PERSONAL EXPENSE TRACKER APPLICATION

IBM-Project-38331-1660378717

NALAYATHIRAN PROJECT

BASED LEARNING ON PROFESSIONAL READINESS FOR INNOVATION, EMPLOYMENT AND ENTERPRENEURSHIP

A PROJECT REPORT BY

SIVASANKARI S(922519104150)

REVATHI P(922519104129)

SRI SRUTHI R(922519104159)

SUWETHA M(922519104169)

TEAM ID : PNT2022TMID33429

INDUSTRY MENTOR : KUSBOO

FACULTY MENTOR :GEETHA S

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

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 AND DISADVANTAGES.

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub and Project Demo Link

1.INTRODUCTION

a. Project Overview:

People today are worried about the consistency of their daily expenses. This is done mostly to maintain track on-users' daily expenses so that monthly expenses can be controlled.We have built an android application titled as “Expense Tracker Application” and this application is used to manage the user‘s daily spending in a more logical and manageable method. This application will assist us in keeping track of spending and reducing the need for manual estimates of daily expenses. This application allows the user to compute his daily total expenses, and the results are saved for each individual user. We must keep the Excel sheets, Word documents, notes, and files for the user's daily and monthly spending, just like the conventional techniques of budgeting. There isn't a fully functional way to easily keep track of our daily expenses. Maintaining a log in diary is a very boring activity that can occasionally become problematic owing to the manual calculations. Considering everything mentioned above, we aim to fulfil the user's needs, requirements by designing an android application which will assist them in lightening their loads. “Expense Tracker Application" is an application that lets the user enter their daily expenses at the end of the day, and to evaluate their expenses.

b.Purpose:

Track and prioritise their extensive range of spending using a customer app will boost productivity and customer happiness. The spending tracking app creates and sends reports that provide accurate data about revenues,

expenditures, budgets, income, balance sheets, etc. provides built-in capabilities to produce reports with understandable graphics and visualisations to acquire insights into the operation of your organisation.

2. LITERATURE SURVEY

a.Existing Problem:

The web application “Expense Tracker” is developed to manage the daily expenses in a more efficient and manageable way. By using this application we can reduce the manual calculations of the daily expenses and keep track of the expenditure. In this application, user can provide his income to calculate his total expenses per day and these results will be stored for each user. The application has the provision to predict the income and expense for the manager using data mining. In this application, there are 3 logins such as admin, manager and staff. Admin has the privilege to add, edit, delete manager, add, edit, delete staff, and to get all custom reports. For Manager, the privileges are to add type of expense, verify expense, add type of income, verify income and generate reports. For staff, the privileges are to add and edit expense, income and calculations, and send for verifications.

b. References:

Manchanda A. (2012). Expense Tracker Mobile Application (Doctoral dissertation, San Diego State University).

c. Problem Statement Definition:

In essence, we can see that today everybody looks to their investment funds. They need that in this period of expansion how they have some control over their expenses and set aside cash for some time later. Clearly, they are looking towards such an application which guides them how they can meet their expenses in a proficient way.

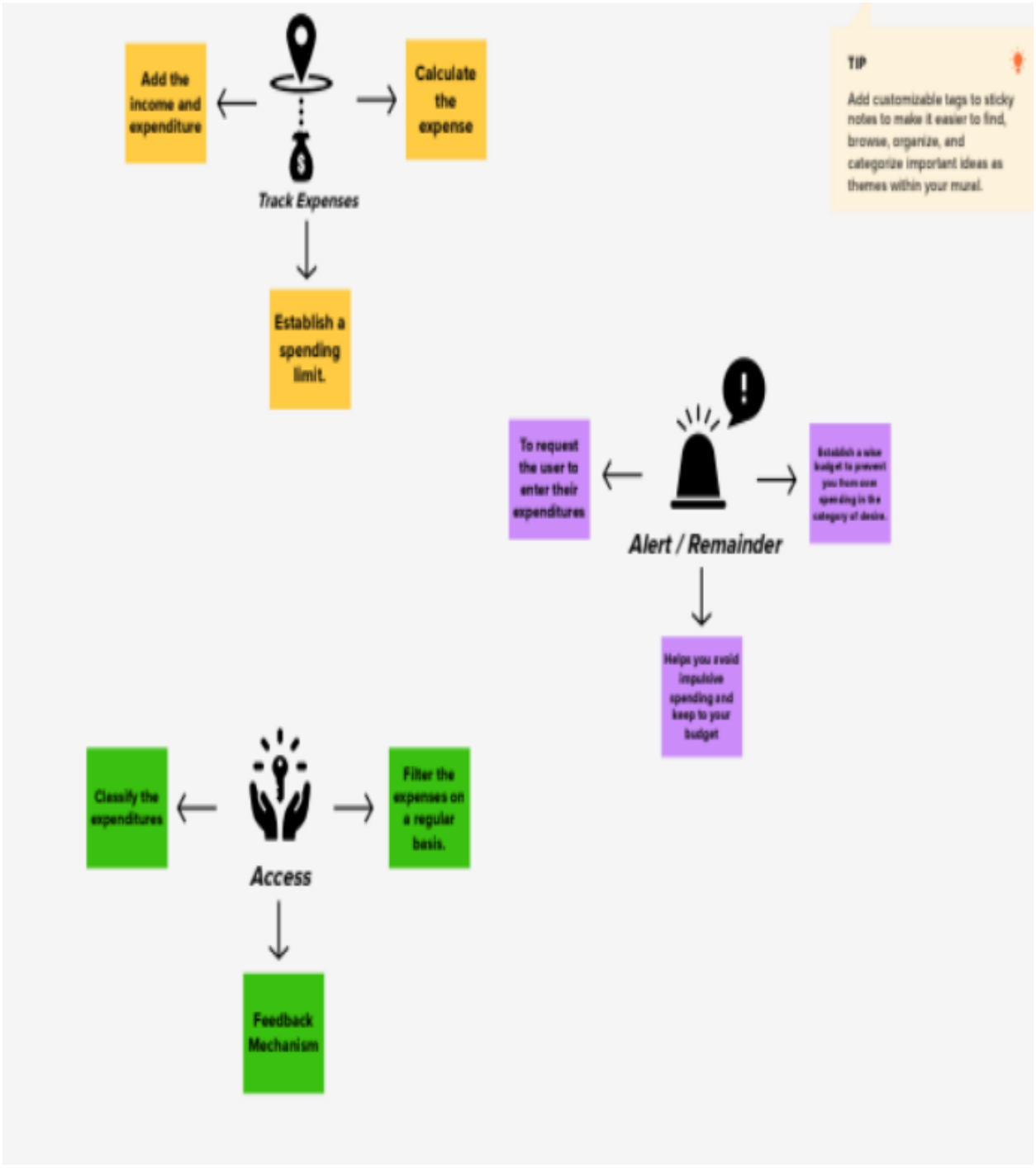
I am	Ms. Sivasankari, a busy employee at xyz company.
I'm trying to	I am trying to Check my daily expense but I was unaware on my expenditure and I don't have enough time to calculate my budget.
But,	Because calculating expenditure manually was uncomfortable for me, I was unable to find the time to calculate it.
Because	I don't feel confident on calculating my expenditure accurately in my daily expenses.
Which makes me feel	By using this application I can maintain my budgets by comparing the previous histories.

Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS-1	Busy Employee	Calculate expenses through application	Because calculating manually was uncomfortable for me.	I don't feel confident on my calculation.	Doubtful, insecure
PS-2	Busy Employee	One must track their expenses to maintain their budgets.	I was unable to find the time to track and compare the previous month expenditure.	I don't have enough time to calculate it.	To me and many around me, doing manual expense tracking is difficult.

3. IDEATION & PROPOSED SOLUTION

a. Empathy Map Canvas

b. Ideation and Brainstorming


Add the income and expenditure
Calculate the expense
Track Expenses
Establish a spending limit.
TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.
To request the user to enter their expenditures
Alert / Remainder
Helps you avoid impulsive spending and keep to your budget
Classify the expenditures
Filter the expenses on a regular basis.
Access
Feedback Mechanism


Pictorially filter the expenses
Add the income and expenditure
Calculate the expense
Add a second source of income
Establish a spending limit.
Establish a wise budget to prevent you from over spending in the category of desire.
Generate Monthly report
Constraints on the budget
Obtain monthly reports in spreadsheet or pdf format.
Classify the expenditures
Feedback Mechanism
Helps you avoid impulsive spending and keep to your budget
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

c. Proposed Solution

For user convenience, this project is being developed on android applications.Because they include an android application anytime they create immediate expenses. One of the biggest problems with keeping personal spending is that We frequently have no idea where the money for everyday expenses goes. Some of the traditional approaches used to address this issue under typical conditions include the use of sticky notes by common users, the use of spread-sheets by proficient persons to track expenses, and the maintenance of huge amounts of information by experts only using ledgers. As this shows that it is various methods used by different people. This makes using this data contrary. There is still complication in areas like there is no assurance for data compatible, there are chances of crucial inputs can be missed and the manual errors may sneak in. The Data recorders are not always handled, and it could be hectic process to have an overall view of those expenses. We think that a practical design and a practical android application can solve these problems. Such an application is capable of keeping track of expenditures, providing a comprehensive view with user-friendly interface, and being enough intelligence to display the history of expenditures indicated in the application.

d. Problem Solution Fit

4.REQUIREMENT ANALYSIS

a. Functional Requirements

Functional Requirements:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration and Confirmation	Form for collecting details and registration can also be done through Gmail Confirmation through Gmail or OTP
FR-2	Login	Enter username and password
FR-3	Calendar	Personal expense tracker application shall allow user to add the data to their expenses.
FR-4	Expense Tracker	This application should graphically represent the expense in the form of report.
FR-5	Report generation	Graphical representation of report must be generated through Message and Gmail
FR-6	Category	This application shall allow users to add categories of their expenses.

b. Non-Functional Requirements

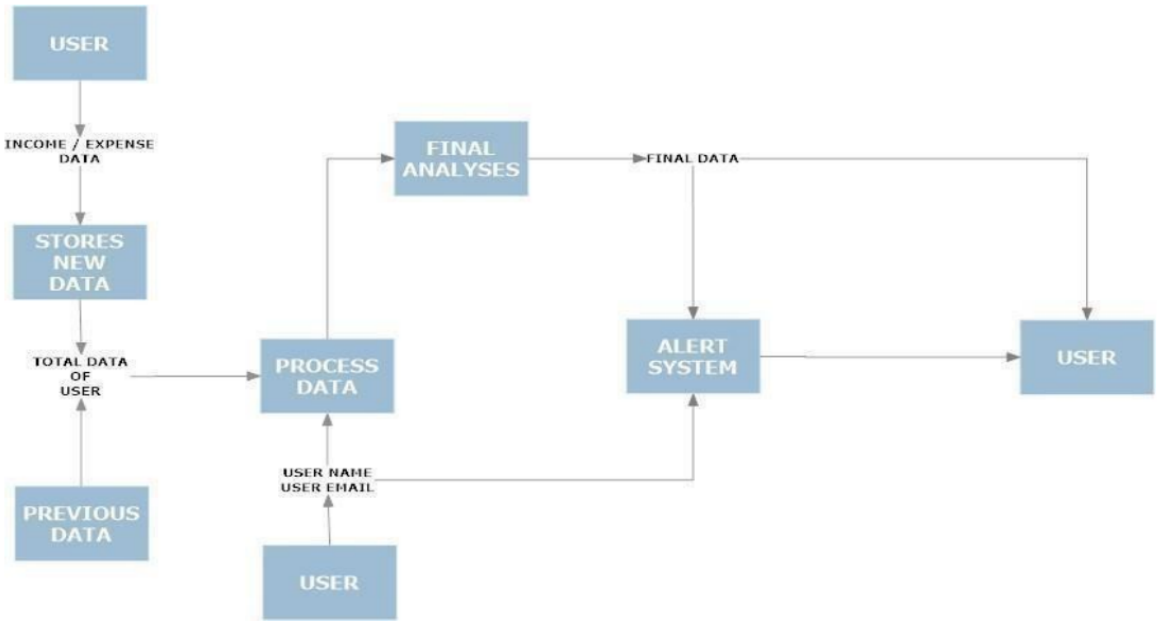
Non-functional Requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Keeps an accurate record of your earnings and outgoings.
NFR-2	**Security**	Apps for tracking spending are thought to be very secure against online criminals.
NFR-3	**Reliability**	Each data record is kept on an effective database structure that is effectively developed. No chance of data loss exists.
NFR-4	**Performance**	Expense kinds include categories and an option. Lightweight database support increases the system's throughput.
NFR-5	**Availability**	The application must have a 100% up-time.
NFR-6	**Scalability**	The capacity to effectively manage escalating demands.

5. PROJECT DESIGN

a. Data Flow Diagrams

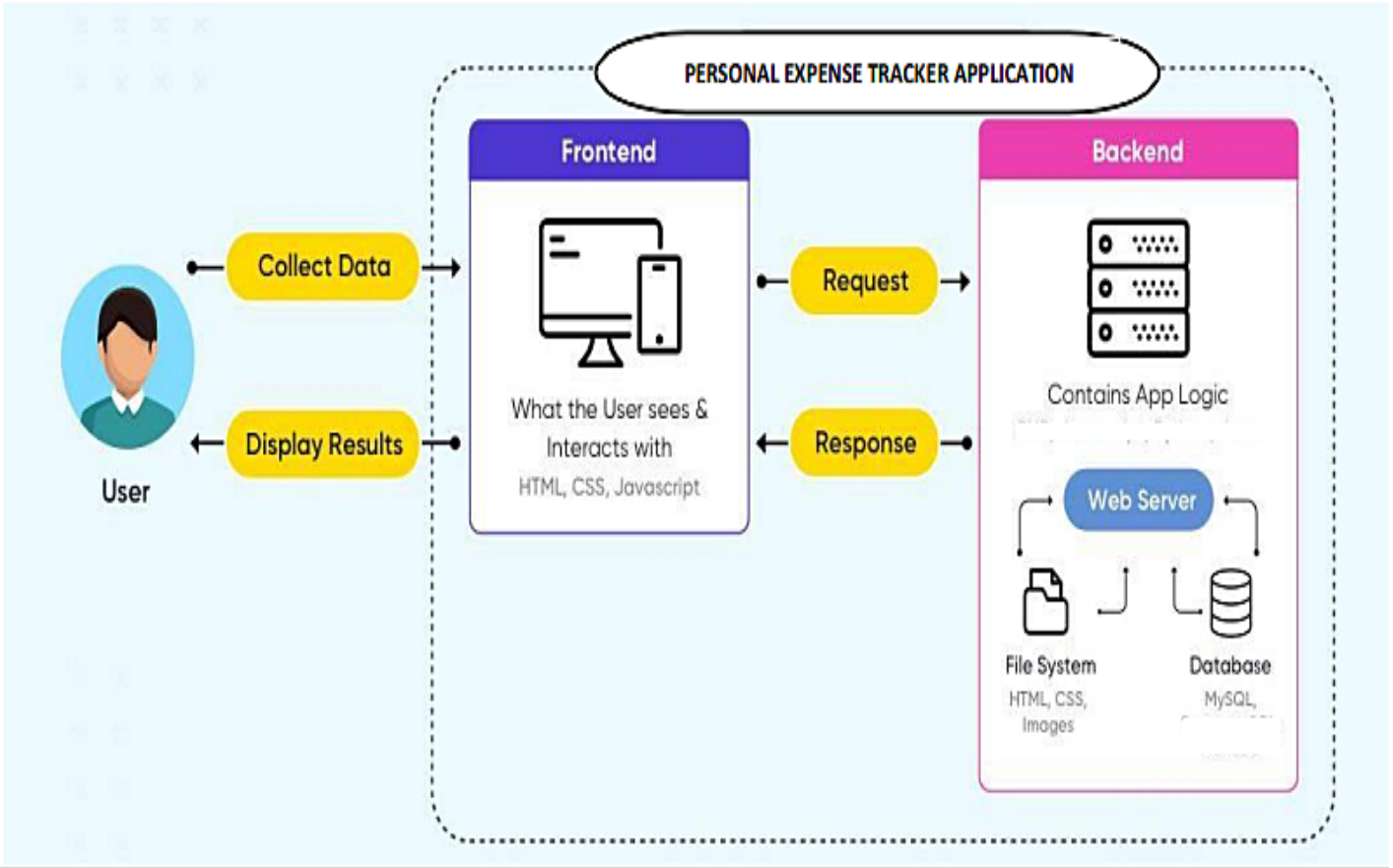
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

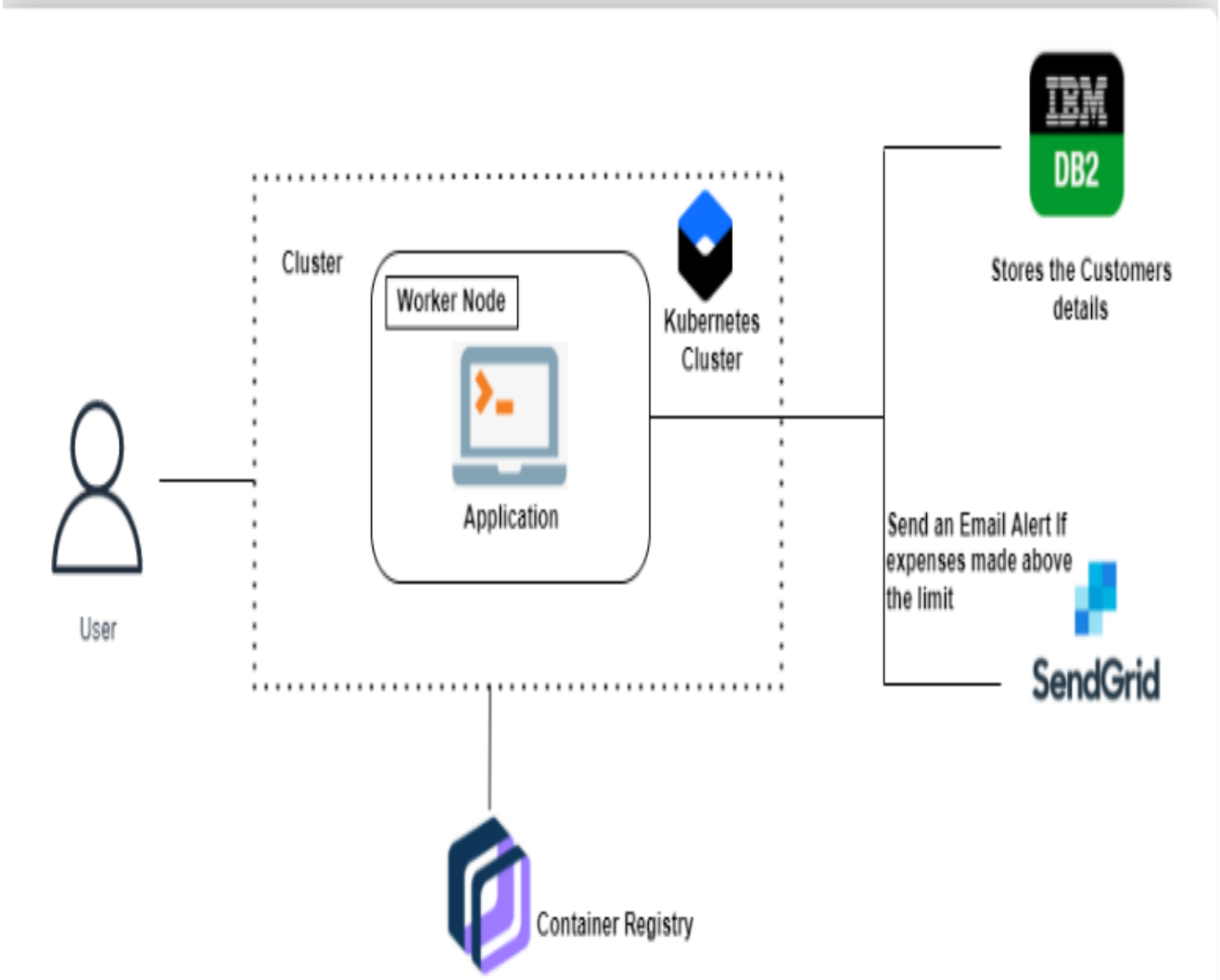


b. Solution and Technical Architecture

Solution architecture, which has numerous sub-processes, is a complicated process that connects business issues with technological solutions. It intends to:

- ➤ Find the best technological solution to address current company issues.
- ➤ Describe to project stakeholders the software's structure, traits, behavior, and other features.
- ➤ Define the project's features, development stages, and solution needs.
- ➤ Give instructions on how the solution should be defined, managed, and delivered to clients.




IBM
DB2
Stores the Customers details
Cluster
Worker Node
Kubernetes Cluster
Application
User
Send an Email Alert If expenses made above the limit
SendGrid
Container Registry

c. User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user & web 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	
		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	
		USN- 3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	
	Login	USN - 4	As a user, I can log into the application by entering email & password	I can access the application	High	
	Dashboard	USN - 5	As a user I can enter my income and expenditure details.	I can view my daily expenses	High	
Customer Care Executive		USN – 6	As a customer care executive I can solve the log in issues and other issues of the application.	I can provide support or solution at any time 24*7	Medium	
Administrator	Application	USN - 7	As a administrator I can upgrade or update the application.	I can fix the bug which arises for the customers and users of the application	Medium	

6. PROJECT PLANNING AND SCHEDULING

a. Sprint Planning and Estimation

Sprint 1

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	Sivasankari
		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Revathi
	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Sri Sruthi
	Dashboard	USN-4	Logging in takes to the dashboard for the logged user.	`2	High	Suwetha
Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only						

Sprint 2

Sprint 2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Sivasankari

	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Revathi
	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Sri Sruthi
		USN-4	Making dashboard interactive with JS	2	High	Suwetha

Sprint 3

Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Sivasankari
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user's query	1	Medium	Revathi
	SendGrid	USN-3	Using SendGrid to send mail to the user about their expenses	1	Low	Sri Sruthi
		USN-4	Integrating both frontend and backend	2	High	Suwetha

Sprint 4

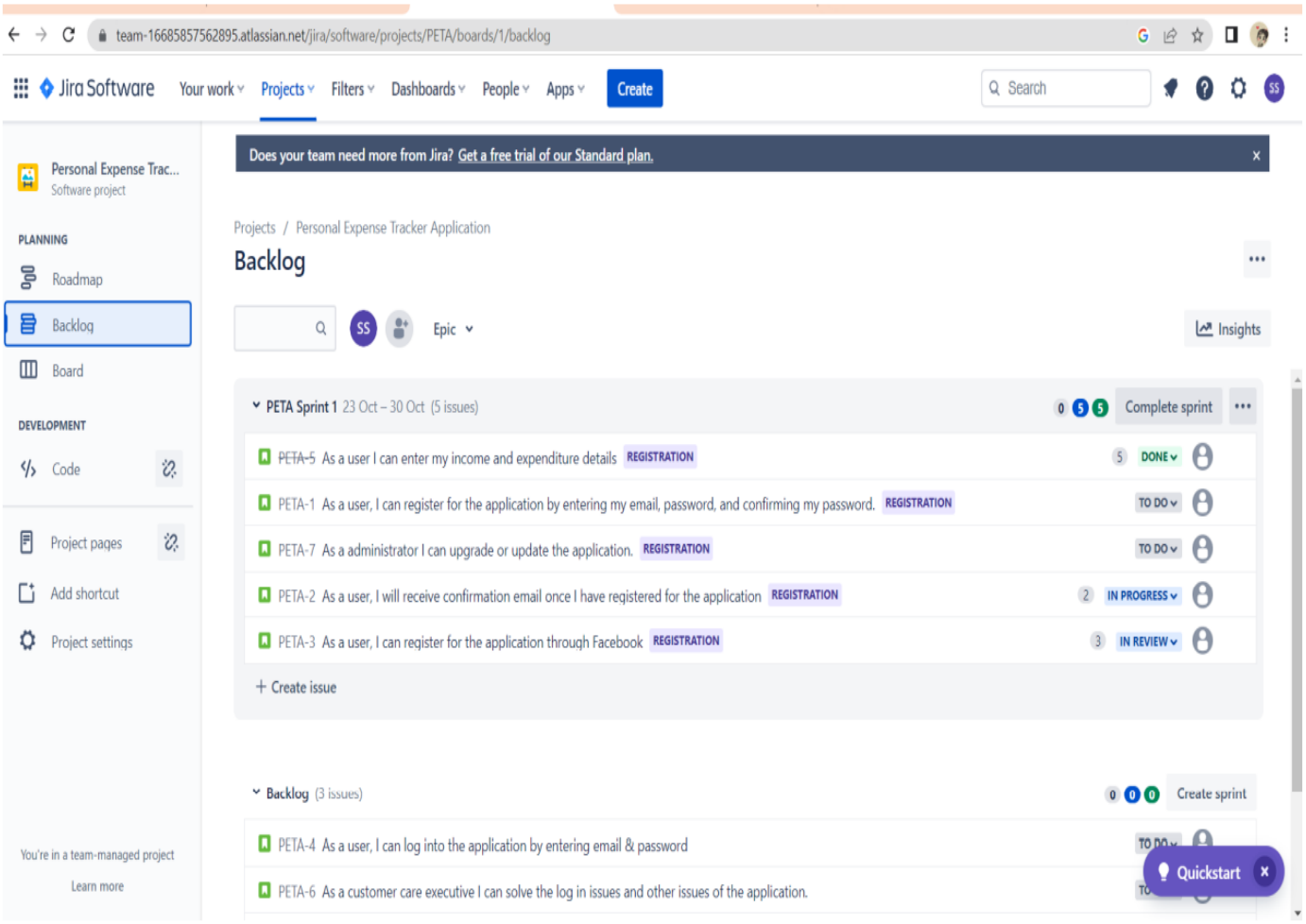
Sprint-4	Docker	USN-1	Creating image of website using docker/	2	High	Sivasankari
	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Revathi
	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Sri Sruthi
	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Suwetha

b. Sprint Delivery Schedule

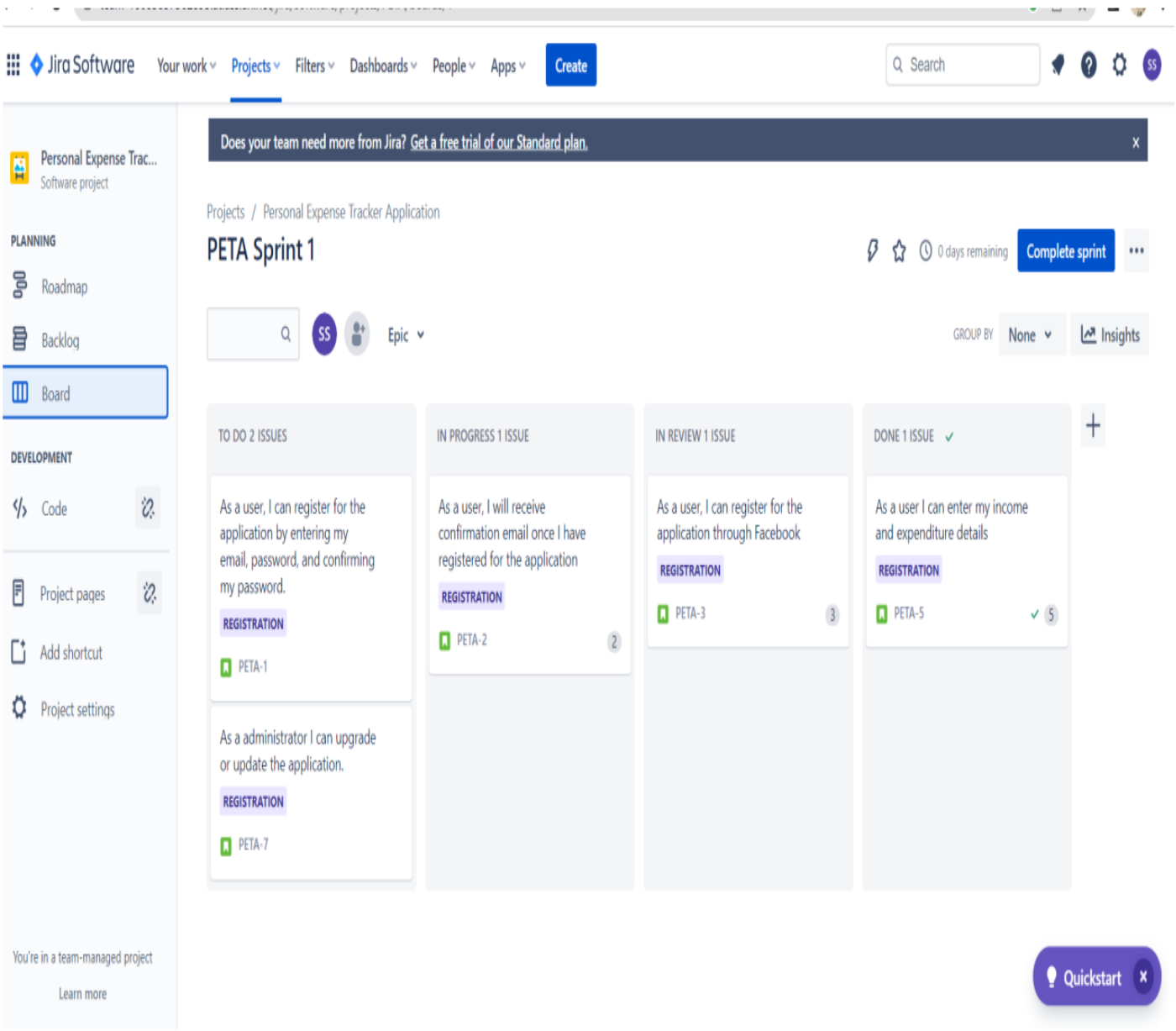
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	23 Oct 2022	28 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	30 Oct 2022	04 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	06 Nov 2022	11 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	13 Nov 2022	18 Nov 2022	20	19 Nov 2022

c. Report from JIRA

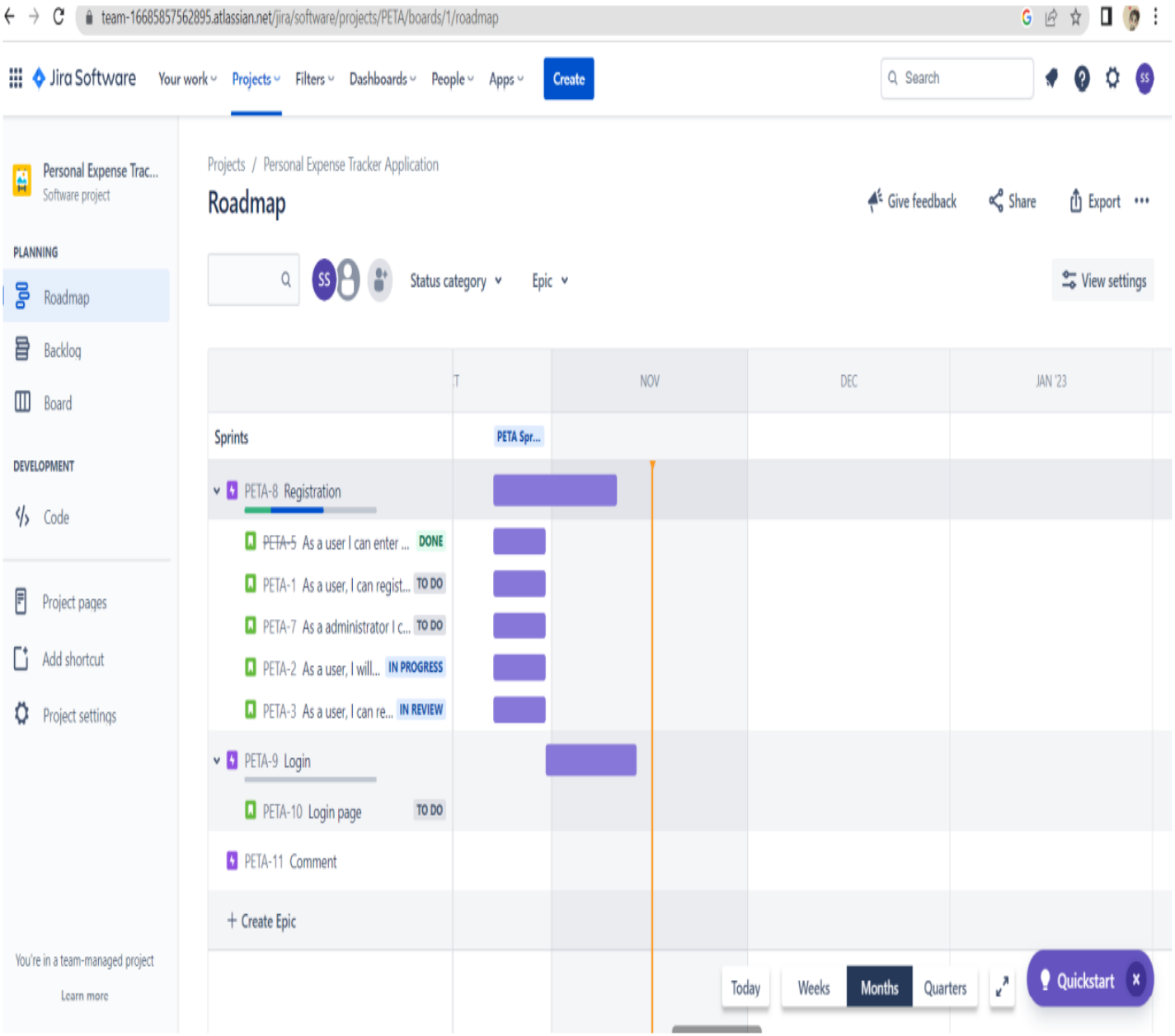
i. Backlog



ii. Board


Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Personal Expense Trac...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Does your team need more from Jira? Get a free trial of our Standard plan.
Projects / Personal Expense Tracker Application
PETA Sprint 1
0 days remaining
Complete sprint
Epic
GROUP BY
None
Insights
TO DO 2 ISSUES
As a user, I can register for the application by entering my email, password, and confirming my password.
REGISTRATION
PETA-1
As a administrator I can upgrade or update the application.
REGISTRATION
PETA-7
IN PROGRESS 1 ISSUE
As a user, I will receive confirmation email once I have registered for the application
REGISTRATION
PETA-2
2
IN REVIEW 1 ISSUE
As a user, I can register for the application through Facebook
REGISTRATION
PETA-3
3
DONE 1 ISSUE
As a user I can enter my income and expenditure details
REGISTRATION
PETA-5
5
Quickstart

iii. Roadmap


team-16685857562895.atlassian.net/jira/software/projects/PETA/boards/1/roadmap
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Personal Expense Trac...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / Personal Expense Tracker Application
Roadmap
Give feedback
Share
Export
Status category
Epic
View settings
NOV
DEC
JAN '23
Sprints
PETA Spr...
PETA-8 Registration
PETA-5 As a user I can enter ... DONE
PETA-1 As a user, I can regist... TO DO
PETA-7 As a administrator I c... TO DO
PETA-2 As a user, I will... IN PROGRESS
PETA-3 As a user, I can re... IN REVIEW
PETA-9 Login
PETA-10 Login page TO DO
PETA-11 Comment
+ Create Epic
Today
Weeks
Months
Quarters
Quickstart

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'



"""

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.appd
omain.cloud;port=31498;protocol=tcpi p;\
PNT2022TMID09631

 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)

 # 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]


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')

 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()
```

PNT2022TMID09631

```
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,
paymode VARCHAR(32) NOT NULL,
category VARCHAR(32) NOT NULL

3.LIMIT

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,userid VARCHAR(32) NOT NULL, limit VARCHAR(32) NOT NULL

8.TESTING:

a.TestCases:

Test case ID	Feature Type	Component	Test Scenerio	Steps To execute	Test Data	Expected Result	Actual Result	Status	Comment	BUG ID	Executed By
Login Page_TC_001	Functional	Home Page	Verify User is able to see the Loginpage up popup When User clicked on my Account button	1.Go to Website 2.Enter valid Username and Password	Username:Sree Password:123456	Login/Signup popup Should display	Working as Expected	Pass	.		Sivasankari
Login Page_TC_002	Functional	Home Page	Verify that the error message is diplayyed when the user enters thewrong credentials	1.Go to Website 2.Enter invalid Username and Password	Username:XXX Password:123456	Error Message should displayed	Working as Expected	Pass	.		Revathi
Login Page_TC_002	UI	Home Page	Verify the UI elements inLoginSignup Page	1.Go to Website 2.Enter Valid credentials 3.Login Page	Username:sree Password:123456	Application should Show below UI elements: a.Email textbox b.Password textbox c.Login button with orange colour d.New customer?Create Account linke.last password?Recovery Password link	Working as Expected	Pass	.		Sri Sruthi

						button with orange colour d.New customer?Cr eate Account linke.last password?R ecovery Password link					
Login Page_TC_ 003	Functio nal	Home Page	Verify user is able to loginto application with valid credentials	1.Go to Website 2.Enter Details and click login	Usernam e:sree Password :123456	User should navigate to User account homepage	Worki ng as Expec ted	Pas s	-		Suwetha
Login Page_TC_ 004	Functio nal	Login Page	Verify user is able to loginto application with invalid credentials	1.Go to Website 2.Enter Details and click login	Usernam e:sree Password :123456	Application should show incorrect email or password validation message	Worki ng as Expec ted	Pas s	-		Sivasank ari
Login Page_TC_ 004	Functio nal	Login Page	Verify user is able to loginto application with invalid credentials	1.Go to Website 2.Enter Details and click login	Usernam e:sree Password :123456	Application should show incorrect email or password validation message	Worki ng as Expec ted	Pas s	-		Revathi
Login Page_TC_ 005	Functio nal	Login Page	Verify user is able to loginto application with invalid credentials	1.Go to Website 2.Enter Details and click login	Usernam e:sree Password :123456	Application should show incorrect email or password validation message	Worki ng as Expec ted	Pas s	-		Sri Sruthi

Login Page_TC_006	Functional	Add Expense Page	Verify Whether the user is able to add Expense or not	1.Add expense name,Date and other details 2.Check if the expense get added	add rent=6000	Application add expenses	Working as Expected	Pass	-		Suwetha

b. User Acceptance Testing

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	10	4	2	8	15
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	9	2	4	11	20
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	0	1	8
Totals	22	14	11	22	51

2. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Interface	7	0	0	7
Login	20	0	0	20
Logout	2	0	0	2
Limit	3	0	0	3
Signup	8	0	0	8
Final Report Output	4	0	0	4

9.RESULTS

a. Performance Metrics

i. Tracking income and expenses: Monitoring the income 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 and 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,

vi. Ecommerce integration: Integrateyour expense trackingapp with your eCommerce store and track your sales through payment 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-38331-1660378717

Project Demo Link:

https://drive.google.com/file/d/1eU4KT-lWO9D_iB4YQJeRNZ49FcwuytyF/view?usp=sharing