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

INTRODUCTON

Project Overview:

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 :

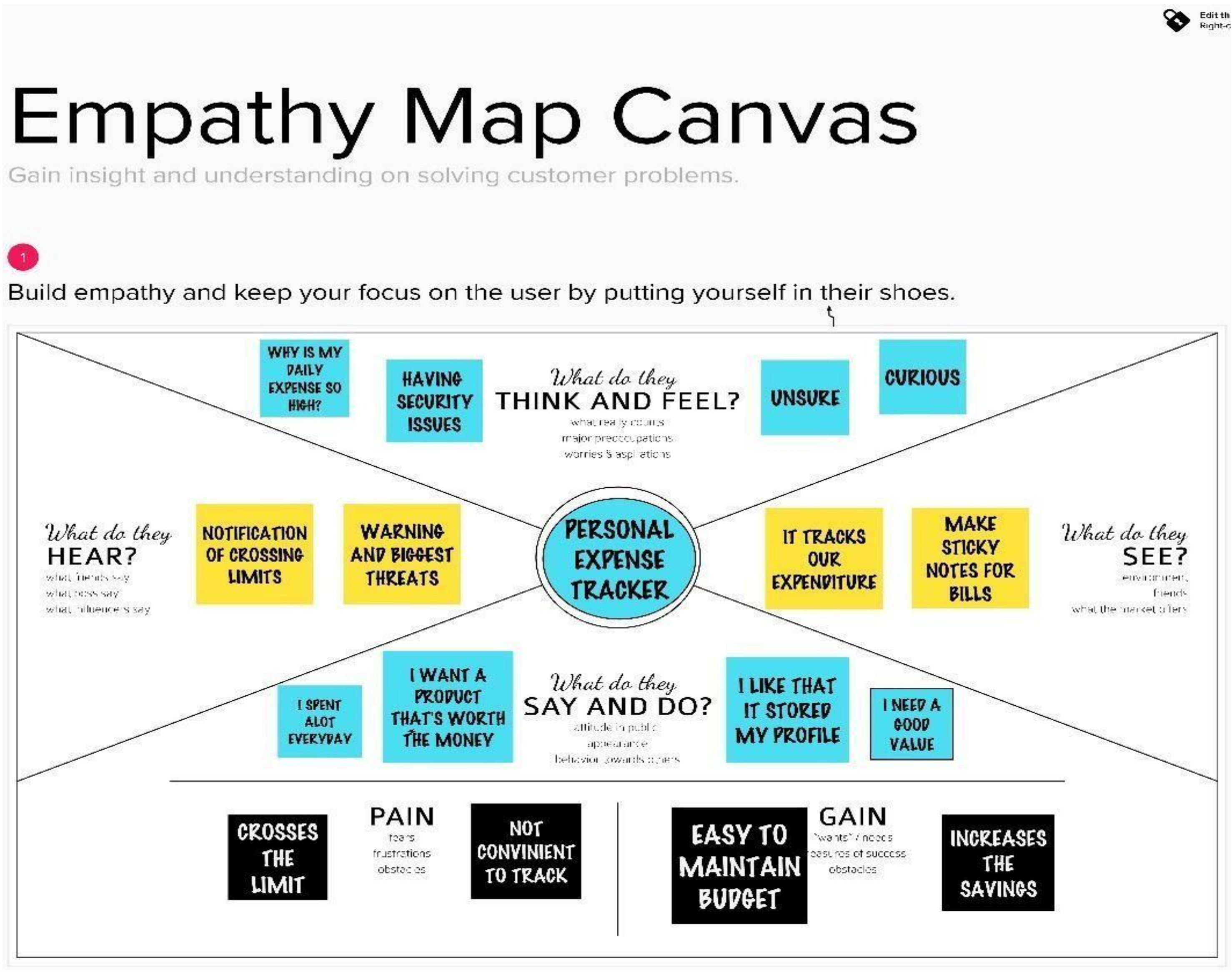
In existing, we need to maintain the Excel sheets, CSV files for the user daily, weekly and monthly expenses and there is no as such complete solution to keep a track of its daily expenses easily. To do so a person as to keep a log in a diary or in a computer system, also all the calculations need to 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.

Ideation and Proposed Solution

Empathy Map Canvas :

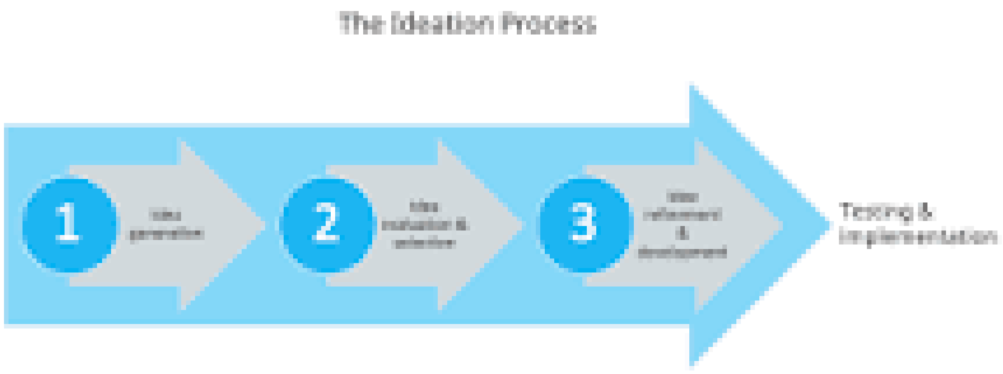


Ideation and Proposed Solution

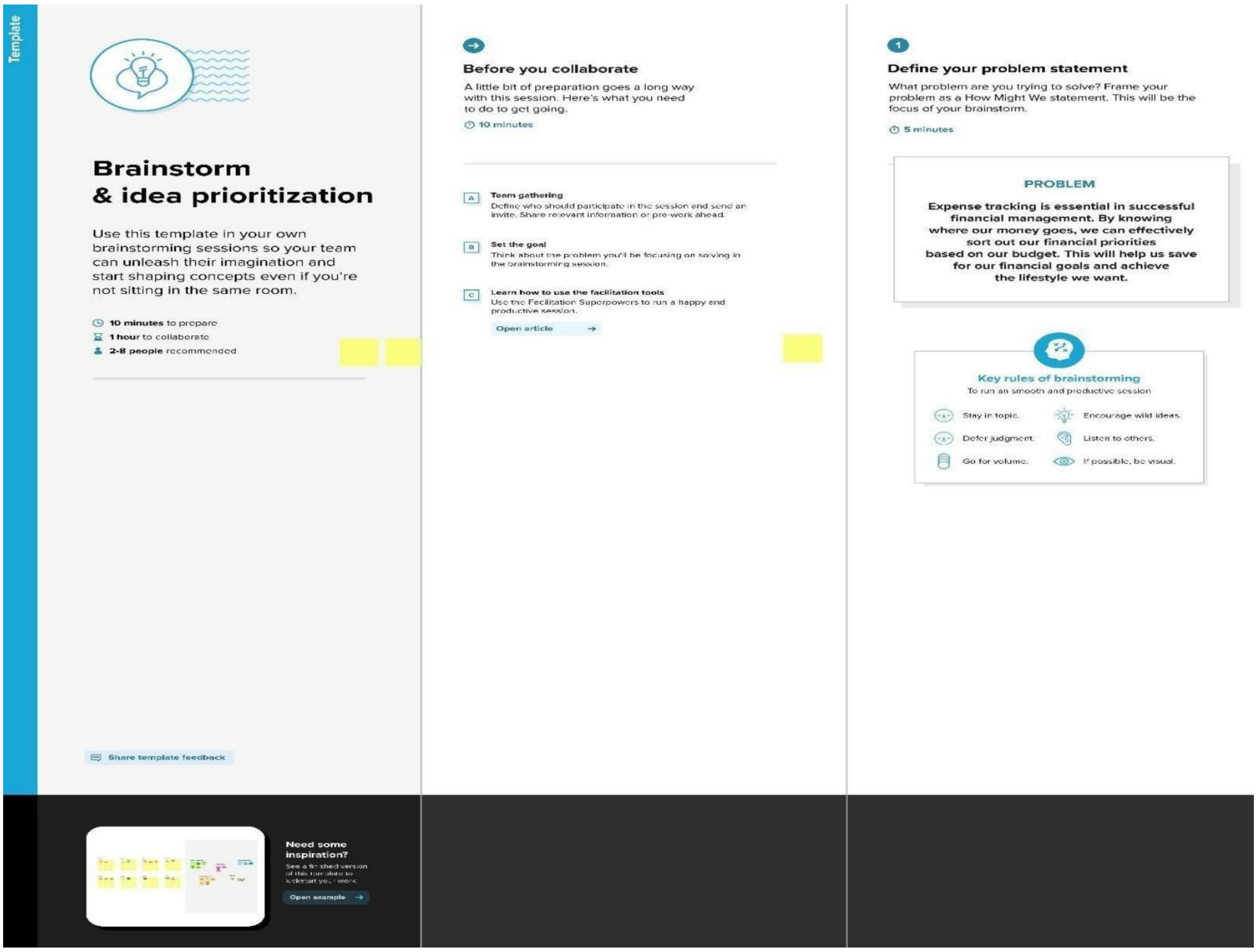
Ideation and Brainstorming :

Brainstorming is a group 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.



Template

Brainstorm
& idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- **10 minutes** to prepare
- **1 hour** to collaborate
- **2-8 people** recommended

Share template feedback

Need some inspiration?
See a finished version of this template to kickstart your work.

Open example →

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

1
Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

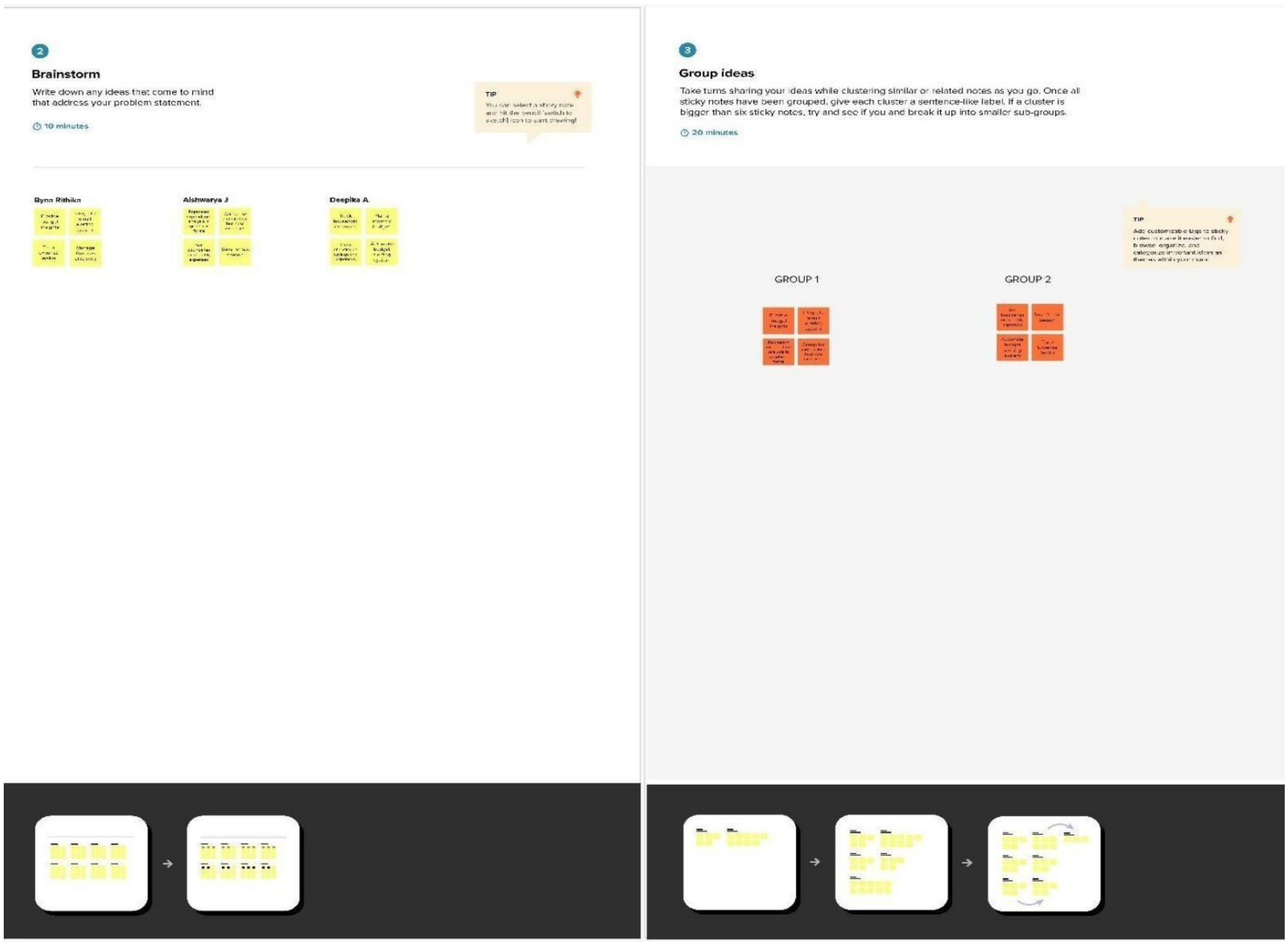
PROBLEM

Expense tracking is essential in successful financial management. By knowing where our money goes, we can effectively sort out our financial priorities based on our budget. This will help us save for our financial goals and achieve the lifestyle we want.

Key rules of brainstorming
To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

2 . Ideation and Proposed Solution



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Byna Rithika

Aishwarya J

Deepika A

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

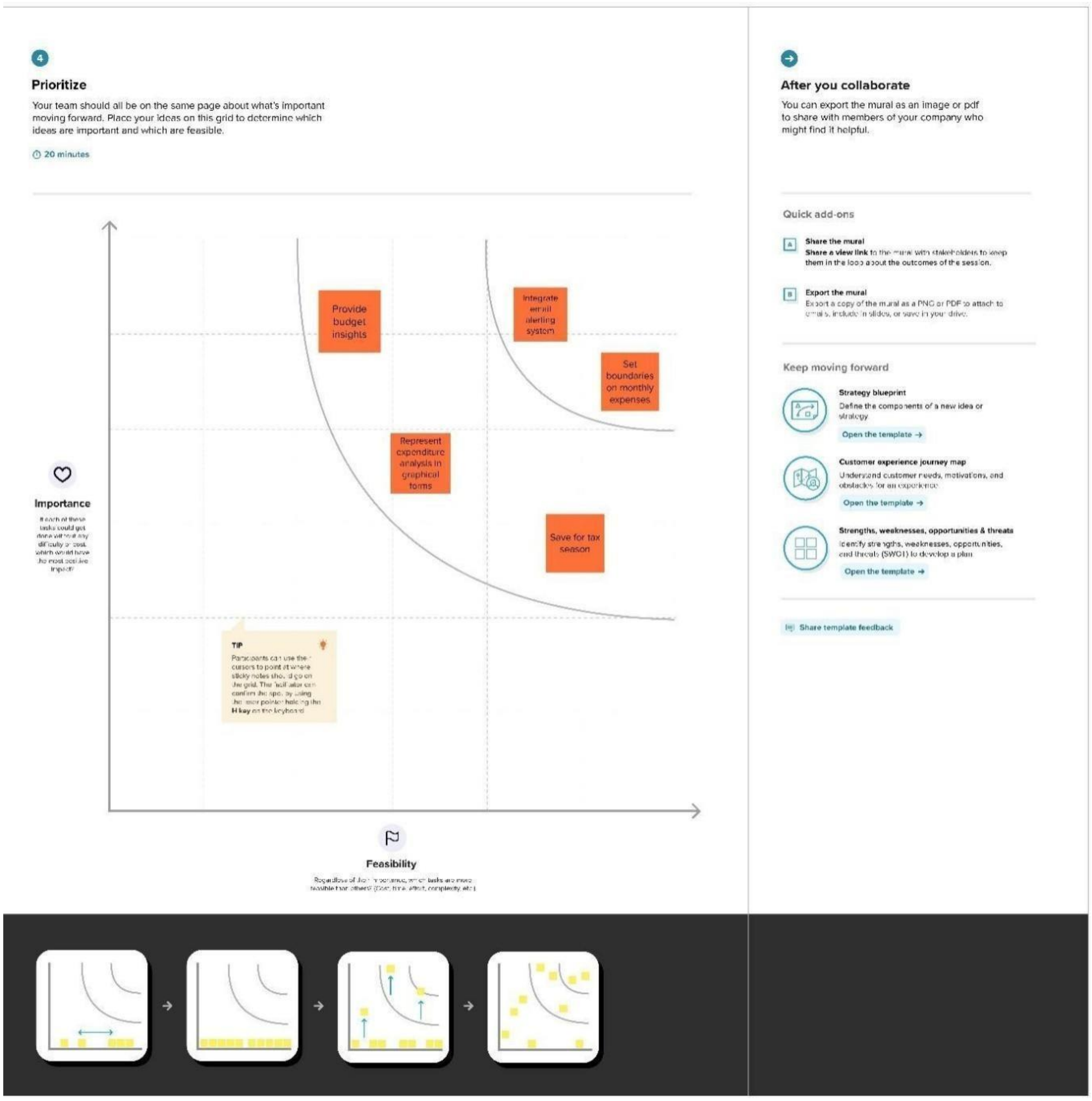
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

GROUP 1

GROUP 2

3.

Ideation and Proposed Solution



4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Provide budget insights

Integrate email alerting system

Set boundaries on monthly expenses

Represent expenditure analysis in graphical forms

Save for tax season

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

Feasibility

Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A **Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B **Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy
Open the template →

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience
Open the template →

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan
Open the template →

Share template feedback

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.

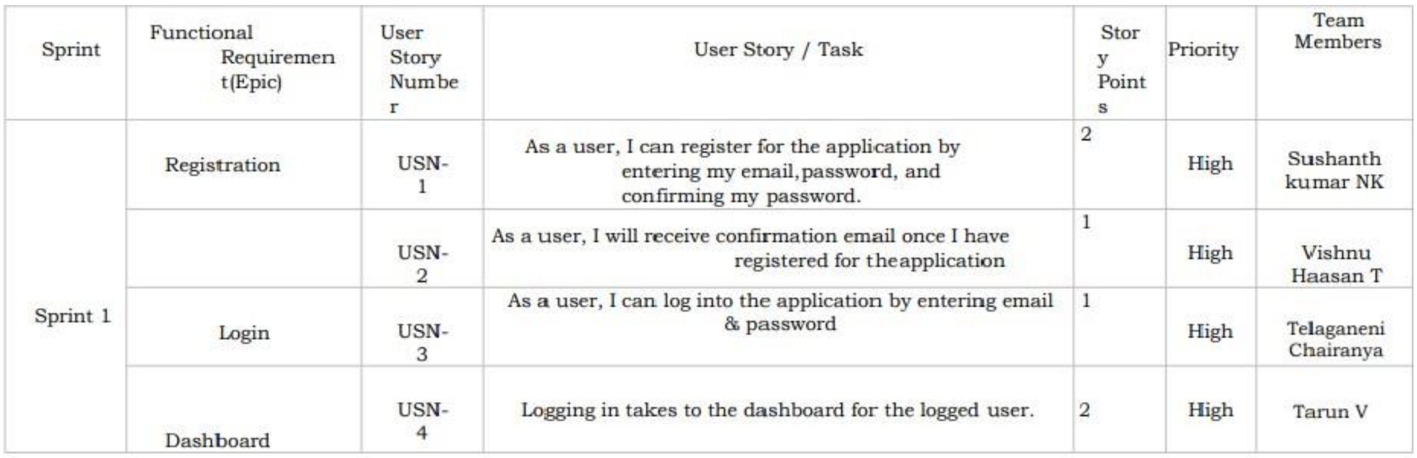
Project Design

Data flow Diagrams :

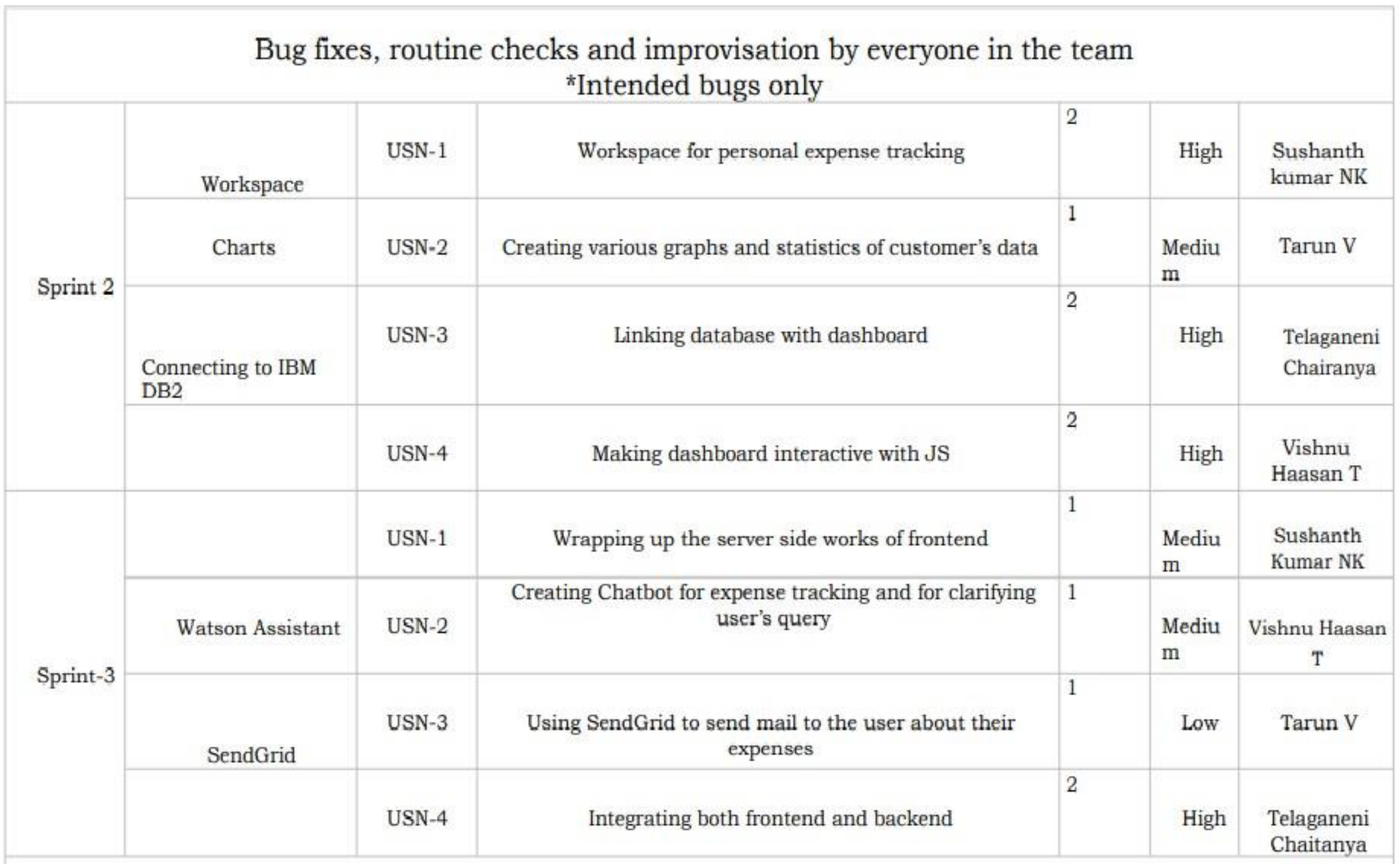
A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. Data flow diagrams provide a straightforward, efficient way for organizations to understand, perfect, and implement new processes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

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	Sushanth kumar NK
		USN-2	As a user, I will receive confirmation email once I have registered for theapplication	1	High	Vishnu Haasan T
	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Telaganeni Chairanya
	Dashboard	USN-4	Logging in takes to the dashboard for the logged user.	2	High	Tarun V



Bug fixes, routine checks and improvisation by everyone in the team
*Intended bugs only

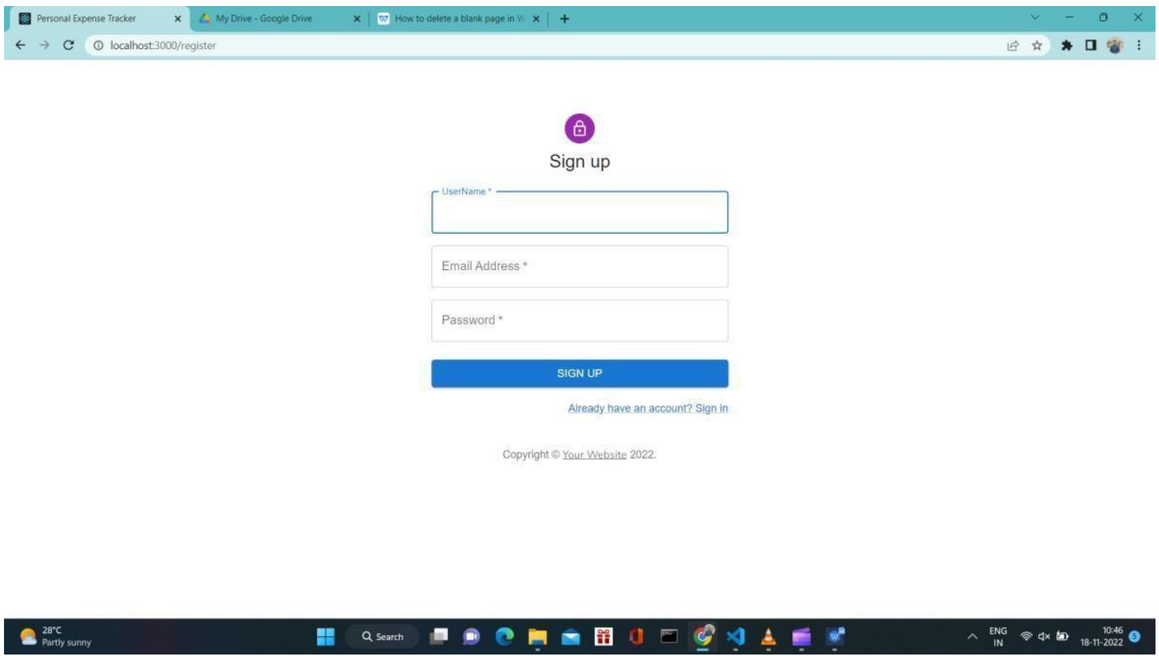
Sprint 2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Sushanth kumar NK
	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Tarun V
	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Telaganeni Chairanya
		USN-4	Making dashboard interactive with JS	2	High	Vishnu Haasan T
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Sushanth Kumar NK
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user's query	1	Medium	Vishnu Haasan T
	SendGrid	USN-3	Using SendGrid to send mail to the user about their expenses	1	Low	Tarun V
		USN-4	Integrating both frontend and backend	2	High	Telaganeni Chaitanya

Sprint-4	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Sushanth Kumar NK
	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Telaganeni Chaitanya
	Docker	USN-1	Creating image of website using docker/	2	High	Vishnu Haasan T
	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Tarun V

Sprint Delivery Schedule:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	23 Oct 2022	27 Oct 2022	20	28 Oct 2022
Sprint-2	20	6 Days	30 Oct 2022	02 Nov 2022	20	03 Nov 2022
Sprint-3	20	6 Days	06 Nov 2022	10 Nov 2022	20	11 Nov 2022
Sprint-4	20	6 Days	13 Nov 2022	18 Nov 2022	20	19 Nov 2022

Register Page :



Sign up

UserName *

Email Address *

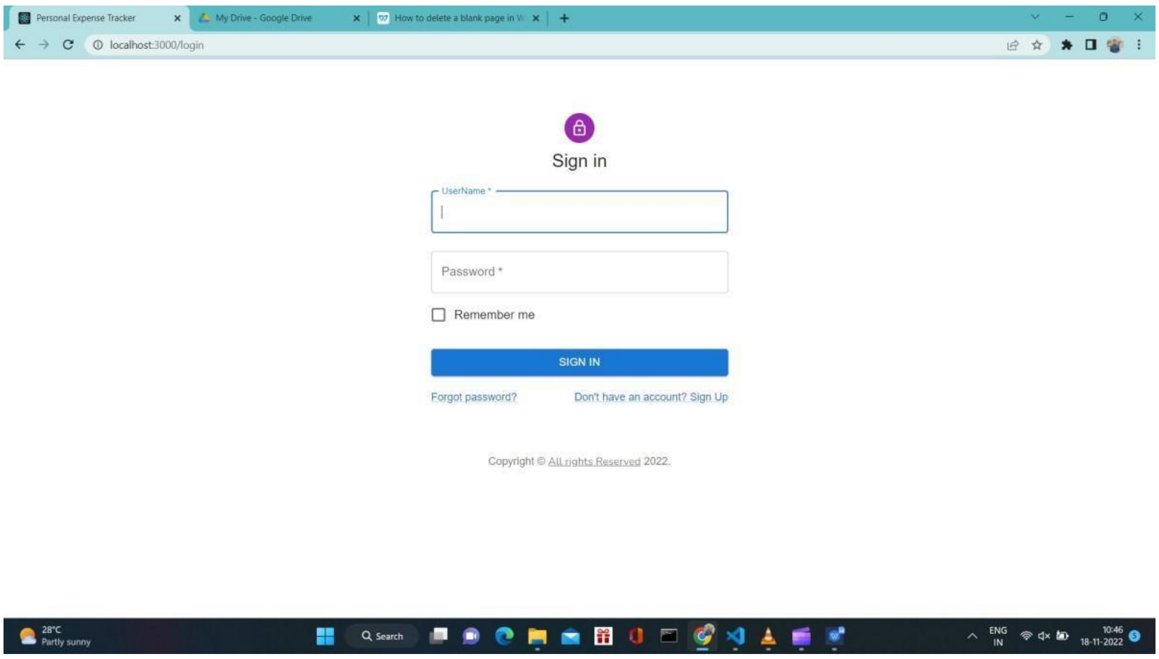
Password *

SIGN UP

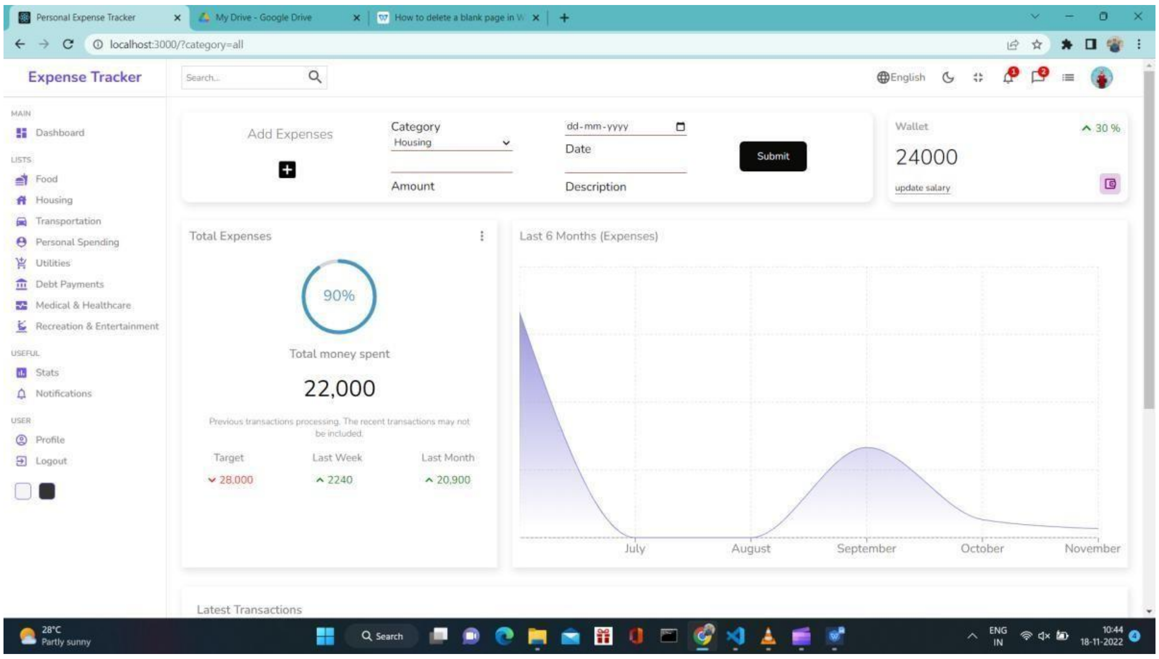
Already have an account? Sign in

Copyright © Your Website 2022.

Sign in :


Personal Expense Tracker
My Drive - Google Drive
How to delete a blank page in W
localhost:3000/login
Sign in
UserName *
Password *
Remember me
SIGN IN
Forgot password?
Don't have an account? Sign Up
Copyright © All rights Reserved 2022.
28°C
Partly sunny
Search
ENG
IN
10:46
18-11-2022

Home Page :



Category: Housing

dd-mm-yyyy

Date

Amount

Description

Wallet 24000 ^ 30 %

Total Expenses

90%

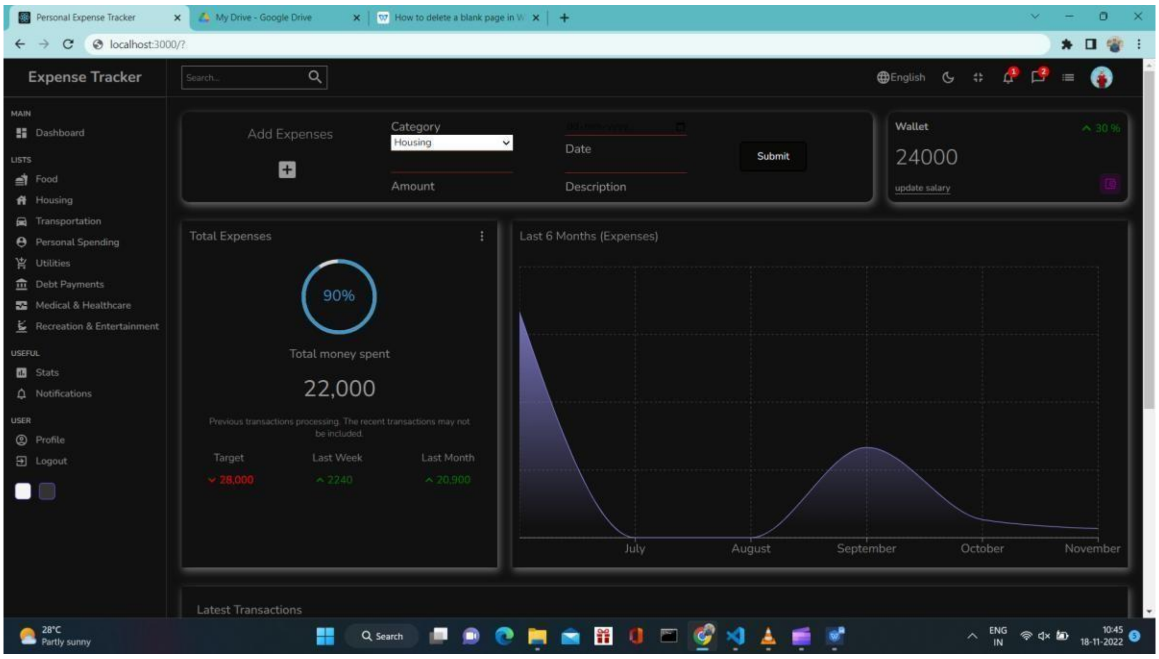
Total money spent

22,000

Previous transactions processing. The recent transactions may not be included.

Target	Last Week	Last Month
28,000	2240	20,900

Dark Mode :



Category: Housing

Date

Amount

Description

Wallet 24000 ^ 30 %

Total Expenses

90%

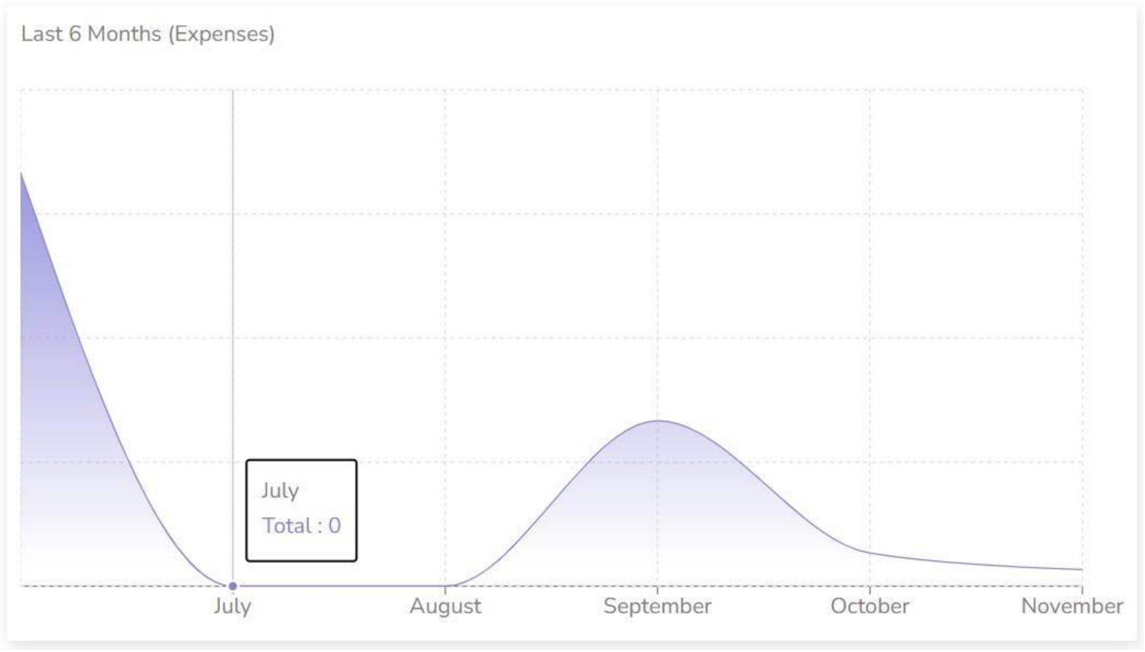
Total money spent

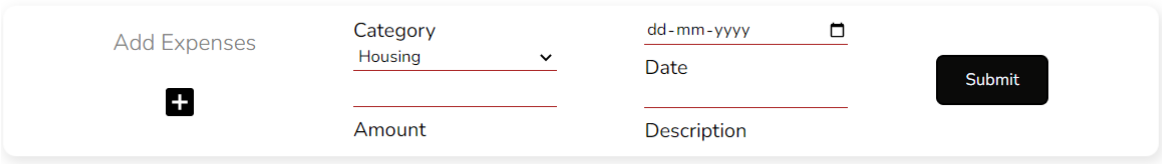
22,000

Previous transactions processing. The recent transactions may not be included.

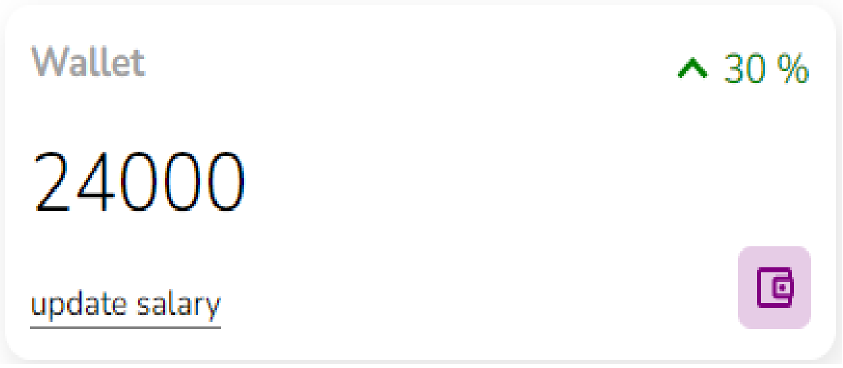
Target	Last Week	Last Month
28,000	2240	20,900

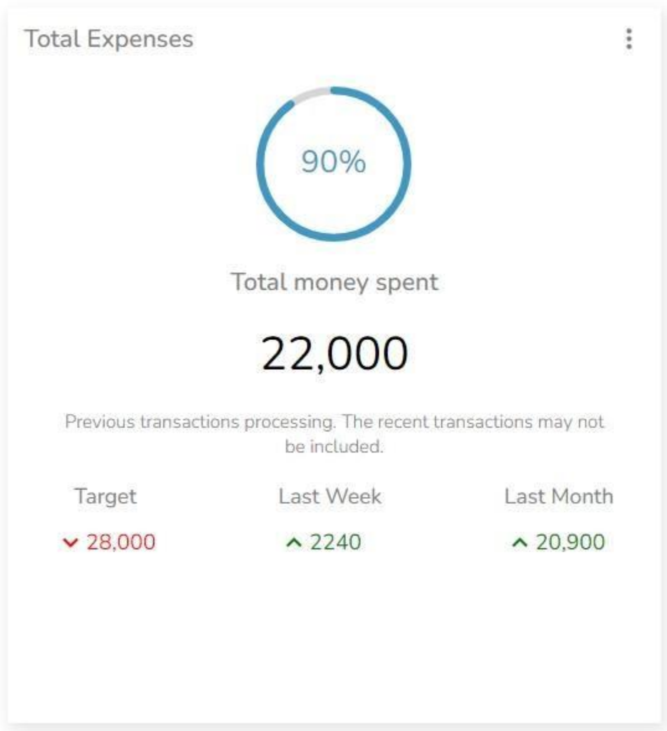
Feature 1 :





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)"

8.Testing

8.1 Test case

TEST CASE ID	PURPOSE	TEST CASES	RESULT
TC1	Authentication	Password with length less than 4 characters	Password cannot be less than 4 characters
TC2	Authentication	User name with length less than 2 characters	User name cannot be less than 2 characters
TC3	Authentication	Valid user name with minimum2 characters	User name accepted
TC4	Authentication	User name left blank	User name cannot be less than 2 characters
TC5	Authentication	Password field left blank	Password cannot be empty
TC6	Authentication	Minimum 4 characters valid password	Password accepted
TC7	Authentication	Password and Confirm Password did not match	Please enter same password
TC8	Authentication	Confirm Password field left blank	Please enter same password

8.2 User Acceptance Testing:

Technical Requirment Document (TSD)

Test Case ID	Test Case Description
TC_001	Verify if user is able to order single product
TC_002	Verify if user is to order multiple products
TC_003	Verify if user can apply single or multiple filters
TC_004	Verify if user can apply different sort by
TC_005	Verify if user is able to pay by Master Card
TC_006	Verify if user is able to pay by Debit Card
TC_007	Verify if user is able to pay fully by reward points
TC_008	Verify if user is able to pay partially by reward points

Results:

Performance Metrics:

● Tracking income and expenses: Monitoring the income and tracking all expenditures (through bank accounts, mobile wallets, and credit & debit cards).

● Transaction Receipts: Capture and organize your payment receipts to keep track of your expenditure.

● Organizing Taxes: Import your documents to the expense tracking app, and it will streamline your income and expenses under the appropriate tax categories.

● Payments & Invoices: Accept and pay from credit cards, debit cards, net banking, mobile wallets, and bank transfers, and track the status of your invoices and bills in the mobile app itself. Also, the tracking app sends reminders for payments and automatically matches the payments with invoices.

● Reports: The expense tracking app generates and sends reports to give a detailed insight about profits, losses, budgets, income, balance sheets, etc.,

● E-commerce integration: Integrate your expense tracking app with your eCommerce store and track your sales through payments received via multiple payment methods.

● Vendors and Contractors: Manage and track all the payments to the vendors and contractors added to the mobile app.

● Access control: Increase your team productivity by providing access control to particular users through custom permissions.

● Track Projects: Determine project profitability by tracking labor costs, payroll, expenses, etc., of your ongoing project.

● Inventory tracking: An expense tracking app can do it all. Right from tracking products or the cost of goods, sending alert notifications when the product is running out of stock or the product is not selling, to purchase orders.

● In-depth insights and analytics: Provides in-built tools to generate reports with easy-tounderstand visuals and graphics to gain insights about the performance of your business.

● Recurrent Expenses: Rely on your budgeting app to track, streamline, and automate all the recurrent expenses and remind you on a timely basis.

Advantages :

1. Improved visibility: Most expense management software includes a dashboard that compiles employee expense data and presents it in an easy-to-understand visual format using charts and other graphics
2. Security: All the Data’s are stored in ibm cloud and db2 so all the data are maintained safely.
3. Month wise Comparison: Using the Expense Manager, you can easily make month on month comparisons of earning, expenses and spending in a more organized manner.
4. Alert Mail: User Receives the alert mail when they exceed the expense limit.

5.Automation: All the calculations are automated. Graph are generated based on the expense made.

6.User Friendly: Expenses can be added easily

Disadvantages :

1. Requires Internet Connection: This web application requires an active internet connection to access.
2. Cost: Using cloud service need some investments. Every time we can't access the cloud freel

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/1Dlon_9RXuVacMcEMkuOiNMBueEl gs0D7/view?usp=sharing