

## PERSONAL EXPENSE TRACKER APPLICATION

**IBM NALAIYATHIRAN (HX8001) PROJECT REPORT**

|  |  |
| --- | --- |
| ***Submitted by*** | |
| **SHRUTHIKA P** | **211719106076** |
| **SNEHA P** | **211719106079** |
| **SNEHA S N** | **211719106080** |
| **VARALAXMI V** | **211719106092** |

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING IN**

**ELECTRONICS AND COMMUNICATION ENGINEERING RAJALAKSHMI INSTITUTE OF TECHNOLOGY**

**ANNA UNIVERSITY: CHENNAI 600 025**

**NOVEMBER 2022**

## ANNA UNIVERSITY: CHENNAI 600 025

**BONAFIDE CERTIFICATE**

Certified that this project report **“PERSONAL EXPENSE TRACKER APPLICATION”** is the bonafide work of **“SHRUTHIKA.P(211719106076), SNEHA.S.N(211719106080), SNEHA.P(211719106079), VARALAXMI.V(211719106092),** who carried out the project work under my supervision.

**SIGNATURE: SIGNATURE:**

**Dr. S. MANJULA, Mrs.S.KALAIVANI,**

**M.E., Ph.D, M.E , Assistant Professor**

**HEAD OF THE DEPARTMENT, MENTOR,**

Dept. of Electronics and Dept. of Electronics and

Communication Engg., Communication Engg.,

Rajalakshmi Institute of Rajalakshmi Institute of

Technology, Technology,

Kuthambakkam Post, Kuthambakkam Post,

Chennai - 600 124 Chennai - 600 124

The viva-voce is held on .

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**INDEX**

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

Source Code

GitHub & Project Demo Link

**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. This

system is made and supervised by the experts and satisfying by the use

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 objective of this project is support to the user to sustain all financial activities like digital automated dairy. This application helps the user to avoid unexpected expenses and bad financial situations.

➢Using this application, users can manage all financial data and track all expense and income category wise.

➢Creating a category and recording all expenses and income under the category.

➢Enable the notification system user get notification daily at a specific time that can help the user insert expense and income.

➢Backup and Restore all information.

➢Report are generated in PDF format in category wise or time period.

**2. LITERATURE SURVEY:**

**2.1. EXISTING PROBLEM:**

After discussing our application functions and comparing them to other existing applications, some features were found lacking. This is a new application that will attract the public user through its features. There are always some challenges. We have to face some challenges as well, since the main purpose of our application is to track the user's expenses.

➢This is a windows-based application, so if a user does not have a laptop or PC then this application will not help them.

➢After getting notifications if a user doesn’t check his source for full information then the main motto of this app will fail.

**2.2. References**

[1] [http://expense-manager.com/how-expense software/](http://expense-manager.com/how-expense%20software/)

[2] <https://www.splitwise.com/terms>

[3] <http://code.google.com/p/socialauthandroid/wiki/Facebook>

[4] <http://code.google.com/p/socialauth-android>

[5] http://www.appbrain.com/app/expensemanager/ com.expensemanager

[6] https://www.xpenditure.com/en?

[7] [http://expense-manager.com/how-expense software/](http://expense-manager.com/how-expense%20software/)

[8] Donn Felker, “Android Application Development for Dummies”, published by For Dummies, 2010.

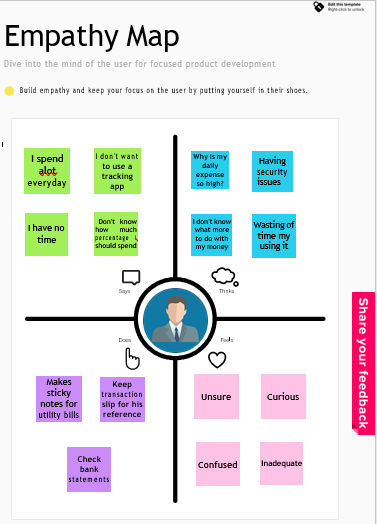
**2.3. Problem Statement Definition:**

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.

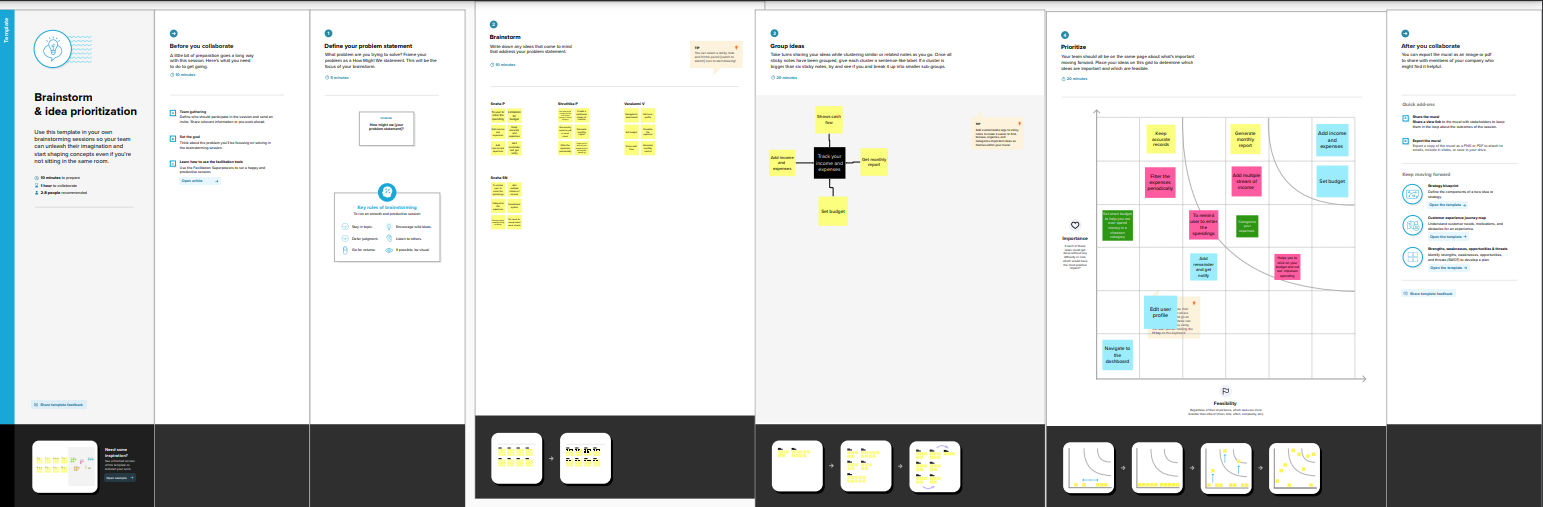
|  |  |
| --- | --- |
| Who does the problem affect? | People getting regular wages. |
| What is the issue? | The paper based expense tracker system does not provide the user portability , existing system only used on paper based records so unable to update anywhere expenses done and unable to update the location of the expense details disruptive that the proposed system. |
| When does the issue occurs? | When the digits could not be recognized correctly. When the transactions are not successful. When the elder people unable to understand the smaller handwritten digits. When the paper based expense tracker records are subjected to fire accident, flood, etc. |
| Where is the issue occurring? | The issue occurs when the person is unable to  track his income and expenditure. |
| Why is it important that we fix theproblem? | By solving this issue those people getting regular wages can track their expenses and  avoid unwanted expenses. |

**3. IDEATION & PROPOSED SOLUTION**

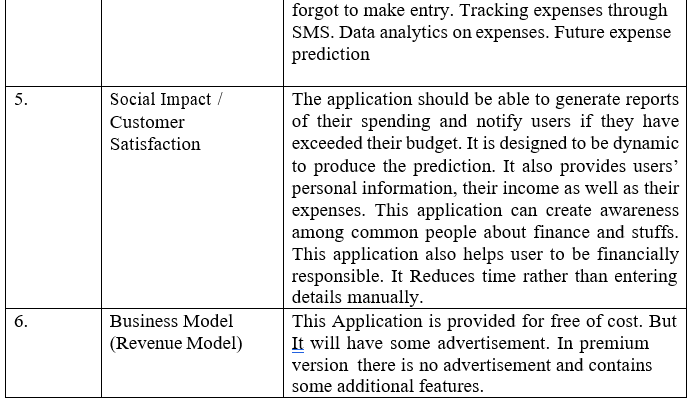
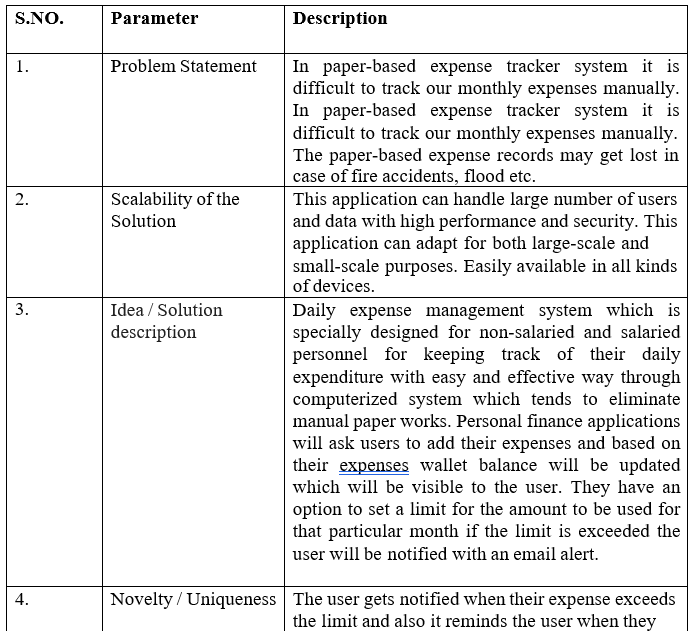
**3.1Empathy Map Canvas**

****

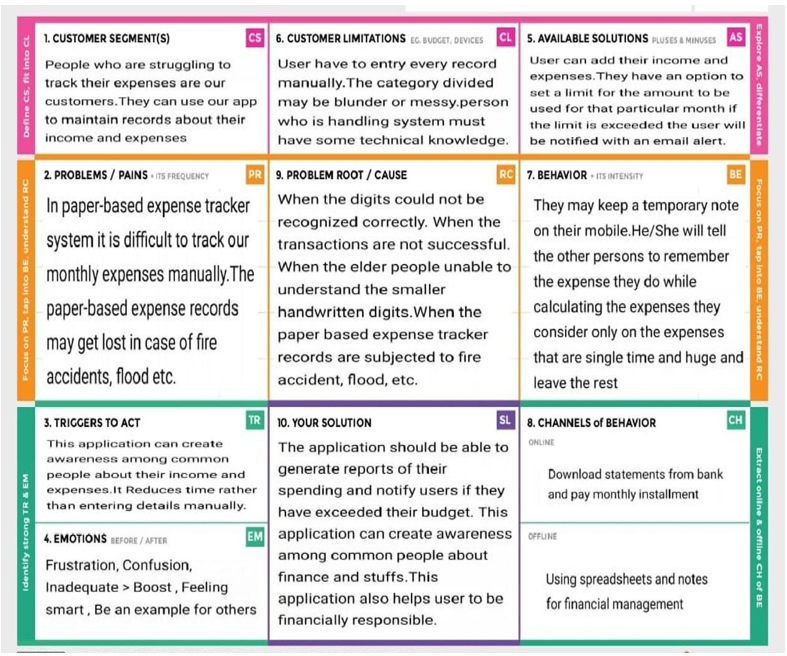
**3.2 Ideation & Brainstorming**

****

* 1. **Proposed Solution**

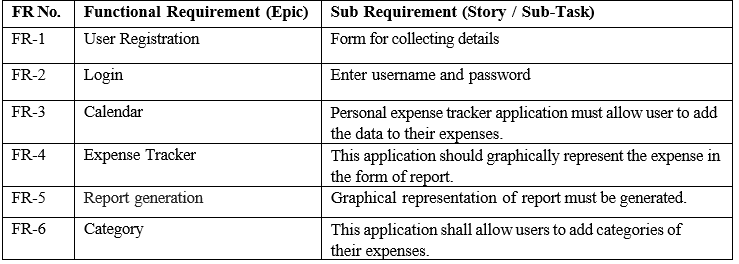
****

* 1. **Solution Fit Problem**

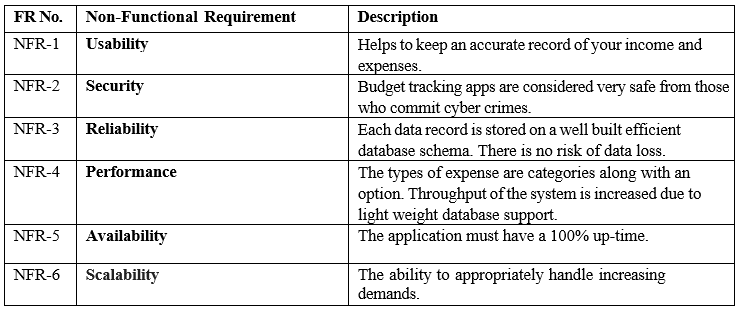
****

**4. REQUIREMENT ANALYSIS**

* 1. **Functional requirement**

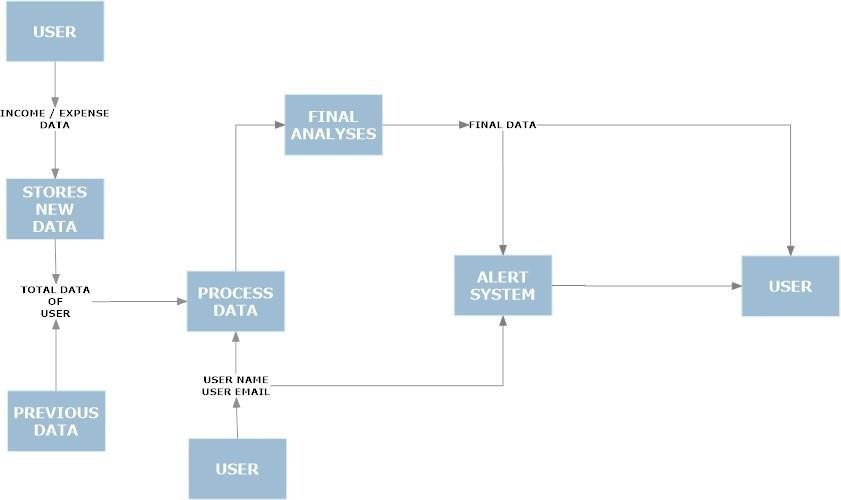
**\**

**14.2Non-Functional requirements**

****

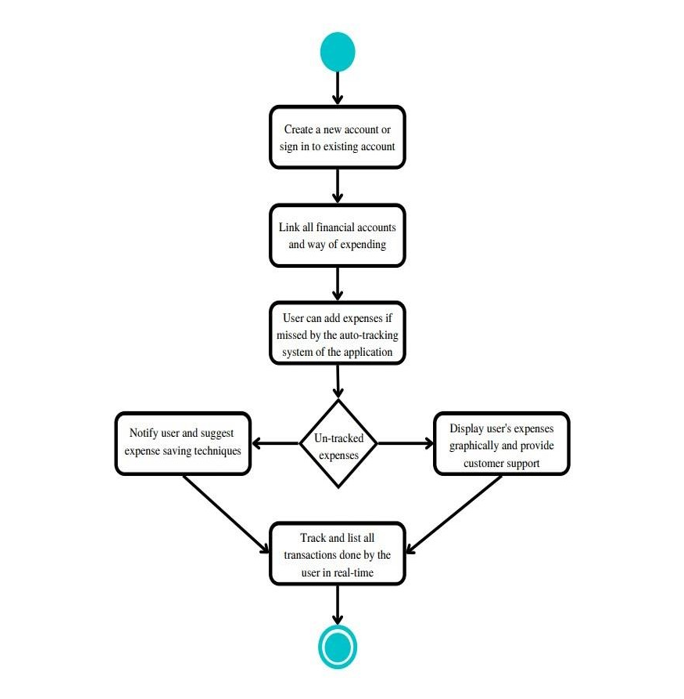
1. **PROJECT DESIGN**
   1. **Data Flow Diagram**

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

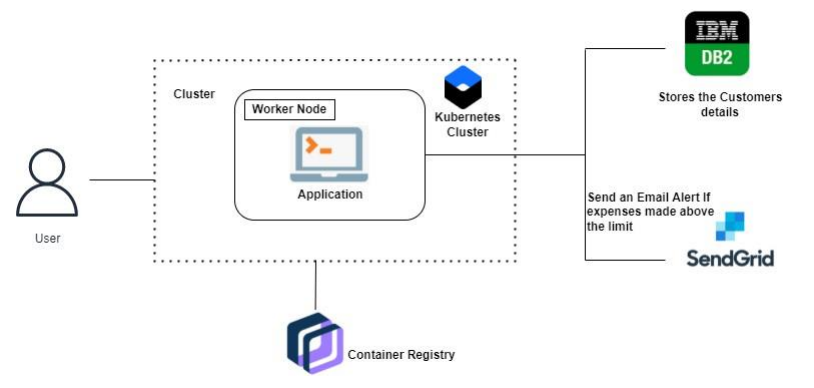


* 1. **Solution & Technical Architecture**

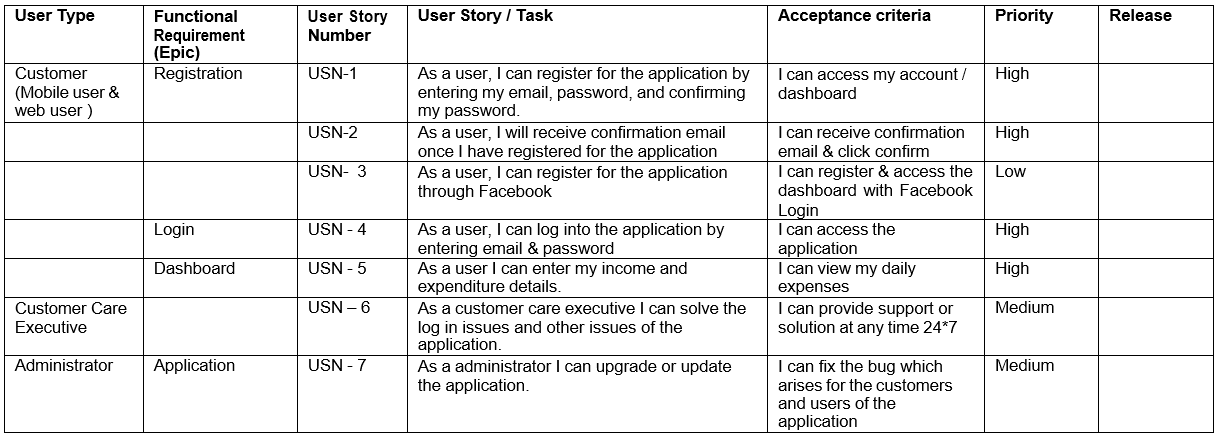
**Solution Architecture:**



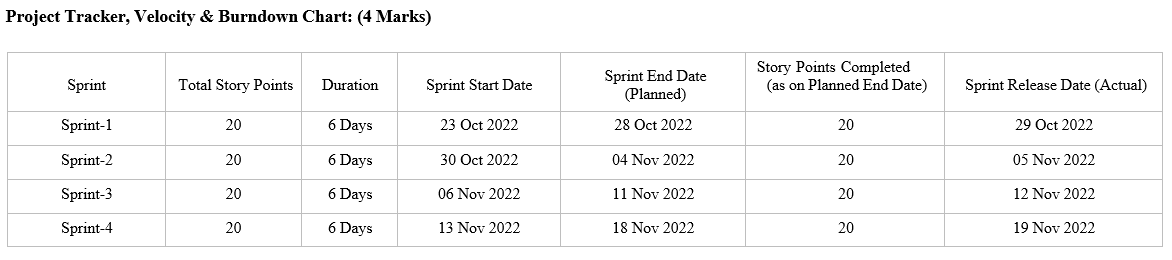
**Technical Architecture:**

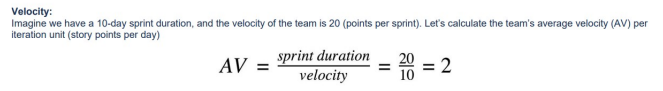


**5.3User Stories:**

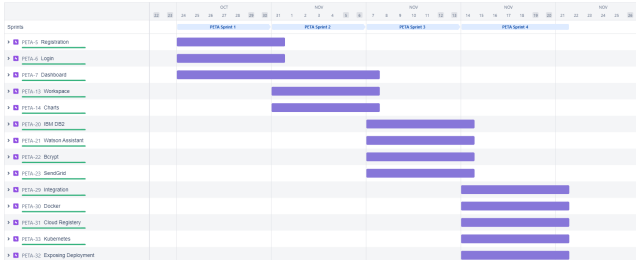
****

1. **PROJECT PLANNING & SCHEDULING**
   1. **Sprint Planning & Estimation & Scheduling:**



****

* 1. **Report from JIRA:**

****

**7.CODING & SOLUTIONING**

**(Explain the features added in the project along with code)**

**7.1 Feature 1:**

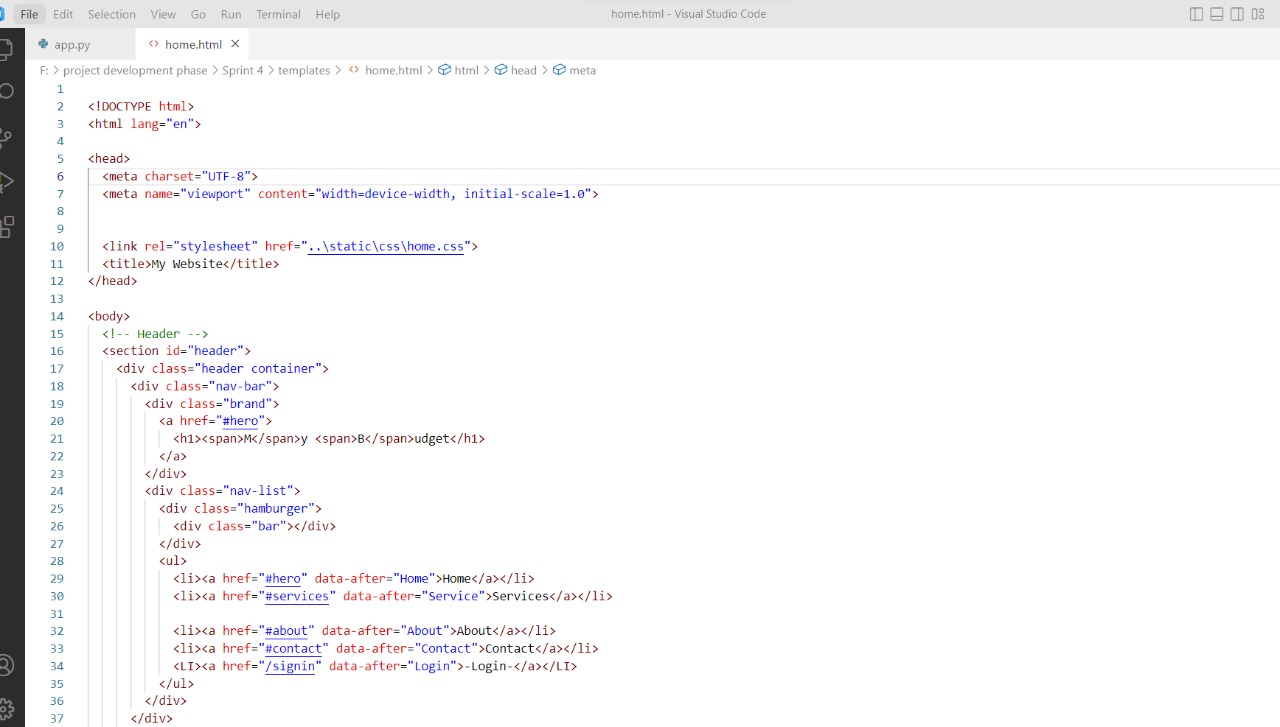
**PYTHON:**

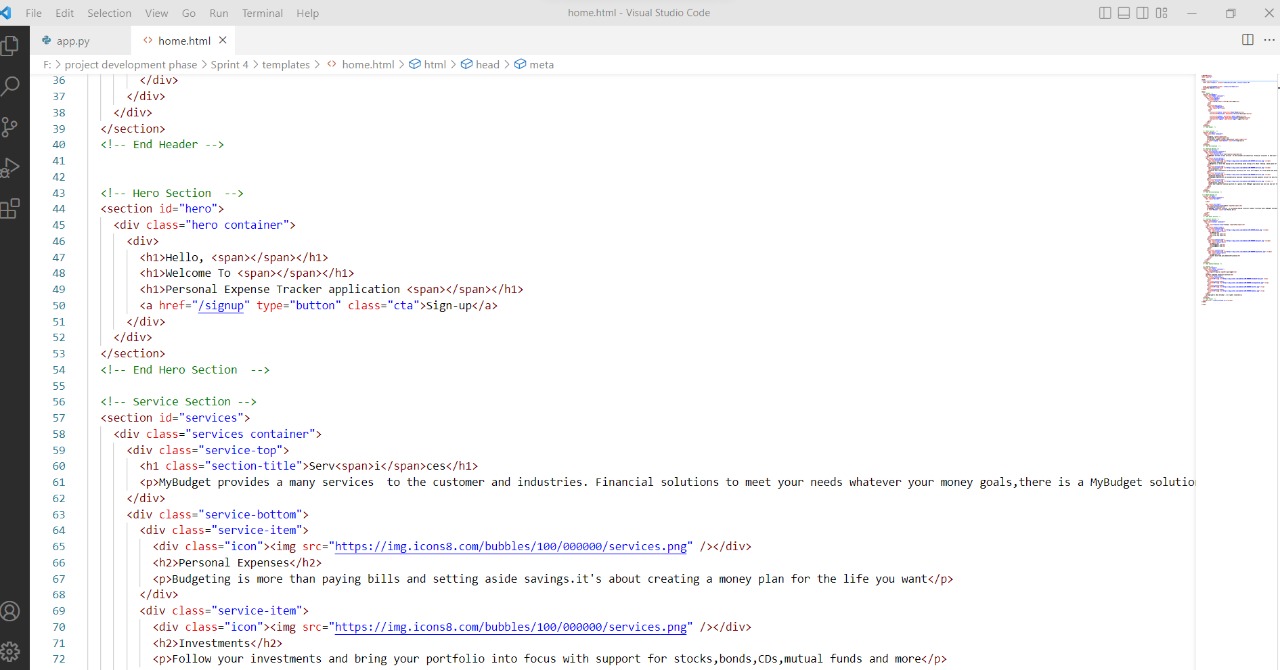
* Python is a widely-used, interpreted, object-oriented, and high-level programming language with dynamic semantics, used for general-purpose programming. It’s everywhere, and people use numerous Python-powered devices on a daily basis, whether they realize it or not.
* Python was created by Guido van Rossum, and first released on February 20, 1991.
* Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell and other scripting languages.
* Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL)
* It is easy to learn – the time needed to learn Python is shorter than for many other languages; this means that it’s possible to start the actual programming fast.
* It is easy to use for writing new software – it’s often possible to write code faster when using Python.
* It is easy to obtain, install and deploy – Python is free, open and multiplatform; not all languages can boast that.
* Programming skills prepare you for careers in almost any industry and are required if you want to continue to more advanced and higher-paying software development and engineering roles.
* Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress

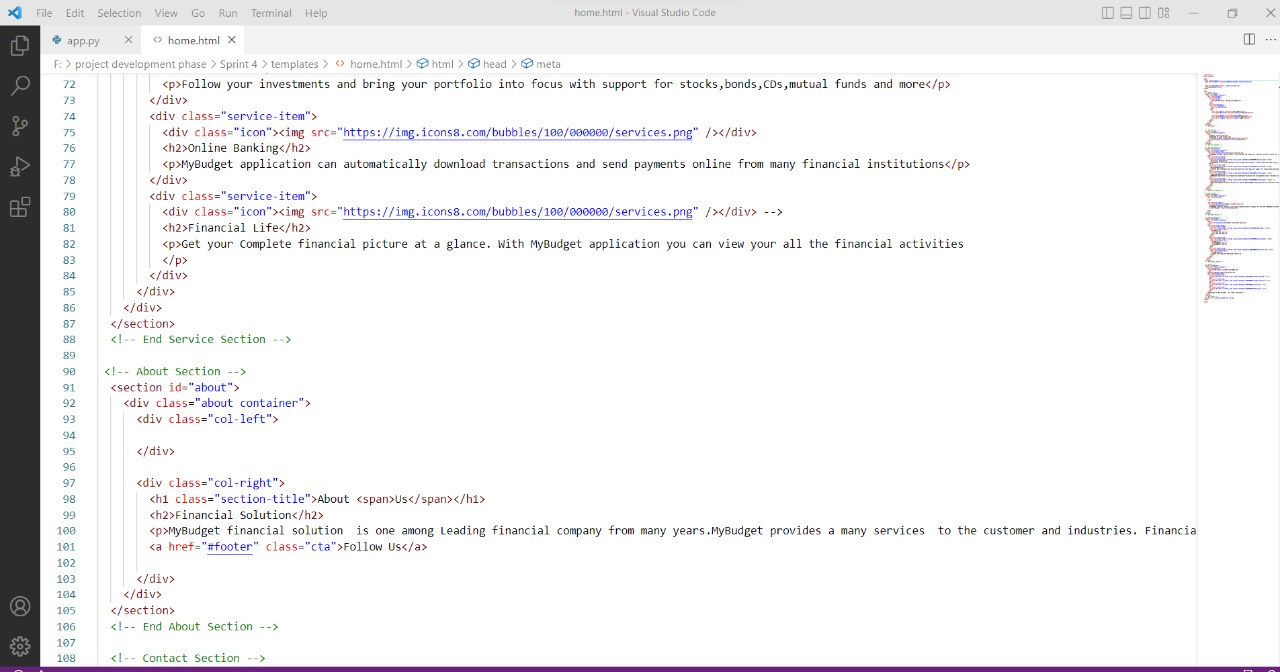
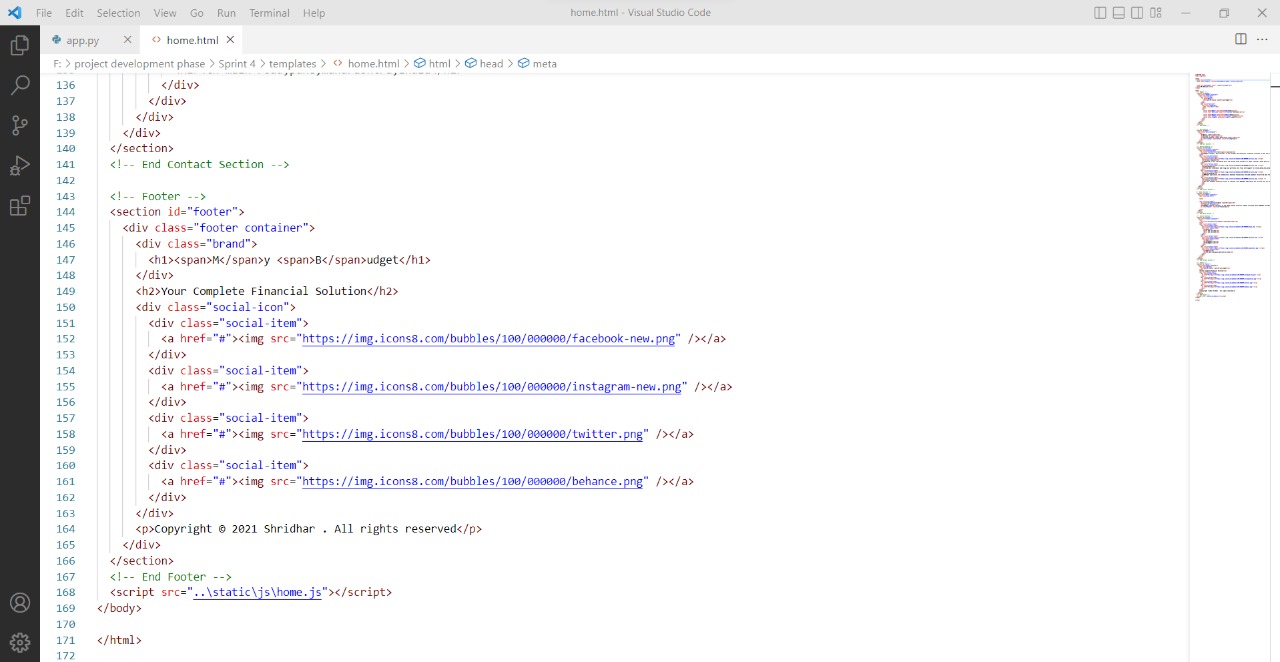
**7.2 Feature 2:**

**FLASK:**

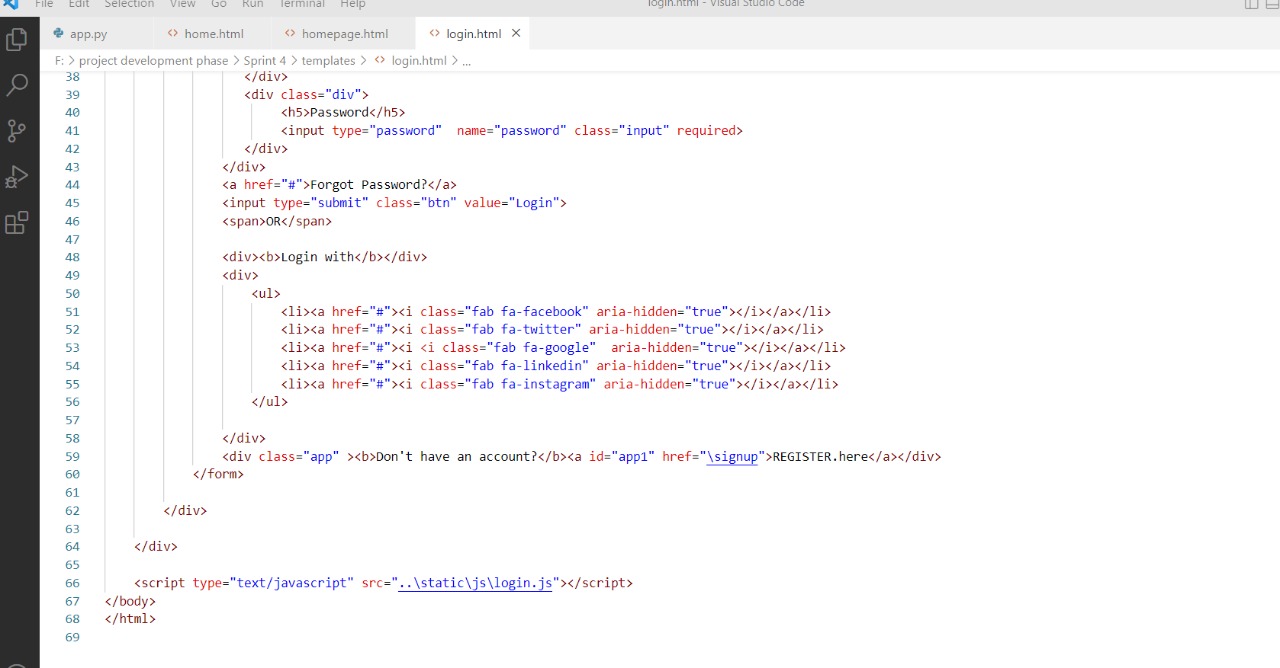
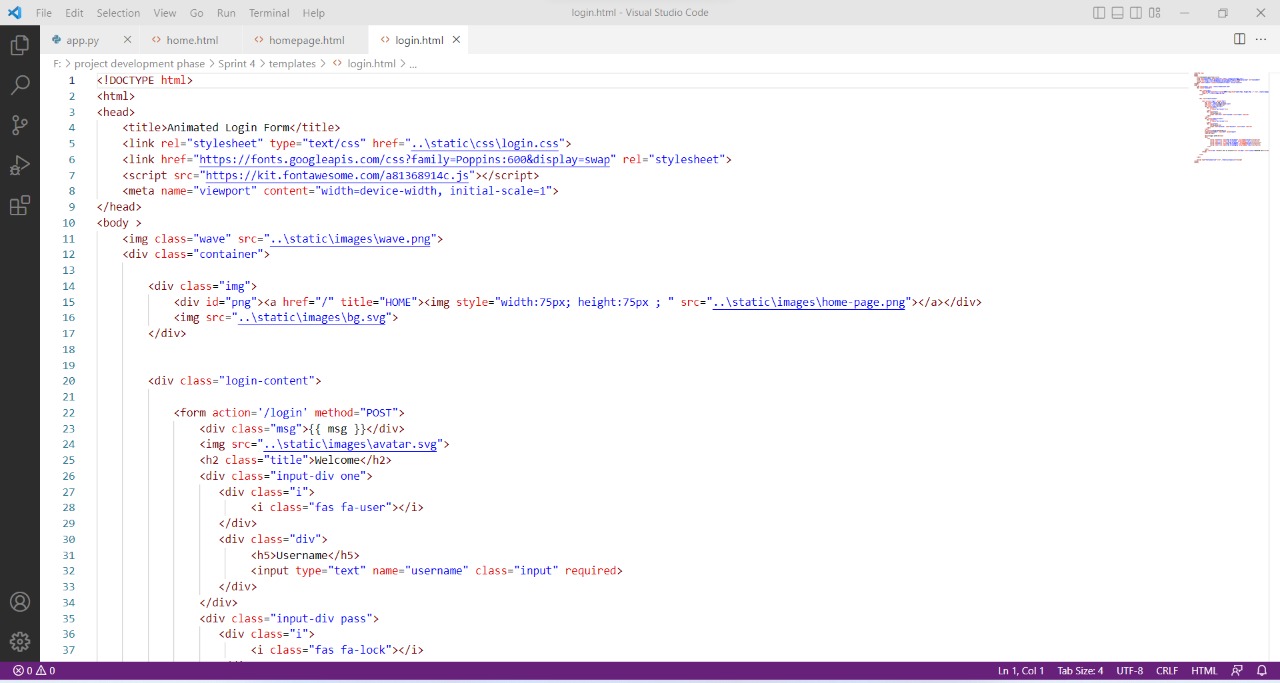
* Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.
* It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself.
* Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.
* Applications that use the Flask framework include Pinterest and LinkedIn.

**Home**

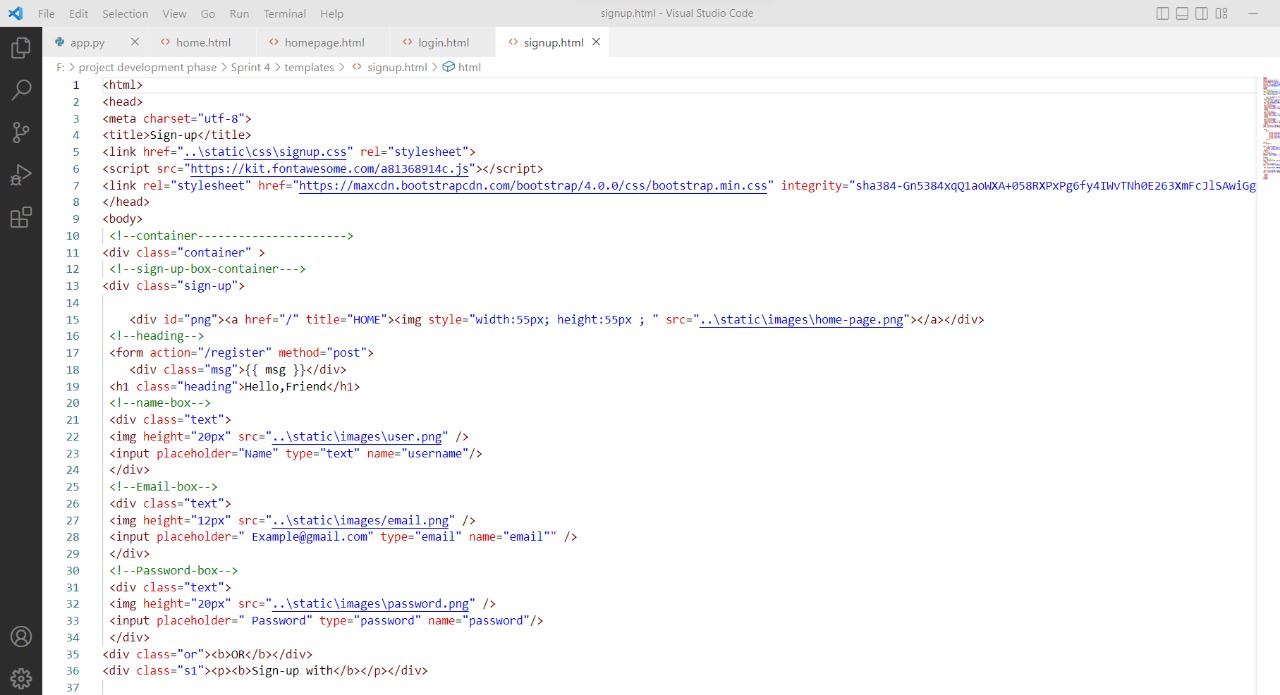
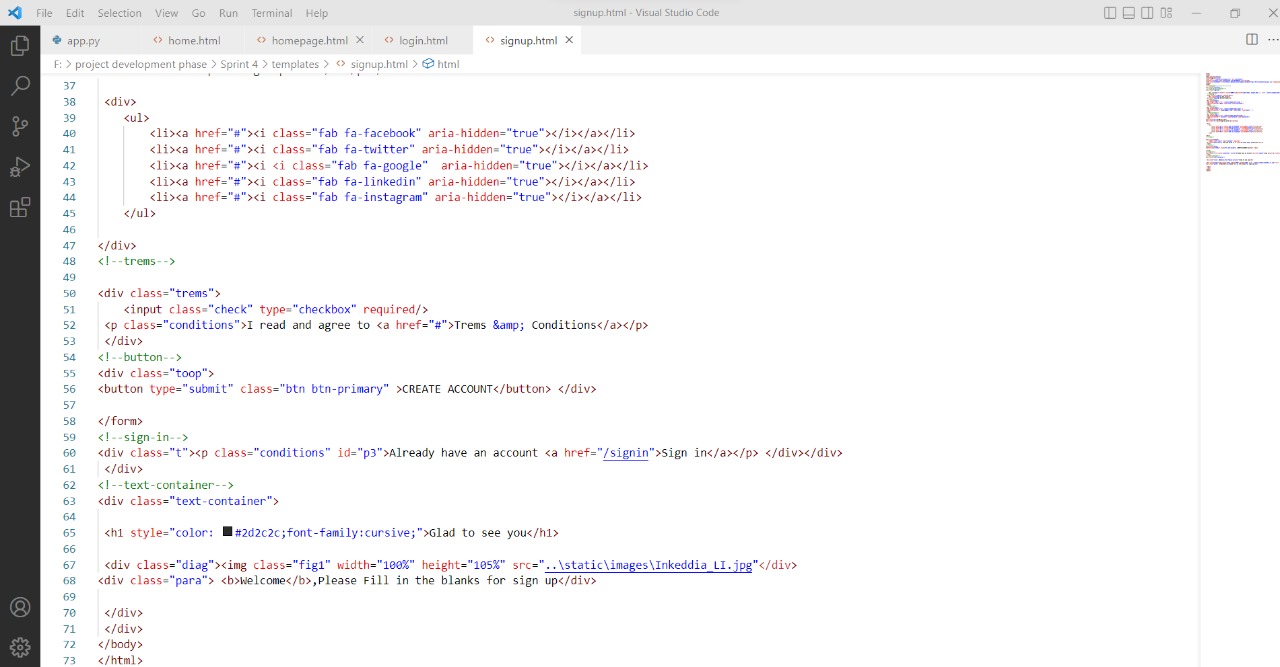
****

****

**Login**

****

**Signup:**

****

**7.3 Database Schema:**

**IBM DB2:**

* DB2 is a database product from IBM.
* It is a Relational Database Management System (RDBMS). DB2 is designed to store, analyze and retrieve the data efficiently.
* DB2 product is extended with the support of Object-Oriented features and non-relational structures with XML.
* Provide a massively parallel processing (MPP) architecture Exploits Hive, HBase and Apache Spark concurrently for best-in-class analytic capabilities.
* Provides low latency support for ad-hoc and complex queries, high performance, and federation capabilities Understands dialects from other vendors and various products 21 from Oracle, IBM® Db2® and IBM Netezza® Enables advanced row and column security.

**KUBERNETES:**

* Kubernetes is also known as 'k8s'.
* Kubernetes is an extensible, portable, and open-source platform designed by Google in 2014.
* It is mainly used to automate the deployment, scaling, and operations of the container based applications across the cluster of nodes.

**8.TESTING**

**8.1. Testing:**

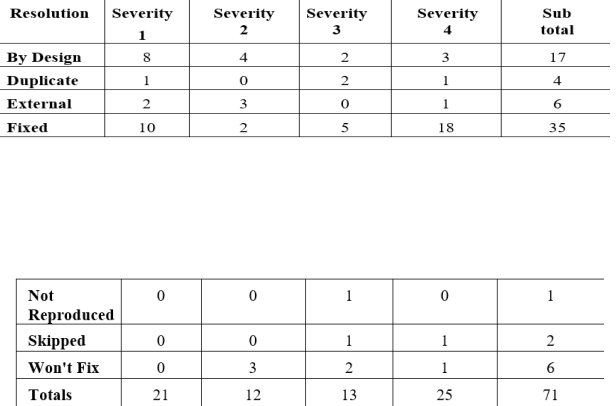
1.Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Plasma Donation Application project at the time of the release to User Acceptance Testing (UAT).

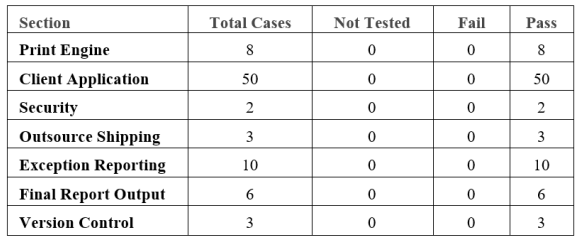
2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how

they were resolved.

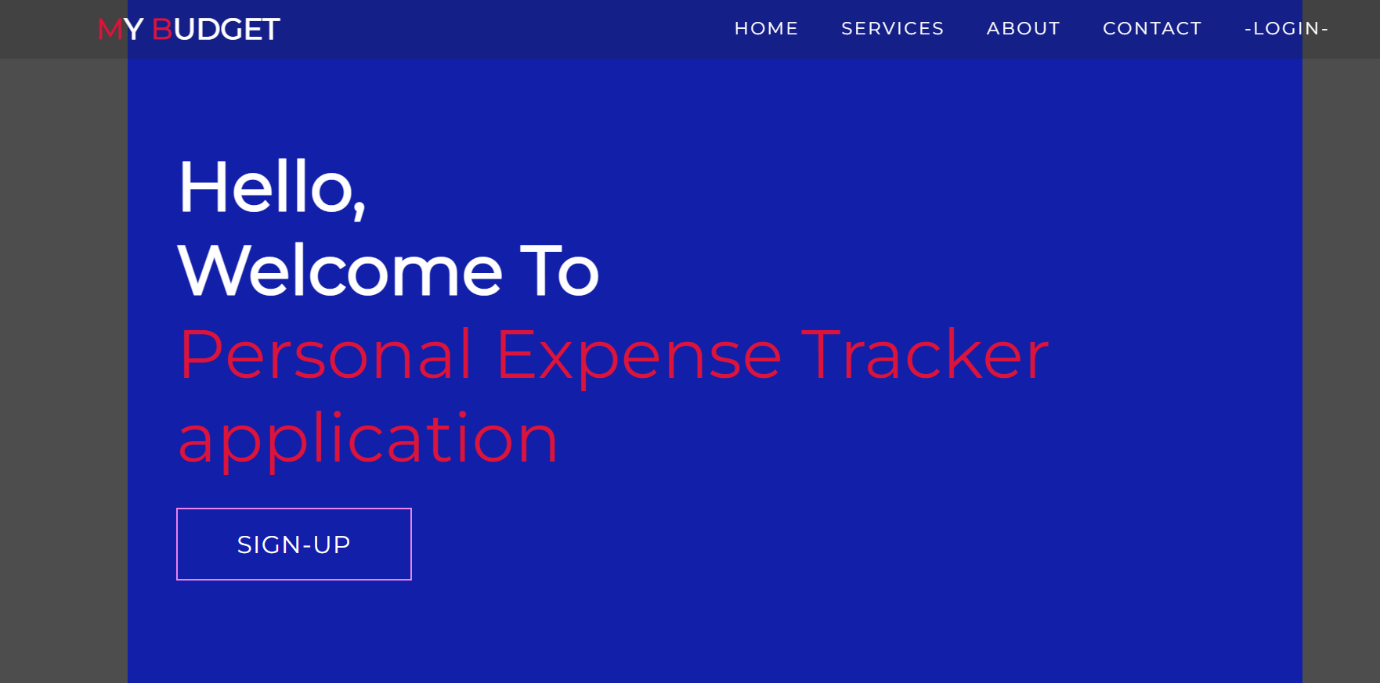
**\**

**8.3 Test case analysis:** This report shows the number of test cases that have passed, failed, and untested.

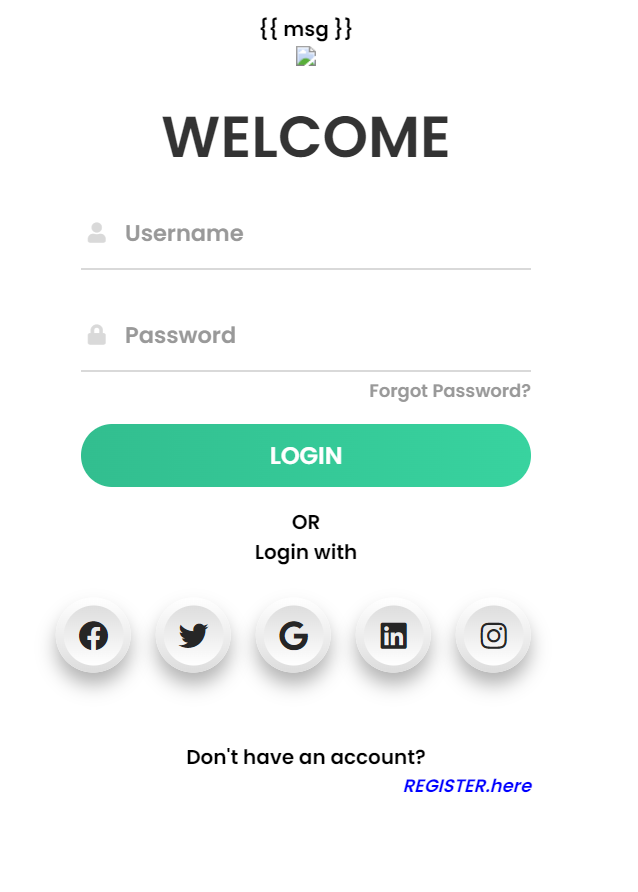
****

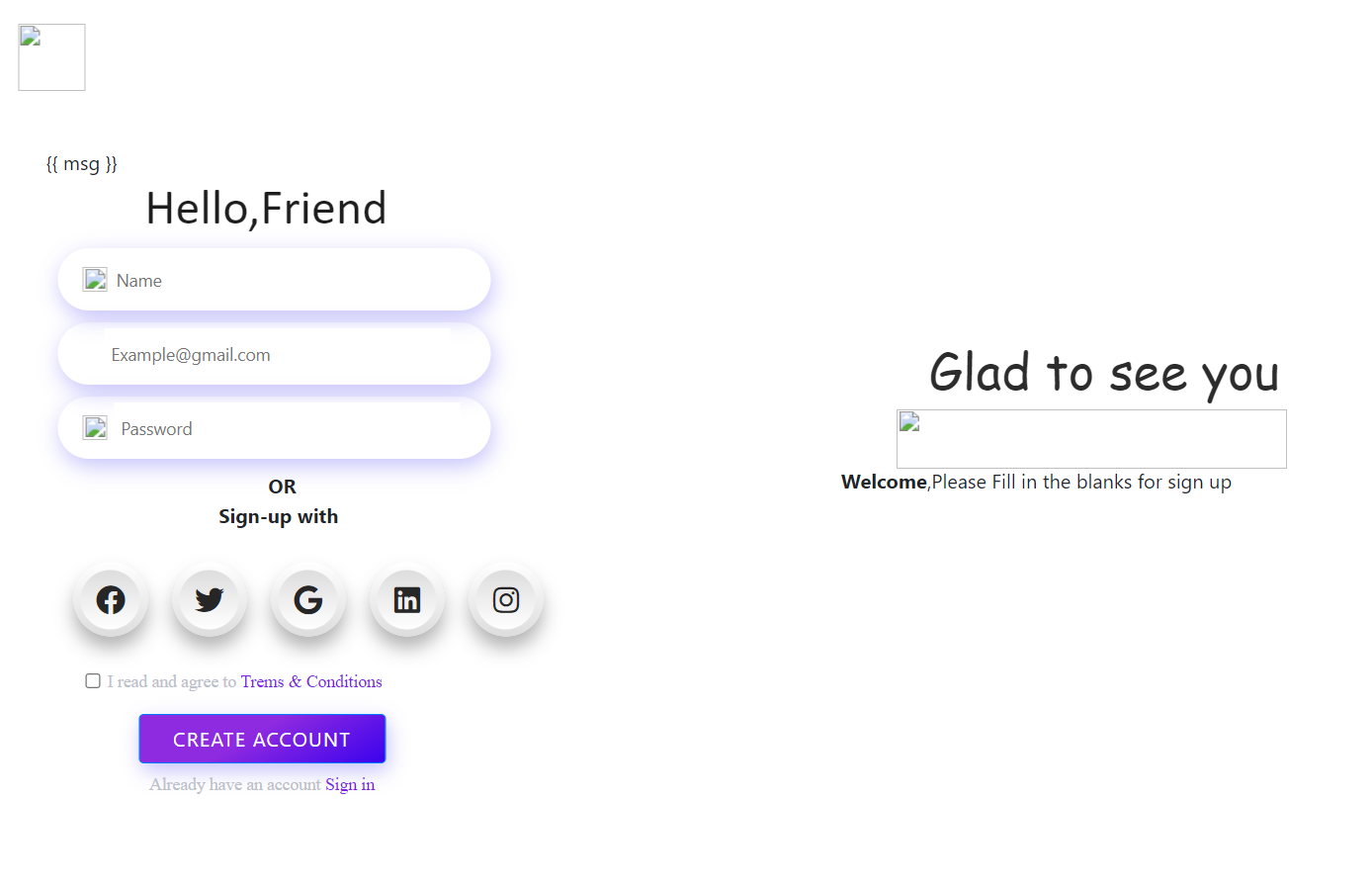
**9.RESULTS**

**9.1Performance Metrics:**

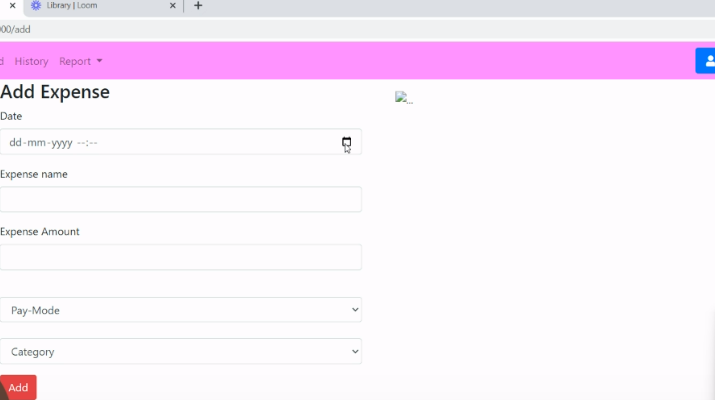
****

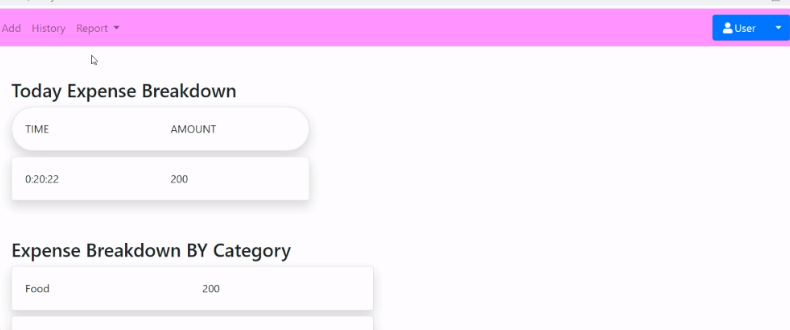
****

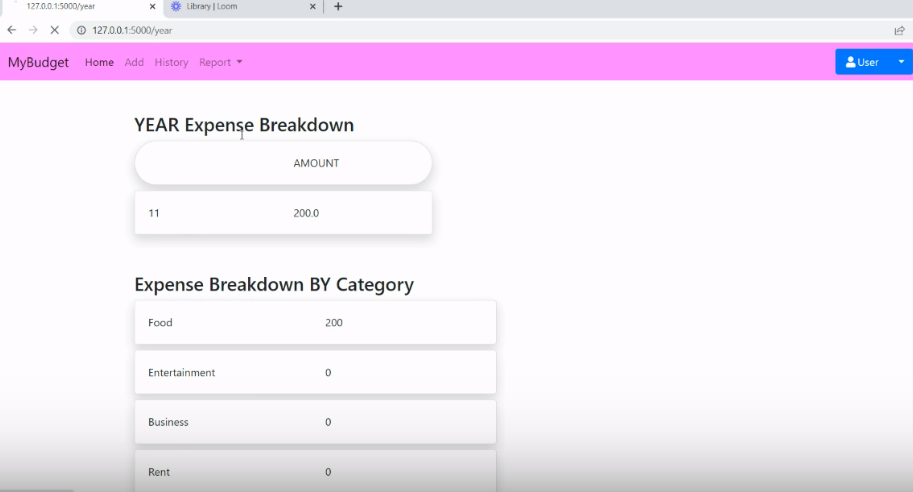
****

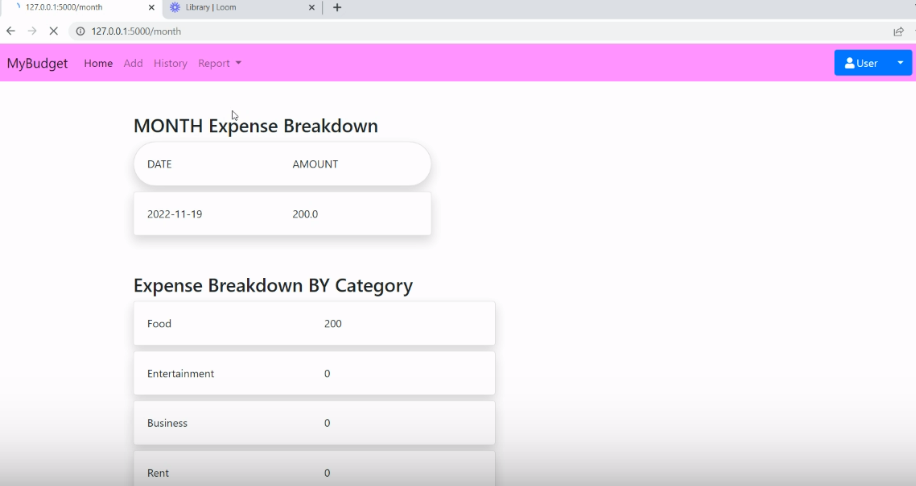
****











**10. ADVANTAGES AND DISADVANTAGES**

**Advantages:**

* **Easy to handle**
* **We can be aware of our expenses**
* **Budget friendly**
* **Easy access**
* **Faster**
* **Can track your expense anywhere, anytime**

**Disadvantages:**

* **Internet connectivity**
* **Budgeting takes time and effort**

**11. CONCLUSION:**

fter making this application we assure that this application will help its

users to manage the cost of their daily expenditure. It will guide them and make them

aware about their daily expenses. It will prove to be helpful for the people who are

frustrated with their daily budget management, irritated because of the amount of

expenses and wish to manage money and to preserve the record of their daily cost

which may be useful to change their way of spending money. In short, this application

will help its users to overcome the wastage of money

Monitoring your everyday expenses can set aside you cash, yet it can likewise help you set your monetary objectives for what’s to come. On the off chance that you know precisely where your sum is going much of a stretch see where a few reductions and bargains can be made. Expense Tracker project is for keeping our day-to-day expenditures will helps us to keep record of our money daily. The project what we have created is work more proficient than the other income and expense tracker. The project effectively keeps away from the manual figuring for trying not to ascertain the pay and cost each month. It’s a user-friendly application.

**12. FUTURE SCOPE:**

1) It will have various options to keep record (for example Food, Travelling Fuel, Salary etc. 2) Automatically it will keep on sending notifications for our daily expenditure.

3) In today’s busy and expensive life, we are in a great rush to make moneys, but at the end of the month we broke off. As we are unknowingly spending money on title and unwanted things. So, we have come over with the plan to follow our profit.

4) Here user can define their own categories for expense type like food, clothing, rent and bills where they have to enter the money that has been spend and likewise can add some data in extra data to indicate the expense.

**13.APPENDIX**

**Source Code:**

import re

from flask import Flask, render\_template, request, redirect, session

from flask\_mysqldb import MySQL

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

app.secret\_key = 'IBM'

app.config['MYSQL\_HOST'] = 'localhost'

app.config['MYSQL\_USER'] = 'root'

app.config['MYSQL\_PASSWORD'] = 'Snehadiya@02'

app.config['MYSQL\_DB'] = 'expense'

mysql = MySQL(app)

@app.route("/home")

def home():

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 += int(x[4])

if x[6] == "food":

t\_food += int(x[4])

elif x[6] == "entertainment":

t\_entertainment += int(x[4])

elif x[6] == "business":

t\_business += int(x[4])

elif x[6] == "rent":

t\_rent += int(x[4])

elif x[6] == "EMI":

t\_EMI += int(x[4])

elif x[6] == "other":

t\_other += int(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("homepage.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("/")

def add():

return render\_template("home.html")

@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)

@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)

@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")

@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)

@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")

@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")

@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 += int(x[4])

if x[6] == "food":

t\_food += int(x[4])

elif x[6] == "entertainment":

t\_entertainment += int(x[4])

elif x[6] == "business":

t\_business += int(x[4])

elif x[6] == "rent":

t\_rent += int(x[4])

elif x[6] == "EMI":

t\_EMI += int(x[4])

elif x[6] == "other":

t\_other += int(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 += int(x[4])

if x[6] == "food":

t\_food += int(x[4])

elif x[6] == "entertainment":

t\_entertainment += int(x[4])

elif x[6] == "business":

t\_business += int(x[4])

elif x[6] == "rent":

t\_rent += int(x[4])

elif x[6] == "EMI":

t\_EMI += int(x[4])

elif x[6] == "other":

t\_other += int(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("month.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 += int(x[4])

if x[6] == "food":

t\_food += int(x[4])

elif x[6] == "entertainment":

t\_entertainment += int(x[4])

elif x[6] == "business":

t\_business += int(x[4])

elif x[6] == "rent":

t\_rent += int(x[4])

elif x[6] == "EMI":

t\_EMI += int(x[4])

elif x[6] == "other":

t\_other += int(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("year.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('/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-22805-1659858493**](https://github.com/IBM-EPBL/IBM-Project-22805-1659858493)

**Project Demo Link:** <https://www.loom.com/share/37356e5afec44c6a9df2b41d809225ab>