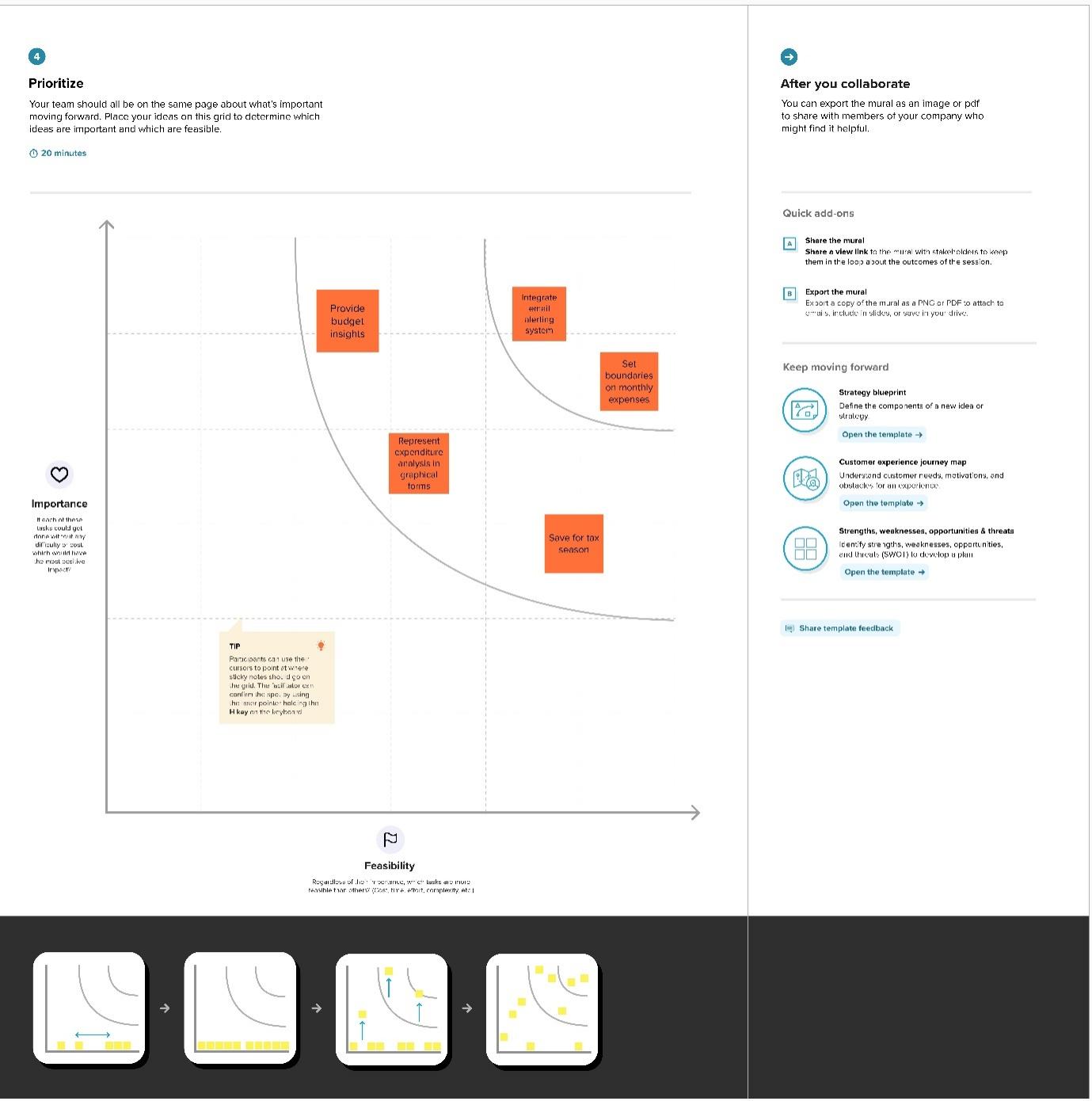
****

**2 . Ideation and Proposed Solution**

****

**3.**

**Ideation and Proposed Solution**

****

**Ideation and Proposed Solution**

**Proposed Solution :**

Expense Tracker is going to be a mobile application so that It can be accessed any time required. This application will have a two-tier architecture: first one is the database tier, where all the data and financial data will be stored. Second it will be the user interface which will support the application user communicate with the system and also store Information in the database. The proposed system should operate offline so it can be accessed at any time without internet availability. The proposed system should provide different categories for the user to select from and they can enter the amount and mode of payment. This system should be able to analyze the information, provide analytics on which category did the user spent most of their money. The proposed system should provide a user interface where the user could store and observe their past expenses.

**Requirement Analysis**

**Functional Requirments :**

**1. Dashboard panel**

The system shall authenticate the user and then display panel based on the particular identified user.

**2. Add bill**

The system shall allow the user to add bill details based on the user's need to track the type of expenses.

**3.Expense planner**

The system should graphically represent the current month figure based current month expenses and user's own budget share.

**4. Expense tracker**

The system should graphically represent the yearly expense numbers in form of report

**5.Add notes**

The system shall allow users to add notes to their expenses.

**Non-Functional Requirments:**

**1. Usability**

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

**2. Reliability**

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

**3. Supportability**

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

**4. Performance**

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

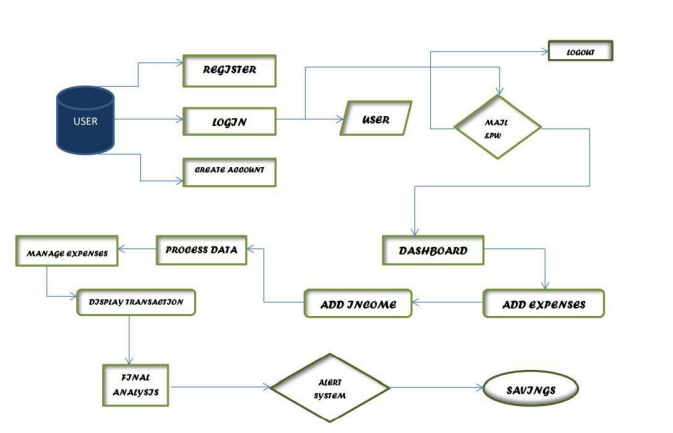
**5. Availability**

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

**Project Design**

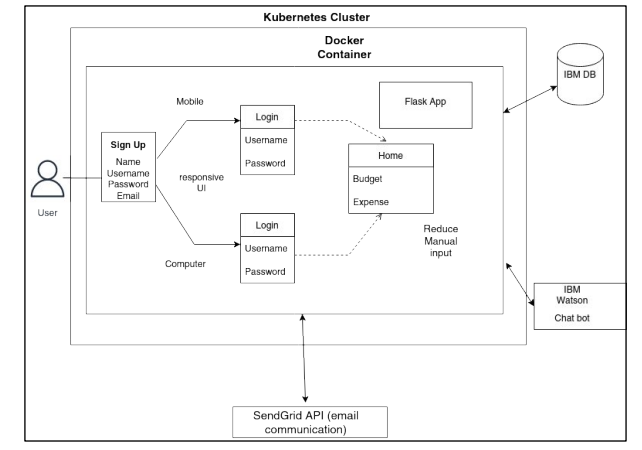
**Data flow Diagrams :**

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. Data flow diagrams provide a straightforward, efficient way for organizations to understand, perfect, and implement new processes or systems. They're visual representations of your process or system, so they make it easy to understand and prune.

****

**Solution & Technical Architecture:**

Technical Architecture (TA) is a form of IT architecture that is used to design computer systems. It involves the development of a technical blueprint with regard to the arrangement, interaction, and interdependence of all elements so that system-relevant requirements are met.



**Project Planning & Scheduling**

**Sprint Plainning & Estimation:**

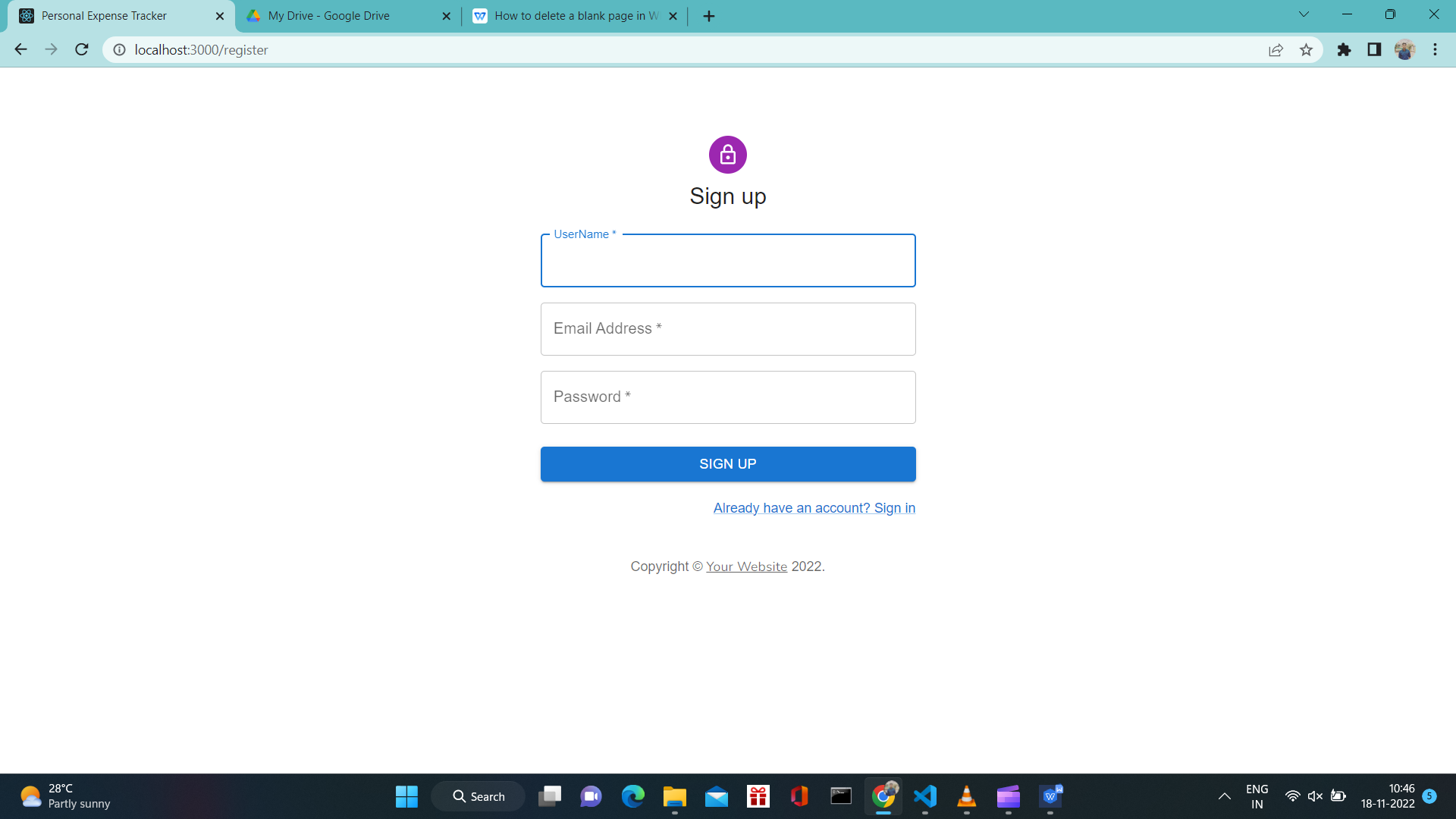
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| **Sprint 1** | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | 2 | High | SushanthKumar.NK  VishnuHassan.T  Telaganeni Chaitanya  Tarun.V |
|  | USN-2 | As a user, I will receive confirmation email once I have registered for the application | 1 | High |
| Login | USN-3 | As a user, I can log into the application by entering email & password | 1 | High |
| Dashboard | USN-4 | Logging in takes to the dashboard for the logged user. | `2 | High |
| *Bug fixes, routine checks and improvisation by everyone in the team \*Intended bugs only* | | | | | | |
| **Sprint 2** | Workspace | USN-1 | Workspace for personal expense tracking | 2 | High | SushanthKumar.NK  Tarun.V  Telaganeni Chaitanya  VishnuHassan.T |
| Charts | USN-2 | Creating various graphs and statistics of customer’s data | 1 | Medium |
| Connecting to IBM DB2 | USN-3 | Linking database with dashboard | 2 | High |
|  | USN-4 | Making dashboard interactive with JS | 2 | High |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint-3** |  | USN-1 | Wrapping up the server side works of frontend | 1 | Medium | SushanthKumar.NK  VishnuHassan.T  Tarun.V  Telaganeni Chaitanya |
| Watson Assistant | USN-2 | Creating Chatbot for expense tracking and for clarifying user’s query | 1 | Medium |
| SendGrid | USN-3 | Using SendGrid to send mail to the user about their expenses | 1 | Low |
|  | USN-4 | Integrating both frontend and backend | 2 | High |
| *Bug fixes, routine checks and improvisation by everyone in the team \*Intended bugs only* | | | | | | |
| **Sprint-4** | Docker | USN-1 | Creating image of website using docker/ | 2 | High | VishnuHassan.T  Tarun.V  SushanthKumar.NK  Telaganeni Chaitanya |
| Cloud Registry | USN-2 | Uploading docker image to IBM Cloud registry | 2 | High |
| Kubernetes | USN-3 | Create container using the docker image and hosting the site | 2 | High |
| Exposing | USN-4 | Exposing IP/Ports for the site | 2 | High |

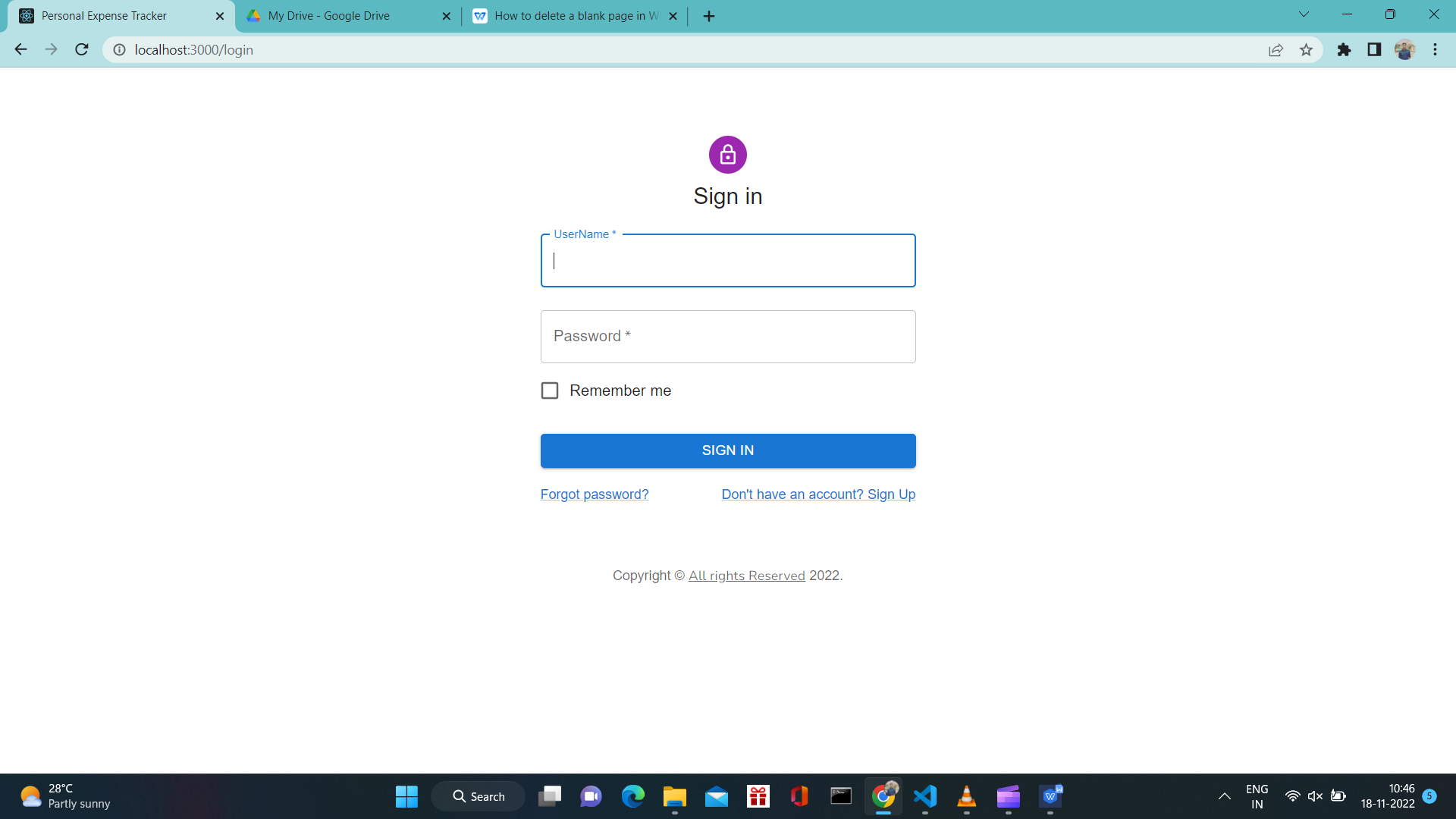
**Sprint Delivery Schedule:**

****

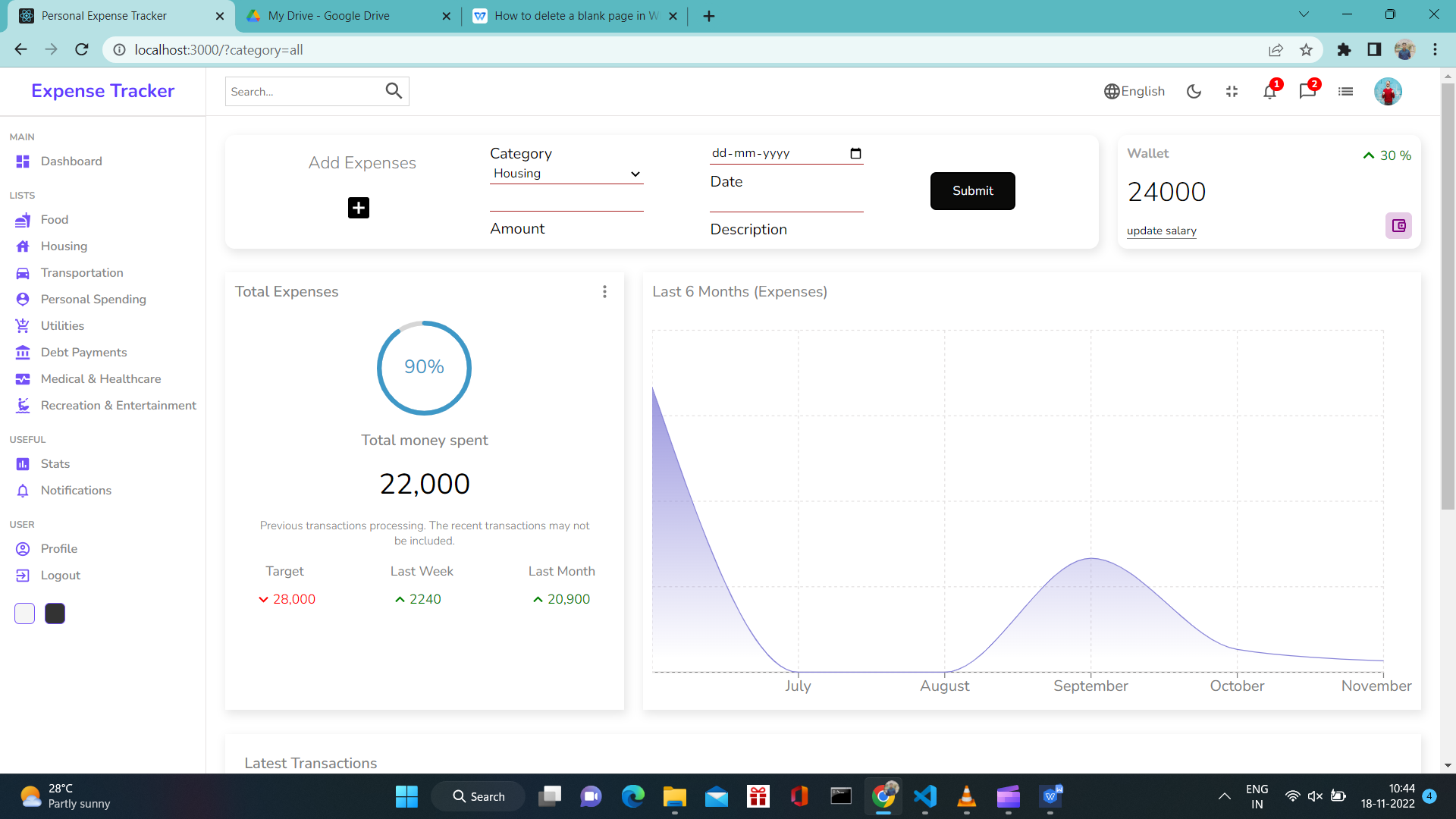
**Register Page :**

****

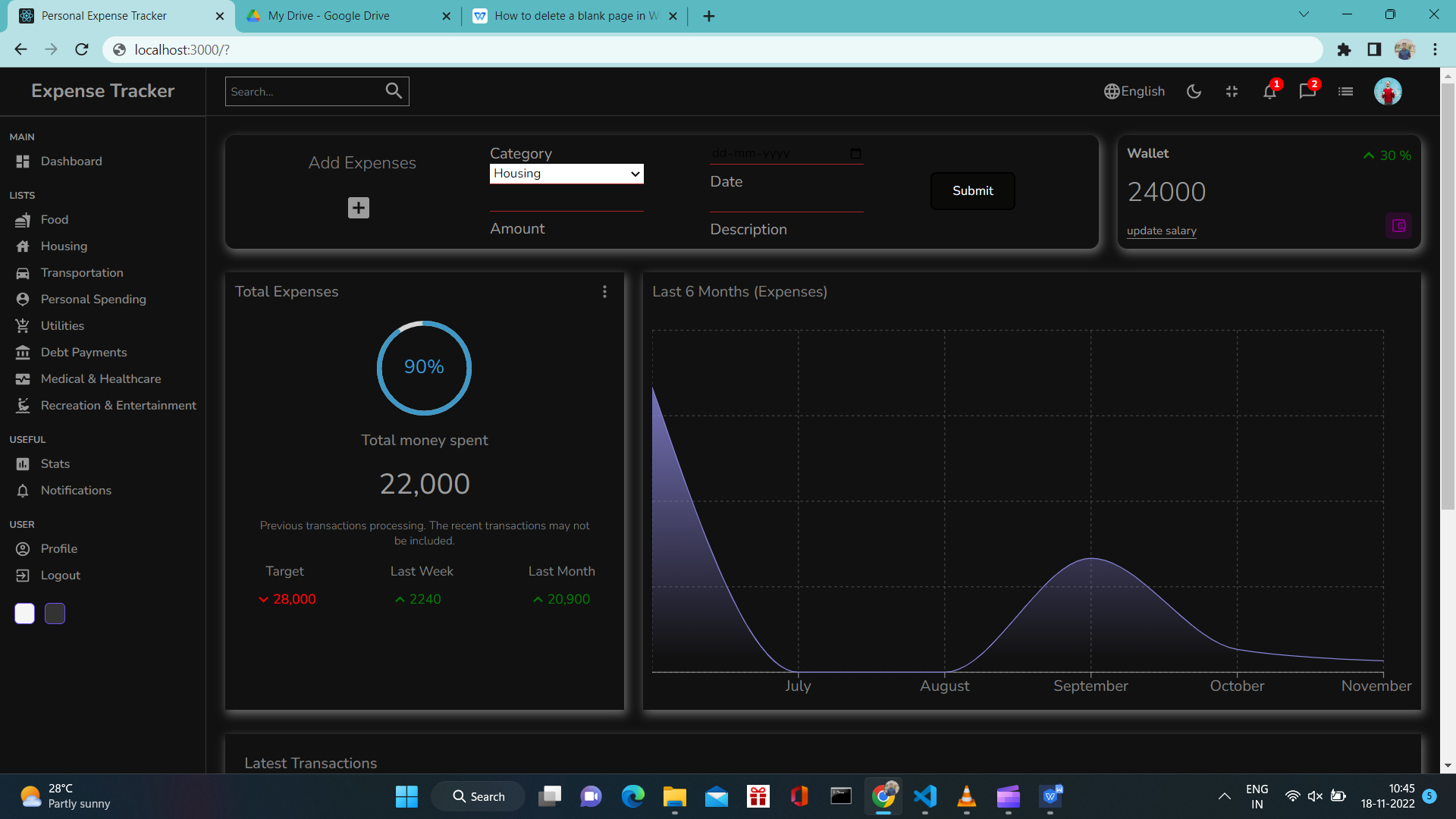
**Sign in :**

****

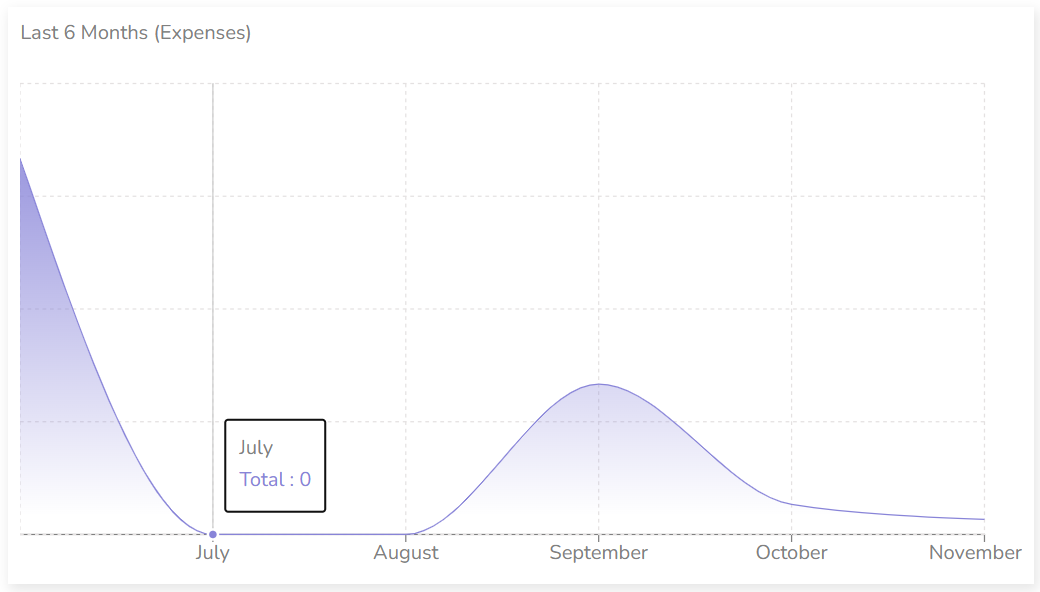
**Home Page :**

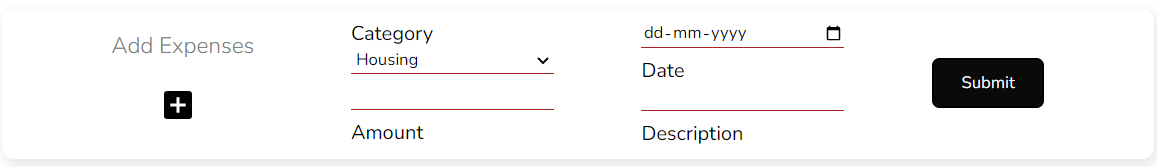
****

**Dark Mode :**

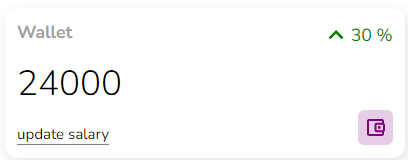
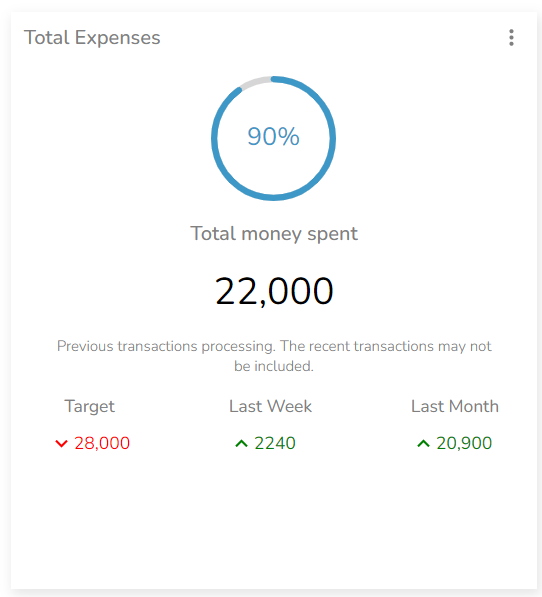
****

**Feature 1 :**





**Feature 2 :**

**Database Schema :**

ExpenseSchema - "CREATE TABLE IF NOT EXISTS EXPENSES (expense\_id INT PRIMARY KEY NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1),ref\_user INT NOT NULL, amount FLOAT NOT NULL, category varchar NOT NULL, description varchar, spent\_date DATE NOT NULL, FOREIGN KEY user\_id (ref\_user) REFERENCES USERS ON DELETE NO ACTION)"

UserSchema - "CREATE TABLE IF NOT EXISTS USERS (user\_id INT PRIMARY KEY NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1) , username varchar NOT NULL UNIQUE, password varchar NOT NULL, email varchar NOT NULL, balance FLOAT NOT NULL, lim FLOAT NOT NULL)"

SalarySchema-"CREATE TABLE IF NOT EXISTS SALARIES (salary\_id INT PRIMARY KEY NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1) , amount FLOAT NOT NULL, update\_date DATE NOT NULL, ref\_user INT NOT NULL, FOREIGN KEY user\_id (ref\_user) REFERENCES USERS ON DELETE NO ACTION)"

**Testing**

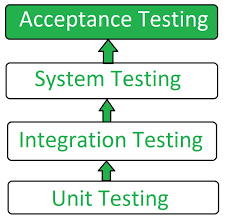
**Test Case:**

A test case is a set of actions performed on a system to determine if it satisfies software requirements and functions correctly. The purpose of a test case is to determine if different features within a system are performing as expected and to confirm that the system satisfies all related standards, guidelines and customer requirements. The process of writing a test case can also help reveal errors or defects within the system.

Test cases are typically written by members of the quality assurance team or the testing team and can be used as step-by-step instructions for each system test. Testing begins once the development team has finished a system feature or set of features. A sequence or collection of test cases is called a test suite.

A test case document includes test steps, test data, preconditions and the postconditions that verify requirements.

**User Acceptance Testing:**



**Results**

**Performance Metrics :**

Performance metrics are used to measure the behavior, activities, and performance of a business. This should be in the form of data that measures required data within a range, allowing a basis to be formed supporting the achievement of overall business goals. Measuring performance through metrics is key to seeing how employees are working, and whether targets are being met.

**Advantages :**

It’s simple to set up and use. When you’re creating your own method of tracking your finances, you first have to figure out how you’re going to do that. Are you going to use pen and paper, or software, or an excel spreadsheet? What are you going to track? How are you going to input that data, and how often are you going to do it? With an automated app, it tracks everything for you in real time. It has a wealth of information, so no matter what data you feel is important to track, it is all there and available for you – you just need to take a look to see it.

It’s instantaneous. The application will track all of your data for you. It doesn’t do it once a week or once a month, like you might if you were doing it manually. All the information is automatically brought right into your account as soon as it is available. That means that you have a day by day way of checking to ensure that you’re on track and moving in the right direction.

**Disadvantages :**

Your information is less secure, and probably being used and sold. If the service is free, then the product is you. Mint.com, like other financial apps, is a free service. They have to pay their bills somehow, so regardless of what their privacy policy may or may not say, just assume that your spending history and trends are going to be recorded and analyzed, by someone, somewhere. Now, you shouldn’t have to worry about credit card fraud or [identity theft](https://maplemoney.com/identity-theft/), these companies are large enough and secure enough that you’ll never have to worry about something like that.Automating everything to do with your finances can make you financially lazy. If your bills are paid automatically and your finances are track automatically, then what is there left for you to do? Not a lot, to be honest. So you might stop [caring about what you’re spending](https://maplemoney.com/control-your-spending-with-a-budget/) and where your money is going. Eventually you may look at your Mint data and realize that you’ve blown your budget over the last two months, but by then it is too late

**Conclusion**

After making this application we assure that this application will help its users to manage the cost of their daily expenditure. It will guide them and aware them about there daily expenses. It will prove to be helpful for the people who are frustrated with their daily budget management, irritated because of amount of expenses and wishes to manage money and to preserve the record of their daily cost which may be useful to change their way of spending money. In short, this application will help its users to overcome the wastage of money

**Source Code**

**Backend:**

from flask import Flask, request, g

from utils.loadenv import LoadEnv

from utils.db import ConnectDB

import utils.db

from loader import SchemaLoader

from schemas.UserSchema import User

from schemas.ExpenseSchema import Expense

from schemas.SalarySchema import Salary

import os

from functools import wraps

import json

from utils.auth import token\_encode, token\_required

from flask\_cors import CORS

from datetime import datetime, timedelta

app = Flask(\_\_name\_\_)

CORS(app=app)

app.secret\_key = "deadman"

app.config['SECRET\_KEY'] = 'niggatarun'

LoadEnv()

JSON\_TYPE = "application/json"

TYPE = 'Content-Type'

TYPE\_OBJ = {TYPE: JSON\_TYPE}

def main():

    try:

        dsn\_hostname = os.getenv('DB\_HOST')

        dsn\_uid = os.getenv('DB\_USER')

        dsn\_password = os.getenv('DB\_PASS')

        dsn\_db = os.getenv('DB\_NAME')

        dsn\_driver = os.getenv('DB\_DRIVER')

        dsn\_port = os.getenv('DB\_PORT')

        dsn\_protocol = os.getenv('DB\_PROTOCOL')

        dsn\_cert = os.getenv('DB\_CERT')

        ConnectDB(dsn\_db=dsn\_db, dsn\_hostname=dsn\_hostname, dsn\_password=dsn\_password, dsn\_port=dsn\_port, dsn\_protocol=dsn\_protocol, dsn\_uid=dsn\_uid, dsn\_driver=dsn\_driver)

        SchemaLoader.CreateAll()

        @app.route("/user", methods=['POST'])

        def AddUser():

            username = request.json['username']

            password = request.json['password']

            email = request.json['email']

            print(username)

            user = User(username=username, password=password, email=email)

            err = user.AddUser()

            print(err)

            if not err:

                return "Unable to Create User", 400

            else:

                return "Successfully Created User", 200

        @app.route("/login", methods=['POST'])

        def LoginUser():

            username = request.json['username']

            password = request.json['password']

            print(username, password)

            user = User(username=username, password=password)

            check, uid = user.LoginUser()

            if check:

                data = {

                    'uid': uid

                }

                token = token\_encode(data=data)

                resp = {

                    "token": token,

                    "error": False

                }

                return json.dumps(resp), 200, TYPE\_OBJ

            else:

                resp = {

                    "error": True

                }

                return json.dumps(resp), 401, TYPE\_OBJ

        @app.route("/checkLogin", methods=['GET'])

        @token\_required

        def CheckLogin():

            return "GOOOD"

        @app.route("/expense", methods=['POST'])

        @token\_required

        def AddExpense():

            user = g.data['uid']

            amount = int(request.json['amount'])

            category = request.json['category']

            description = request.json['description']

            date = request.json['date']

            expense = Expense(amount=amount, category=category, description=description,date=date, user=user)

            err = expense.AddExpense()

            if not err:

                return "Unable To Add Expense", 400

            else:

                return "Expense Added", 200

        @app.route('/queryexpense', methods=['GET'])

        @token\_required

        def QueryExpense():

            print({TYPE: JSON\_TYPE})

            st = request.json['start\_time']

            end = request.json['end\_time']

            categories = request.json['category']

            uid = g.data['uid']

            if categories == []:

                l, err = Expense.QueryExpenses(start\_time=st, end\_time=end, id=uid)

            else:

                l, err = Expense.QueryExpenses(start\_time=st, end\_time=end, category=categories, id=uid)

            if not err:

                obj = {

                    "error": True

                }

                return json.dumps(obj=obj), 404, {TYPE: JSON\_TYPE}

            resArr = []

            for i in l:

                dt = i[5].strftime("%d-%m-%Y")

                print(dt)

                exp = Expense.NewDict(id=i[0], user=i[1], amount=i[2], category=i[3], description=i[4], date=dt)

                resArr.append(exp.\_\_dict\_\_)

            obj = {

                "error": False,

                "expenses": resArr

            }

            return json.dumps(obj=obj), 200, {TYPE: JSON\_TYPE}

        @app.route('/salary', methods=['POST'])

        @token\_required

        def AddSalary():

            sal = request.json['amount']

            date = request.json['date']

            uid = g.data['uid']

            salary = Salary(user=uid, amount=sal, date=date)

            err = salary.AddSalary()

            if not err:

                return "Unable to Add Salary", 400

            else:

                return "Added Salary", 200

        @app.route('/balance', methods=['GET'])

        @token\_required

        def GetSalary():

            uid = g.data['uid']

            d, err = User.GetBalance(id=uid)

            if not err:

                return "Unable to fetch salary", 404

            else:

                resp = {

                    "balance": d[0][0],

                    "limit": d[0][1]

                }

                return json.dumps(resp), 200, TYPE\_OBJ

        def ObjToStr(date : datetime) -> str:

            return date.strftime('%Y-%m-%d')

        @app.route('/expenses', methods=['POST'])

        @token\_required

        def GetExpenses():

            date = request.json['date']

            category = request.json['category']

            print(date, category)

            uid = g.data['uid']

            neededD = datetime.strptime(date, '%Y-%m-%d')

            arr = []

            subArr = []

            for i in range(6):

                if i == 0:

                    prev = neededD - timedelta(days=30)

                    subArr.append(prev)

                    arr.append([ObjToStr(neededD), ObjToStr(prev)])

                else:

                    prev = subArr[-1] - timedelta(days=30)

                    arr.append([ObjToStr(subArr[-1]), ObjToStr(prev)])

                    subArr.append(prev)

            respJson = {

                "months": {

                    "1": [],

                    "2": [],

                    "3": [],

                    "4": [],

                    "5": [],

                    "6": []

                }

            }

            k = 1

            for i in (arr):

                l, err = Expense.QueryExpenses(start\_time=i[1], end\_time=i[0], category=category, id=uid)

                if not err:

                    respJson['months'][str(k)] = []

                    k += 1

                resArr = []

                for j in l:

                    dt = j[5].strftime("%d-%m-%Y")

                    print(dt)

                    exp = Expense.NewDict(id=j[0], user=j[1], amount=j[2], category=j[3], description=j[4], date=dt)

                    resArr.append(exp.\_\_dict\_\_)

                respJson['months'][str(k)] = resArr

                k += 1

            d, \_ = User.GetBalance(id=uid)

            bal = {

                "balance": d[0][0],

                "limit": d[0][1]

            }

            respJson['balanceObj'] = bal

            return json.dumps(respJson), 200, TYPE\_OBJ

        app.run(host='0.0.0.0',port=5000)

    except KeyboardInterrupt as e:

        print("LUL")

        utils.db.Connection.close()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Frontend:**

import Sidebar from "../../components/sidebar/Sidebar";

import Navbar from "../../components/navbar/Navbar";

import "./home.scss";

import Widget from "../../components/widget/Widget";

import Featured from "../../components/featured/Featured";

import Chart from "../../components/chart/Chart";

import Table from "../../components/table/Table";

import AddBoxIcon from '@mui/icons-material/AddBox';

import { useEffect, useRef } from "react";

import axios from 'axios'

import { useState } from "react";

const Home = () => {

  const monthArr = [

    'January',

    'February',

    'March',

    'April',

    'May',

    'June',

    'July',

    'August',

    'September',

    'October',

    'November',

    'December'

  ]

  const cateref=useRef();

  const amountref=useRef();

  const descref=useRef();

  const dateref=useRef();

  const token=localStorage.getItem('token')

  const [mainObj, setMainObj] = useState({});

  const [graphObj, setgrObj] = useState([

    {name : monthArr[0], Total: 12000},

    {name : monthArr[1], Total: 21000},

    {name : monthArr[2], Total: 15000},

    {name : monthArr[3], Total: 17000},

    {name : monthArr[4], Total: 11000},

    {name : monthArr[5], Total: 16000}

  ])

  const GetTot = (obj) => {

    let total = 0

    for(const e of obj) {

      console.log(e)

      total += e['amount']

    }

    return total

  }

  useEffect(() => {

    let newObj = []

    let curr = (new Date()).getMonth()

    for(const i in mainObj['months']) {

      newObj.unshift({name: monthArr[curr], Total: GetTot(mainObj['months'][i])})

      curr -= 1

      curr = curr % 12

    }

    setgrObj(newObj)

  }, [mainObj])

  function Query() {

    return new URLSearchParams(window.location.search);

  }

  const format = (date) => {

    return `${date.getFullYear()}-${date.getMonth()+1}-${date.getDate()}`

  }

  useEffect(() => {

    const currDate = new Date()

    const formattedDate = format(currDate)

    let category = Query().get('category');

    const headers = {

      'token': token,

    }

    const data = {

      'date': formattedDate,

      category

    }

    console.log(data)

    console.log(headers)

    axios.post('/expenses', data, {headers}).then(res => {

      setMainObj(res.data)

    }).catch(err => {

      console.log(err)

    })

  }, [])

  const handleClick = () => {

    const category = cateref.current.value

    const amount = amountref.current.value

    const date = dateref.current.value

    const description = descref.current.value

    const reset = () => {

      cateref.current.value = "Housing"

      amountref.current.value = 0

      dateref.current.value = ""

      descref.current.value = ""

    }

    const data = {

      category,

      amount,

      date,

      description

    }

    const headers = {

      'token': token

    }

    console.log(data)

    axios.post('/expense', data,{headers}).then(res => {

      if(res.status == 200) {

        console.log(res.data)

        reset()

      }

    }).catch(err => {

      console.log(err)

    })

  };

  return (

    <div className="home">

      <Sidebar />

      <div className="homeContainer">

        <Navbar />

        <div className="widgets">

          <div className="addexpense">

            <div className="title">

            <h1>Add Expenses</h1>

            <AddBoxIcon className="icon" onclick={handleClick}/>

            </div>

            <div className="inputdetails">

              <div className="first">

              <label htmlFor="category">Category</label>

              <select name="category" id="category" ref={cateref}>

                <option value="Housing">Housing</option>

                <option value="Transportation">Transportation</option>

                <option value="Food">Food</option>

                <option value="Utilities">Utilities</option>

                <option value="Insurance">Insurance</option>

                <option value="Healthcare">Healthcare</option>

                <option value="Repayments">Repayments</option>

                <option value="Personal">Personal</option>

                <option value="Recreation">Recreation</option>

                <option value="Miscellaneous">Miscellaneous</option>

            </select>

              <input   type="number" id="amount"  ref={amountref}/>

              <label htmlFor="amount">Amount</label>

              </div>

              <div className="second">

              <input   type="date" id="Date" ref={dateref}/>

              <label htmlFor="Date">Date</label>

              <input   type="text" id="desc" ref={descref}/>

              <label htmlFor="desc">Description</label>

              </div>

              <button className="button" onClick={handleClick}>Submit</button>

            </div>

          </div>

          <Widget type="balance" amount={2000} diff={30}/>

        </div>

        <div className="charts">

          <Featured />

          <Chart title="Last 6 Months (Expenses)" aspect={2 / 1} data={graphObj} />

        </div>

        <div className="listContainer">

          <div className="listTitle">Latest Transactions</div>

          <Table />

        </div>

      </div>

    </div>

  );

};

export default Home;

import Home from "./pages/home/Home";

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

import List from "./pages/list/List";

import Single from "./pages/single/Single";

import New from "./pages/new/New";

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

import { productInputs, userInputs } from "./formSource";

import "./style/dark.scss";

import { useContext } from "react";

import { DarkModeContext } from "./context/darkModeContext";

import SignUp from "./pages/Register/register";

function App() {

  const { darkMode } = useContext(DarkModeContext);

  return (

    <div className={darkMode ? "app dark" : "app"}>

      <BrowserRouter>

        <Routes>

          <Route path="/">

            <Route index element={<Home />} />

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

            <Route path="register" element={<SignUp />} />

            <Route path="users">

              <Route index element={<List />} />

              <Route path=":userId" element={<Single />} />

              <Route

                path="new"

                element={<New inputs={userInputs} title="Add New User" />}

              />

            </Route>

            <Route path="products">

              <Route index element={<List />} />

              <Route path=":productId" element={<Single />} />

              <Route

                path="new"

                element={<New inputs={productInputs} title="Add New Product" />}

              />

            </Route>

          </Route>

        </Routes>

      </BrowserRouter>

    </div>

  );

}

export default App;

**Github Link :**

**https://github.com/IBM-EPBL/IBM-Project-24647-1659946700**

**Project Demo Link:**

**https://drive.google.com/file/d/197JTuNYydjjlaKTjw6L6pplWdM3mG4yr/view?usp=share\_link**