PERSONAL EXPENSE TRACKER APPLICATION

NALAIYA THIRAN PROJECTBASED LEARNING ON PROFESSIONAL READLINESS FOR INNOVATION, EMPLOYNMENT AND ENTERPRENEURSHIP

A PROJECT REPORT
BY

PRABHA C (713519CEIT026**)**
MANOJ KUMAR R (713519CEIT017**)**
SRIRAM A (713519CEIT038**)**
RAMYA (713519CEIT028**)**

B. TECH.INFORMATION TECHNOLOGY

SNS COLLEGE OF ENGINEERING-641107

INDEX

11. FUTURE SCOPE

12. APPENDIX

    a. Source code

    b. GitHub & Project Demo Link

TEAM ID: PNT2022TMID07838

INDUSTRY MENTOR: KUSHBOO

FACULTY MENTOR: ASHOK 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

i. 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 portionof 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

i. Personal finance management is an important part of people's lives. How ever, everyone does not have the knowledge or time to manage theirfinances in a proper manner. And, even if a person has time and knowledge, they do not bother with tracking their expenses as they findit tedious and time-consuming. Now, you don't have to worry about managing your expenses, as you can get access to an expense trackerthat will help in the active management of your finances. Also known as expense manager and money manager, an expense tracker a software or application that helps to keep an accurate record of yourmoney 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.

ii. People tend to overspend without realizing, and this can prove to be disastrous. Using a dailyexpense manager can help you keep track of how much you spend every day and on what. At theend do the month, you will have a clear picture where your money is going. 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 tohelp them out. Before you decide to go in for a money manager, it is important to decide the type you want.

2. **Literature Survey**

Intelligent Online Budget Tracker:

The development of this application has been conducted in a stepwise manner using the well-defined methodology, RUP, customized according to the requirements of the system. Most of the goals set at the start of the development phase have been met. Security problems like web security or network security have also been treated in the design and development of the system, thus increasing the reliability of the system. Quality management issues have also been handled satisfactorily.

Online Income and Expense Tracker:

This project is work more efficient than the other income and expense tracker. The project successfully avoids the manual calculation for calculating the income and expense per month. The modules are developed efficiently and also in an attractive manner.

Family Expense Manager Application:

As the result, the user can make use of this application in his/her daily life. After being used it can be a part of daily life to update and view daily expenses and family expenses. This helps to keep track of expenses & manage it for the user as they are busy in their daily routine, they are not able to keep track of their incomes & expenses.

Personalized Expense Managing Assistant Using Android:

Some of the features are like enabling users to register to the application using an existing email or social network account, it will synchronize the user's profile information to the application. Apart from this, the application can be used to gather samples of data related to user's expenses with consents and use those sample data as parameters to assess patterns of spending. Using some data mining techniques expenses can be classified and can be used in market analysis and planning.

Mobiwik Expense Tracking Application:

Mobiwik came up with a new feature in their app called Expense Manager. With this feature, you can track and manage your expenditures(expenses), savings, reminders and bill payments. This is a personal budget management app that tracks your expenditures and income and gives you recommendations to make you economically strong. The main idea of developing this feature for giving users a clear picture that how much they are spending and where they are spending and when. We remind them to pay their utilities and card bills before the due date by using the same platform in just one tap, instead of going any other way. Also serving them by giving saving tips for their good future investment.

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, from http://www.geocities.com/CollegePark/Library/3954/svghbes.pdf
2. Central Statistics Office. (2001). Household budget survey. Government of Ireland. Retrieved August 15, 2006, from http://www.cso.ie/releasespublications/documents/housing/hbs.pdf
3. European Countries. (2004). Household budget surveys in candidate countries: Methodological analysis 2003. European Countries. Luxembourg. Retrieved February 19,2007, http://europa.eu.int/estatref/info/sdds/en/hbs/hbs_meth2003_cand_countries.pdf
4. International Research Journal of Engineering and Technology (IRJET)
5. https://ww.researchgate.net/publication/237448489_Intelligent_Online_Budget_Tracker (Bekaroo, Girish & Sunhaloo, Sameer. (2007). Intelligent Online Budget Tracker.)
6. https://www.irjet.net/archives/V6/i3/IRJET-V6I31110.pdf
7. M N Rajaprabha 2017 IOP Conf. Ser.: Mater. Sci. Eng. 263 042050
8. https://easychair.org/publications/preprint/73S7
9. https://medium.com/@rajotiya.ravi2/case-study-of-expense-tracking-app-get-daily-alerts-of- your-expense-a0561526973d

3. PROBLEM STATEMENT

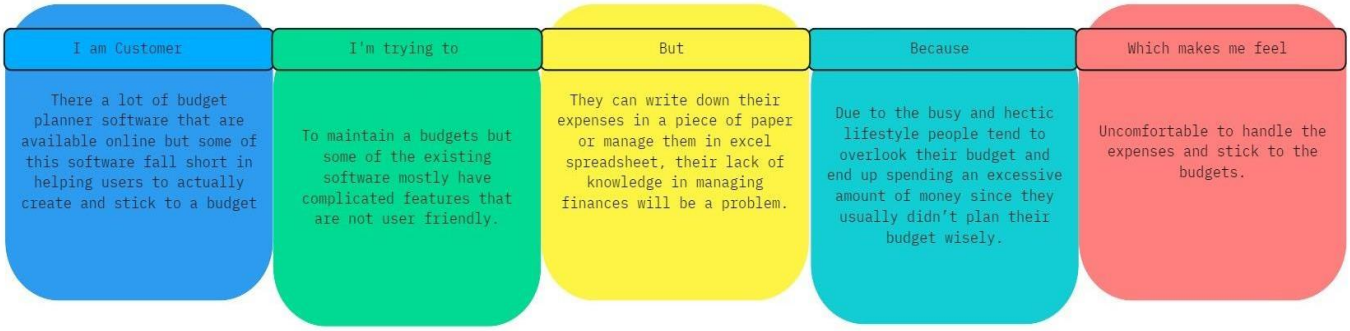
Customer Problem Statement Template:

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people willlove.

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they

I am	Describe customer with 3-4 key characteristics - *who are they?*	Describe the customer and their attributes here
I'm trying to	*List their outcome or "job" the care about - what are they trying to achieve?*	List the thing they are trying to achieve here
but	Describe what problems or barriers stand in the way – *what bothers them most?*	Describe the problems or barriers that get in the way here
because	Enter the "root cause" of why the problem or barrier exists – *what needs to be solved?*	Describe the reason the problems or barriers exist
which makes me feel	Describe the emotions from the customer's point of view – *how does it impact them emotionally?*	Describe the emotions the result from experiencing the problems or barriers

perceive your product or service.



4. CUSTOMER PROBLEM STATEMENT:

Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to helps teams better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants

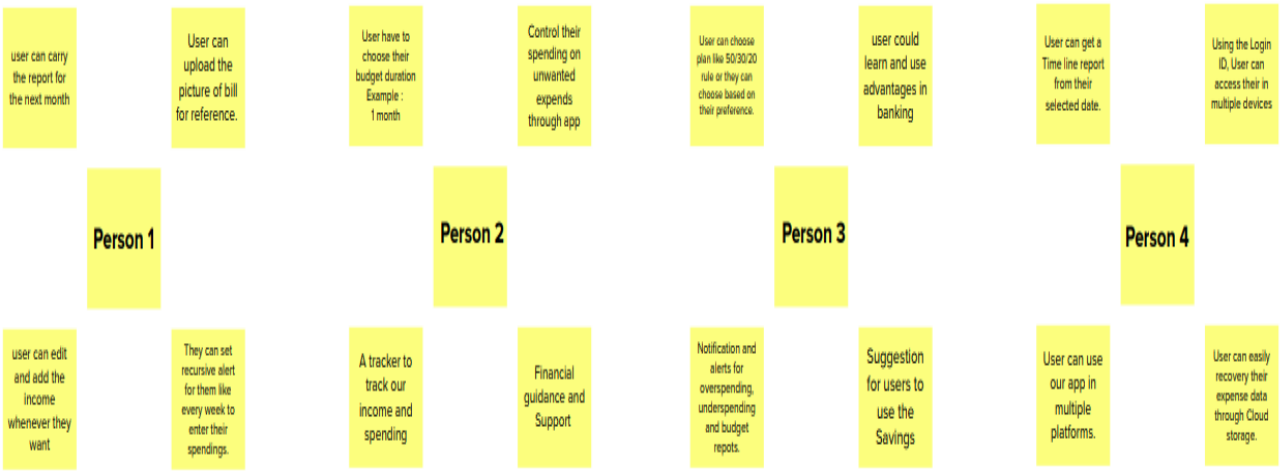
consider things from the user's perspective along with his or her goals and challenges.

Brainstorm & Idea Prioritization Template:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

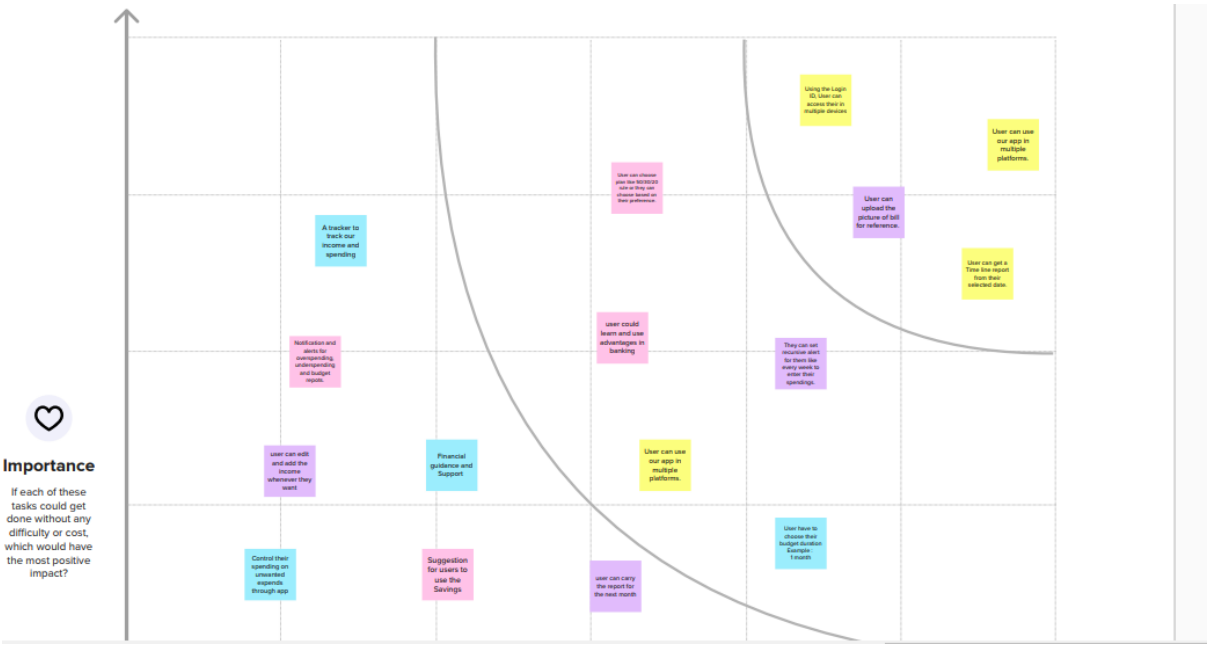
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Brainstorm:




user could learn and use advantages in banking
user can carry the report for the next month
Financial guidance and Support
User can choose plan like 50/30/20 rule or they can choose based on their preference.
User have to choose their budget duration Example : 1 month
User can use our app in multiple platforms.
user can edit and add the income whenever they want
User can get a Time line report from their selected date.
User can upload the picture of bill for reference.
Notification and alerts for overspending, underspending and budget repots.
User can easily recovery their expense data through Cloud storage.
They can set recursive alert for them like every week to enter their spendings.
A tracker to track our income and spending
Suggestion for users to use the Savings
Control their spending on unwanted expends through app
Using the Login ID, User can access their in multiple devices

Idea Prioritization:



Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Keeping Proper track of our daily expenses is becoming challenging in today's world. Without the proper money management knowledge people overspend on their wants instead of focusing on their needs. Especially when using online applications for purchasing their requirements consumers tend to over spend. This problem leads to improper distribution of their daily expenses. Without proper knowledge on managing money poor are becoming poorer and rich are becoming richer.
2.	Idea / Solution description	An attempt to develop an app to manage our daily expenses and give us insights on managing our money would be a good idea. This app will be able to track expenses on various online platforms and apps. The app can help with proper budgeting and give alerts when the user over spends or crosses the limit previously set by them. This will lead to proper spending habits and make them knowledgeable about money management. IBM cloud can be used to handle the data safely.
3.	Novelty / Uniqueness	The speciality for the app will be the data security with IBM cloud being used for data storage and this app genuinely helps with the money management. The proper and personalized budgeting of the user's money leads them to trust the app and they wouldn't have to worry about their expenditure on unnecessary things.

4.	Social Impact / Customer Satisfaction	People using the app will be becoming better at their spending habits and will be able to save more than their peers who are not using the app. This application aims to improve the users' savings sustainably and steadily which leads them to trust the app without worrying about their money.
5.	Business Model (Revenue Model)	This application leads to a business model, the user can be suggested the right products to buy based on their budget and this can lead to targeted business approaching the right consumers. The model leads to systematic and structured expenses of the user and also leads to predictive analysis of the future expenses of the consumer. This model makes the user more careful with expenses as they are provided with the money management insights.
6.	Scalability of the Solution	This application can be created as a multi user model nationwide. The model can also be modified based on the country's law on applications and data security which leads to international implementation of this application by maintaining proper gateway rules. This app when developed for multiple nations can be modified to their requirements. The app can also be modified for a particular group of people or organization.

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?

Predominantly Engineers who are just starting to earn and manage their personal finance. Typically from middle and lower class family, who badly need financial discipline .

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions?

The impulse buying and lacking to awareness to look into bigger picture

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem

Totally shunning to spend even on necessities under the impression that the spending could result in bad financial position.

The existing solutions are otherwise over complicated and designed to extract data from user.

Manual physical logging in time consuming

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

- Logging expenses into categories
- Show historical stats
- Generate insightful charts
- Alert user to imbibe good discipline

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?

Lack of proper education in financial literacy in school education. More children are not given pocket money to learn by spending/wasting less / saving.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?

Get frustrated and fall into debt traps by taking unpayable loans for unnecessary items leading to increase in mental stress

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

Functional Requirements:

Following are the functional requirements of the proposed solution.

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

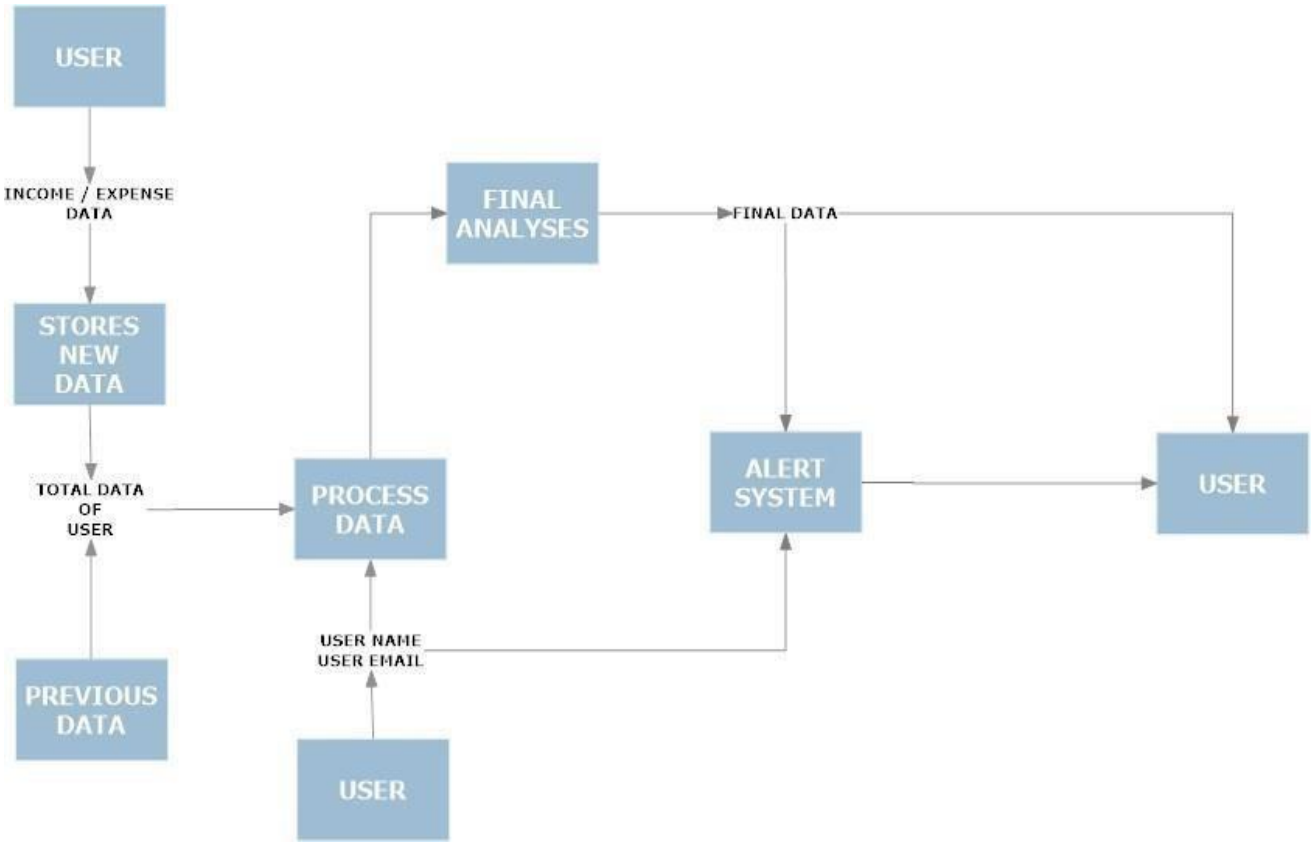
Non-functional Requirements:

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

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.

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



User Stories

Use the below template to list all the user stories for the product.

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 emailonce 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	

SPRINT DELIVERY PLAN:

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	14 Oct 2022	20 Oct 2022	20	21 Oct 2022
Sprint-2	20	6 Days	22 Oct 2022	28 Oct 2022	20	29 Oct 2022
Sprint-3	20	6 Days	30 Oct 2022	05 Nov 2022	20	06 Nov 2022
Sprint-4	20	6 Days	7 Nov 2022	13 Nov 2022	20	14 Nov 2022

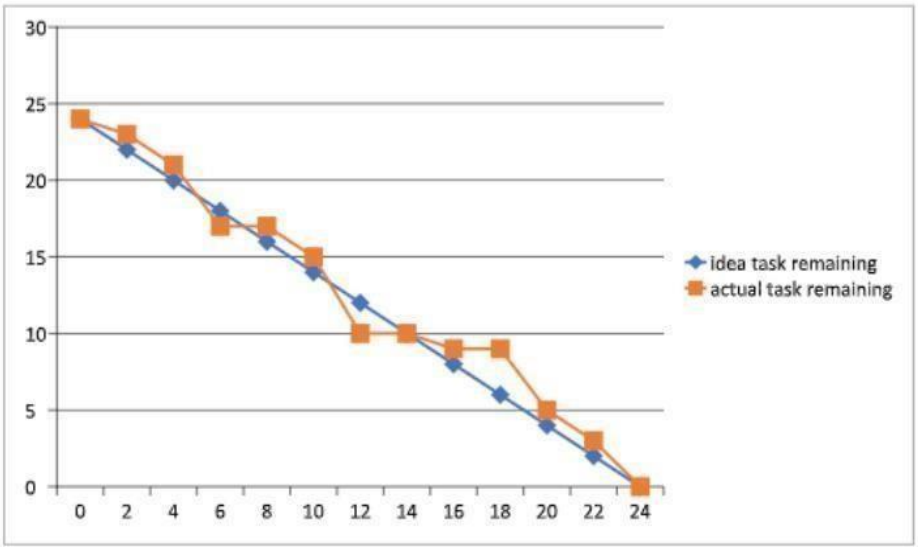
Velocity:

Imagine we have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

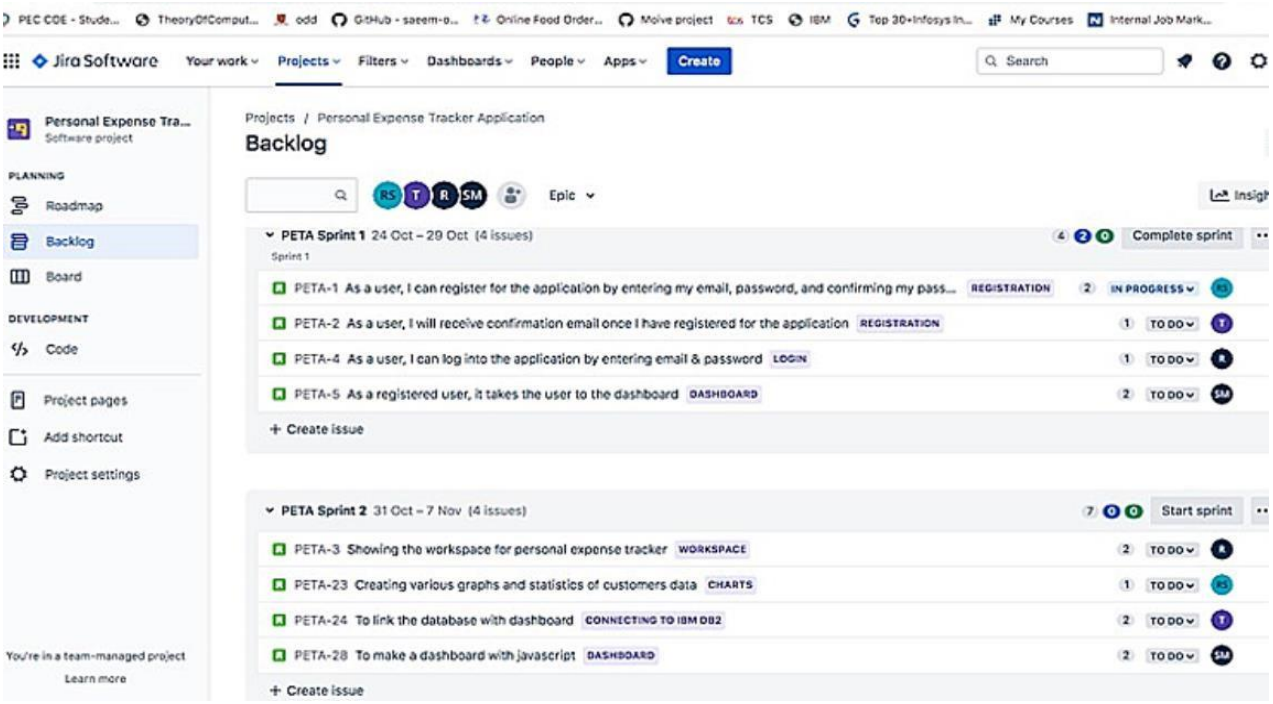
$$AV = \text{sprint duration} / \text{velocity} = 20/6 = 3.3$$

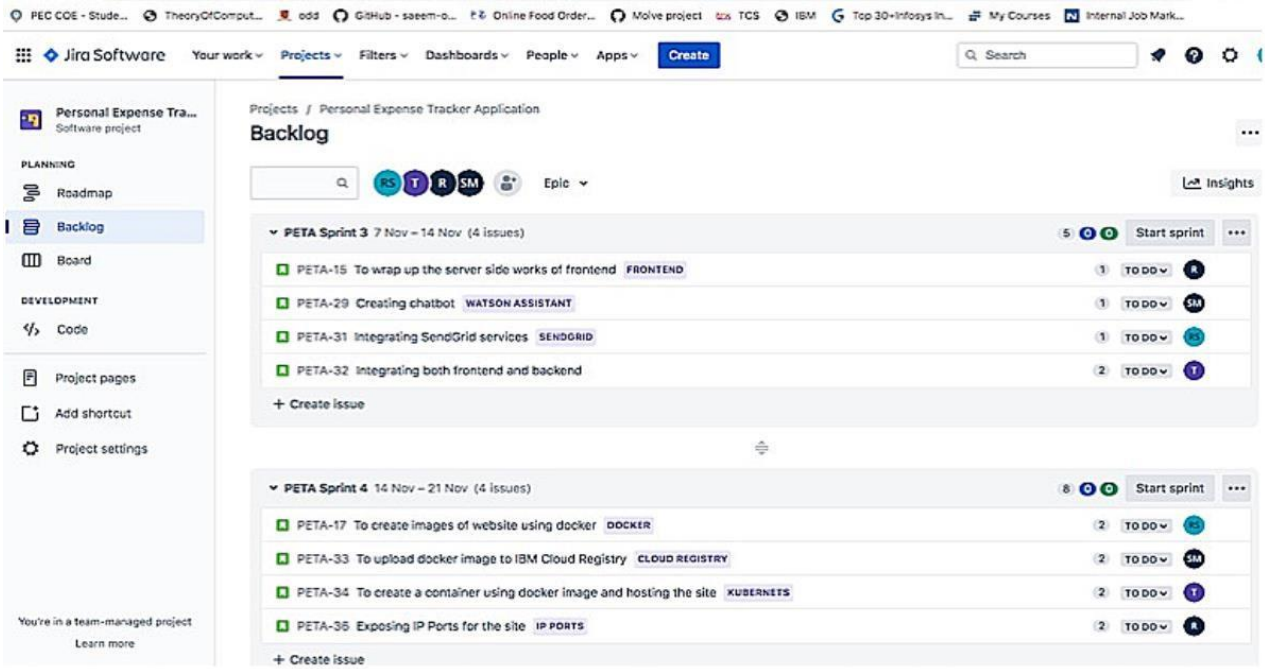
Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologiessuch as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

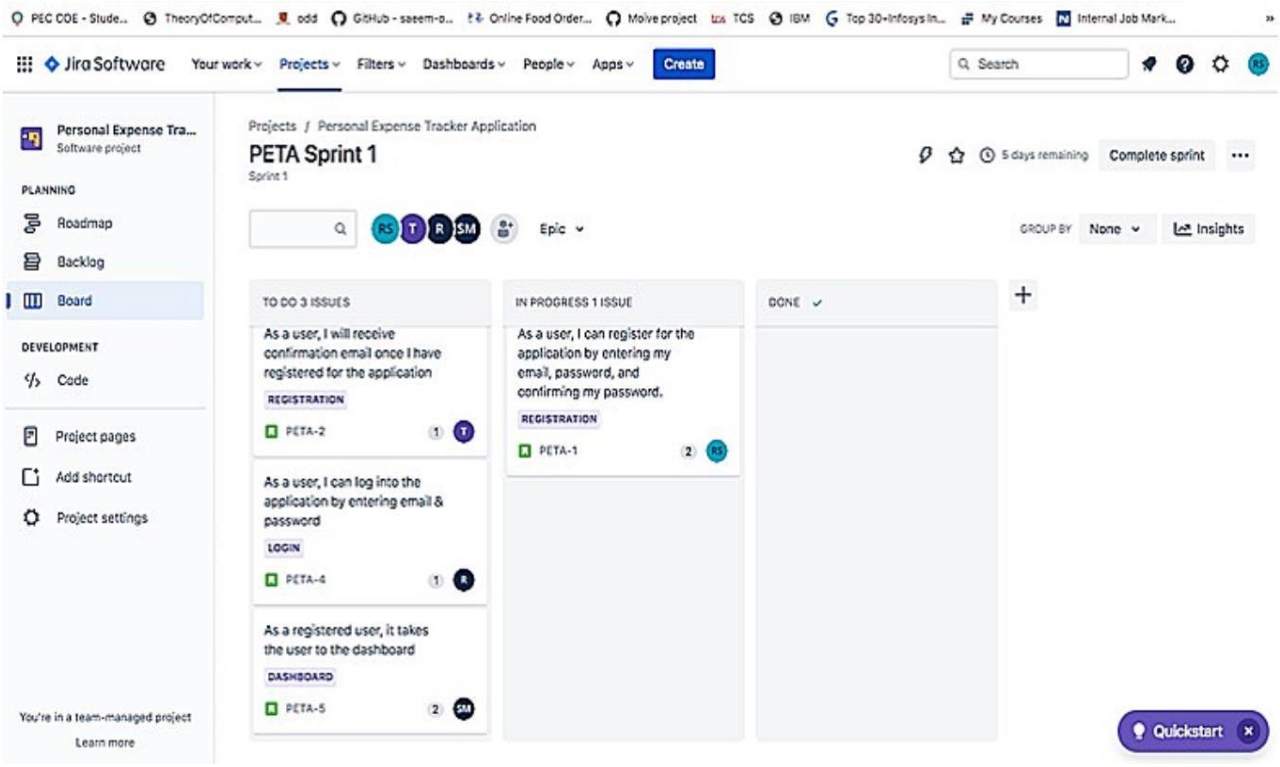


Reports from Jira:

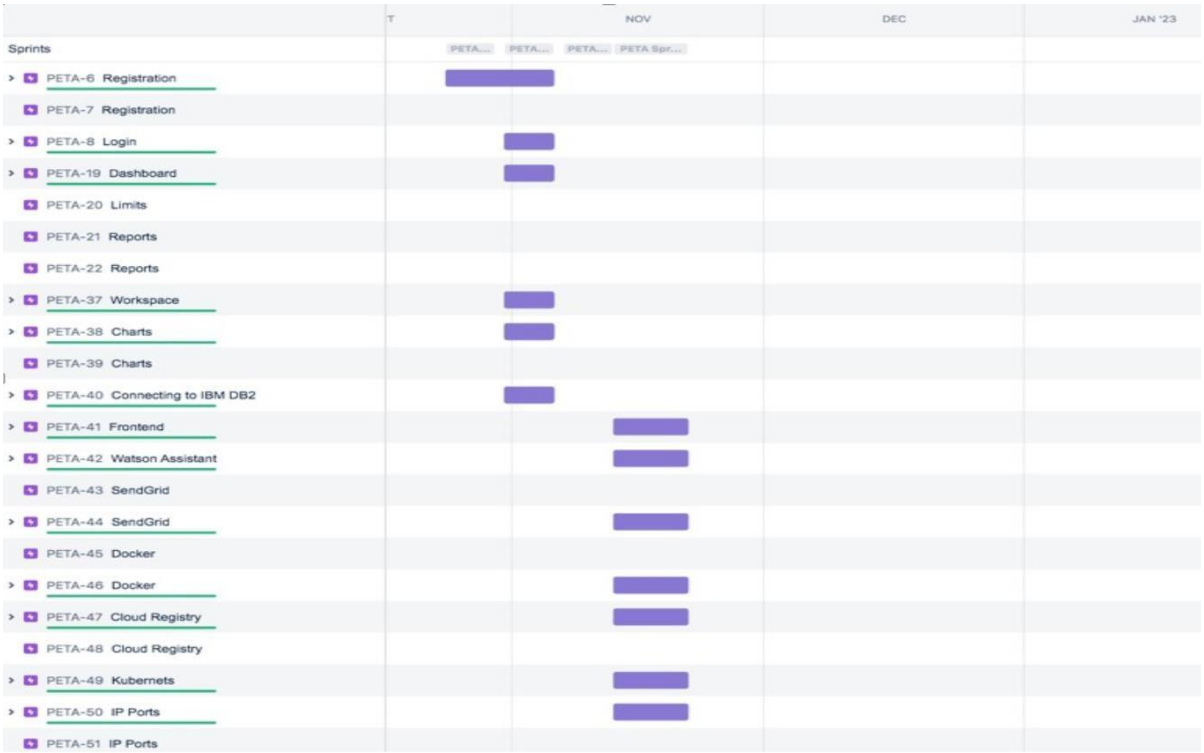




i. Board



i. Road Map



1. 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.cur
sorsimport re

from
flask_db2
import DB2
import
ibm_db
import ibm_db_dbi
from sendemail import sendgridmail,sendmail

# from gevent.pywsgi import
WSGIServerimport 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.appdomai
n.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.appdomai

n.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;protoco
l=tcpi p;\
            uid=sbb93800;pwd=wobsVLm6ccFxcNLe;securit
    y=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.r
oute("/h
ome")
def
home():
    return render_template("homepage.html")


@
a
p
p.
ro
ut
e(
"/
```

```
")
d
ef
a
d
d(
):
    return render_template("home.html")



#SIGN--UP--OR--REGISTER



@app.ro
ute("/sig
nup")def
signup():
    return render_template("signup.html")



@app.route('/register', methods =['GET', 'POST'])
d
    e
    f
    r
    e
    g
    i
    s
    t
    e
    r
    (
    )
    :
```

```
m
s
g
=
'
'
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(s
tmt)print(result)
account =
ibm_db.fetch_row(st
mt)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.ro
ute("/sig
nin")def
signin():
    return render_template("login.html")
```

```
@app.route('/login',methods
=['GET', 'POST'])def login():
    g
    l
    o
    b
    a
    l
    u
    s
    e
    r
    i
    d
    m
    s
    g
    =
    '
    '



    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(st
        mt)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['logg
            edin'] = True
            session['id'] =
            dictionary["ID"]
            userid =
            dictionary["ID"]
            session['username'] =
            dictionary["USERNAME"]
            session['email'] =
            dictionary["EMAIL"]


            return
        redirect('/ho
        me')else:
            msg = 'Incorrect username / password !'


    return render_template('login.html', msg = msg)
#ADDING ---DATA
```

```
@app
.route
("/add
")def
addin
g():
    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
    =
    dat
    e[0
    :10
    ]
    p2
    =
    dat
    e[1
    1:1
    3]
    p3
```

```
    =
    dat
    e[1
    4:]
    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
    dictionar
    y !=
    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
      i
      n
      e
      x
      p
      e
      n
      s
      e
      :
      t
      o
      t
      a
      l
      +
      =
      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)
    r
    o
    w
    =
    [
    ]
    s
    =
    0
    while
        dictionar
        y !=
        False:
        temp = []
        temp.append(dictionary["LI
        MITSS"])
        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['em
                ail'])


    return redirect("/display")
```

```
#DISPLAY---graph


@app.rou
te("/displ
ay")def
display():
    print(session["username"],session['id'])


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


    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + "
ORDERBY date DESC"
    res =
    ibm_db.exec_immediate(ibm_db_conn,
    param)dictionary =
    ibm_db.fetch_assoc(res)
    expense = []
    while
        dictionar
        y !=
        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
    = " + idres =
    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 = " + idres =
    ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    while
```

```
        dictionar
        y !=
        False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"
        ])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENS
        ENAME"])
        temp.append(dictionary["AMOUN
        T"])
        temp.append(dictionary["PAYMO
        DE"])
        temp.append(dictionary["CATEGO
        RY"]) 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
    =
    dat
    e[0
    :10
    ]
    p2
    =
    dat
    e[1
    1:1
    3]
    p3
    =
    dat
    e[1
    4:]
    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_para
    m(stmt, 6, id)
    ibm_db.execute(st
    mt)


    print('successfu
```

```
        lly updated')
        return
        redirect("/displ
        ay")
#limit
@app.r
oute("/
limit" )
def
limit():
         return redirect('/limitn')


@app.route("/limitnum" , methods
= ['POST' ])def limitnum():
      if request.method ==
           "POST": number=
           request.form['numb
           er']
          #  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.r
oute("/l
imitn")
```

```
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)
    r
    o
    w
    =
    [
    ]
    s
    =
    "
    /
    -
    "
    while
        dictionar
        y !=
        False:
        temp = []
        temp.append(dictionary["LI
        MITSS"])
        row.append(temp)
        dictionary =
        ibm_db.fetch_assoc(res)s
        = temp[0]
```

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


#REPORT


@app.r

oute("/t

oday")

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

          dictionary

          1 != 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'])
+ " ANDDATE(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
          dictionar
          y !=
          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_foo
      d=0
      t_ent
      ertai
      nme
      nt=0
      t_bu
      sines
      s=0
      t_ren
      t=0
```

```
t_EMI=0
t_other=0



for
    x
    i
    n
    e
    x
    p
    e
    n
    s
    e
    :
    t
    o
    t
    a
    l
    +
    =
    x
    [
    4
    ]
    if x[6]
        ==
        "foo
        d":
        t_fo
        od
        +=
        x[4]

    elif x[6] ==
```

```
    "entertainm
    ent":
    t_entertain
    ment  +=
    x[4]

elif x[6]
    ==
    "busin
    ess":
    t_busi
    ness
    +=
    x[4]
elif
    x
    [
    6
    ]
    =
    =
    "
    r
    e
    n
    t
    "
    :
    t
    _
    r
    e
    n
    t
    +
    =
    x
```

[
4
]

elif
x
[
6
]
=
=
"
E
M
I
"
:
t
_
E
M
I
+
=
x
[
4
]

elif
x[
6]
=
=
"o
th
er

":

t_

ot

he

r

+

=

x[

4]

print(total)

print(t_food)

print(t_entert

ainment)

print(t_busin

ess)

print(t_rent)

p

r

i

n

t

(

t

_

E

M

I

)

p

r

i

n

t

(

t

```
_
o
t
h
e
r
)
```

```
    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.ro
ute("/mo
nth")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
        dictionary
        1 != False:
        temp = []
        temp.append(diction
        ary1["DT"])
        temp.append(diction
        ary1["TOT"])
        texpense.append(tem
        p) 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
        dictionar
        y !=
        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)



t
o
t
a
l
=
0
t
_
f
o
o
d
=
0
t_ent
ertai
nme
nt=0
t_bu
sines
s=0
t_ren
t=0
t_EMI=0
t_other=0
for
    x
    i
    n
    e
    x
    p
```

```
e
n
s
e
:
t
o
t
a
l
+
=
x
[
4
]
if x[6]
    ==
    "foo
    d":
    t_fo
    od
    +=
    x[4]

elif x[6] ==
    "entertainm
    ent":
    t_entertain
    ment  +=
    x[4]

elif x[6]
    ==
    "busin
    ess":
    t_busi
```

ness

+=

x[4]

elif

x

[

6

]

=

=

"

r

e

n

t

"

:

t

_

r

e

n

t

+

=

x

[

4

]

elif

x

[

6

]

=

=

"
E
M
I
"
:
t
_
E
M
I
+
=
x
[
4
]

elif
x[
6]
=
=
"o
th
er
":
t_
ot
he
r
+
=
x[
4]

print(total)

```
    print(t_fo
    od)
    print(t_ent
    ertainmen
    t)
    print(t_bu
    siness)
    print(t_re
    nt)
    print(t_E
    MI)
    print(t_ot
    her)



    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
        dictionary
        1 != False:
        temp = []
        temp.append(diction
        ary1["MN"])
        temp.append(diction
        ary1["TOT"])
        texpense.append(tem
        p) print(temp)
        dictionary1 = ibm_db.fetch_assoc(res1)



  #  cursor = mysql.connection.cursor()
  # cursor.execute('SELECT * FROM expenses WHERE userid =
% s ANDYEAR(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'])
+ " ANDYEAR(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
        dictionar
        y !=
        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)
```

```
t
o
t
a
l
=
0
t
_
f
o
o
d
=
0
t_ent
ertai
nme
nt=0
t_bu
sines
s=0
t_ren
t=0
t_EMI=0
```

```
t_other=0



for
    x
    i
    n
    e
    x
    p
    e
    n
    s
    e
    :
    t
    o
    t
    a
    l
    +
    =
    x
    [
    4
    ]
    if x[6]
        ==
        "foo
        d":
        t_fo
        od
        +=
        x[4]


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

```
elif x[6]
    ==
    "busin
    ess":
    t_busi
    ness
    +=
    x[4]
elif
    x
    [
    6
    ]
    =
    =
    "
    r
    e
    n
    t
    "
    :
    t
    _
    r
    e
    n
    t
    +
    =
    x
    [
    4
    ]

elif
```

x
[
6
]
=
=
"
E
M
I
"
:
t
_
E
M
I
+
=
x
[
4
]

elif
x[
6]
=
=
"o
th
er
":
t_
ot
he
r

```
            +
            =
            x[
            4]


    print(total)


    print(t_fo
    od)
    print(t_ent
    ertainmen
    t)
    print(t_bu
    siness)
    print(t_re
    nt)
    print(t_E
    MI)
    print(t_ot
    her)




    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('logge
  din', None)
  session.pop('id',
  None)
```

```
    session.pop('userna
    me', 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:

```
apiVer
sion:
apps/v
1kind:
Deplo
yment
metad
ata:
   name: sakthi-flask-node-
deploymentspec:
   r
   e
   p
   l
   i
   c
   a
   s
   :
   1
   s
   e
```

```
l
e
c
t
o
r
:
  matchLabels:
    a
pp:
flas
kno
de
tem
plat
e:
  metadata:
    labels:
      a
  pp:
  flas
  kno
  de
  spe
  c:
    containers:
    - name: flasknode
      image: icr.io/sakthi_expense_tracker2/flask-
      template2imagePullPolicy: Always
      ports:
      - containerPort: 5000
```

flask-service.yaml:

```
a
p
i
V
```

e
r
s
i
o
n
:
v
1
k
i
n
d
:
S
e
r
v
i
c
e
m
e
t
a
d
a
t
a
:
  name:
flask-app-
servicespec:
  selector:
    a
  p
  p:

flask-appports:
- name: http
  protocol: TCP
  port:

8

0

targ

etPort:

5000

type:

LoadB

alancer

manife

st.yml:

applications:

- name: Python Flask App IBCMR

2022-10-19random-route: true

me

mo

ry:

512

M

dis

k_q

uot

a:

1.5

G

sendemail.py:

```
import smtplib
import
sendgrid
as sg
import
os
from sendgrid.helpers.mail import Mail, Email,
To, ContentSUBJECT = "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.co
    m', 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 objectmail_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

T

a

b

l

e

s

:

1

.

A

d

m

i

n

:

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,username VARCHAR(32) NOT NULL, emailVARCHAR(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, amountVARCHAR(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 Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	BUG ID	Executed By
LoginPage_TC_OO 1	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	-		Kavinaya
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	-		Afra
LoginPage_TC_OO 2	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	-		Abdul Waseem
LoginPage_TC_OO 3	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	-		Jayasri
LoginPage_TC_OO 4	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	-		Afra
LoginPage_TC_OO 4	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	-		Kavinaya
LoginPage_TC_OO 5	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	-		Abdul Waseem
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	-		Jayasri

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

8.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. Payments & Invoices: Accept and pay from credit cards, debit cards, net banking, mobile wallets, and bank transfers, and trackthe status of your invoices and bills in the mobile app itself. Also, the trackingapp sends

remindersfor payments an d automatically matches the payments with invoices.

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

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

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

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

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

ix. Inventory tracking: An expense tracking app can do it all. Rightfrom 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.

x. 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.

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

9. 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 methodssuch 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.

10.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 ofexperience 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.

11 .FUTURE

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

1. The application is unable to maintain the backup of data once it isuninstalled.

2. This application does not provide higher decision capability.

To further enhance the capability of this application, we recommend the followingfeatures 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.

12.APPENDIX

Project Demo link- https://drive.google.com/file/d/1evdqDWIf_6uGsLhPDaZj3VPyWvdzZqYg/view?usp=share_link
GitHub link- https://github.com/IBM-EPBL/IBM-Project-17808-1659676539