PERSONAL EXPENSE TRACKER APPLICATION

IBM-Project-39456-1660449676

NALAIYA THIRAN PROJECT BASED LEARNING ON PROFESSIONAL READINESS FOR INNOVATION EMPLOYNMENT AND ENTERPRENEURSHIP

A PROJECT REPORT BY

KABILESH G(922519104060)
ARAVINDHU N(922519104009)
GOWTHAM C(922519104050)
KISHORE KUMAR R(922519104301)

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE

V.S.B.ENGINEERING COLLEGE,KARUR -639111

INDEX

1. **INTRODUCTION**
    a. Project Overview
    b. Purpose
2. **LITERATURE SURVEY**
    a. Existing problem
    b. References
    c. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
    a. Empathy Map Canvas
    b. Ideation & Brainstorming
    c. Proposed Solution
    d. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
    a. Functional requirement
    b. Non-Functional requirements
5. **PROJECT DESIGN**
    a. Data Flow Diagrams
    b. Solution & Technical Architecture
    c. User Stories
6. **PROJECT PLANNING & SCHEDULING**
    a. Sprint Planning & Estimation
    b. Sprint Delivery Schedule
    c. Reports from JIRA

7.CODING&SOLUTIONING(Explain the features added in the project along with code)

a. Feature 1

b. Feature 2

c. Database Schema (if Applicable)

8. **TESTING**

a. Test Cases

b. User Acceptance Testing

9. **RESULTS**

a. Performance Metrics

10.ADVANTAGES & DISADVANTAGES

11.CONCLUSION

12.FUTURE SCOPE

13. **APPENDIX**

Source Code
GitHub & Project Demo Link

INTRODUCTION

a. Project Overview

TEAM ID : PNT2022TMID33342

INDUSTRYMENTOR : Kusboo

FACULTY MENTOR : K SENTHIL KUMAR.

Skills Required:

IBM Cloud, HTML, Javascript, IBM Cloud Object Storage, Python- Flask, Kubernetes, Docker, IBM DB2, IBM Container Registry

1. **INTRODUCTION**

a. Project Overview

This project is based on expense tracking .This project aims to create an easy, faster and smooth cloud application .For better expense tracking we developed our project that will help the users a lot .Most of the people cannot track their expenses and income leading to facing money crisis ,so this application can help people to track their expense day to day and make life stress free .Money is the most valuable portion of our daily life and without money we will not last one day on earth .So using the daily expense tracker application is important to lead a happy family. It helps the user to avoid unexpected expenses and bad financial situations. It will save time and provide a responsible lifestyle.

b. Purpose

Personal finance management is an important part of people's lives.

However, every one does not have the knowledge or time to manage their finances in a proper manner. And, even if a person has time and knowledge, they do not bother with tracking their expenses as they find it tedious and timeconsuming. Now, you don't have to worry about managing your expenses, as you can get access to an expense tracker that will help in the active management of your finances. Also known as expense manager and money manager, an expense tracker is a software or application that helps to keep an accurate record of your money inflow and outflow. Many people in India live on a fixed income, and they find that towards the end of the month they don't have sufficient money to meet their needs. While this problem can arise due to low salary, invariably it is due to poor money management skills.

People tend to overspend without realizing, and this can prove to be disastrous. Using a daily expense manager can help you keep track of how much you spend every day and on what. At the end of the month, you will have a clear picture where your money is going g. This is one of the best ways to get your expenses under control and bring some semblance of order to your finances .Today, there are several expense manager applications in the market. Some are paid managers while others are free. Even banks like ICICI offer their customers expense tracker to help them out. Before you decide to go in for a money manager ,it is important to decide the type you want.

2. **LITERATURE SURVEY**

a. Existing problem

In a study conducted by Forresterin 2016 surveying small and medium businesses (SMBs) across the world, 56% companies reported expense management as being the biggest challenge for their finance departments.

In another survey conducted by Level Research in 2018 in North

America,respondents reported the following pain points in expense management before adopting automation:

i. Manual entry and routing of expense reports (62%)

ii. Lack of visibility into spend data (42%)

iii. Inability to enforce travel policies (29%)

iv. Lost expense reports (24%)

v. Lengthy expense approval system and reimbursement cycles(23%)

REFERENCES:

1. Access Consultants. (1998). the final report on the analysis of the household budget and expenditure survey for St. Vincent and the Grenadines. Atlanta GA. Retrieved August 15,2006.

2. European Countries. (2004). Household budget surveys in candidate countries: Methodological analysis 2003. European Countries. Luxembourg. Retrieved February 19,2007.

3. M N Rajaprabha 2017 IOP Conf. Ser.: Mater. Sci. Eng. 263 042050.

3.Problem Statement Definition

I am	User, need to reduce unwanted expenses and save money.
I'm trying to	Needs to monitor and control expenses in day-to-day life.
But,	It takes long time to load and I couldn't get time to analyse every expense I made. And reading or viewing each expenses every time was not confortable for me in eachtime.
Because	I am not comfortable with listing each day expenses and can't find an appropriate way to handle it.
Which makes me feel	I consider listing a expense manually are frustrated to use the application.

Customer Problem Statement :

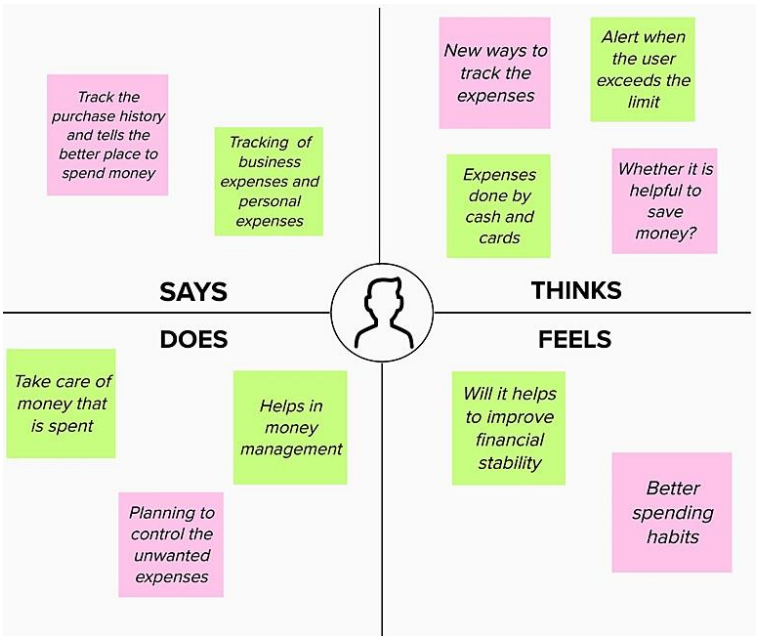
The personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management. Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert.

Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS-1	User	Control daily expenses	Hard to analyse expenses	Didn't have necessary data	Disappointed
PS-2	User	Overview expenses	All expenses are not reviewed.	Lack of necessary data	Annoying

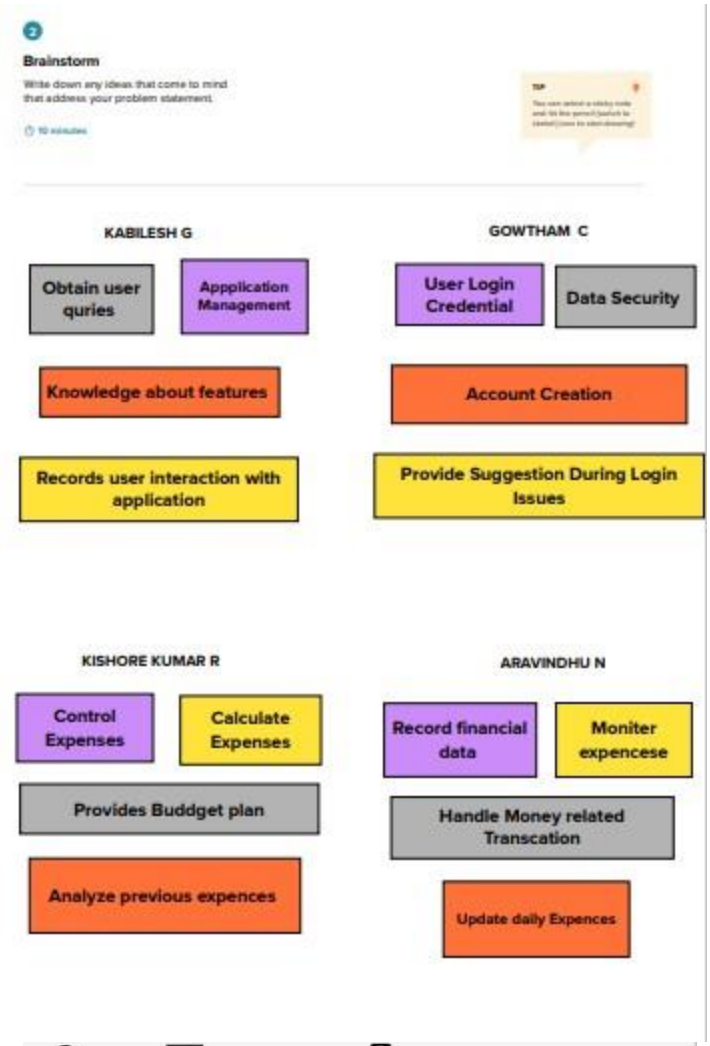
1. **IDEATION & PROPOSED SOLUTION**

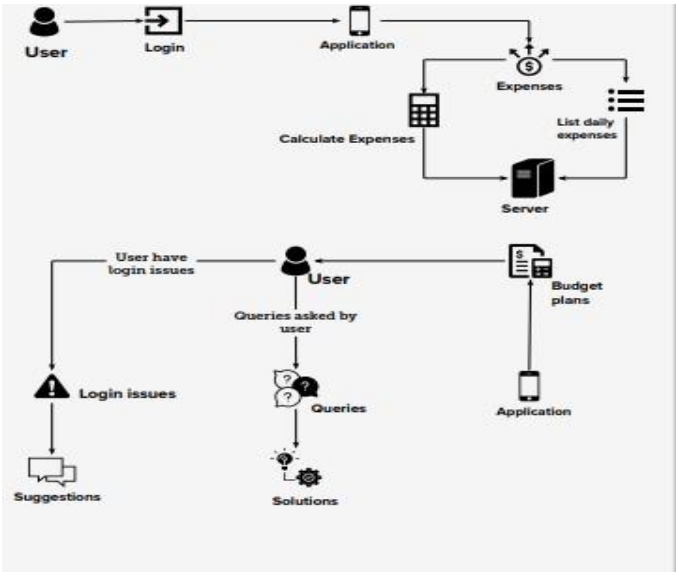
    a. Empathy Map Canvas

TEAM Gathering,collaboration and select the problem:

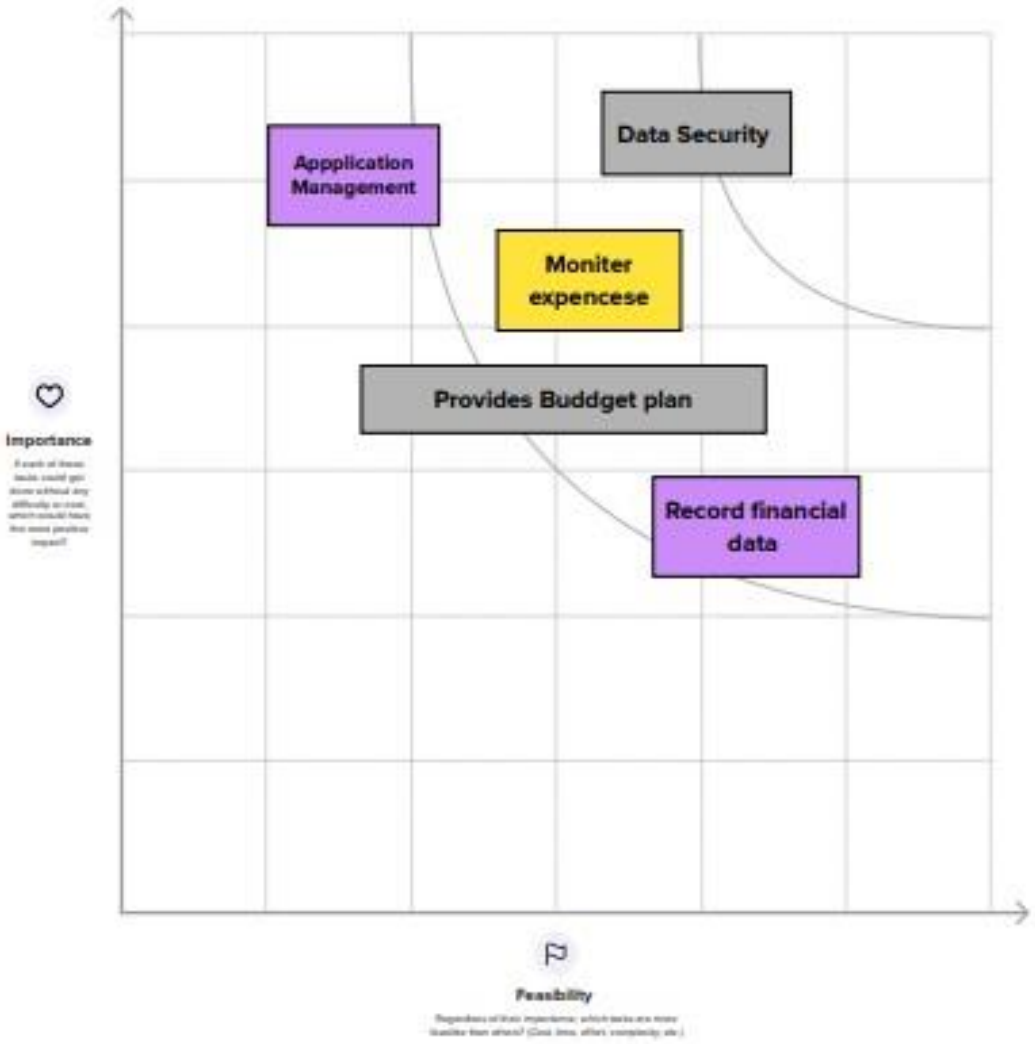


BRAINSTROMING,IDEA LISTING AND GROUPING





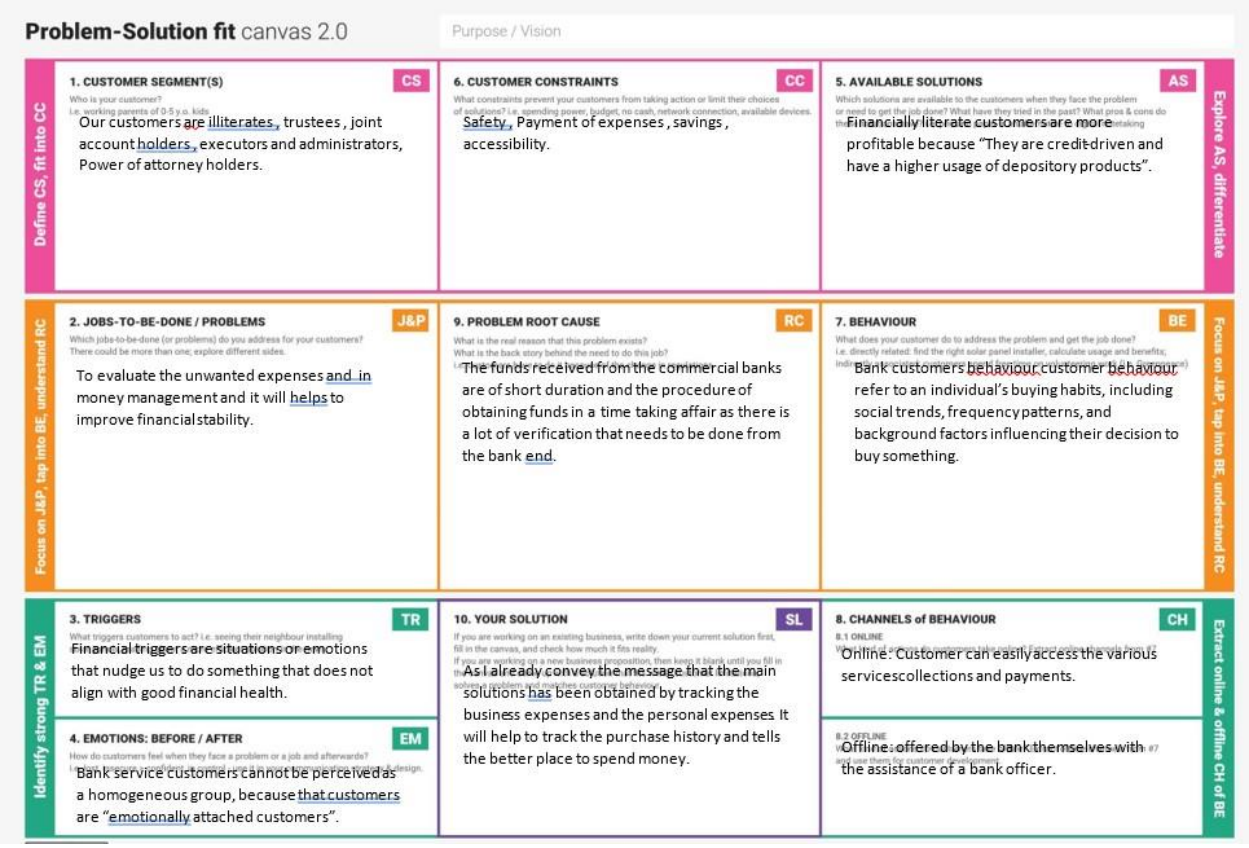
IDEA PRIORITIZATION:


Application Management
Data Security
Moniter expencese
Provides Buddget plan
Record financial data
Importance
Feasibility

Proposed Solu on Template:

S. No.	Parameter	Descrip on
1.	Problem Statement (Problem to be solved)	Personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management.
2.	Idea / Solu on descrip on	New ways to track the expenses/Better spending habits.
3.	Novelty / Uniqueness	Approval of expenditure in real me.
4.	Social Impact / Customer Sa sfac on	Take care of money that is spent/Helps in money management.
5.	Business Model (Revenue Model)	Planning to control the unwanted expenses.
6.	Scalability of the Solu on	Tracking of business and personal expenses.

d.Problem Solution fit :



Func onal Reqerments:

Sub Requirement (Story / Sub-Task)	Func onal Requirement (Epic)	FR No.
Registra on through Applica on Registra on through Gmail *to prove their identity*	User Registra on	FR-1

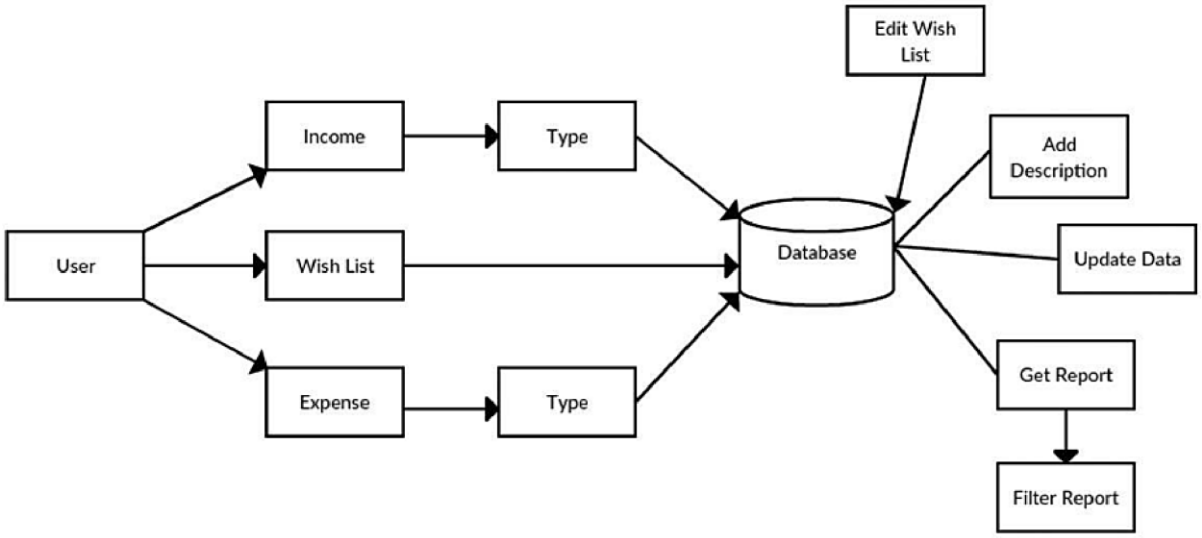
Confirma on via Email Confirma on via OTP	User Confirma on	FR-2
Data to be registered in the app	User monthly expense tenta ve data	FR-3
Data to be registered in the app	User monthly income data	FR-4
Alert through E-mail Alert through SMS	Alert/ No fica on	FR-5
Planning and Tracking of user expense vs budget limit	User Budget Plan	FR-6

Non-Func onal Requirement :

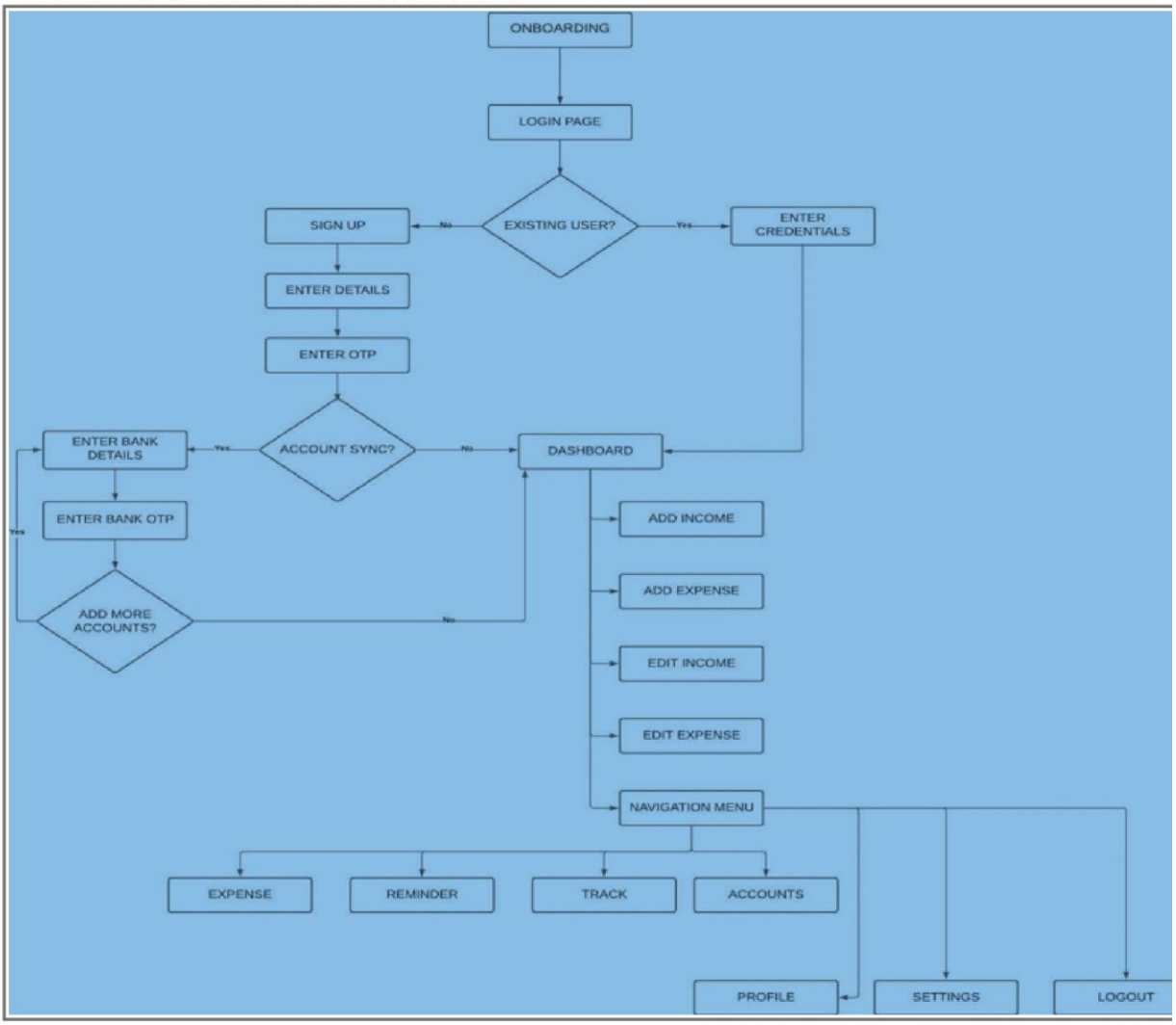
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Effectiveness, efficiency and overall satisfaction of the user while interacting with our application.
NFR-2	**Security**	Authentication, authorization, encryption of the application.
NFR-3	**Reliability**	Probability of failure-free operations 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**	Without near 100% availability, application reliability and the user satisfaction will affect the solution.
NFR-6	**Scalability**	Capacity of the application to handle growth, especially in handling more users.

5. PROJECT DESIGN

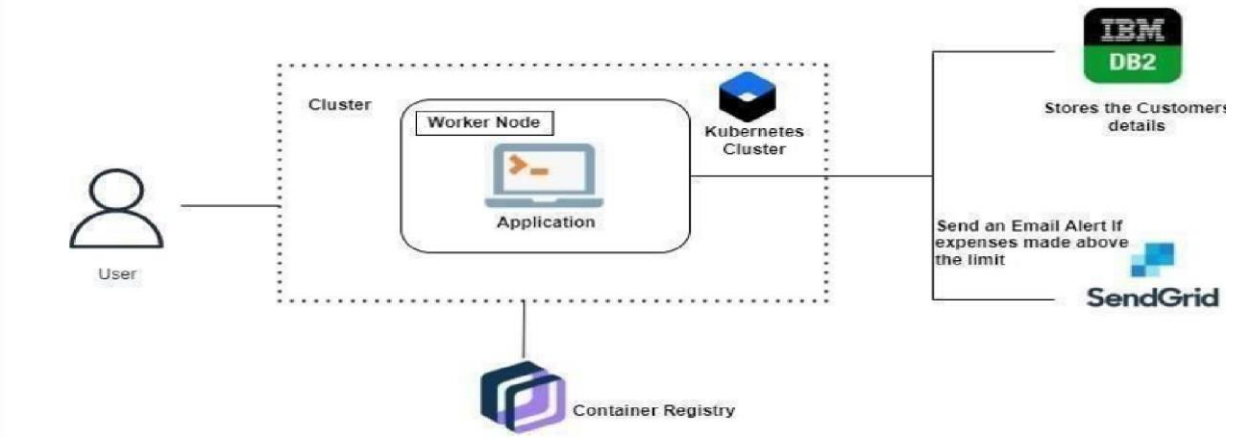
a.**DataFlow Diagram**



TECHNICAL ARCHITECTURE DIAGRAM



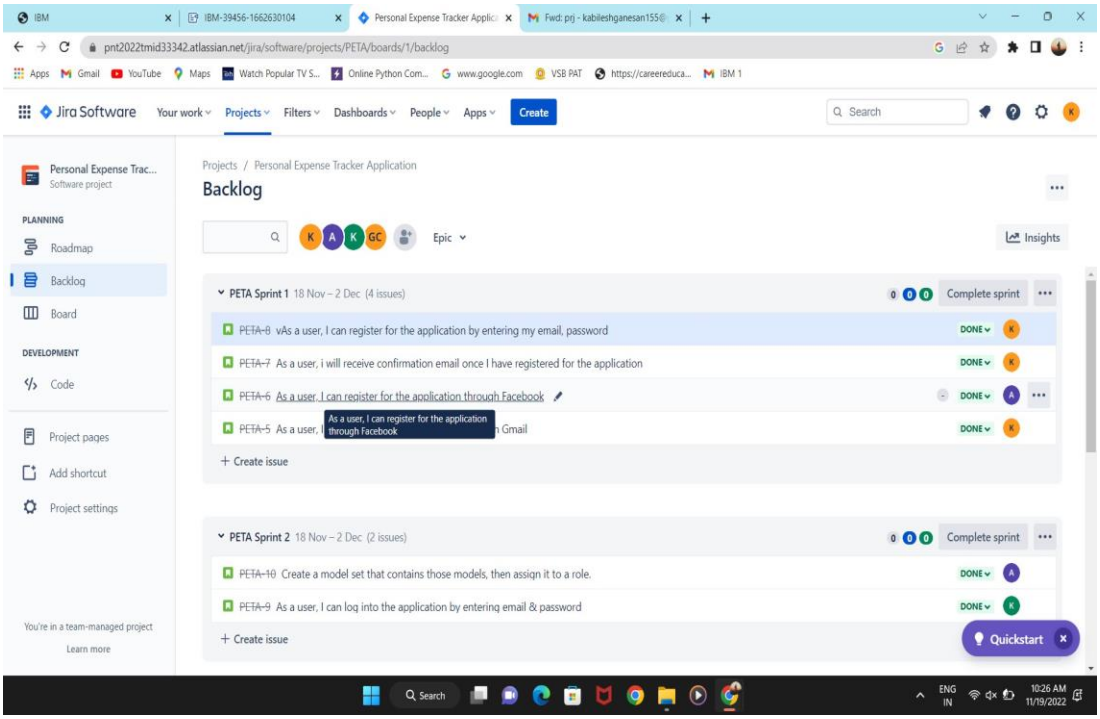
SOLUTION ARCHITECTURE DIAGRAM

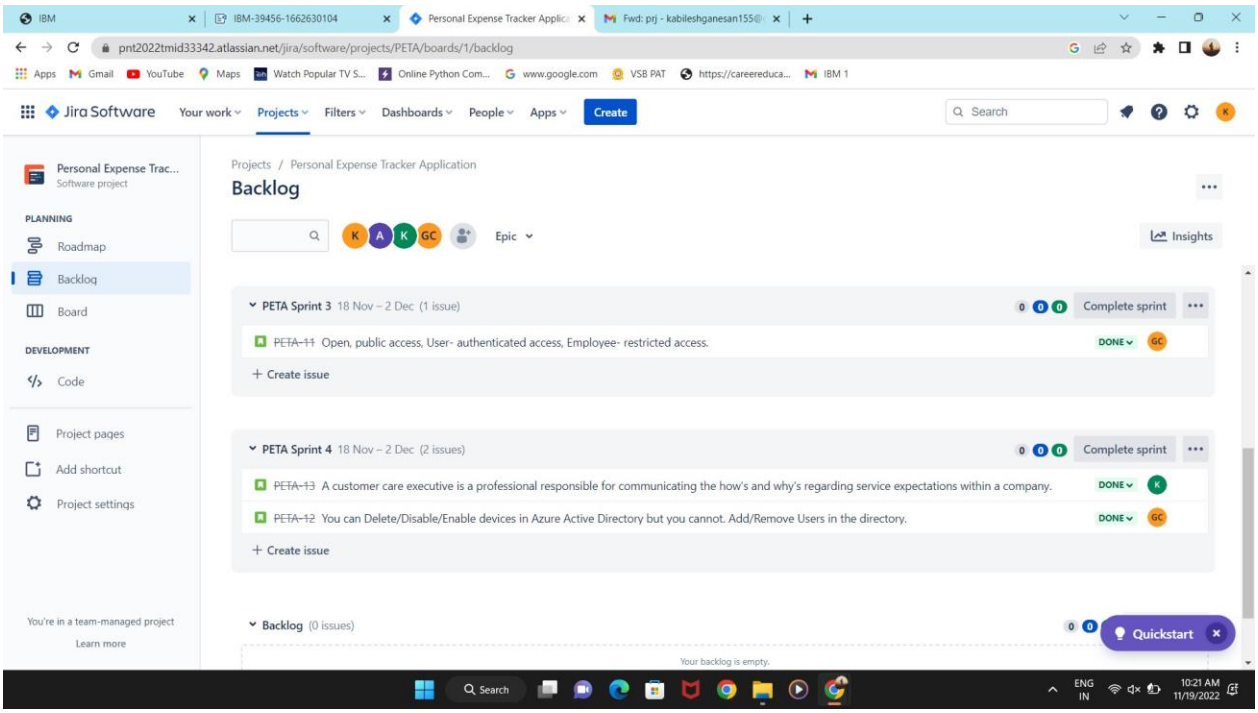


Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

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	ARAVINDHU
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	ARAVINDHU
Sprint-2		USN-3	As a user, I can for the application through Facebook	2	Low	ARAVINDHU
Sprint-1		USN-4	As a user, I can register for the application through G mail	2	Medium	KISHORE KUMAR
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	1	High	KISHORE KUMAR
Sprint -2	Dashboard	USN-6	As a user after logged in, I wished to see my wallet page.	1	Low	KISHORE KUMAR
Sprint-2		USN-7	As a user, I can add expense under expense page.	2	High	KISHORE KUMAR
Sprint-3	Backend	USN-8	As a developer, I need to create back end database for storing information.	1	High	GOWTHAM, KABILESH
Sprint-3		USN-9	As a developer, automate the mail to send alert when expense reach the limit.	1	Medium	GOWTHAM, KABILESH
Sprint-4	Containerization & Testing	UNS-10	As a developer, Need to container the project in the professional way to work every where.	2	High	GOWTHAM, KABILESH
Sprint	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Story Points**	**Priority**	**Team Members**
Sprint-4		USN-11	As a developer, test the project to check whether the project correctly work or not.	2	High	GOWTHAM, KABILESH





7. CODING & SOLUTIONING

app.py:

```
# -*- 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 sendemail import sendgridmail,sendmail  #
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'PNT2022TMID09631
"""
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=tcpi
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'])PNT2022TMID09631  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)
# sendmail("hello sakthi","sivasakthisairam@gmail.com")  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]PNT2022TMID09631  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 of Rs.
" + s + "/- !!!" + "\n" + "Thank you, " + "\n" + "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') PNT2022TMID09631
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)
```

deployment.yaml:

```
apiVersion: apps/v1  kind:
Deployment  metadata:
name: sakthi-flask-node-deployment
spec:
replicas: 1
selector:
matchLabels:  app:
flasknode
template:
metadata:  labels:
```

app: flasknode
spec: containers: -
name: flasknode
image: icr.io/sakthi_expense_tracker2/flask-template2
imagePullPolicy: Always ports:
- containerPort: 5000

flask-service.yaml:

apiVersion: v1 kind:
Service metadata:
name: flask-app-service
spec:
selector: app: flask-
app ports: - name:
http protocol: TCP
port: 80 targetPort:
5000 type:
LoadBalancer

manifest.yml:

applications:
- name: Python Flask App IBCMR 2022-10-19
random-route: true memory: 512M disk_quota:
1.5G **sendemail.py:**

```
import smtplib  import
sendgrid as sg  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("il.tproduct8080@gmail.com", "oms@1Ram")
s.login("tproduct8080@gmail.com", "lxixbmpnexbkiemh")
message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)  #
s.sendmail("il.tproduct8080@gmail.com", email, message)
s.sendmail("il.tproduct8080@gmail.com", email, message)
s.quit()
def sendgridmail(user,TEXT):
# from_email = Email("shridhartp24@gmail.com")  from_email
= Email("tproduct8080@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)
```

Database Schema

Tables :

1.Admin:

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,username VARCHAR(32) NOT NULL, email VARCHAR(32) NOT NULL,password VARCHAR(32) NOT NULL

2.Expense:

id INT NOT NULL GENERATED ALWAYS AS IDENTITY, userid INT NOT NULL, date TIMESTAMP(12) NOT NULL,expensename VARCHAR(32) NOT NULL, amount VARCHAR(32) NOT NULL,

8.TESTING:a.TestCases:

Acceptance Testing

UAT Execution & Report Submission

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	BUG ID
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button	1. Go to website 2. Enter Valid username and password	Username: Kavi password: 123456	Login/Signup popup should display	Working as expected	Pass	-	
Loginpage_TC_002	Functional	Home Page	Verify that the error message is displayed when the user enters the wrong credentials	1. Go to website 2. Enter Invalid username and password	Username: XXXX Password: 12345	Error message should displayed	Working as expected	Pass	-	
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup	1.Go to website 2.Enter valid credentials 3.Click Login	Username: Kavi password: 123456	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	-	
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials	1. Go to website 2. Enter details and click login	Username: Kavi password: 123456	User should navigate to user account homepage	Working as expected	Pass	-	
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials	1. Go to website 2. Enter details and click login	Username: Kavi password: 123456	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass	-	
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials	1. Go to website 2. Enter details and click login	Username: Kavi password: 123456	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass	-	
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with InValid credentials	1. Go to website 2. Enter details and click login	Username: Kavi password: 123456	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass	-	
AddExpensePage_TC_OO6	Functional	Add Expense page	Verify whether user is able to add expense or not	1. Add date, expense name and other details 2.Check if the expense gets added	add rent = 6000	Application adds expenses	Working as expected	Pass	-	

1. Purpose of Document

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	5
Client Application	51	0	0	50
Security	2	0	0	1
Outsource Shipping	2	0	0	4

1. Defect Analysis

This report shows the number of 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	1	4	2	3	20

Duplicate	1	2	3	0	4
External	2	0	0	1	6
Fixed	11	0	4	20	37
Not Reproduced	0	1	1	0	0
Skipped	10	0	1	1	2
Won't Fix	0	5	0	0	8
Totals	25	12	11	25	76

2. Test Case Analysis

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

Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9.RESULTS

a. Performance Metrics

i.

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

ii.

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

iii.

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

Payments & Invoices: Accept and pay from credit cards, debit cards, net banking, mobile wallets, and bank transfers, and track the status of your invoices and bills in the mobile app itself. Also, the trackingapp sendsremindersfor payments an d automatically matches the payments with invoices. v.

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

Ecommerce integration: Integrateyour expense trackingapp wit h your eCommerce store and track your sales through payments received via multiple payment methods. vii.

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

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

ix.

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

x.

Inventory tracking: An expense tracking app can do it all. Right from tracking products or the cost of goods, sending alert

notifications when the product is running out of stock or the product is not selling, to purchase orders. xi.

In-depth insights and analytics: Provides in-built tools to generate reports with easy-to- understand visuals and graphics to gain insights about the performance of yourbusiness. xii.

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

10. ADVANTAGES & DISADVANTAGES

1. **Achieve your business goals** with a tailored mobile app that perfectly fits your business.
2. **Scale-up** at the pace your business is growing.
3. Deliver an **outstanding** customer experience through additional control over the app.
4. Control the **security** of your business and customer data
5. Open **direct marketing channels** with no extra costs with methods such aspush notifications.
6. **Boost the productivity** of all the processes within theorganization.
7. Increase **efficiency** and **customer satisfaction** with an app aligned to their needs.
8. **Seamlessly integrate** with existing infrastructure.
9. Ability to provide **valuable insights**.
10. Optimize sales processes to generate **more revenue** through enhanced data collection.

11. **CONCLUSION**

From this project, we are able to manage and keep tracking the daily expenses as well as income. While making this project, we gained a lot of experience of working as a team. We discovered various predicted and unpredicted problems and we enjoyed a lot solving them as a team. We adopted things like video tutorials, text tutorials, internet and learning materials to make our project complete.

12. FUTURE

The project assists well to record the income and expenses in general. However, this project has some limitations:

1. The application is unable to maintain the backup of data once it is uninstalled.
2. This application does not provide higher decision capability.

To further enhance the capability of this application, we recommend the following features to be incorporated into the system: 3. Multiple language interface.

4. Provide backup and recovery of data.
5. Provide better user interface for user.
6. Mobile apps advantage.

13. APPENDIX

Source Code Github Link : https://github.com/IBM-EPBL/IBM-Project-39456-1660449676

Project Demo Link:

https://drive.google.com/file/d/10PyHe3N0yuHxhQYexnMwqS4lO3Nohm8/view?usp=sharing