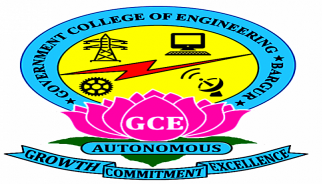
NALAIYA THIRAN PROJECT – EXPERIENTIAL PROJECT BASED LEARNING

PERSONAL EXPENSE TRACKER APPLICATION



TEAM ID: PNT2022TMID06613

BATCH: B1-1M3E

TEAM MEMBERS:

ANEESH KUMAR D - 6107191105
DINESH KUMAR G - 6107191112
EZHUMALAI K - 6107191113
GOKUL P - 6107191115

INDUSTRY MENTOR NAME: Ms. KUSHBOO

FACULTY MENTOR NAME: LOGANAYAGI K

CHAPTER-1
INTRODUCTION

1.1 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 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. Most of the people cannot track their expenses and income one way they face a money crisis, in this case daily expense tracker can help the people to track income-expense day to day and making life tension free. Money is the most valuable portion of our daily life and without money we will not last one day on the earth. So using the daily expense tracker application is important to load a happy family. Daily expense tracker helps the user to avoid unexpected expenses and bad financial situations. This Project will save time and provide a responsible lifestyle.

1.2 Purpose

The main reason you should track your expenses is **to identify and eliminate wasteful spending habits in your financial life**. Moreover, consistently tracking your expenses will help you maintain control of your finances, and promote better financial habits like saving and investing. Recording your expenses daily can ensure that you are financially aware all year long and not just during tax season. Knowing where your money is going and how much you’re spending can improve your spending habits. Plus, you’ll have a better idea of where you can allocate money to positively impact your bottom line.

CHAPTER-2

LITERATURE SURVEY

a. Existing problem

As a user We face many difficulties in our daily file. In our daily life money is the most important portion and without it we cannot last one day on earth but if we keep on track all financial data then we can overcome this problem. Most of the people cannot track their expenses and income one way they face the money crisis and depression. This situation motivates us to make an android app to track all financial activities. Using the Daily Expense Tracker user can be tracking expenses day to day and making life tension free.

b. References

Paper 1
Title: Application for Predictive Recommendation and Visualization of Personal Expenses
Authors: *Darsh Shah, Sanay Shah, Ritik Savani, Dr. Bhavesh Patel, Ashwini Deshmukh*

Paper 2
Title: Utilization of QR Code for Tracking Digital Expenses
Authors: Farhan Putra Salsabil1, Gerardo Axel Lwiantoro2, Gregorius A. N. Aditantyo3

Paper 3
Title: A Novel Expense Tracker using Statistical Analysis
Authors: *Muskaan Sharma, Ayush Bansal, Dr. Raju Ranjan, Shivam Sethi*

Paper 4
Title: Daily Expense Tracker
Authors: Shivam Mehra, Prabhat Parashar

Paper 5
Title: A Review on Budget Estimator Android Application Authors: *Namita Jagtap, Priyanka Joshi, Aditya Kamble*

Paper 6
Title: STUDENT EXPENSE TRACKING APPLICATION

Authors: Saumya Dubey1, Pragya Dubey2, Rigved Rishabh Kumar 3, Aaisha Khatoon4

Paper 7
Title: Time Scheduling and Finance Management: University Student Survival Kit
Authors:*J.L. Yeo, P.S. JosephNg, K.A. Alezabi, H.C. Eaw, K.Y. Phan (2020, IEEE)*

Paper 8
Title: Development of an Application for Expense Accounting
Authors: *Denis E. Yurochkin; Anton A. Horoshiy; Saveliy A. Karpukhin(2021, IEEE)*

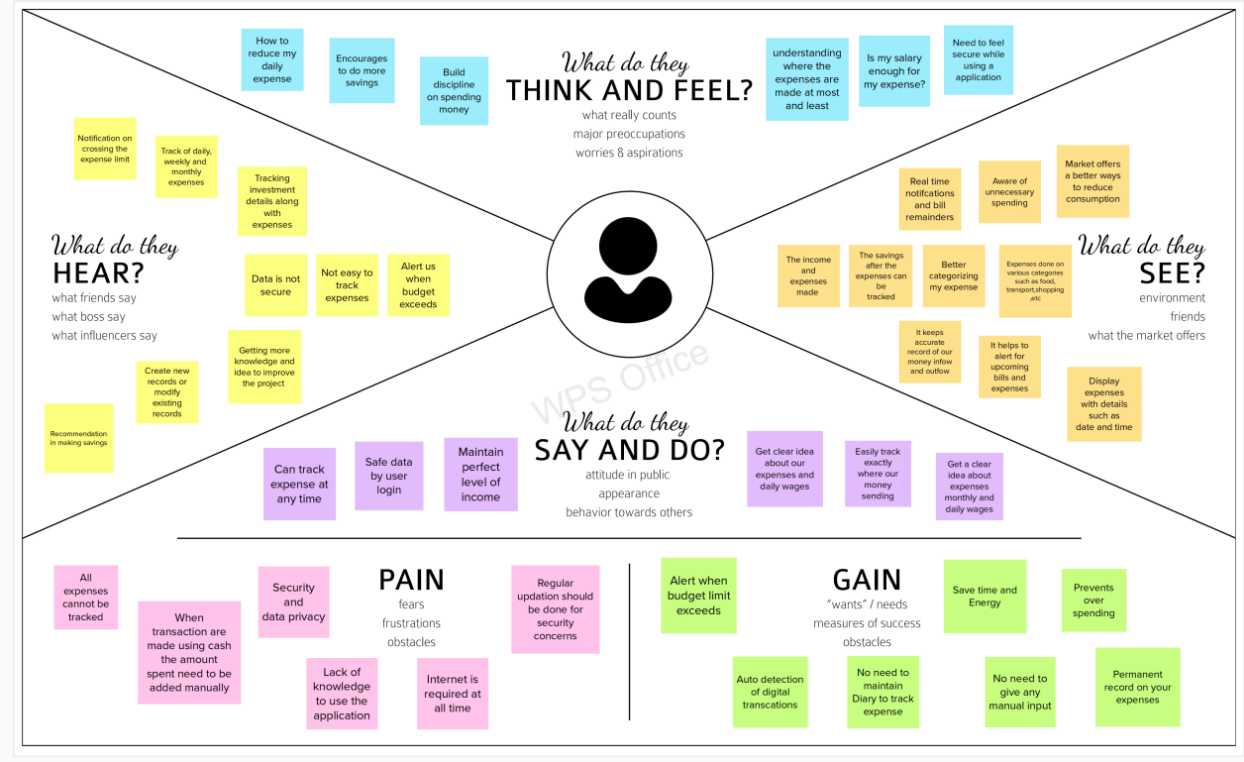
c. **Problem Statement Definition**

Many organizations have their own system to record their income and expenses, which they feel is the main key point of their business progress. It is good habit for a person to record daily expenses and earning but due to unawareness and lack of proper applications to suit their privacy, lacking decision making capacity people are using traditional note keeping methods to do so. Due to lack of a complete tracking system, there is a 2 constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.

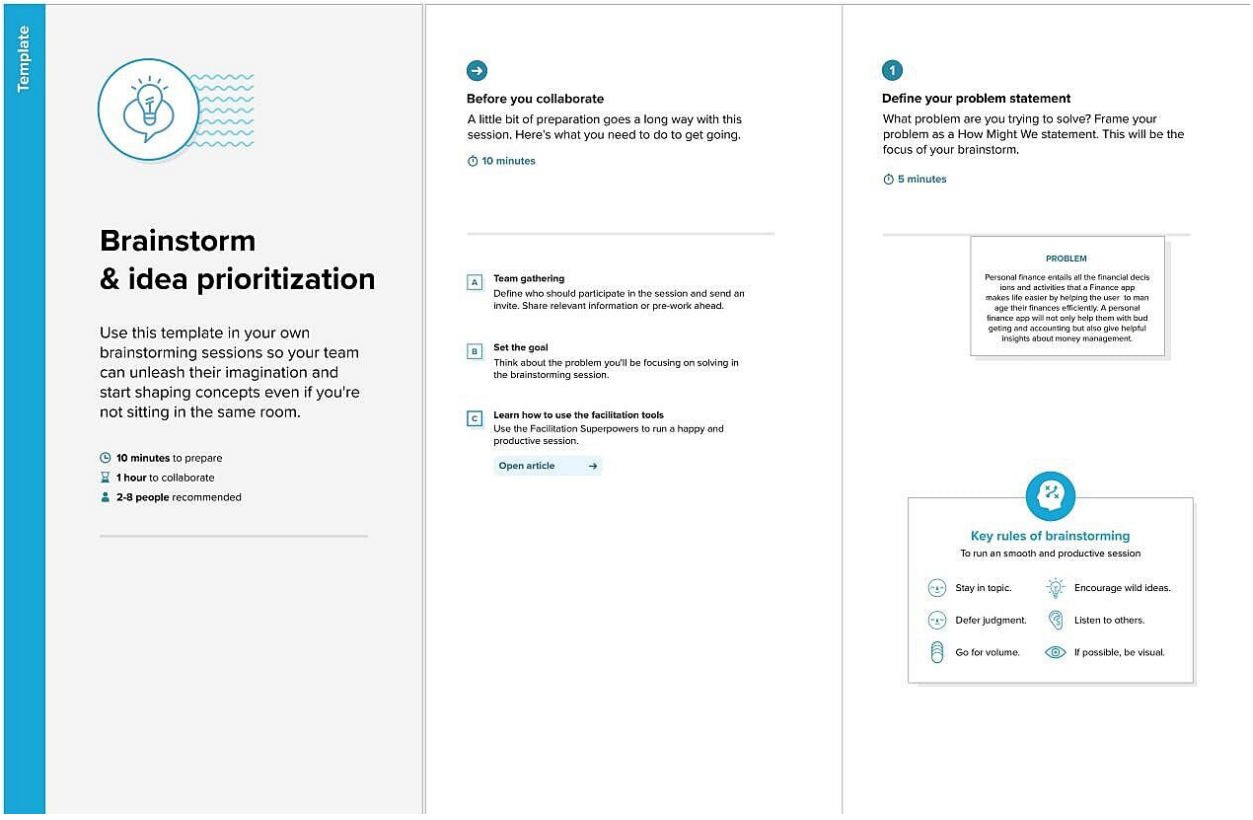
CHAPTER-3

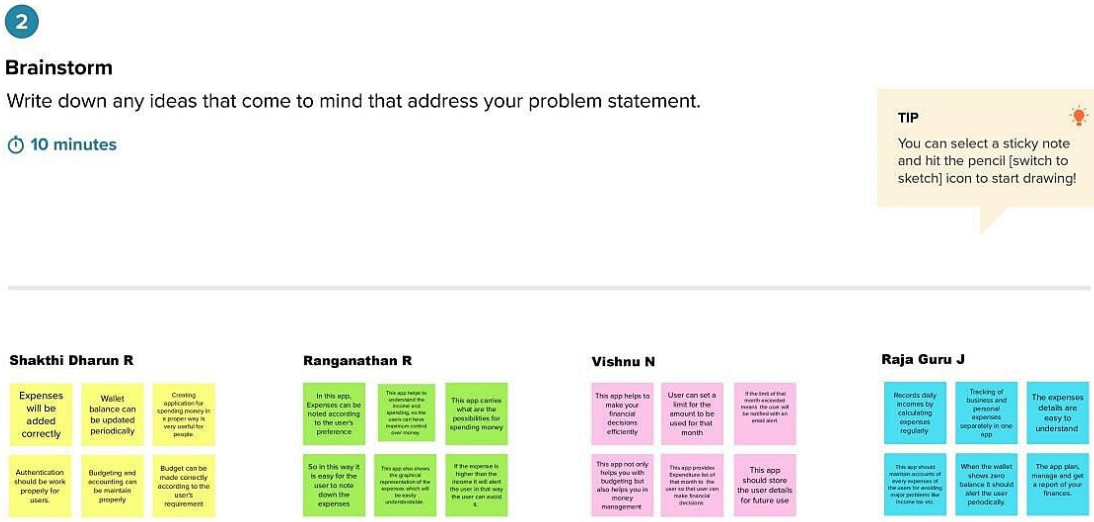
IDEATION & PROPOSED SOLUTION

a. Empathy Map Canvas



b. **Ideation & Brainstorming**



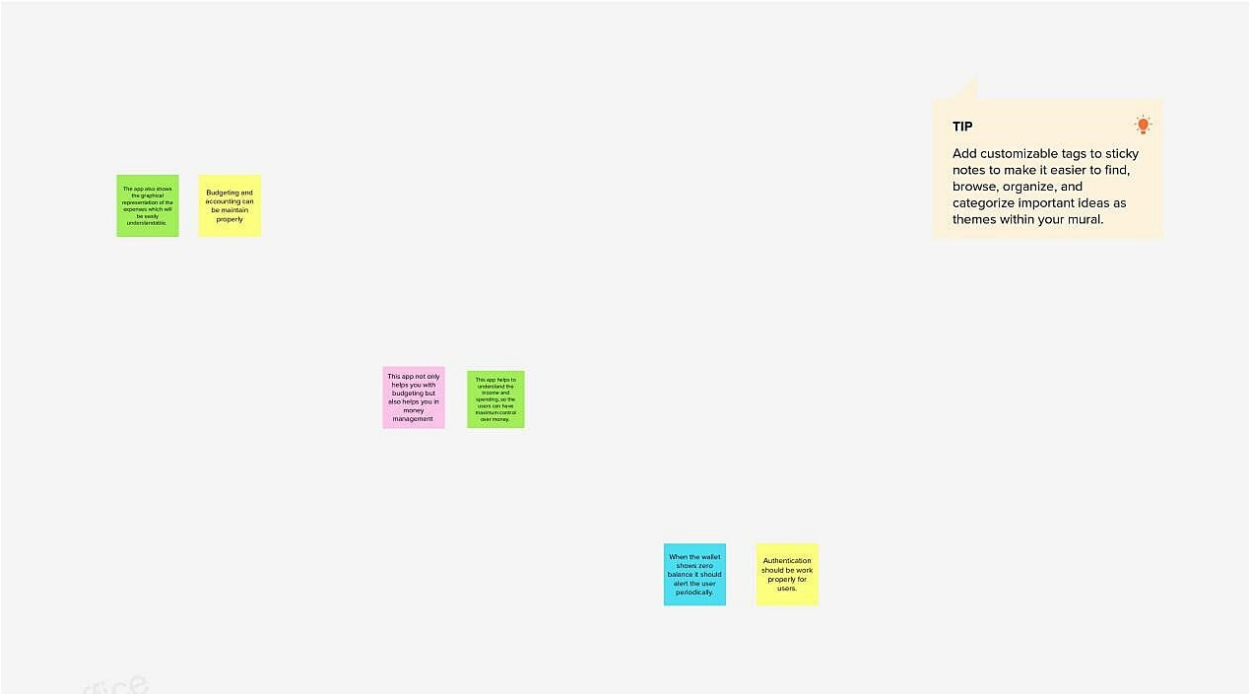


3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, 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

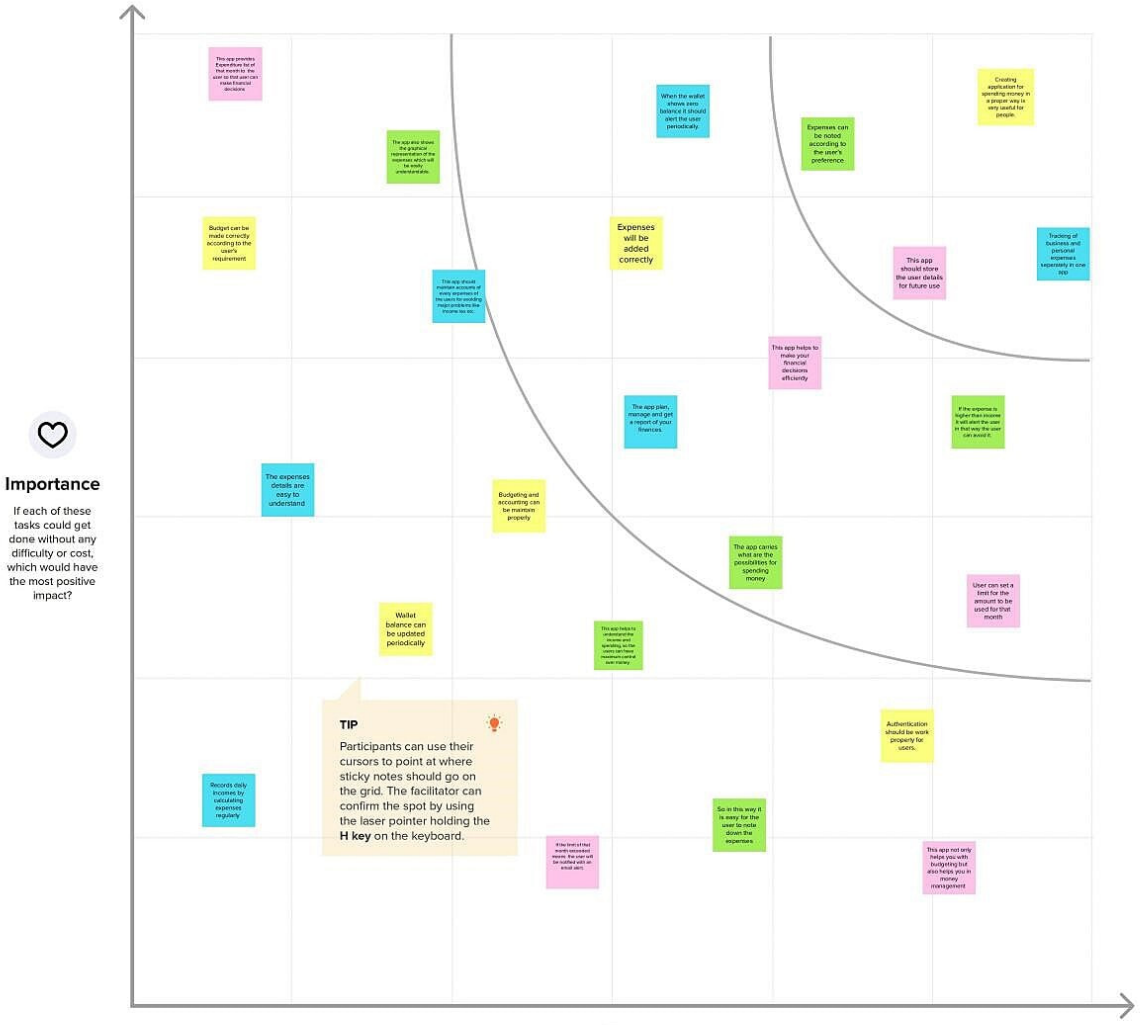


4

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



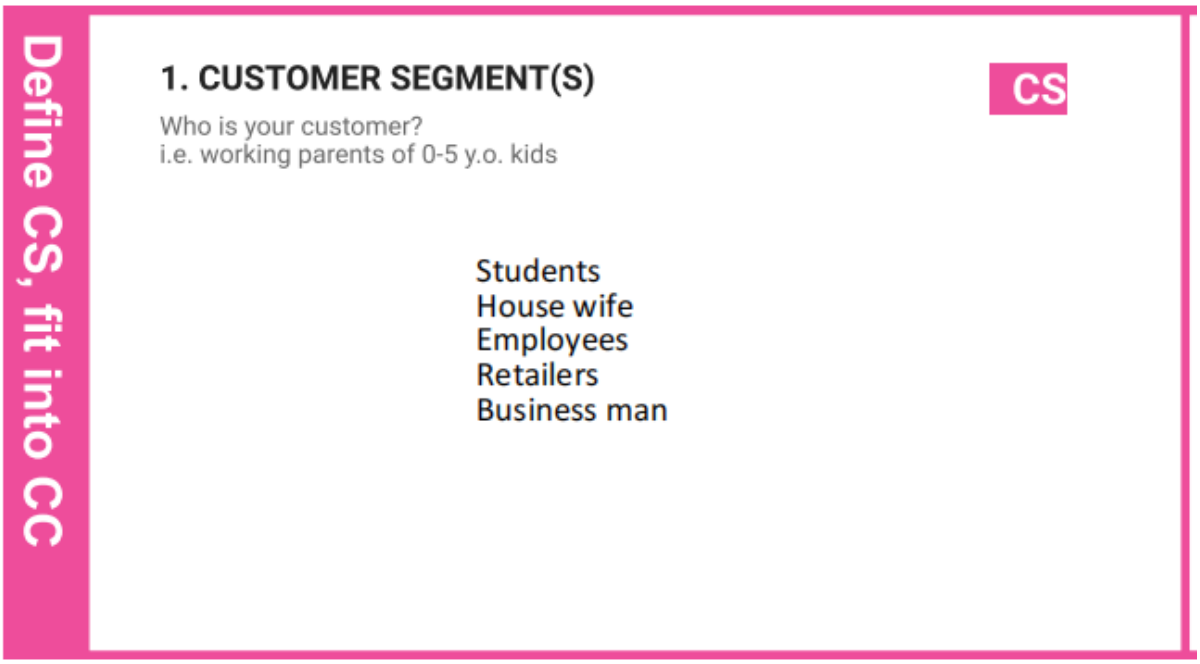
c. **Proposed Solution**

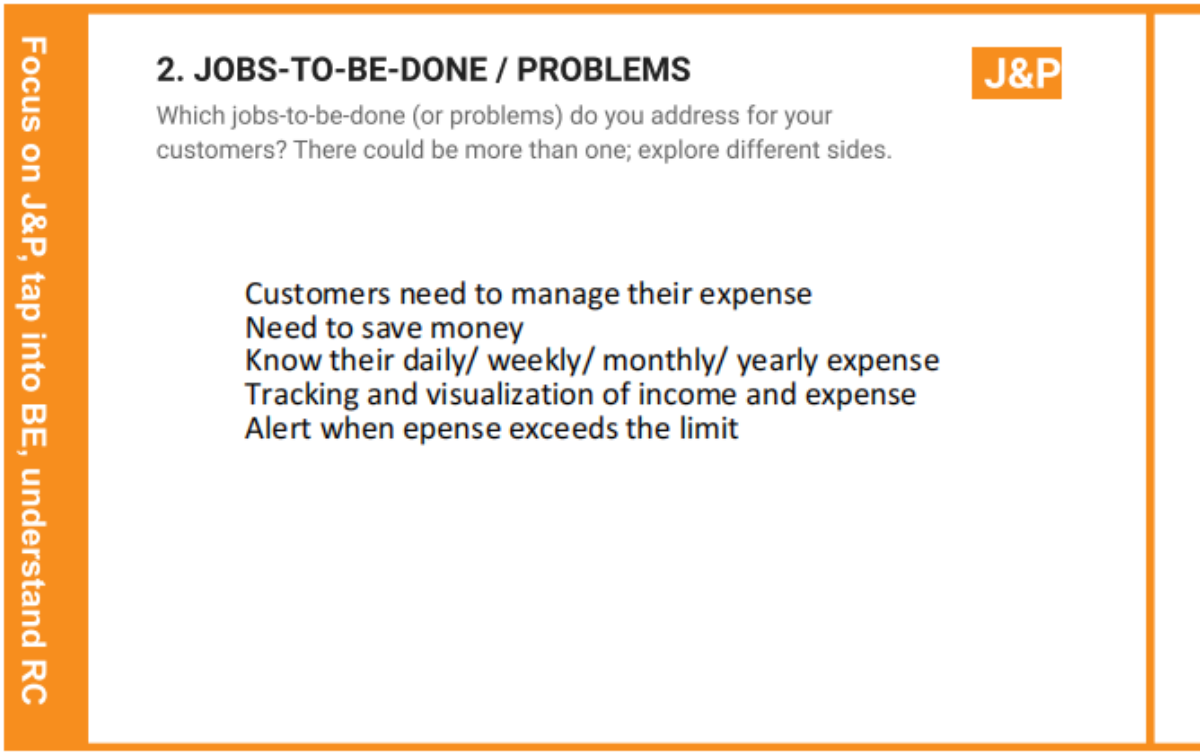
S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Modern life offers a excessive options of services and goods for consumers. As a result, people's expenses have gone up dramatically, e.g., compared to a decade ago, and the cost of living has been increasing day by day. Thus it becomes essential to keep a check on expenses in order to live a good life with a proper budget set up.
2.	Idea / Solution description	Our goal is to create an personal expense tracking application where user can be tracking all financial activities and view previous income and expense report. 1. User can easily review the reports daily, weekly, monthly or yearly. 2. User can update or delete records. 3. User can get notification daily. 4. Create Category and Change currency. 5. User can also change Notification time and modify some features. 6. Add Expense and Income

3.	Novelty / Uniqueness	Personal Expense Tracker is a simple application. Writing in a user's pocket handbook every time during income and expense is not convenient and it is easy because of this application, the user will be able to manage the money through his/her smart devices without any hassle under any circumstances. Users just need to enter income and expense and the app calculates it for users. This application is very easy, fast and secure with money calculation and online mode service
4.	Social Impact / Customer Satisfaction	1. Eliminate paper, automatically route expense report to the user and reducing time & cost of processing which help customer to save their time. 2. Software reduces like-hood of data entry errors and can find duplicate entries so that customer don't do any extra payments 3. Reporting and analytics provide real time insight into expense by user as category which enable customer to spend their time on higher value task
5.	Business Model (Revenue Model)	Freemium Model

6.	Scalability of the Solution	1. Payment mode embedded with the app 2. The application can be used to collect samples of data related to user's expenses with permissions and use those sample data as parameters to evaluate patterns of spending. 3. Using some data mining technique expenses can be classified and can be used in market analysis and planning. 4. This application will not only helps users to manage their expenses but also help marketing executives to plan marketing according to the needs of users.

d. **Problem Solution fit**





Identify strong TR & EM

3. TRIGGERS

TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

Customers are triggered when they see their expense is higher than their income.
When they came to know about more productive and efficient way to manage their expense.
When they want to save the money

4. EMOTIONS: BEFORE / AFTER

EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

Before:
Feel like spendthrift
Spending money on unwanted things
Incapable to manage money
Not saving for future

After:
Feel like thrifty
Capable to manage money
Saving for future plans
Using income on important works

Explore AS, differentiate

5. AVAILABLE SOLUTIONS

AS

Which solutions are available to the customers when they face the problem
or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

Customer in past tried a lot of things to manage their expense like sticky notes, spreadsheet and ledger that cause confusion, data inconsistency problems while recording and splitting of expenses
All they need is a person to maintain their expense and show their statistics. But all people can’ t afford a separate person to manage their own expenses. So in this modern world they need a app which make their management easier.

6. CUSTOMER CONSTRAINTS

CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

Internet Dependence
Reduced Speed
Internet reliance
Security
Restricted Functionality
Availability
Web Issues
Browser Support

7. BEHAVIOUR

BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

Customers is given instant access to online chat features with knowledgeable staff members
Customers is given option to email the application support team with questions or concerns and make sure the response time is fast
Give customers the option of using the phone and talking to a live operator because some consumers are averse to or unfamiliar with virtual communication forms.

Focus on J&P, tap into BE, understand RC

8. CHANNELS of BEHAVIOUR

CH

8.1 ONLINE

What kind of actions do customers take online? Extract online channels from #7

8.2 OFFLINE

What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

Online:
Social Media Marketing
Google Advertisement
Youtube Advertisement

Offline:
Newspaper
Radio
Existing Customer Recommendation

Identify strong TR & EM

9. PROBLEM ROOT CAUSE

RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

People dont check their spending and create a budget, theyl have no control whatsoever on money. Instead, money control them, and theyl either have perpetual lack of funds or will end up steeped in debt.

So many people don’ t have great financial management skills, they willl not know how to categorize your expenses.

When they don’ t keep a watch on your spending, there will be short of money, always. This will stress them out.

People are spending money frivolously, they not have money to set financial goals.

10. YOUR SOLUTION

SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

This application help the users to add their expenses so they can get an analysis of their expenditure in graphical form. They have option to set the limit of amount to be used for a particular month and if the limit exceed the user will be notified with alert message.

CHAPTER -4

REQUIREMENT ANALYSIS

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Calendar	Personal expense tracker application shall allow user to add the data to their expenses.
FR-4	User monthly income data	Data to be registered in the app
FR-5	Alert/ Notification	Alert through E-mail Alert through SMS
FR-6	User Budget Plan	Planning and tracking of user expense vs budget limit

4.2 Non-functional Requirements:

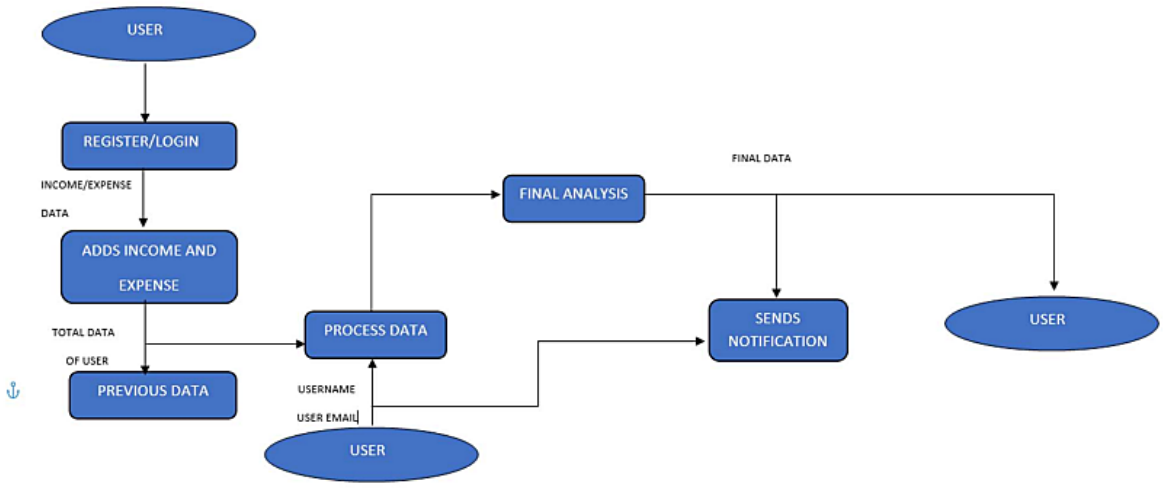
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Expense Tracker is highly reusable. The application can be modified for just about anything!
NFR-2	**Security**	Authentication, Authorization, Encryption, Application security testing. More security of the customer data and bank account details.
NFR-3	**Reliability**	Each data record is stored on a well built efficient database schema. There is no risk of data loss.
NFR-4	**Performance**	Application performance monitoring (APM)
NFR-5	**Availability**	It is available all the time, no time constraint
NFR-6	**Scalability**	Capacity of the application to handle growth, especially in handling more users.

CHAPTER-5

PROJECT DESIGN

a. Data Flow Diagrams



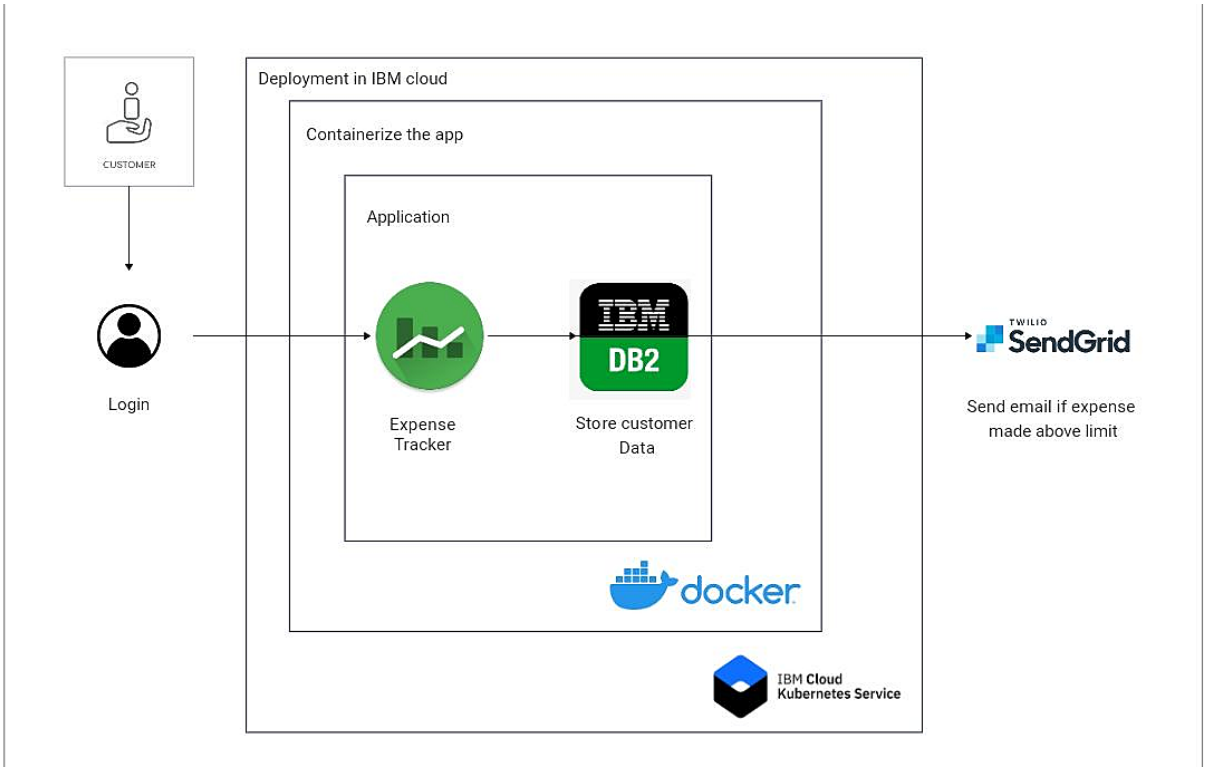
b. Solution & Technical Architecture

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g., Web UI, Mobile App, etc.	HTML, CSS, Python flask
2	Registration	User register in the application to start the process	HTML, CSS, Python flask, IBM cloud, IBM Container registry
3	Login	User login to their account	HTML, CSS, Python flask, IBM cloud, IBM Container registry
4	Wallet page	User can add their expenses in the wallet	HTML, CSS, Python flask, IBM cloud, IBM Container registry

5	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
6	Email alert	User can be notified when their expenses cross the limit in the wallet	Kubernetes, IBM container registry, SendGrid
7	Graphical view	User can able to see their monthly expenses in a graph format	IBM cloud object storage, IBM container registry, HTML, CSS

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Docker and Kubernetes are the open source frameworks	Docker, Kubernetes
2	Security Implementations	IBM DB2 is used for the security control	IBM DB2
3	Scalable Architecture	This architecture connects the three dimensions like processing, storage and connectivity between the user and the system	Python flask, IBM container registry
4	Availability	It is always available	Python flask and IBM cloud
5	Performance	The application can perform well user can experience the fast while using the application	Python flask and IBM cloud


CUSTOMER
Login
Deployment in IBM cloud
Containerize the app
Application
Expense Tracker
IBM DB2
Store customer Data
docker
IBM Cloud Kubernetes Service
TWILIO SendGrid
Send email if expense made above limit

CHAPTER-6

PROJECT PLANNING & SCHEDULING

a. Sprint Planning & 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	Gokul P
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Aneeshkumar D
			As a user, I can register for the application through Gmail	1	High	Ezhumalai K
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Dineshkumar G
Sprint-1	Dashboard	USN-4	As a user, I can add income and expenses in the application	2	High	Gokul P

Sprint-2	Charts	USN-2	Creating various graphs and statistics of customer’s data	1	Medium	Ezhumalai K
Sprint-2	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Aneeshkumar D
Sprint-2		USN-4	Making dashboard interactive with JS	2	High	Gokul
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Dineshkumar G
Sprint-3	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user’s query	1	Medium	Aneeshkumar D
Sprint-3	SendGrid	USN-3	Using SendGrid to send mail to the user about	1	Low	Dineshkumaar D
Sprint-3		USN-4	Integrating both frontend and backend	2	High	Gokul P
Sprint-4	Docker	USN-1	Creating image of website using docker	2	High	Dineshkumar G
Sprint-4	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Ezhumalai K
Sprint-4	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Aneeshkumar D

Sprint-4	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Gokul P

b. **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	04-11-2022	09-11-2022	20	09-11-2022
Sprint-2	20	6 Days	05-11-2022	10-11-2022	20	10-11-2022
Sprint-3	20	6 Days	06-11-2022	11-11-2022	20	11-11-2022
Sprint-4	20	6 Days	07-11-2022	12-11-2022	20	12-11-2022

CHAPTER-7
CODING & SOLUTIONING

Feature 1

User Login

The ***login page*** allows a user to gain access to an application by entering their username and password or by authenticating using a social media login.

Code:

```
<!DOCTYPE html>
<html>
<head>
	<title>Login Form</title>
	<link rel="stylesheet" type="text/css" href="..\static\css\login.css">
	<link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap" rel="stylesheet">
	<script src="https://kit.fontawesome.com/a81368914c.js"></script>
	<meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body >
	<img class="wave" src="..\static\images\wave.png">
	<div class="container">

		<div class="img">
			<div id="png"><a href="/" title="HOME"><img style="width:75px; height:75px ; " src="..\static\images\home-page.png"></a></div>
			<img src="..\static\images\bg.svg">
		</div>


		<div class="login-content">

			<form action='/login' method="POST">
				<div class="msg"></div>
				<img src="..\static\images\avatar.svg">
				<h2 class="title">Welcome</h2>
		<div class="input-div one">
		  <div class="i">
				<i class="fas fa-user"></i>
		  </div>
		  <div class="div">
				<h5>Username</h5>
				<input type="text" name="username" class="input" required>
		  </div>
```

```
                </div>
                <div class="input-div pass">
                  <div class="i">
                        <i class="fas fa-lock"></i>
                  </div>
                  <div class="div">
                        <h5>Password</h5>
                        <input type="password"  name="password" class="input" required>
                  </div>
                </div>
                <a href="#">Forgot Password?</a>
                <input type="submit" class="btn" value="Login">
                                <span>OR</span>

                                <div><b>Login with</b></div>
                                <div>
                                        <ul>
                                                <li><a href="#"><i class="fab fa-facebook" aria-
hidden="true"></i></a></li>
                                                  <li><a href="#"><i class="fab fa-twitter" aria-
hidden="true"></i></a></li>
                                                  <li><a href="#"><i <i class="fab fa-google"
aria-hidden="true"></i></a></li>
                                                  <li><a href="#"><i class="fab fa-linkedin" aria-
hidden="true"></i></a></li>
                                                  <li><a href="#"><i class="fab fa-instagram"
aria-hidden="true"></i></a></li>
                                        </ul>

                                </div>
                                <div class="app" ><b>Don't have an account?</b><a
id="app1" href="\signup">REGISTER.here</a></div>
        </form>

      </div>

   </div>

   <script type="text/javascript" src="..\static\js\login.js"></script>
</body>
```

```
</html>
```

User Register

A signup page (also known as a registration page) **enables users and independently register and gain access to the application**.

```
<html>
<head>
<meta charset="utf-8">
<title>Sign-up</title>
<link href="..\static\css\signup.css" rel="stylesheet">
<script src="https://kit.fontawesome.com/a81368914c.js"></script>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
 <!--container---------------------->
<div class="container" >
 <!--sign-up-box-container--->
<div class="sign-up">

   <div id="png"><a href="/" title="HOME"><img style="width:55px; height:55px ; "
src="..\static\images\home-page.png"></a></div>
 <!--heading-->
 <form action="/register" method="post">
   <div class="msg"></div>
 <h1 class="heading">Hello,Friend</h1>
 <!--name-box-->
 <div class="text">
 <img height="20px" src="..\static\images\user.png" />
 <input placeholder="Name" type="text" name="username"/>
 </div>
 <!--Email-box-->
 <div class="text">
 <img height="12px" src="..\static\images/email.png" />
 <input placeholder=" Example@gmail.com" type="email" name="email"" />
 </div>
```

```
 <!--Password-box-->
 <div class="text">
 <img height="20px" src="..\static\images\password.png" />
 <input placeholder=" Password" type="password" name="password"/>
 </div>
<div class="or"><b>OR</b></div>
<div class="s1"><p><b>Sign-up with</b></p></div>

 <div>
   <ul>
      <li><a href="#"><i class="fab fa-facebook" aria-hidden="true"></i></a></li>
      <li><a href="#"><i class="fab fa-twitter" aria-hidden="true"></i></a></li>
      <li><a href="#"><i <i class="fab fa-google"  aria-hidden="true"></i></a></li>
      <li><a href="#"><i class="fab fa-linkedin" aria-hidden="true"></i></a></li>
      <li><a href="#"><i class="fab fa-instagram" aria-hidden="true"></i></a></li>
   </ul>

</div>
<!--trems-->

<div class="trems">
   <input class="check" type="checkbox" required/>
 <p class="conditions">I read and agree to <a href="#">Trems &amp;amp; Conditions</a></p>
 </div>
<!--button-->
<div class="toop">
<button type="submit" class="btn btn-primary" >CREATE ACCOUNT</button> </div>

</form>
<!--sign-in-->
<div class="t"><p class="conditions" id="p3">Already have an account <a
href="/signin">Sign in</a></p> </div></div>
 </div>
<!--text-container-->
<div class="text-container">

 <h1 style="color: #2d2c2c;font-family:cursive;">Glad to see you</h1>

 <div class="diag"><img class="fig1" width="100%" height="105%"
src="..\static\images\Inkeddia_LI.jpg"</div>
```

 Welcome,Please Fill in the blanks for sign up

</body>
</html>

Add Expense
In this page user can add their daily expense to the application

```
<div class="container">
   <div class="row">
      <div class="col-md-6">
         <h3>Add Expense</h3>
         <form action="/addexpense" method="POST">

            <div class="form-group">
               <label for="">Date</label>
               <input class="form-control" type="datetime-local" name="date"
id="date"></div>
            <div class="form-group"> <label for="">Expense name</label>
               <input class="form-control" type="text" name="expensename"
id="expensename">
            </div>
            <div class="form-group">
               <label for="">Expense Amount</label>
                <input class="form-control"  type="number" min="0" name="amount"
id="amount">
            </div>

            <div class="form-group">
               <label for=""></label>
               <select class="form-control" name="paymode" id="paymode">
                  <option  selected hidden>Pay-Mode</option>
                  <option name="cash" value="cash">cash</option>
                  <option name="debitcard" value="debitcard">debitcard</option>
                  <option name="creditcard" value="creditcard">creditcard</option>
                  <option name="epayment" value="epayment">epayment</option>
                  <option name="onlinebanking"
```

```
value="onlinebanking">onlinebanking</option>

                </select>




              <div class="form-group">
                <label for=""></label>
                <select class="form-control" name="category" id="category">

                    <option selected hidden>Category</option>
                   <option   name = "food" value="food">food</option>
                   <option name = "entertainment"
value="entertainment">Entertainment</option>
                   <option name = "business" value="business">Business</option>
                   <option name ="rent" value="rent">Rent</option>
                   <option name = "EMI" value="EMI">EMI</option>
                   <option name = "other" value="other">other</option>

                </select>
              </div>



              <input class="btn btn-danger" type="submit" value="Add" id="">

            </form>

            <div  style="position: relative; left: 590px; top: -460px;" class="imagge">
               <img width="90%" height="90%" src="../static/images/addd.jpg" class="rounded
float-left" alt="...">

              </div>

        </div>
    </div>

</div>
```

{% endblock %}

Home Page:
User can find the login option, signup option and Basic information about the application

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">


  <link rel="stylesheet" href="..\static\css\home.css">
  <title>My Website</title>
</head>

<body>
  <!-- Header -->
  <section id="header">
    <div class="header container">
      <div class="nav-bar">
        <div class="brand">
          <a href="#hero">
            <h1><span>M</span>y <span>B</span>udget</h1>
          </a>
        </div>
        <div class="nav-list">
          <div class="hamburger">
            <div class="bar"></div>
          </div>
          <ul>
            <li><a href="#hero" data-after="Home">Home</a></li>
            <li><a href="#services" data-after="Service">Services</a></li>

            <li><a href="#about" data-after="About">About</a></li>
            <li><a href="#contact" data-after="Contact">Contact</a></li>
            <LI><a href="/signin" data-after="Login">-Login-</a></LI>
          </ul>
```

```
      </div>
    </div>
  </div>
 </section>
 <!-- End Header -->


 <!-- Hero Section  -->
 <section id="hero">
  <div class="hero container">
    <div>
      <h1>Hello, <span></span></h1>
      <h1>Welcome To <span></span></h1>
      <h1>Expense Tracker Web application <span></span></h1>
      <a href="/signup" type="button" class="cta">Sign-up</a>
    </div>
  </div>
 </section>
 <!-- End Hero Section  -->

 <!-- Service Section -->
 <section id="services">
  <div class="services container">
    <div class="service-top">
      <h1 class="section-title">Serv<span>i</span>ces</h1>
      <p>MyBudget provides a many services  to the customer and industries. Financial solutions to meet your needs whatever your money goals,there is a MyBudget solution to help you reach them </p>
    </div>
    <div class="service-bottom">
      <div class="service-item">
        <div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
        <h2>Personal Expenses</h2>
        <p>Budgeting is more than paying bills and setting aside savings.it's about creating a money plan for the life you want</p>
      </div>
      <div class="service-item">
        <div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
```

```
        <h2>Investments</h2>
        <p>Follow your investments and bring your portfolio into focus with support for stocks,bonds,CDs,mutual funds and more</p>
      </div>
      <div class="service-item">
        <div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
        <h2>Online Banking</h2>
        <p>MyBudget application can automatically download transactions and send payments online from many financial institutions</p>
      </div>
      <div class="service-item">
        <div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
        <h2>Financial Life</h2>
        <p>Get your Complete financial picture at a glance. With MyBudget application you can view your all the financial activities
        </p>
      </div>
    </div>
  </div>
 </section>
 <!-- End Service Section -->

<!-- About Section -->
 <section id="about">
  <div class="about container">
   <div class="col-left">
    <div class="about-img">
     <img src="..\static\images\img-2.png" alt="img">
     <div><h2>MyBudget ceo,Shridhar </h2></div>
    </div>
   </div>

   <div class="col-right">
    <h1 class="section-title">About <span>Us</span></h1>
    <h2>Financial Solution</h2>
    <p>MyBudget financial solution  is one among Leading financial company from many years.MyBudget provides a many services  to the customer and industries. Financial solutions to meet your needs whatever your money goals,there is a MyBudget solution to help you
```

```
reach them.u can Contact our service center for further information and also follow our social media for update on new services </p>
        <a href="#footer" class="cta">Follow Us</a>

      </div>
    </div>
  </section>
  <!-- End About Section -->

  <!-- Contact Section -->
  <section id="contact">
    <div class="contact container">
      <div>
        <h1 class="section-title">Contact <span>info</span></h1>
      </div>
      <div class="contact-items">
        <div class="contact-item">
          <div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/phone.png" /></div>
          <div class="contact-info">
            <h1>Phone</h1>
            <h2>+1 234 123 1234</h2>
            <h2>+1 234 123 1234</h2>
          </div>
        </div>
        <div class="contact-item">
          <div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/new-post.png" /></div>
          <div class="contact-info">
            <h1>Email</h1>
            <h2>info@gmail.com</h2>
            <h2>abcd@gmail.com</h2>
          </div>
        </div>
        <div class="contact-item">
          <div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/map-marker.png" /></div>
          <div class="contact-info">
            <h1>Address</h1>
            <h2>4th main-road,Bengaluru,Karnataka,India</h2>
```

```
        </div>
       </div>
     </div>
   </div>
  </section>
  <!-- End Contact Section -->

  <!-- Footer -->
  <section id="footer">
   <div class="footer container">
    <div class="brand">
     <h1><span>M</span>y <span>B</span>udget</h1>
    </div>
    <h2>Your Complete Financial Solution</h2>
    <div class="social-icon">
     <div class="social-item">
      <a href="#"><img src="https://img.icons8.com/bubbles/100/000000/facebook-
new.png" /></a>
     </div>
     <div class="social-item">
      <a href="#"><img src="https://img.icons8.com/bubbles/100/000000/instagram-
new.png" /></a>
     </div>
     <div class="social-item">
      <a href="#"><img src="https://img.icons8.com/bubbles/100/000000/twitter.png"
/></a>
     </div>
     <div class="social-item">
      <a href="#"><img src="https://img.icons8.com/bubbles/100/000000/behance.png"
/></a>
     </div>
    </div>
    <p>Copyright © 2021 Shridhar . All rights reserved</p>
   </div>
  </section>
  <!-- End Footer -->
  <script src="..\static\js\home.js"></script>
</body>

</html>
```

Dashboard:
Here the user can find the add expense tab, limit tab and report tab

```
{% extends 'base.html' %}

{% block body %}
<style>

    H1 {
        position: relative;
        right: -790PX;
        top: -400PX;
        color: RED;

    }

p{
position: relative;
right: -800px;
top: -350px;
font-family:monospace;
}

span{
    position: relative;
right: -800px;
top: -360px;
}


   .ccc {
        position: relative;
        top:80px;
        left:-100px;

    }



    </style>
```

```
   <div id=aa class="container">
   <div class="ccc">
      <img src="../static/images/images (13).jpeg">
      <h1>LET START JOURNEY</h1>
      <P>MyBudget web application helps<br> you to maintain budget<br>
          and analyse the expense</P>
      <img height="13%" width="13%" style="position: relative; top: -690PX; left: -60PX;"
src="../static/images/istockphoto-943300706-612x612.jpg">
   </div>
   <span  class="btn btn-outline-dark">Let's Begin</span>
</div>

{% endblock %}
```

Limit Page:

Here the user can set the monthly limit of the expense

```
{% extends 'base.html' %}

{% block body %}

<p> Currently your MONTHLY limit is ₹ {{y}} </p>


<form action="/limitnum" method="POST">

   <p> ENTER the MONTHLY LIMIT to avoid over EXPENSES</p> <br/>
   <input type="number" name="number" required/> <span>
      <button class="btn btn-warning" type="submit">ENTER</button>
   </span>




</form>
{% endblock %}
```

Edit Page:

The edit page is used to change any expense that entered mistakenly

```
{% extends 'base.html' %}

{% block body %}
<div class="container">
   <div class="row">
      <div class="col-md-6">
         <h3>Edit Expense</h3>
         <form action="/update/{{expenses[0]}}" method="POST">


               <input type="hidden" class="form-control"   name="" value =
"{{expenses[0]}}" id="">

            <div class="form-group">
               <label for="">Date</label>
               <input class="form-control"  type="datetime-local" name="date"
value="{{expenses[2]}}"  id="date"></div>




          <script type="text/javascript">
                var d = new Date(value="{{expenses[2]}}" );
                var elem = document.getElementById("date");
                elem.value = d.toISOString().slice(0,16);
               </script>



            <div class="form-group"> <label for="">Expense name</label>
               <input class="form-control" type="text" name="expensename"
value="{{expenses[3]}}" id="expensename">
            </div>
            <div class="form-group">
               <label for="">Expense Amount</label>
                <input class="form-control"  type="number" min="0" name="amount"
value="{{expenses[4]}}" id="amount">
            </div>

            <div class="form-group">
```

```
                <label for=""></label>
                <select class="form-control" name="paymode" value="{{expenses[5]}}"
id="paymode">
                    <option selected hidden>{{expenses[5]}}</option>
                    <option value="cash">cash</option>
                    <option value="debitcard">debitcard</option>
                    <option value="creditcard">creditcard</option>
                    <option value="epayment">epayment</option>
                    <option value="onlinebanking">onlinebanking</option>

                </select>

              <div class="form-group">
                <label for=""></label>
                <select class="form-control" name="category" value="{{expenses[6]}}"
id="category">

                     <option  selected hidden>{{expenses[6]}}</option>
                    <option value="food">food</option>
                    <option value="entertainment">Entertainment</option>
                    <option value="business ">Business</option>
                    <option value="rent">Rent</option>
                    <option value="EMI">EMI</option>
                    <option value="other">other</option>

                </select>
              </div>

              <input class="btn btn-danger" type="submit" value="Update" id="">

          </form>
       </div>
    </div>

</div>

    {% endblock %}
```

Mail Alert:

This feature is used to send mail to user when the expense exceed the limit

Code:

```
import smtplib
import sendgrid
import os
from sendgrid.helpers.mail import Mail, Email, To, Content
SUBJECT = "expense tracker"
s = smtplib.SMTP('smtp.gmail.com', 587)

def sendmail(TEXT,email):
    print("sorry we cant process your candidature")
    s = smtplib.SMTP('smtp.gmail.com', 587)
    s.starttls()
    s.login("shakthidharun@gmail.com", "pass123")
    message  = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)
    s.sendmail("shakthidharun@gmail.com", email, message)
    s.quit()
def sendgridmail(user,TEXT):

    from_email = Email("shakthidharun@gmail.com")
    to_email = To(user)
    subject = "Sending with SendGrid is Fun"
    content = Content("text/plain",TEXT)
    mail = Mail(from_email, to_email, subject, content)

    # Get a JSON-ready representation of the Mail object
    mail_json = mail.get()
    # Send an HTTP POST request to /mail/send
    response = sg.client.mail.send.post(request_body=mail_json)
    print(response.status_code)
    print(response.headers)
```

Flask App:

Python flask framework is used to run the application

Code:

```
# -*- coding: utf-8 -*-
"""
Spyder Editor
```

```
This is a temporary script file.
"""

from flask import Flask, render_template, request, redirect, session
from flask_mysqldb import MySQL
import MySQLdb.cursors
import re


app = Flask(__name__)
app.secret_key = 'a'
app.config['MYSQL_HOST'] = 'remotemysql.com'
app.config['MYSQL_USER'] = 'D2DxDUPBii'
app.config['MYSQL_PASSWORD'] = 'r8XBO4GsMz'
app.config['MYSQL_DB'] = 'D2DxDUPBii'
mysql = MySQL(app)


#HOME--PAGE
@app.route("/home")
def home():
    return render_template("homepage.html")

@app.route("/")
def add():
    return render_template("home.html")


#SIGN--UP--OR--REGISTER

@app.route("/signup")
def signup():
    return render_template("signup.html")

@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
```

```
        password = request.form['password']
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM register WHERE username = % s', (username, ))
        account = cursor.fetchone()
        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            cursor.execute('INSERT INTO register VALUES (NULL, % s, % s, % s)', (username,
email,password))
            mysql.connection.commit()
            msg = 'You have successfully registered !'
            return render_template('signup.html', msg = msg)

 #LOGIN--PAGE

@app.route("/signin")
def signin():
    return render_template("login.html")

@app.route('/login',methods =['GET', 'POST'])
def login():
    global userid
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM register WHERE username = % s AND password =
% s', (username, password ),)
        account = cursor.fetchone()
        print (account)

        if account:
            session['loggedin'] = True
            session['id'] = account[0]
```

```
            userid=  account[0]
            session['username'] = account[1]

            return redirect('/home')
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)


#ADDING----DATA


@app.route("/add")
def adding():
    return render_template('add.html')

@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():

    date = request.form['date']
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category = request.form['category']
    cursor = mysql.connection.cursor()
    cursor.execute('INSERT INTO expenses VALUES (NULL,  % s, % s, % s, % s, % s, % s)',
(session['id'] ,date, expensename, amount, paymode, category))
    mysql.connection.commit()
    print(date + " " + expensename + " " + amount + " " + paymode + " " + category)

    return redirect("/display")

#DISPLAY---graph

@app.route("/display")
def display():
    print(session["username"],session['id'])

    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM expenses WHERE userid = % s AND date ORDER BY
```

```
`expenses`.`date` DESC',(str(session['id'])))
    expense = cursor.fetchall()
    return render_template('display.html' ,expense = expense)

#DELETE--DATA

@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
def delete(id):
     cursor = mysql.connection.cursor()
     cursor.execute('DELETE FROM expenses WHERE  id = {0}'.format(id))
     mysql.connection.commit()
     print('deleted successfully')
     return redirect("/display")


#UPDATE---DATA

@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM expenses WHERE  id = %s', (id,))
    row = cursor.fetchall()

    print(row[0])
    return render_template('edit.html', expenses = row[0])




@app.route('/update/<id>', methods = ['POST'])
def update(id):
  if request.method == 'POST' :

      date = request.form['date']
      expensename = request.form['expensename']
      amount = request.form['amount']
      paymode = request.form['paymode']
      category = request.form['category']

      cursor = mysql.connection.cursor()
```

```
    cursor.execute("UPDATE `expenses` SET `date` = % s , `expensename` = % s , `amount`
= % s, `paymode` = % s, `category` = % s WHERE `expenses`.`id` = % s ",(date,
expensename, amount, str(paymode), str(category),id))
    mysql.connection.commit()
    print('successfully updated')
    return redirect("/display")

 #LIMIT
@app.route("/limit" )
def limit():
     return redirect('/limitn')

@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
   if request.method == "POST":
      number= request.form['number']
      cursor = mysql.connection.cursor()
      cursor.execute('INSERT INTO limits VALUES (NULL, % s, % s) ',(session['id'],
number))
      mysql.connection.commit()
      return redirect('/limitn')


@app.route("/limitn")
def limitn():
  cursor = mysql.connection.cursor()
  cursor.execute('SELECT limitss FROM `limits` ORDER BY `limits`.`id` DESC LIMIT 1')
  x= cursor.fetchone()
  s = x[0]


  return render_template("limit.html" , y= s)

#REPORT

@app.route("/today")
def today():
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT TIME(date)   , amount FROM expenses  WHERE userid = %s
```

```
AND DATE(date) = DATE(NOW()) ',(str(session['id'])))
    texpense = cursor.fetchall()
    print(texpense)

    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM expenses WHERE userid = % s AND DATE(date) =
DATE(NOW()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
    expense = cursor.fetchall()

    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0

    for x in expense:
        total += x[4]
        if x[6] == "food":
            t_food += x[4]

        elif x[6] == "entertainment":
            t_entertainment  += x[4]

        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]

        elif x[6] == "EMI":
            t_EMI  += x[4]

        elif x[6] == "other":
            t_other  += x[4]

    print(total)

    print(t_food)
    print(t_entertainment)
```

```
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)
    return render_template("today.html", texpense = texpense, expense = expense,  total =
total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )


@app.route("/month")
def month():
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT DATE(date), SUM(amount) FROM expenses WHERE userid=
%s AND MONTH(DATE(date))= MONTH(now()) GROUP BY DATE(date) ORDER BY
DATE(date) ',(str(session['id'])))
    texpense = cursor.fetchall()
    print(texpense)

    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM expenses WHERE userid = % s AND
MONTH(DATE(date))= MONTH(now()) AND date ORDER BY `expenses`.`date`
DESC',(str(session['id'])))
    expense = cursor.fetchall()

    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0

    for x in expense:
        total += x[4]
        if x[6] == "food":
            t_food += x[4]

        elif x[6] == "entertainment":
```

```
            t_entertainment  += x[4]

        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]

        elif x[6] == "EMI":
            t_EMI  += x[4]

        elif x[6] == "other":
            t_other  += x[4]

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)
    return render_template("today.html", texpense = texpense, expense = expense,  total =
total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )

@app.route("/year")
def year():
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT MONTH(date), SUM(amount) FROM expenses WHERE
userid= %s AND YEAR(DATE(date))= YEAR(now()) GROUP BY MONTH(date) ORDER
BY MONTH(date) ',(str(session['id'])))
    texpense = cursor.fetchall()
    print(texpense)

    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM expenses WHERE userid = % s AND
YEAR(DATE(date))= YEAR(now()) AND date ORDER BY `expenses`.`date`
DESC',(str(session['id'])))
```

```
expense = cursor.fetchall()

total=0
t_food=0
t_entertainment=0
t_business=0
t_rent=0
t_EMI=0
t_other=0


for x in expense:
    total += x[4]
    if x[6] == "food":
        t_food += x[4]

    elif x[6] == "entertainment":
        t_entertainment  += x[4]

    elif x[6] == "business":
        t_business  += x[4]
    elif x[6] == "rent":
        t_rent  += x[4]

    elif x[6] == "EMI":
        t_EMI  += x[4]

    elif x[6] == "other":
        t_other  += x[4]

print(total)

print(t_food)
print(t_entertainment)
print(t_business)
print(t_rent)
print(t_EMI)
print(t_other)

return render_template("today.html", texpense = texpense, expense = expense,  total =
```

```
total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )

#log-out

@app.route('/logout')

def logout():
   session.pop('loggedin', None)
   session.pop('id', None)
   session.pop('username', None)
   return render_template('home.html')

if __name__ == "__main__":
      app.run(debug=True)
```

CHAPTER-8

TESTING

8.1 Test Cases

Test case ID	Feature Type	Component	Test Scenario
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with InValid credentials
AddExpensePage_TC	Functional	ld Expense Pa	whether user is able to add expense

Pre-Requisite	Steps To Execute	Test Data
None	1. Go to website 2. Home page appears	Username: shakthidharun@gmail.com password: password123
Home	1.Go to website 2.Enter details and click login	Username: shakthidharun@gmail.com password: password123
Username & password	1.Go to website 2.Enter details and click login	Username: shakthidharun@gmail.com password: password123
Username & password	1.Go to website 2.Enter details and click login	Username: shakthidharun@gmail.com password: password123
Login first	1.Go to website 2.Enter details and click login	Username: shakthidharun@gmail.com password: password123
Login first	1.Go to website 2.Enter details and click login	Username: shakthidharun@gmail.com password: password123
Have some expense to add	1. Add date, expense name and other details 2.Check if the expense gets added	add electricity bill = 6000

Expected Result	Actual Result	Status
Login/Signup popup should display	Working as expected	Pass
Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.New customer? Create account link e.Last password? Recovery password link	Working as expected	Pass
User should navigate to user account homepage	Working as expected	Pass
Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass
Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass
Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass
Application adds expenses	Working as expected	Pass

8.2 User Acceptance Testing

Defect Analysis

This reportshows the numberof resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	9	3	1	2	15
Duplicate	2	1	2	1	6
External	1	2	0	3	6
Fixed	10	2	4	11	27
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	22	13	11	19	65

Test Case Analysis

This report shows the numberof test cases that have passed, failed,and untested

Section	Total Cases	Not Tested	Fail	Pass
Interface	13	0	3	10
Login	20	0	3	17
Logout	5	0	0	5
Add Expense	15	0	2	13
Limit	5	0	0	5
Report	5	1	2	3
Sign up	3	0	1	2

CHAPTER-9

RESULTS

a. **Performance Metrics**

Leadtime - 4 months
Cycle time - 10 Min
Team velocity

ADVANTAGES & DISADVANTAGES

Advantages

When if comes to personal finance, being out of control is not something anybody would strive for. There's nothing financially worse than feeling like you don't have any idea what's going on with your money.

The good news is, when you make an effort to record every financial transaction you make, you are essentially, taking the reins on anything and everything involving your money. At any one time, you will know exactly how much money is sitting in your bank account, and how much you can spend.

In other words, when you track your expenses, you take complete control over your finances.

If you have any plans on saving, investing, getting out of debt, or building wealth, what is holding you accountable. I mean, we can all set financial goals, and have financial dreams, but if you aren't tracking your expenses, there is nothing to hold you accountable when you make a bad financial decision.

Tracking your expenses holds you accountable to your future financial goals. And in the long run, that can be the difference between broke and wealthy.

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**, these companies are large enough and secure enough that you'll never have to worry about something like that. Just recognize that your information, most likely anonymous, will be used and potentially even sold. Personally, I have no problem with that, but if you do, then make sure you avoid these types of services

CHAPTER-10
CONCLUSION

Recording your expenses daily can ensure that you are financially aware all year long and not just during tax season. Knowing where your money is going and how much you're spending can improve your spending habits. So, using the daily expense tracker application is important to load a happy family. Daily expense tracker helps the user to avoid unexpected expenses and bad financial situations. This Project will save time and provide a responsible lifestyle.

FUTURE SCOPE

Year-on-year, modern expense management software undergone a continuous evolution from traditional back-office function to strategic internal set of processes. But would it be sufficient to meet the needs of next-gen companies? Have you ever thought how the next-generation software should look like? As the requirements of companies evolve continuously, the software should undergo a series of changes to meet the growing needs of next generation companies.

The next-generation travel and expense (T & E) management apps should not only just accelerate the expense management process but also should come with mobile and cloud integration capabilities that add tremendous value to the business bottom line. Future T & E management software should be able to provide greater visibility into spending and standardize critical procedures.

APPENDIX

Source Code

```
from flask import Flask, render_template, request, redirect, session
from flask_mysqldb import MySQL
import MySQLdb.cursors
import re

app = Flask(__name__)

app.secret_key = 'a'

app.config['MYSQL_HOST'] = 'remotemysql.com'
app.config['MYSQL_USER'] = 'D2DxDUPBii'
app.config['MYSQL_PASSWORD'] = 'r8XBO4GsMz'
app.config['MYSQL_DB'] = 'D2DxDUPBii'

mysql = MySQL(app)

#HOME--PAGE
@app.route("/home")
def home():
    return render_template("homepage.html")


@app.route("/")
def add():
    return render_template("home.html")

#SIGN--UP--OR--REGISTER

@app.route("/signup")
def signup():
    return render_template("signup.html")

@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
```

```
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM register WHERE username = % s', (username, ))
        account = cursor.fetchone()
        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            cursor.execute('INSERT INTO register VALUES (NULL, % s, % s, % s)', (username,
email,password))
            mysql.connection.commit()
            msg = 'You have successfully registered !'
            return render_template('signup.html', msg = msg)


 #LOGIN--PAGE

@app.route("/signin")
def signin():
    return render_template("login.html")

@app.route('/login',methods =['GET', 'POST'])
def login():
    global userid
    msg = ''


    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM register WHERE username = % s AND password = % s',
(username, password ),)
```

```
        account = cursor.fetchone()
        print (account)

        if account:
            session['loggedin'] = True
            session['id'] = account[0]
            userid=  account[0]
            session['username'] = account[1]

            return redirect('/home')
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)

#ADDING----DATA

@app.route("/add")
def adding():
    return render_template('add.html')

@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():

    date = request.form['date']
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category = request.form['category']

    cursor = mysql.connection.cursor()
    cursor.execute('INSERT INTO expenses VALUES (NULL,  % s, % s, % s, % s, % s, % s)',
(session['id'] ,date, expensename, amount, paymode, category))
    mysql.connection.commit()
    print(date + " " + expensename + " " + amount + " " + paymode + " " + category)

    return redirect("/display")
```

```
#DISPLAY---graph

@app.route("/display")
def display():
    print(session["username"],session['id'])
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM expenses WHERE userid = % s AND date ORDER BY
`expenses`.`date` DESC',(str(session['id'])))
    expense = cursor.fetchall()

    return render_template('display.html' ,expense = expense)


#DELETE--data

@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
def delete(id):
     cursor = mysql.connection.cursor()
     cursor.execute('DELETE FROM expenses WHERE  id = {0}'.format(id))
     mysql.connection.commit()
     print('deleted successfully')
     return redirect("/display")


#UPDATE---DATA

@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM expenses WHERE  id = %s', (id,))
    row = cursor.fetchall()

    print(row[0])
    return render_template('edit.html', expenses = row[0])


@app.route('/update/<id>', methods = ['POST'])
def update(id):
  if request.method == 'POST' :
```

```
    date = request.form['date']
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category = request.form['category']

    cursor = mysql.connection.cursor()

    cursor.execute("UPDATE `expenses` SET `date` = % s , `expensename` = % s , `amount` =
% s, `paymode` = % s, `category` = % s WHERE `expenses`.`id` = % s ",(date, expensename,
amount, str(paymode), str(category),id))
    mysql.connection.commit()
    print('successfully updated')
    return redirect("/display")


 #LIMIT
@app.route("/limit" )
def limit():
     return redirect('/limitn')

@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
   if request.method == "POST":
      number= request.form['number']
      cursor = mysql.connection.cursor()
      cursor.execute('INSERT INTO limits VALUES (NULL, % s, % s) ',(session['id'], number))
      mysql.connection.commit()
      return redirect('/limitn')

@app.route("/limitn")
def limitn():
   cursor = mysql.connection.cursor()
   cursor.execute('SELECT limitss FROM `limits` ORDER BY `limits`.`id` DESC LIMIT 1')
   x= cursor.fetchone()
   s = x[0]

   return render_template("limit.html" , y= s)
```

```
#REPORT

@app.route("/today")
def today():
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT TIME(date)   , amount FROM expenses  WHERE userid = %s AND
DATE(date) = DATE(NOW()) ',(str(session['id'])))
    texpense = cursor.fetchall()
    print(texpense)

    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM expenses WHERE userid = % s AND DATE(date) =
DATE(NOW()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
    expense = cursor.fetchall()

    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0


    for x in expense:
        total += x[4]
        if x[6] == "food":
            t_food += x[4]

        elif x[6] == "entertainment":
            t_entertainment  += x[4]

        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]

        elif x[6] == "EMI":
            t_EMI  += x[4]
```

```
        elif x[6] == "other":
            t_other  += x[4]

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)

    return render_template("today.html", texpense = texpense, expense = expense,  total = total ,
                        t_food = t_food,t_entertainment =  t_entertainment,
                        t_business = t_business,  t_rent =  t_rent,
                        t_EMI =  t_EMI,  t_other =  t_other )



@app.route("/month")
def month():
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT DATE(date), SUM(amount) FROM expenses WHERE userid= %s AND MONTH(DATE(date))= MONTH(now()) GROUP BY DATE(date) ORDER BY DATE(date) ',(str(session['id'])))
    texpense = cursor.fetchall()
    print(texpense)

    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM expenses WHERE userid = % s AND MONTH(DATE(date))= MONTH(now()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
    expense = cursor.fetchall()

    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
```

```
    t_EMI=0
    t_other=0


    for x in expense:
        total += x[4]
        if x[6] == "food":
            t_food += x[4]

        elif x[6] == "entertainment":
            t_entertainment  += x[4]

        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]

        elif x[6] == "EMI":
            t_EMI  += x[4]

        elif x[6] == "other":
            t_other  += x[4]

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)

    return render_template("today.html", texpense = texpense, expense = expense,  total =
total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )

@app.route("/year")
def year():
```

```
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT MONTH(date), SUM(amount) FROM expenses WHERE userid= %s
AND YEAR(DATE(date))= YEAR(now()) GROUP BY MONTH(date) ORDER BY MONTH(date)
',(str(session['id'])))
    texpense = cursor.fetchall()
    print(texpense)

    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM expenses WHERE userid = % s AND YEAR(DATE(date))=
YEAR(now()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
    expense = cursor.fetchall()

    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0


    for x in expense:
        total += x[4]
        if x[6] == "food":
            t_food += x[4]

        elif x[6] == "entertainment":
            t_entertainment  += x[4]

        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]

        elif x[6] == "EMI":
            t_EMI  += x[4]

        elif x[6] == "other":
            t_other  += x[4]
```

```
    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)
    return render_template("today.html", texpense = texpense, expense = expense,  total =
total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )



#LOGOUT

@app.route('/logout')

def logout():
  session.pop('loggedin', None)
  session.pop('id', None)
  session.pop('username', None)
  return render_template('home.html')

if __name__ == "__main__":
   app.run(debug=True)
```

GitHub Link

https://github.com/IBM-EPBL/IBM-Project-33777-1660226663