NALAIYA THIRAN -IBM PROJECT

Bacth No: B2-2M4E

KONGU ENGINEERING COLLEGE

(An Autonomous Institution)

Department of Computer Science and Engineering

Team ID	PNT2022TMID04480
Project Name	Project - Personal Expense Tracker

Project Guide:

Industry mentor: Kusboo

Faculty Mentor: Malliga S

	NAME	REGISTER NO
TEAM LEADER	SOWBHAKIAN E S	19CSR189
TEAM MEMBER 1	CHANDEEP G	19CSR020
TEAM MEMBER 2	SARAN KUMAR S	19CSR178
TEAM MEMBER 3	VISHNU T	19CSR226

INDEX

1.INTRODUCTION

1. Project Overview

Personal Expense Tracker is a web application that allows you to track the daily expense of the user and help them to keep track of their expenses daily, monthly, weekly and yearly basis. It will also create a digital records for the user's income and various expenses spent by the user is calculated. It also gets the input from the user as how much income earned and the date of the income earned and further creates a transaction entry and sums up all the total income. Further we can give voice commands and it is responsive. It will be very helpful for the users to manage their needs and they can spend in a better way by keeping track of the application daily basis.

2. Purpose

The main purpose of personal expense tracker application is used to keep track of expenses based on the user income and how much they spent and they can keep track of their expenses daily, monthly, weekly and yearly basis.

2.LITERATURE SURVEY

1.Existing problem

The problem of current generation population is that they can't remember where all of the money they earned have gone and ultimately have to live while sustaining the little money they have left for their essential needs. In this time there is no such perfect solution which helps a person to track their daily expenditure easily and efficiently and notify them about the money shortage they have. For doing so have to maintain long ledgers or computer logs to maintain such data and the calculation is done manually by the user, which may generate error leading to losses. Not having a complete tracking

2.References

Anant Prakash Singh

Expense Tracker is a day-to-day expense management system designed to easily and efficiently track the daily expenses of unpaid and unpaid staff through a computerized system that eliminates the need for manual paper tasks that systematically maintain records and easily accesses data stored by the user. We have tried to design the window applicationin such a way that the user does not have to bother using this application without much effort. End users with Windows running devices can use this software. The language databases we use to develop this system. This application is a GUI (Graphics User Interface) based application. If you are a window user, you can download the application and work accordingly. This system is used by any person to control his income expenditure from daily to annual basics. And to keep an eye on their spending. This app is very easy to use and multi-language. The main feature of this app is that you can track by day and category. You can use it according to your category.

Uday Pratap Singh

Tracking daily expenses is not so contemporary. Many traditional and technical approaches have been invented to track expenses and budgets with their performance. In the past and today, we write our amount in the register to calculate harvest or savings. Not only this, many desktop and mobile applications have been developed for this purpose. Quicken and Microsoft Money were the first desktop applications developed many years ago, but can't getfame and name in the digital world. Personal capital and the Dollar Bird app are used to display spending on maps or graphs with a calendar in-build system. Quick Book was an app for the limited business holder to wrap up their consolidated business.

Praphulla S. Kherade

We are building an android application named "Expense Tracker". As the name suggests, this project is an android app that is used to track the daily expenses of the user. It is like digital record keeping which keeps the records of expenses done by a user. The application keeps the track of the Income and Expenses of the user on a day-to-day basis. This application takes the income of a user and manages its daily expenses so that the user can save money. If you exceed the daily expense allowed amount it will give you a warning, so that you don't spend much and that specific day. If you spend less money than the daily expense allowed amount, the money left after spending is added to the user's savings. The application generates a report of the expenses at each end of the month. The amount saved can be used for celebrating festivals, Birthdays, or Anniversaries.

Prof Miriam Thomas

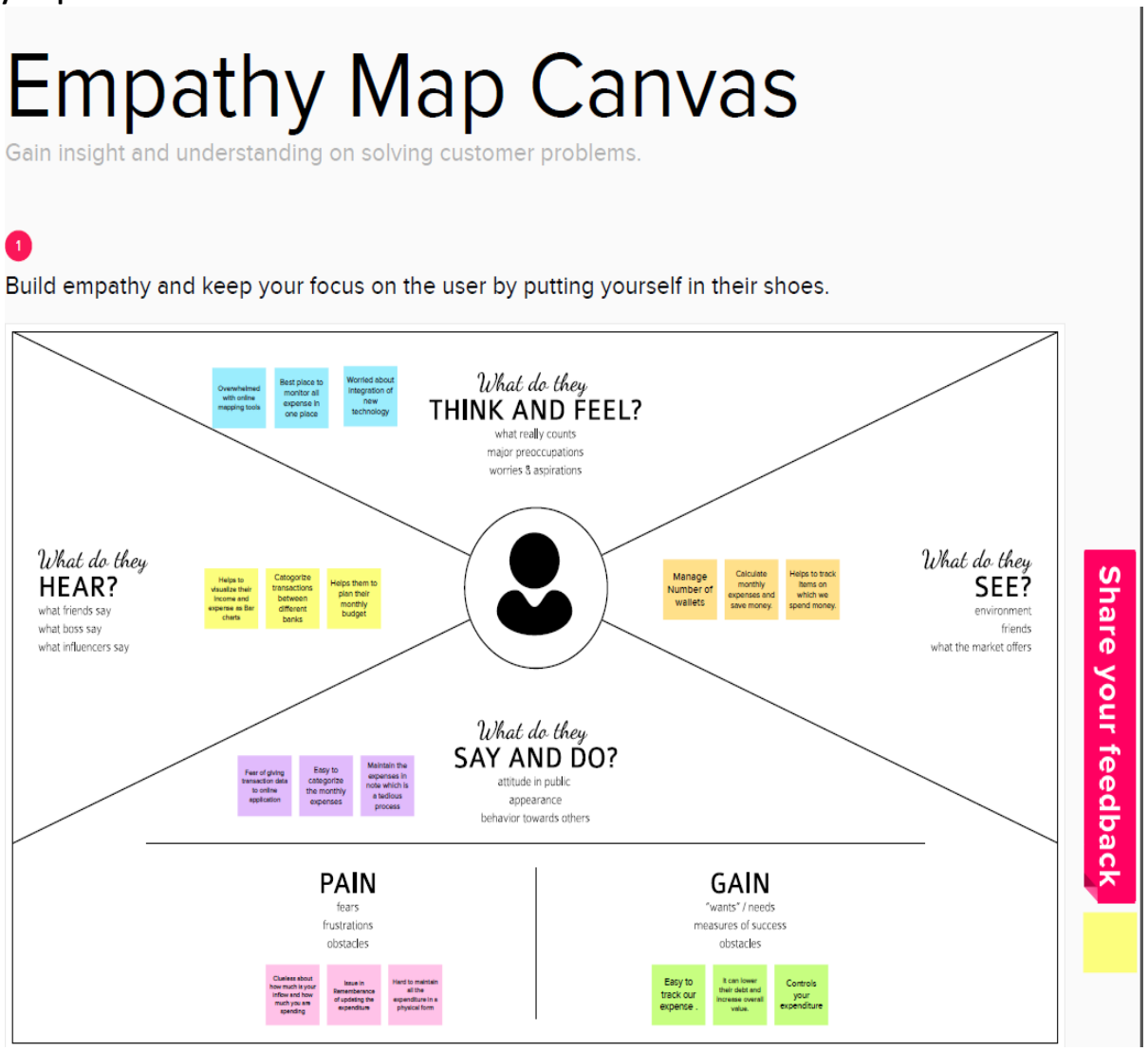
To reduce manual calculations, we propose an application. This application allows users tomaintain a digital automated diary. Each user will be required to register on the system at registration time, the user will be provided id, which will be used to maintain the record of each unique user. Expense Tracker application which will keep a track of Income-Expenseof a user on a day-to-day basis.

3.Problem Statement Definition:

This Expense Tracker is a web application that facilitates the users to keep track and manage their personal as well as business expenses. This application helps the users to keep a digital diary. It will keep track of a user's income and expenses on a daily basis. The user will be able to add his/her expenditures instantly and can review them anywhere and anytime with the help of the internet. He/she can easily import transactions from his/her mobile wallets without risking his/her information and efficiently protecting his/her privacy. This expense tracker provides a complete digital solution to this problem.
Excel sheets do very little to help in tracking Furthermore, they don't have the advanced functionality of preparing graphical visuals automatically. Not only it will save the time of the people but also it will assure error free calculations. The user just has to enter the income and expenditures and everything else will be performed by the system. Keywords: Expense Tracker, budget, planning, savings, graphical visualization of expenditure.

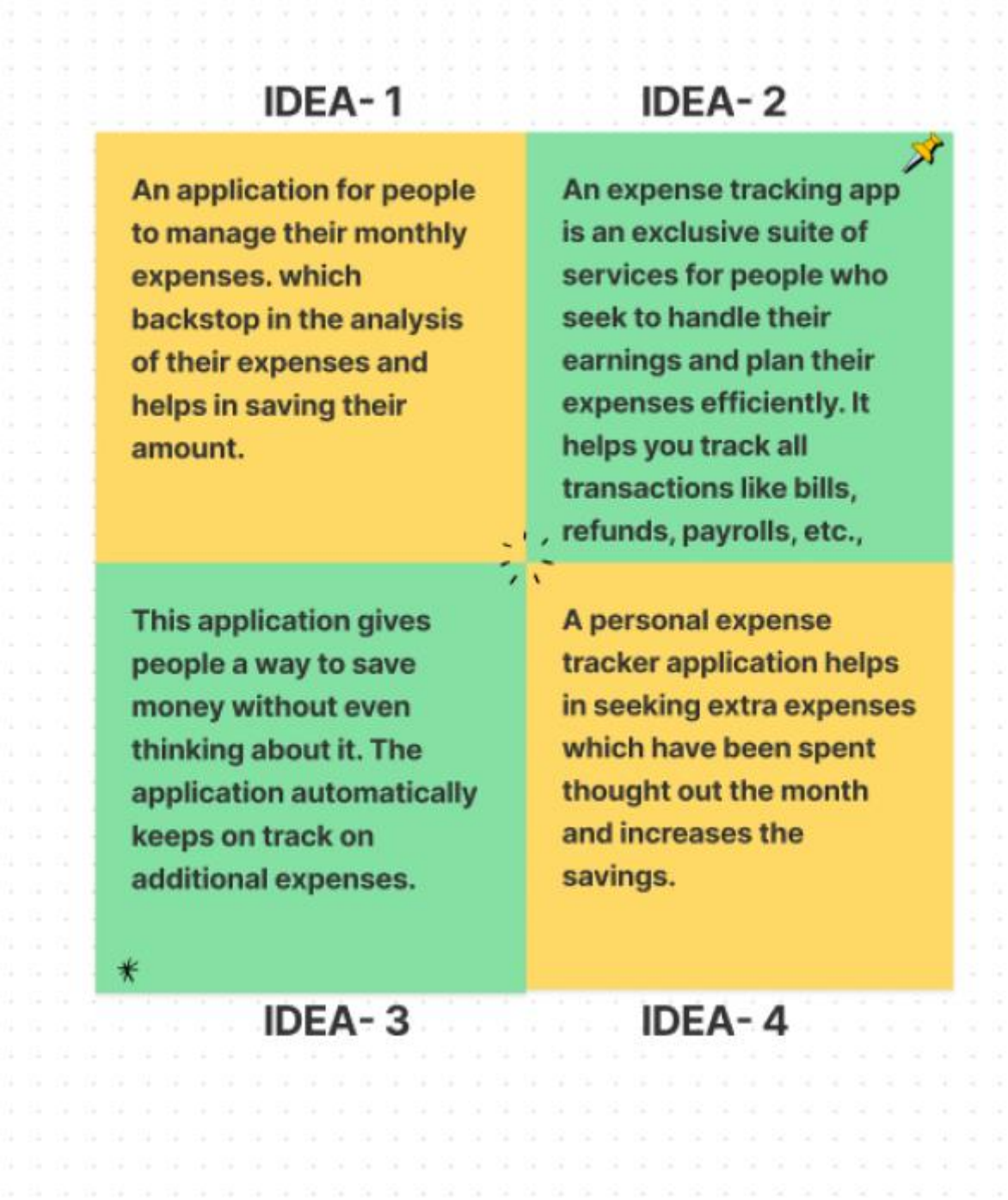
3.IDEATION & PROPOSED SOLUTION

1.Empathy Map Canvas



2.Ideation & Brainstorming:

IDEATION



2

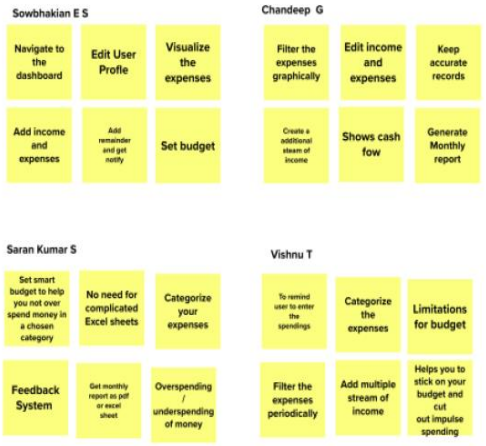
Brainstorm

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!

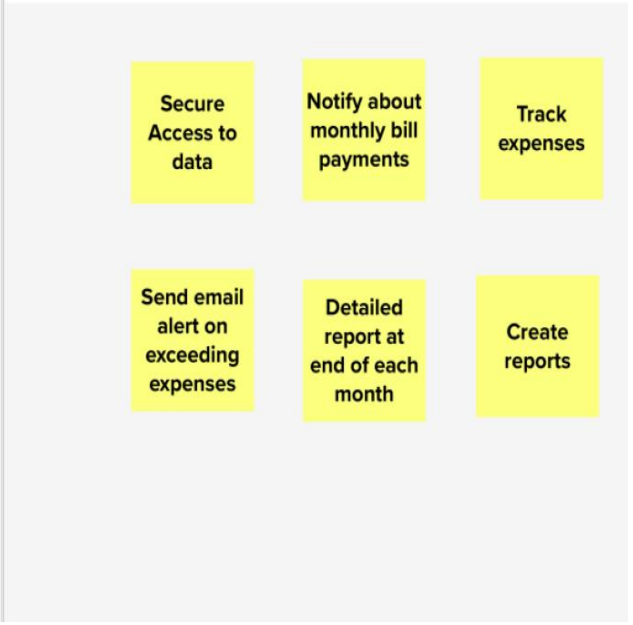


3

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.

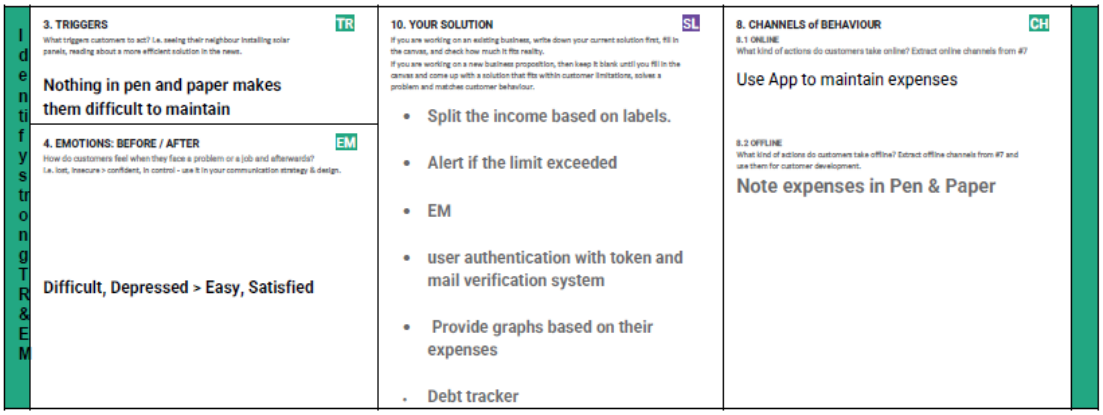
20 minutes

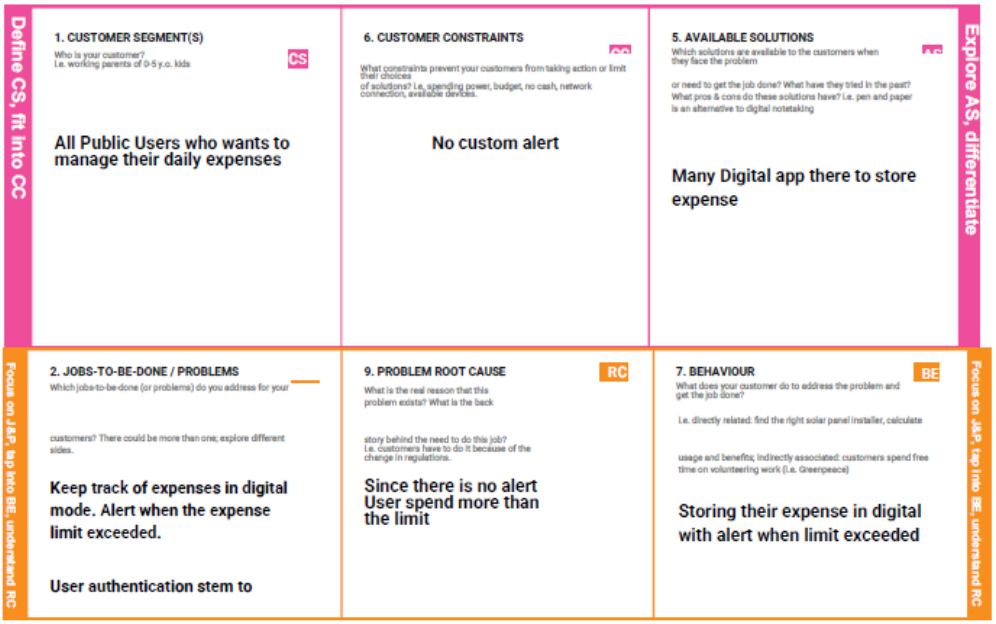


3. Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The issue is that consumers struggle to keep track of their monthly expenses and refrain from compulsive spending.
2.	Idea / Solution description	The user of the personal cost tracker programmer can input their earnings and outgoing costs. The expense wallet is consequently updated. One could get a graphic representation of the cost breakdown. Additionally, if the budget's monthly maximum is exceeded, the user is informed.
3.	Novelty / Uniqueness	The user of the personal spending tracker application is assisted not only with accounting and budgeting, but also with money management insights through analysis. If the monthly limit is violated, the user is also informed.
4.	Social Impact / Customer Satisfaction	The personal expenditure tracker tool not only helps the user with accounting and budgeting, but it also offers analysis-based insights into money management. The useris also informed if the monthly cap is reached.
5.	Business Model (Revenue Model)	There may be a free and paid version of the application, and the user can choose to upgrade to the paid version in order to access more features. The premium edition might also be ad-free.
6.	Scalability of the Solution	This application can be made available to commercial organizations in addition to just individuals.

4.Problem Solution fit





4.REQUIREMENT ANALYSIS:

1.Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement	Sub Requirement
FR-1	User Registration	Registration via Application Registration via E-mail
FR-2	User Confirmation	Confirmation via Email
FR-3	User monthly expense tentative data	Data to be registered in the app
FR-4	User monthly income data	Data to be registered in the app
FR-5	Alert / Notification	Alert via E-mail
FR-6	User Budget Plan	Planning and Tracking of user expenses vs. budget limit

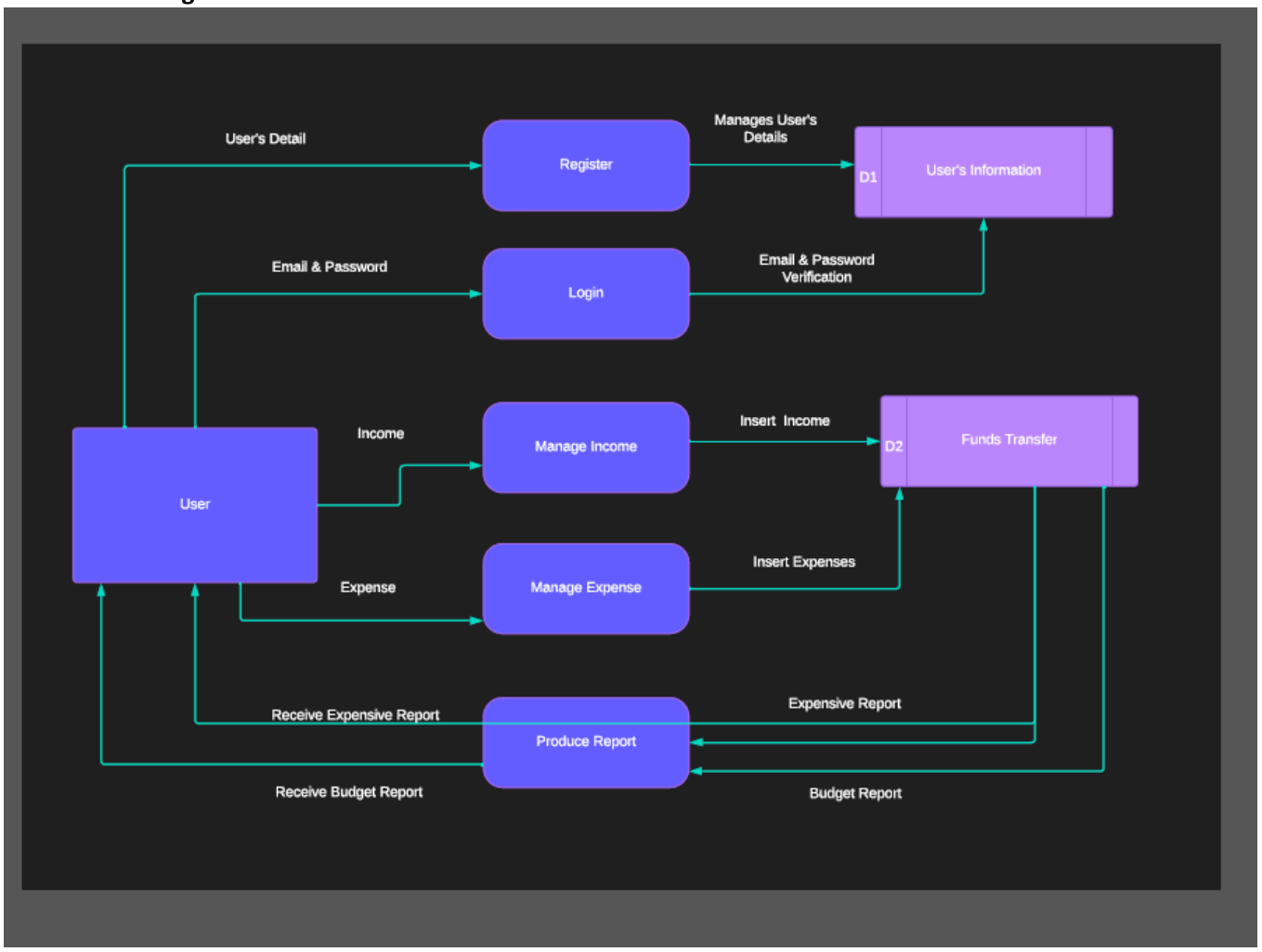
2.Non-functional Requirements:

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

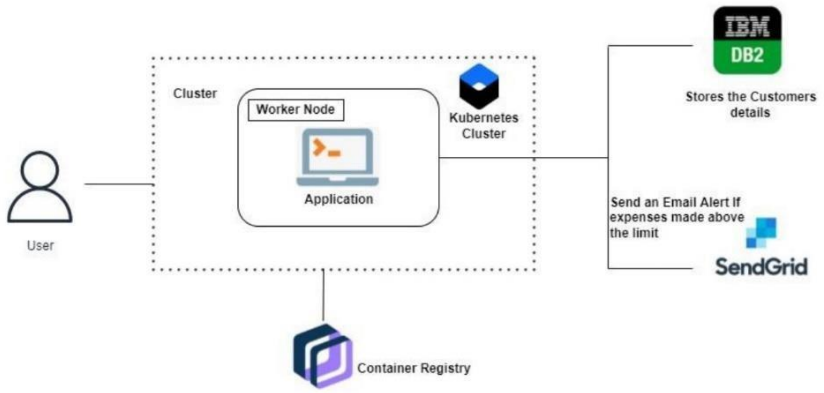
NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	Effectiveness, efficiency, and overall satisfaction of the user while interacting with the application.
NFR-2	Security	Authentication and encryption of the application.
NFR-3	Reliability	Probability of failure-free operations in a specified environment for a specified time.
NFR-4	Performance	How the application is functioning and how responsive the application is to the end-users.
NFR-5	Availability	In spite of the lack of an active internet connectionall features of the application are accessible. Synchronization of data cannot be done.
NFR-6	Scalability	The capacity of the application to handle growth,especially in handling more users.

5.PROJECT DESIGN:

1.Data Flow Diagrams



2.Solution & Technical Architecture



3.User Stories

User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Gmail	I can access the application	Medium	Sprint-1
	Login	USN-4	As a user, I can log into the application by entering email & password	I can enter into the application by using the registered email and password	High	Sprint-1
	Dashboard	USN-5	As a user, I can administrator I can enter into the dashboardto add my income and expenditures	I can view my daily, monthly and yearly expenses	High	Sprint-1
Customer Care Executive	Alerts and Messages	USN-6	As a Customer Care Executive, I can send Alerts and messages to the user	I can send alerts when the user exceeds the expense limit	High	Sprint-1
Administrator	Application	USN-7	As an upgrade orupdate the application.	The bug can be fixed for the application's customers and users	High	Sprint-1

6.PROJECT PLANNING & SCHEDULING:

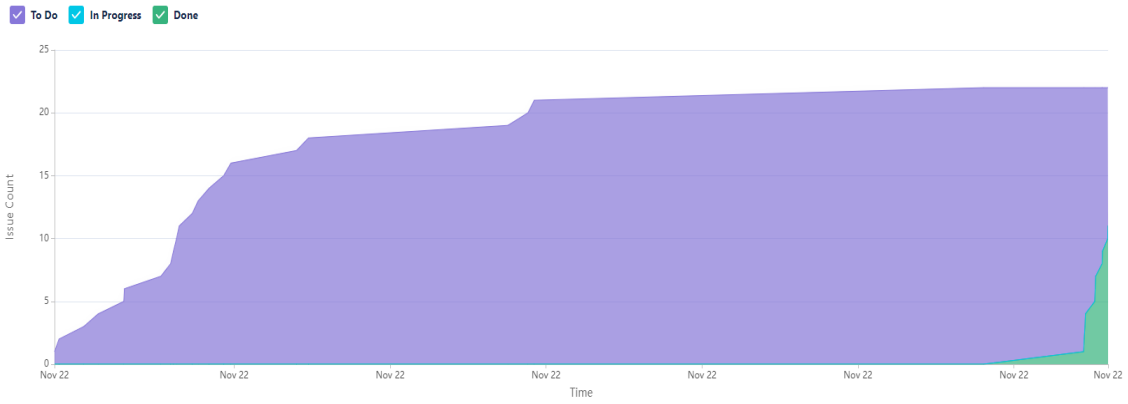
1.Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration (MobileUser)	USN-1	In order to register for the application, I must first enter my email address and then confirmmy password.	4	High	Sowbhakian ES
Sprint-1	Registration	USN-2	The application will send me a confirmationemail once User have registered as a user	3	High	Saran Kumar S
Sprint-1	Registration	USN-3	Through my Gmail account, I can register for theapplication	2	Medium	Chandeep G
Sprint-1	Login	USN-4	The application allows me to log in by enteringmy email address and password	3	High	Chandeep G Vishnu T
Sprint-2	Registra tion (Web User)	USN-5	It is possible for me to register for the application by entering my email address,password, and confirming my password	4	High	Vishnu T
Sprint-2	Registration	USN-6	Once I have registered for the application, Userwill receive a confirmation email	3	High	Saran Kumar S
Sprint-2	Registration	USN-7	User can able to register for the applicationusing my Gmail account	2	Medium	Chandeep G
Sprint-2	Login	USN-8	The application allows User to log in byentering my email address and password	3	High	Sowbhakian ES
Sprint-3	Expense Update	USN-9	As a user, I can add my wallet balance andadd or delete my expenses.	6	High	Vishnu T
Sprint-3	Expense Update	USN-10	As a user, I have the ability to update myexpenses on a regular basis.	6	Medium	Saran Kumar S
Sprint-4	Email alert	USN-11	As a user, I can set an expense limit out ofthe Wallet balance.	6	High	Sowbhakian ES
Sprint-4	Email alert	USN-12	As a user, I will be notified by e-mail if Ihave reached the set limit.	6	High	Chandeep G

2. Sprint Delivery Schedule

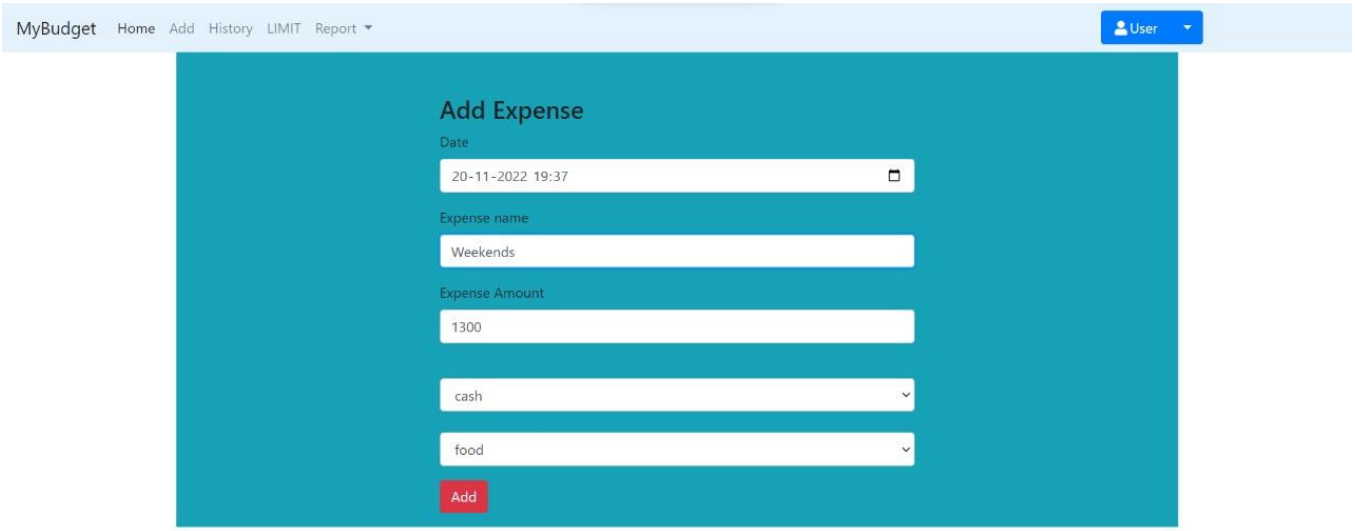
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

3.Reports from JIRA:

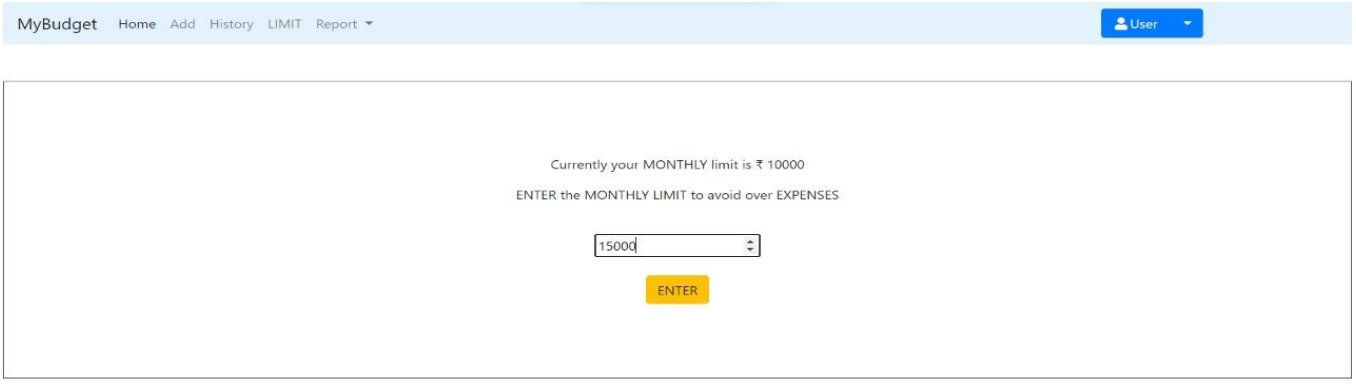


7.CODING & SOLUTIONING:

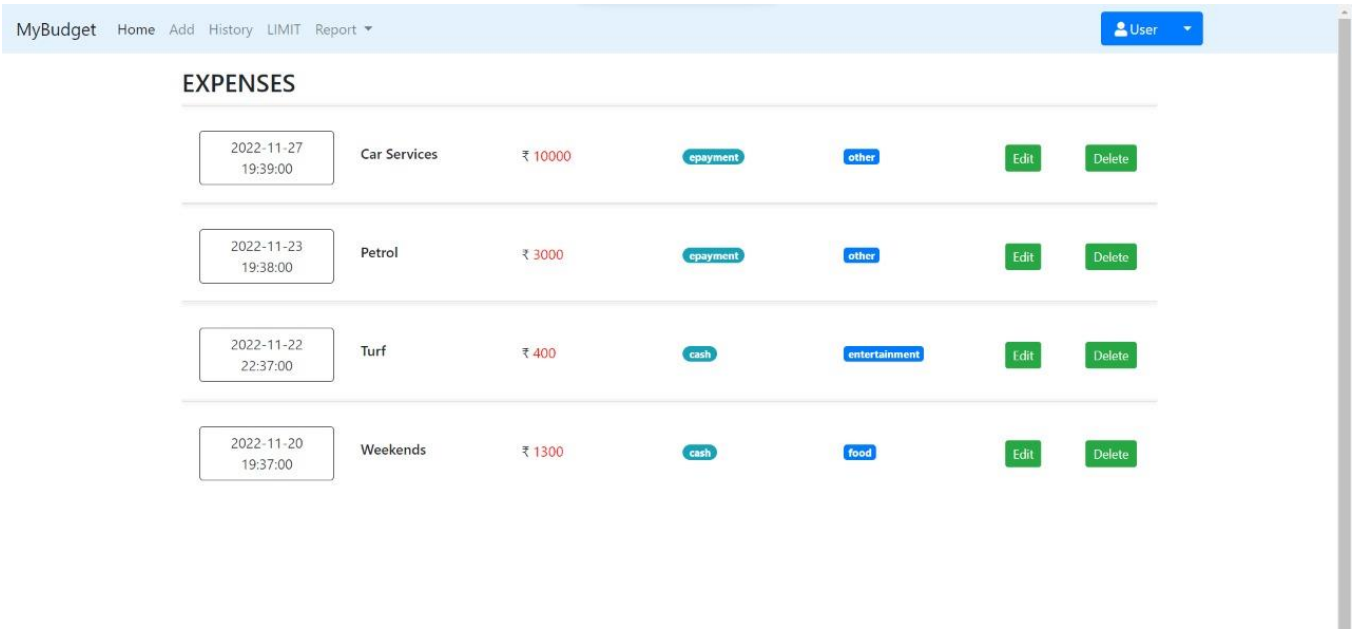
1. Add Expense



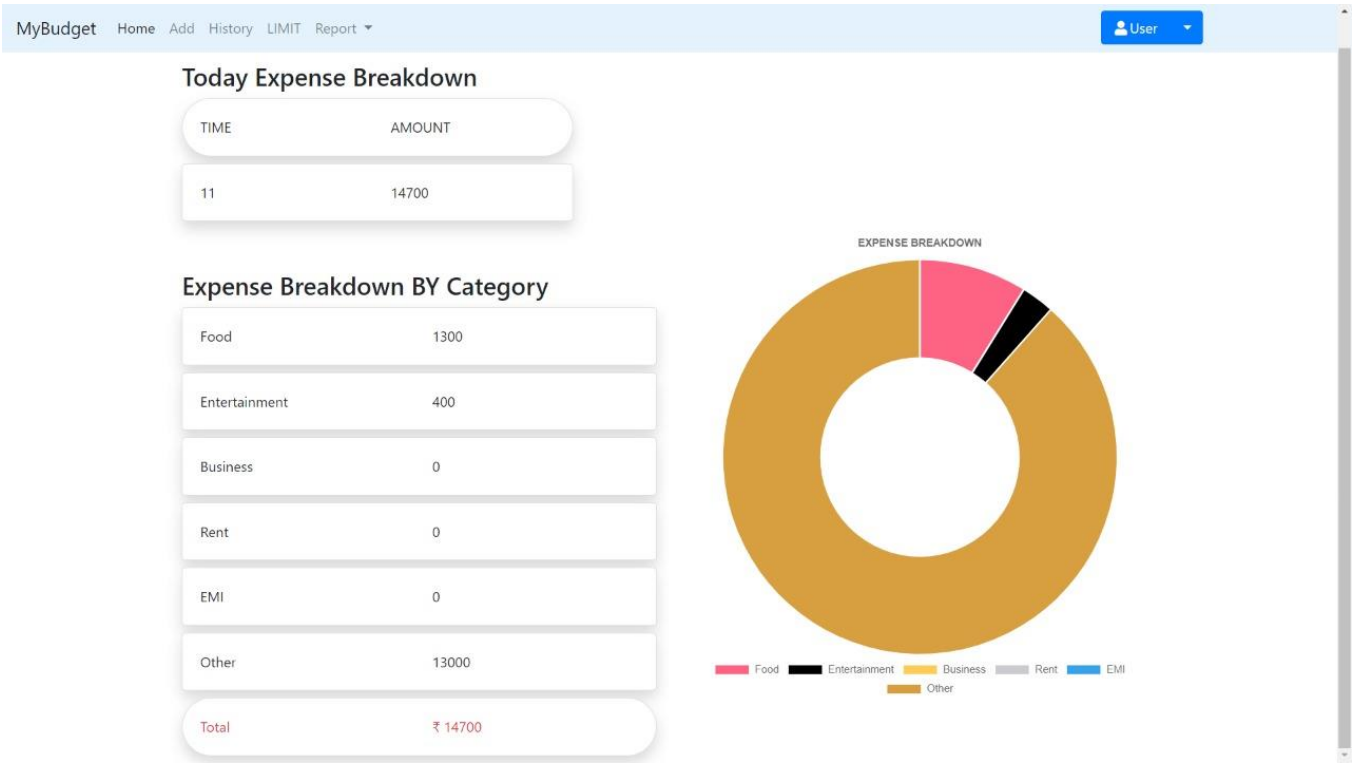
2. Add Income



3. Change Budget



4.Expense Breakdown



```
<!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>E</span>xpence <span>T</span>racker</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="#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>Personal Expense Tracker 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>Founders, CSE-C Last Benchers </h2>
</div>
</div>
</div>

<div class="col-right">
<h1 class="section-title">About <span>Us</span></h1>
<h2>Financial Solution</h2>
<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>+91 888888888</h2>
<h2>+91 999999999</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>demo@gmail.com</h2>
<h2></h2>
</div>
</div>
</div>
</div>
</section>
<!-- End Contact Section -->

<!-- Footer -->
<section id="footer">
<div class="footer container">
```

```
<div class="brand">
<h1><span>E</span>xpence <span>T</span>racker</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/behance.png" /></a>
</div>
</div>
<p> Expense Tracker </p>
</div>
</section>
<!-- End Footer -->

<script src="..\static\js\home.js"></script>
</body>

</html>



{% extends 'base.html' %} {% block body %}
<div class="pt-5 mb-5">
<div class="border border-dark">
<form action="/limitnum" method="POST" class="text-center">
<div class="pt-5 mt-5">
<p> Currently your MONTHLY limit is ₹ {{y}} </p>
<p> ENTER the MONTHLY LIMIT to avoid over EXPENSES</p> <br/>
<input type="number" name="number" required/> <span> <br><br>
<button class="btn btn-warning" type="submit">ENTER</button>
<br><br><br><br><br>
</span>
</div>
</form>
</div>
</div>
{% endblock %}


<!DOCTYPE html>
<html>
<head>
```

```
<title>Animated 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">{{ 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">
<div>
<ul>
</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>
```

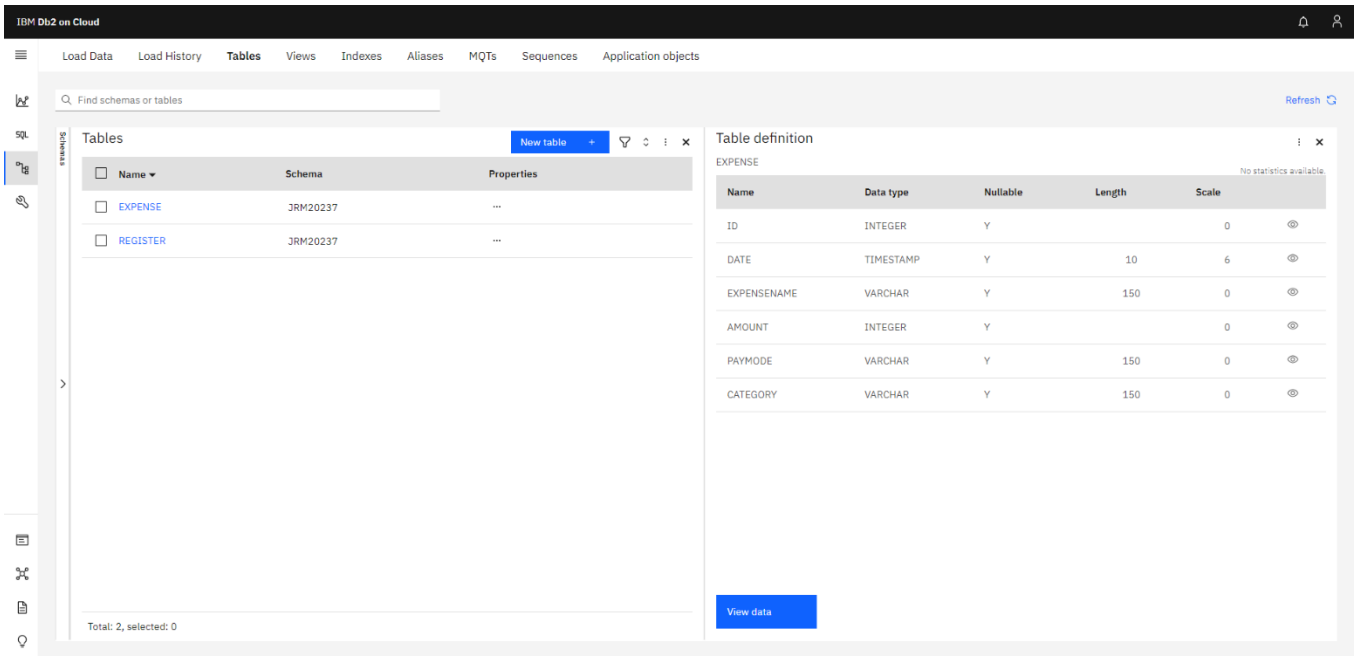
```
</body>
</html>



{% extends 'base.html' %} {% block body %}
<div class="container bg-info pt-5">
<div class="row">
<div class="col-3"></div>
<div class="col-md-6">
<h3> Expense Tracker</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> -->
</div>
```

```
</div>
<div class="col-3"></div>
</div>
{% endblock %}
```

5.DataBase Schema:



8.TESTING

1.Test Cases

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Sub Total
By Design	5	2	1	2	10
Duplicate	1	2	0	0	3
External	3	1	0	0	4
Fixed	5	0	1	0	6
Not Reproduced	0	3	0	1	4
Skipped	1	2	1	1	5
Won't Fix	0	1	0	2	3
Totals	15	11	3	6	35

2.User Acceptance Testing

Section	Test Cases	Not Tested	Fail	Pass
Print Engine	6	0	0	6
Client Application	36	0	0	36
Security	7	0	0	7
Outsource Shipping	10	0	0	10
Exception Reporting	2	0	0	2
Final Report Output	16	0	0	16
Version Control	5	0	0	5

9.RESULTS

1.Performance Metrics

- Tracking income and expenses: Observing the pay and following all consumptions (through ledgers, versatile wallets, and credit and check cards)
- Exchange Receipts: Catch and sort out your instalment receipts to monitor your consumption.
- Sorting out Duties: Import your archives to the cost following application, and it will smooth out your pay and costs under the suitable assessment classes.
- Instalments and Solicitations: Acknowledge and pay from Visas, check cards, net banking, versatile wallets, and bank moves, and track the situation with your solicitations and bills in the portable application itself. Additionally, the following application sends reminders for instalments and naturally coordinates the instalments with solicitations.
- Reports: The cost following application creates and sends reports to give an itemized understanding about benefits, misfortunes, spending plans, pay, monetary records, and so on.,
- Online business combination: Coordinate your cost following application with your Internet business store and track your deals through instalments got by means of various instalment strategies.
- Merchants and Workers for hire: Oversee and follow every one of the instalments to the sellers and project workers added to the versatile application.
- Access control: Increment your group efficiency by giving access control to specific clients through custom authorizations.
- Track Activities: Decide project benefit by following work costs, finance, costs, and so forth, of your continuous venture.
- Stock following: A cost following application can do everything. Right from following items or the cost of merchandise, sending ready notices when the item is running unavailable or the item isn't selling, to buy orders.
- Top to bottom experiences and examination: Gives in-constructed devices to produce reports with easy to-comprehend visuals and illustrations to acquire bits of knowledge about the presentation of your business.
- Intermittent Costs: Depend on your planning application to follow, smooth out, and robotize every one of the repetitive costs and remind you on a convenient premise.

10.ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

One of the significant aces of following spending is continuously monitoring the condition of one's individual budgets. Following what you spend can assist you with adhering to your financial plan, in an overall way, however in every class like lodging, food, transportation and gifts. While a con is that physically following all money that is spent can be disturbing as well as tedious, a genius is that doing this naturally can be fast and basic. Another genius is that numerous programmed spending following programming programs are accessible free of charge. Having the program on a hand-held gadget can be a principal genius since it tends to be checked prior to spending happens to make certain of the accessible financial plan.

DISADVANTAGES:

A con with any framework used to follow spending is that one might begin doing it then tighten until it's overlooked all together. However, this is a gamble for any new objective, for example, attempting to get in shape or stopped smoking. In the event that an individual first makes a financial plan arrangement, places cash in reserve funds prior to spending any each new payroll interval or month, the following objective can help. Along these lines, following spending and ensuring all receipts are represented just should be done more than once per month. Indeed, even with consistent following of one's ways of managing money, there is no assurance that monetary objectives will be met. Albeit this can be viewed as a con of following spending, it very well may be changed into an ace if one makes up their psyche to continue to attempt to deal with all funds appropriately.

11.CONCLUSION:

From this venture, we can oversee and continue to follow the everyday costs as well as pay. While making this task, we acquired a great deal of involvement of functioning collectively. We found different anticipated and unpredicted issues and we partook in a great deal tackling them collectively. We embraced things like video instructional exercises, text instructional exercises, web and learning materials to make our task total.

12.FUTURE SCOPE

The venture helps well to keep the pay and costs overall. In any case, this task has a few constraints:

- The application can't keep up with the reinforcement of information once it is uninstalled.
- This application doesn't give higher choice capacity.

To additional upgrade the ability of this application, we prescribe the accompanying elements to be integrated into the framework:

- Different language interface.
- Give reinforcement and recuperation of information.
- Give better UI to client.
- Portable applications advantage.

13.APPENDIX:

```
# -*- 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
from flask_db2 import DB2
import ibm_db
import ibm_db_dbi
# from gevent.pywsgi import WSGIServer
import os
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'
"""
dsn_hostname = "3883e7e4-18f5-4afe-be8c-
   fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"
dsn_uid = "sbb93800"
dsn_pwd = "wobsVLm6ccFxcNLe"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "31498"
dsn_protocol = "tcpip"

dsn = (
"DRIVER={0};"
"DATABASE={1};"
"HOSTNAME={2};"
"PORT={3};"
```

```
"PROTOCOL={4};"
"UID={5};"
"PWD={6};"
).format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol, dsn_uid, dsn_pwd)
"""
# app.config['DB2_DRIVER'] = '{IBM DB2 ODBC DRIVER}'
app.config['database'] = 'bludb'
app.config['hostname'] = '3883e7e4-18f5-4afe-be8c-
   fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'
app.config['port'] = '31498'
app.config['protocol'] = 'tcpip'
app.config['uid'] = 'sbb93800'
app.config['pwd'] = 'wobsVLm6ccFxcNLe'
app.config['security'] = 'SSL'
try:
mysql = DB2(app)

conn_str='database=bludb;hostname=3883e7e4-18f5-4afe-be8c-
   fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=31498;protocol=tcpip;\
uid=sbb93800;pwd=wobsVLm6ccFxcNLe;security=SSL'
ibm_db_conn = ibm_db.connect(conn_str,'','')

print("Database connected without any error !!")
except:
print("IBM DB Connection error   :    " + DB2.conn_errormsg())
# app.config['']
# 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 = ''
    print("Break point1")
    if request.method == 'POST' :
    username = request.form['username']
    email = request.form['email']
    password = request.form['password']
    print("Break point2" + "name: " + username + "------" + email + "------" + password)
    try:
    print("Break point3")
    connectionID = ibm_db_dbi.connect(conn_str, '', '')
    cursor = connectionID.cursor()
    print("Break point4")
    except:
    print("No connection Established")
    # cursor = mysql.connection.cursor()
    # with app.app_context():
    #     print("Break point3")
    #     cursor = ibm_db_conn.cursor()
    #     print("Break point4")
    print("Break point5")
    sql = "SELECT * FROM register WHERE username = ?"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.execute(stmt)
    result = ibm_db.execute(stmt)
    print(result)
    account = ibm_db.fetch_row(stmt)
    print(account)
    param = "SELECT * FROM register WHERE username = " + "\'" + username + "\'"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    print("---- ")
    dictionary = ibm_db.fetch_assoc(res)
    while dictionary != False:
    print("The ID is : ", dictionary["USERNAME"])
    dictionary = ibm_db.fetch_assoc(res)
    # dictionary = ibm_db.fetch_assoc(result)
```

```
# cursor.execute(stmt)
# account = cursor.fetchone()
# print(account)
# while ibm_db.fetch_row(result) != False:
#     # account = ibm_db.result(stmt)
#     print(ibm_db.result(result, "username"))
# print(dictionary["username"])
print("break point 6")
if account:
msg = 'Username 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:
sql2 = "INSERT INTO register (username, email,password) VALUES (?, ?, ?)"
stmt2 = ibm_db.prepare(ibm_db_conn, sql2)
ibm_db.bind_param(stmt2, 1, username)
ibm_db.bind_param(stmt2, 2, email)
ibm_db.bind_param(stmt2, 3, password)
ibm_db.execute(stmt2)
# 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)
    sql = "SELECT * FROM register WHERE username = ? and password = ?"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.bind_param(stmt, 2, password)
    result = ibm_db.execute(stmt)
    print(result)
    account = ibm_db.fetch_row(stmt)
    print(account)
    param = "SELECT * FROM register WHERE username = '" + "\'" + username + "\'" + " and password = '" +
        "\'" + password + "\'"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    if account:
    session['loggedin'] = True
    session['id'] = dictionary["ID"]
    userid = dictionary["ID"]
    session['username'] = dictionary["USERNAME"]
    session['email'] = dictionary["EMAIL"]
    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']
  print(date)
    p1 = date[0:10]
```

```
p2 = date[11:13]
p3 = date[14:]
p4 = p1 + "-" + p2 + "." + p3 + ".00"
print(p4)
# 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)


sql = "INSERT INTO expenses (userid, date, expensename, amount, paymode, category) VALUES (?, ?,
   ?, ?, ?, ?)"
stmt = ibm_db.prepare(ibm_db_conn, sql)
ibm_db.bind_param(stmt, 1, session['id'])
ibm_db.bind_param(stmt, 2, p4)
ibm_db.bind_param(stmt, 3, expensename)
ibm_db.bind_param(stmt, 4, amount)
ibm_db.bind_param(stmt, 5, paymode)
ibm_db.bind_param(stmt, 6, category)
ibm_db.execute(stmt)
print("Expenses added")
# email part
param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND MONTH(date) =
MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
res = ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res)
expense = []
while dictionary != False:
temp = []
temp.append(dictionary["ID"])
temp.append(dictionary["USERID"])
temp.append(dictionary["DATE"])
temp.append(dictionary["EXPENSENAME"])
temp.append(dictionary["AMOUNT"])
temp.append(dictionary["PAYMODE"])
temp.append(dictionary["CATEGORY"])
expense.append(temp)
print(temp)
dictionary = ibm_db.fetch_assoc(res)
```

```
    total=0
    for x in expense:
    total += x[4]
    param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER BY id DESC
       LIMIT 1"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = 0
    while dictionary != False:
    temp = []
    temp.append(dictionary["LIMITSS"])
    row.append(temp)
    dictionary = ibm_db.fetch_assoc(res)
    s = temp[0]
    if total > int(s):
    msg = "Hello " + session['username'] + " , " + "you have crossed the monthly limit Team Personal
       Expense Tracker."
    #sendmail(msg,session['email'])
    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()
    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
    temp = []
    temp.append(dictionary["ID"])
    temp.append(dictionary["USERID"])
    temp.append(dictionary["DATE"])
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
```

```
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    expense.append(temp)
    print(temp)
    dictionary = ibm_db.fetch_assoc(res)
    return render_template('display.html' ,expense = expense)
#delete---the--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()
    param = "DELETE FROM expenses WHERE  id = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)
  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()
 param = "SELECT * FROM expenses WHERE  id = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    while dictionary != False:
    temp = []
    temp.append(dictionary["ID"])
    temp.append(dictionary["USERID"])
    temp.append(dictionary["DATE"])
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    row.append(temp)
    print(temp)
    dictionary = ibm_db.fetch_assoc(res)
```

```
    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()
    p1 = date[0:10]
    p2 = date[11:13]
    p3 = date[14:]
    p4 = p1 + "-" + p2 + "." + p3 + ".00"
    sql = "UPDATE expenses SET date = ? , expensename = ? , amount = ?, paymode = ?, category = ?
        WHERE id = ?"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, p4)
    ibm_db.bind_param(stmt, 2, expensename)
    ibm_db.bind_param(stmt, 3, amount)
    ibm_db.bind_param(stmt, 4, paymode)
    ibm_db.bind_param(stmt, 5, category)
    ibm_db.bind_param(stmt, 6, id)
    ibm_db.execute(stmt)
  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()
    sql = "INSERT INTO limits (userid, limitss) VALUES (?, ?)"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, session['id'])
    ibm_db.bind_param(stmt, 2, number)
    ibm_db.execute(stmt)
    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]

    param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER BY id DESC
       LIMIT 1"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = " /-"
    while dictionary != False:
    temp = []
    temp.append(dictionary["LIMITSS"])
    row.append(temp)
    dictionary = ibm_db.fetch_assoc(res)
    s = temp[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)
```

```
param1 = "SELECT TIME(date) as tn, amount FROM expenses WHERE userid = " + str(session['id']) + "
AND DATE(date) = DATE(current timestamp) ORDER BY date DESC"
res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
dictionary1 = ibm_db.fetch_assoc(res1)
texpense = []
while dictionary1 != False:
temp = []
temp.append(dictionary1["TN"])
temp.append(dictionary1["AMOUNT"])
texpense.append(temp)
print(temp)
dictionary1 = ibm_db.fetch_assoc(res1)
#   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()
param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND DATE(date) =
   DATE(current timestamp) ORDER BY date DESC"
res = ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res)
expense = []
while dictionary != False:
temp = []
temp.append(dictionary["ID"])
temp.append(dictionary["USERID"])
temp.append(dictionary["DATE"])
temp.append(dictionary["EXPENSENAME"])
temp.append(dictionary["AMOUNT"])
temp.append(dictionary["PAYMODE"])
temp.append(dictionary["CATEGORY"])
expense.append(temp)
print(temp)
dictionary = ibm_db.fetch_assoc(res)
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)


    param1 = "SELECT DATE(date) as dt, SUM(amount) as tot FROM expenses WHERE userid = " +
        str(session['id']) + " AND MONTH(date) = MONTH(current timestamp) AND YEAR(date) =
        YEAR(current timestamp) GROUP BY DATE(date) ORDER BY DATE(date)"
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
```

```
dictionary1 = ibm_db.fetch_assoc(res1)
texpense = []

while dictionary1 != False:
temp = []
temp.append(dictionary1["DT"])
temp.append(dictionary1["TOT"])
texpense.append(temp)
print(temp)
dictionary1 = ibm_db.fetch_assoc(res1)



#   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()

param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND MONTH(date) =
    MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
res = ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res)
expense = []
while dictionary != False:
temp = []
temp.append(dictionary["ID"])
temp.append(dictionary["USERID"])
temp.append(dictionary["DATE"])
temp.append(dictionary["EXPENSENAME"])
temp.append(dictionary["AMOUNT"])
temp.append(dictionary["PAYMODE"])
temp.append(dictionary["CATEGORY"])
expense.append(temp)
print(temp)
dictionary = ibm_db.fetch_assoc(res)



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)

param1 = "SELECT MONTH(date) as mn, SUM(amount) as tot FROM expenses WHERE userid = " +
    str(session['id']) + " AND YEAR(date) = YEAR(current timestamp) GROUP BY MONTH(date) ORDER BY
    MONTH(date)"
res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
dictionary1 = ibm_db.fetch_assoc(res1)
texpense = []

while dictionary1 != False:
temp = []
temp.append(dictionary1["MN"])
temp.append(dictionary1["TOT"])
texpense.append(temp)
print(temp)
dictionary1 = ibm_db.fetch_assoc(res1)



#   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()

param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND YEAR(date) =
    YEAR(current timestamp) ORDER BY date DESC"
res = ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res)
expense = []
```

```
while dictionary != False:
temp = []
temp.append(dictionary["ID"])
temp.append(dictionary["USERID"])
temp.append(dictionary["DATE"])
temp.append(dictionary["EXPENSENAME"])
temp.append(dictionary["AMOUNT"])
temp.append(dictionary["PAYMODE"])
temp.append(dictionary["CATEGORY"])
expense.append(temp)
print(temp)
dictionary = ibm_db.fetch_assoc(res)


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)
        session.pop('email', None)
        return render_template('home.html')
        port = os.getenv('VCAP_APP_PORT', '8080')
        if __name__ == "__main__":
        app.secret_key = os.urandom(12)
  app.run(debug=True, host='0.0.0.0', port=port)
```

Project Demonstration Video Link :

https://drive.google.com/file/d/17yqYIEs_uiOJwojV1scxedG6iU6ZoEm7/view

Git Hub:

https://github.com/IBM-EPBL/IBM-Project-22628-1659855406

Project Deployment Link:

https://personalexpensetracker-sweet-okapi-yh.mybluemix.net/