PNT2022TMID33421

PERSONAL EXPENSE TRACKER APPLICATION

IBM-Project-25691-1659970868

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

A PROJECT REPORT

BY

SATHISH KUMAR (922519104142)

SANJITH S (922519104137)

SUDHARSAN N (922519104164)

SENTHILKUMAR G (922519104144)

INDEX

1. INTRODUCTION

a. Project Overview

TEAM ID : PNT2022TMID09631

INDUSTRY MENTOR : Kusboo

FACULTY MENTOR : K Johny Elma

Skills Required:

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

1. INTRODUCTION

a. Project Overview

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

b. Purpose

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

People tend to overspend without realizing, and this can prove to be disastrous. Using a daily expense managercan help you keep track of how much you spend every day and on what. At the endof the month, you will have a clear picturewhere your moneyis 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 to helpthem out. Beforeyou decide to go in for a money manager,it is important to decidethe type you want.

2. LITERATURE SURVEY

a. Existing problem

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

In another survey conducted by Levvel Research in 2018 in North America, respondents reported the following pain points in expense management before adopting automation:

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

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

iii. Inability to enforce travel policies (29%)

iv. Lost expense reports (24%)

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

b. References

S.No	TITLE	PROPOSED WORK	TOOLS USED/ ALGORITHM	TECHNOLOGY	ADVANTAGES/ DISADVANTAGES
1.	**EXPENSE MANAGER APPLICATION. (2020)**	To Develop A Moblie Application That Keeps Record Of User Personal Expenses Contribution In Group Expenditure Top Investment Options View Of The Current Stock Market ,Read Authenticated Financial News	Android Studio	Cloud Application	Advantages: ➢ Keeps Track All Of Your Daily Transactions, Keeps Track Of Your Money Lent Or Borrowed. Disadvantages: ➢ Occupy Lot Of Space.
2.	**A NOVEL EXPENSE TRACKER USING STATISTICAL ANALYSIS. (2021)**	To Maintain And Manage Data Of Daily Expenditure In A More Precise Way.	SQL Lite	Cloud Application	Advantages: ➢ Its Suggest You With The Most Effective Investment Options. Disadvantages: ➢ The Work Done Being Is Not Accurate.

S.No	TITLE	PROPOSED WORK	TOOLS USED/ ALGORITHM	TECHNOLOGY	ADVANTAGES/ DISADVANTAGES
3.	**EXPENSE TRACKER. (2021)**	Facilitates The User To Keep Track And Manage Their Personal As Well As Business Expenses.	Android OS	Cloud Application	Advantages: ➢ Become Aware Of Poor Spending Habits And Take Care Of Your Finances Saving And Investment. Disadvantages: ➢ Searching And Referencing Is Difficult And Time-consuming.
4.	**EXPENSE TRACKER. (May 2021)**	The Application Keeps The Track Of The Income And Expenses Both Of User On A Day To Day Bases	Java	Cloud Application	Advantages: ➢ The Project Effectively Keeps Away From The Manual Figuring. Disadvantages: ➢ Report Generation Is A Tedious Process.

3. Problem Statement Definition

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

Personal Expense Tracker Application:

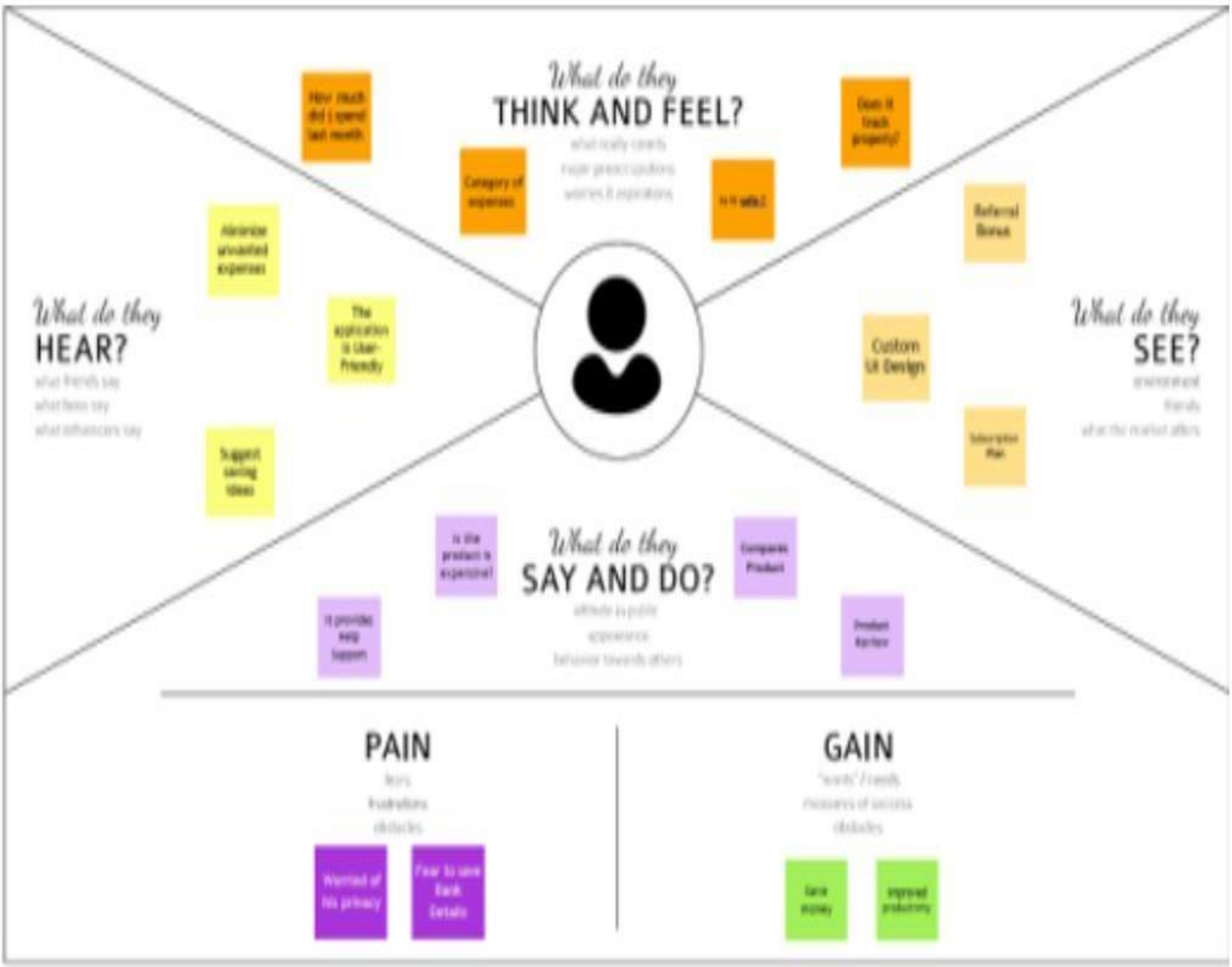
Customer Problem Statement :

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

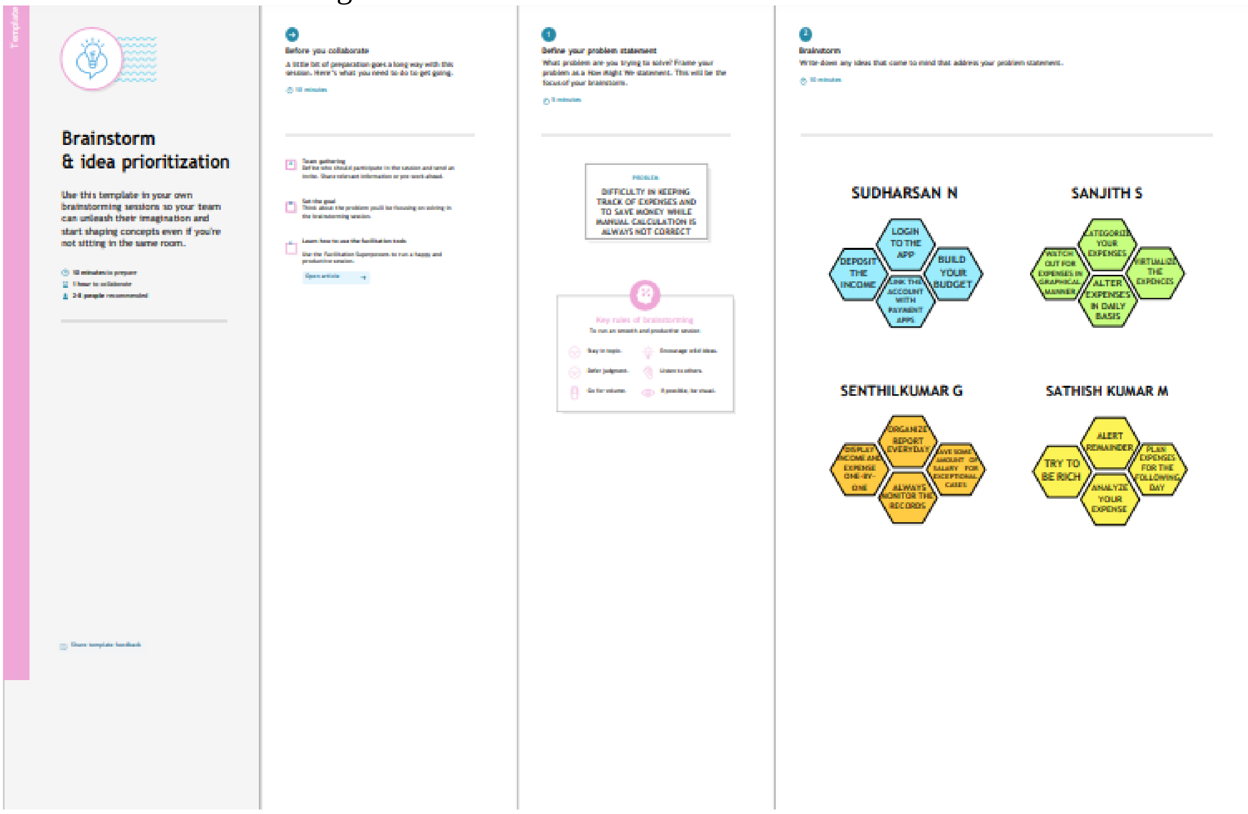
Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Student	Manage my expenses	It is very difficult	There is no proper app to warn me regarding my expense	Frustrated
PS-2	IT employee	Reduce my expense	I am not able to keep track of my expens e	I cant see the app whoch satisfies my needs	Annoyed

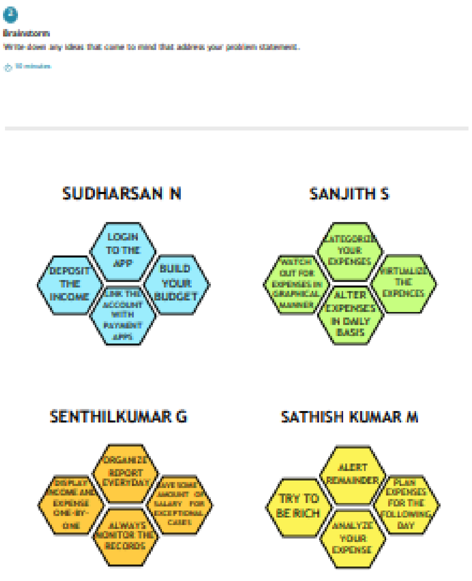
1. IDEATION & PROPOSED SOLUTION

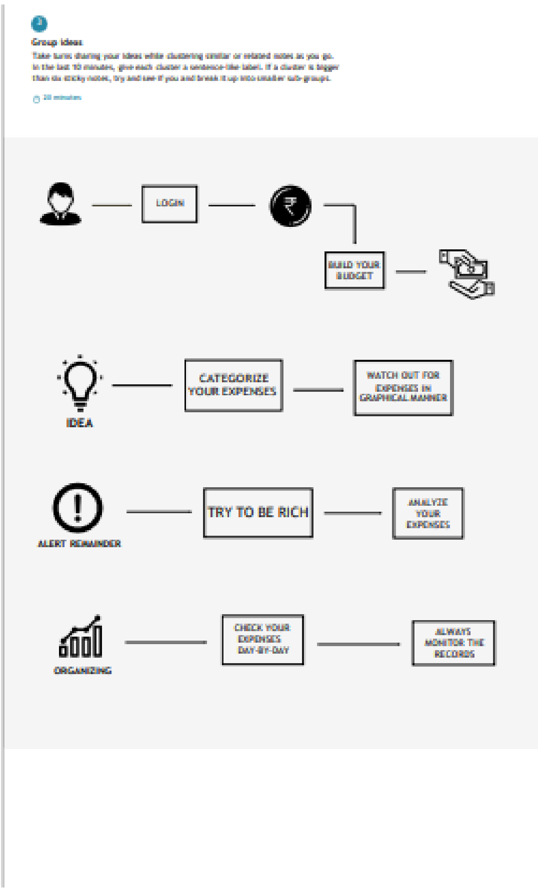
a. Empathy Map Canvas



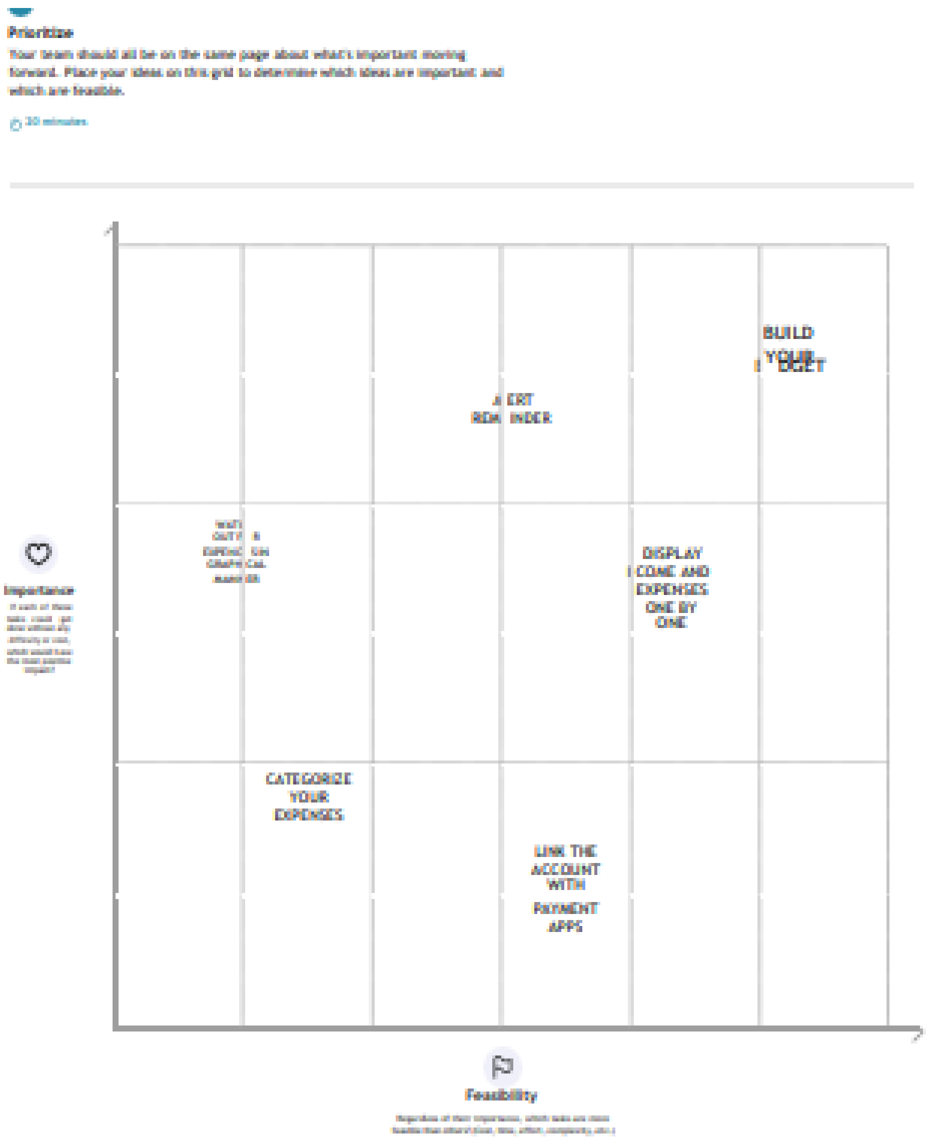
B. Ideation & Brainstorming




Brainstorm
Write down any ideas that come to mind that address your problem statement.
10 minutes
SUDHARSAN N
LOGIN TO THE APP
DEPOSIT THE INCOME
BUILD YOUR BUDGET
LINK THE ACCOUNT WITH PAYMENT APPS
SANJITH S
CATEGORIZE YOUR EXPENSES
WATCH OUT FOR EXPENSES IN GRAPHICAL MANNER
VIRTUALIZE THE EXPENCES
ALTER EXPENSES IN DAILY BASIS
SENTHILKUMAR G
ORGANIZE REPORT EVERYDAY
DISPLAY INCOME AND EXPENSE ONE-BY-ONE
SAVE SOME AMOUNT OF SALARY FOR EXCEPTIONAL CASES
ALWAYS MONITOR THE RECORDS
SATHISH KUMAR M
ALERT REMAINDER
TRY TO BE RICH
PLAN EXPENSES FOR THE FOLLOWING DAY
ANALYZE YOUR EXPENSE


Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go.
In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger
than six sticky notes, try and see if you and break it up into smaller sub-groups.
20 minutes
LOGIN
BUILD YOUR BUDGET
IDEA
CATEGORIZE YOUR EXPENSES
WATCH OUT FOR EXPENSES IN GRAPHICAL MANNER
ALERT REMAINDER
TRY TO BE RICH
ANALYZE YOUR EXPENSES
ORGANIZING
CHECK YOUR EXPENSES DAY-BY-DAY
ALWAYS MONITOR THE RECORDS

PNT2022TMID33421



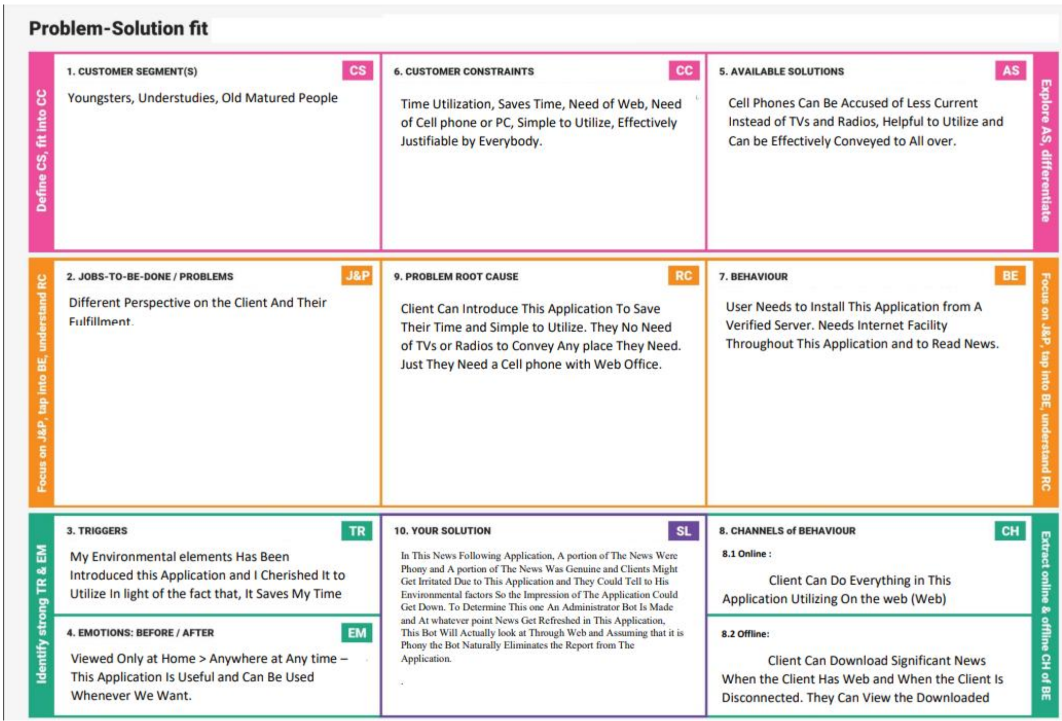
PNT2022TMID33421

c.Proposed Solution

S.No.	Parameter	Description
1.	ProblemStatement(Problemtobe solved)	Many organizations have their own system to record their income and expenses, which they feel is the main key point of their business progress. It is good habit for a person to record daily expenses and earning but due to unawareness and lack of proper applications to suit their privacy, lacking decision making capacity people are using traditional note keeping methods to do so. Due to lack of a complete tracking system, there is a 2 constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.
2.	Idea/Solutiondescription	We are building an android application named as "Expense Tracker". As the name suggests, this project is an android app which is used to track the daily expenses of the user. It is like digital recordkeeping which keeps the records of expenses done by a user. The application keeps the track of the Income and Expenses both of user on a day-to-day basis. This application takes the income of a user and manage its daily expenses so that the user can save money. If you exceed daily expense allowed amount it will give you a warning, so that you don't spend much and that specific day. If you spend less money than the daily expense allowed amount, the money left after spending is added into user's savings. The application generates report of the expenses of each end of the month. The amount saved can be used for celebrating festivals, Birthdays or Anniversary.
3.	Novelty/ Uniqueness	It will have various options to keep record (for example Food, Travelling Fuel, Salary etc.). Automatically it will keep on sending notifications for our daily expenditure. In today's busy and expensive life, we are in a great rush to make moneys, but at the end of the month we broke off. As we are unknowingly spending money on title and unwanted things. So, we have come over with the plan to follow our profit. Here user can define their own categories for expense type like food, clothing, rent and bills where they have to enter the money that has been

		to indicate the expense.
4.	SocialImpact/ CustomerSatisfaction	Money is the significant source of stress for nearly two-thirds of Americans. Fortunately, you don't have to wait until income increases or your debts are gone to enjoy relief- the mere act of planning ahead can reap immediate benefits. Here are some simple strategies you can use to better track and budget your expenses.
5.	BusinessModel(RevenueModel)	A well-known personal expense tracker, Mint is also a simple tool for smaller businesses and freelancers to track where money is going. It lets you create budgets and goals within the app, and track your credit score. You can access all of this data through an easy-to-read dashboard, so you know your standing at any time.
6.	Scalabilityofthe Solution	Monitoring your everyday expenses can set aside you cash, yet it can likewise help you set your monetary objectives for what's to come. On the off chance that you know precisely where your sum is going much of a stretch see where a few reductions and bargains can be made. Expense Tracker project is for keeping our day-to-day expenditures will help us to keep record of our money daily. The project what we have created is work more proficient than the other income and expense tracker. The project effectively keeps away from the manual figuring for trying not to ascertain the pay and cost each month. It's a user-friendly application.

d.Problem Solution fit



4. REQUIREMENT ANALYSI

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 Registration through Gmail *to prove their identity*
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

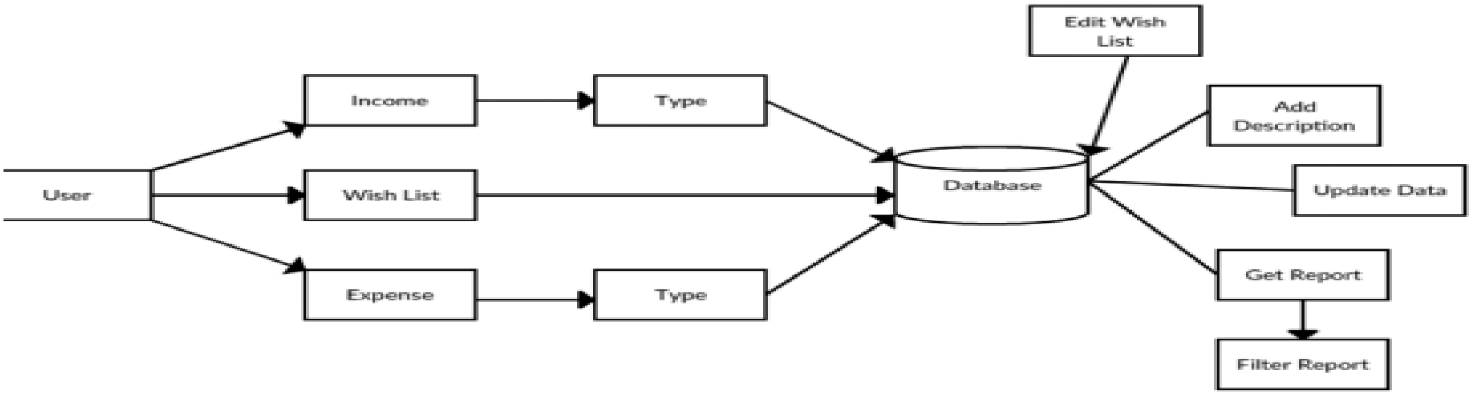
Non-functional Requirements:

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.

5. PROJECT DESIGN

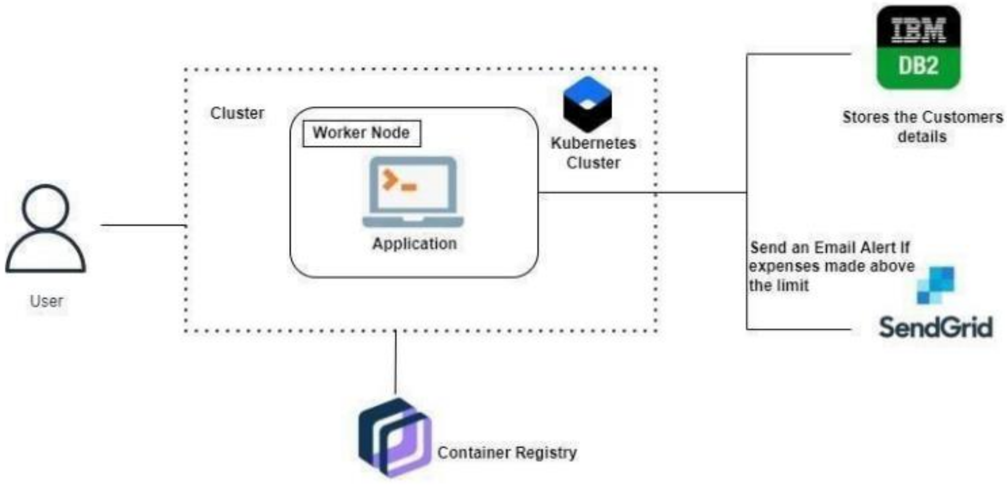
a. Data Flow Diagrams



PNT2022TMID33421

]

B.Solution & Technical Architecture



PNT2022TMID33421

C.**User Stories**

User Type	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Acceptance criteria**	**Priority**
Customer (web user)	Registration	USN-1	As a user, I can register for the application by entering mail id and password	I can access my account/ dashboard	High
		USN-2	As a user,I will receive a confirmation email once I have registered for the email and click application	I can receive a confirmation email	High
		USN-3	As a user, I can access using mail	I can register through mail	Low
	Login	USN-4	As a user, I can login application by entering application using email and password	I can access the application	High
	Dashboard	USN-5	As a user,I can view 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 login issue and other issues of the solution at any application	I can provide support	Medium
Administrator	Application	USN-7	As an administrator,I can upgrade or update the application	I can fix the bug	Medium

6. PROJECT PLANNING & SCHEDULING

a. Sprint planning and estimation

Sprint Delivery planning:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	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.	I can access my account / dashboard	High	SATHISHKUMAR M SANJITH S
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	SUDHARSAN N
Sprint-1		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	SENTHILKUMAR G
Sprint-1		USN-4	As a user, I can register for the application through Gmail	I can receive confirmation email & click confirm	Medium	SATHISHKUMAR M
Sprint-2	Login	USN-5	As a user, I can log into the application by entering email & password	I can access my account / dashboard	High	SUDHARSAN N
Sprint-2	Dashboard	USN-6	Create a model set that contains those models, then assign it to a role.	Assign that group to the appropriate roles on the Roles page	High	SANJITH S
Sprint-3	Identity-Aware	USN-7	Open, public access, User-authenticated access, Employee-restricted access.	Company public website. App running on the company intranet. App with access to customer private information.	High	SATHISHKUMAR M
Sprint-4	Communication	USN-8	A customer care executive is a professional responsible for communicating the how's and why's regarding service expectations within a company.	For how to tackle customer queries.	Medium	SENTHILKUMAR G
Sprint-4	Device management	USN-9	You can Delete/Disable/Enable devices in Azure Active Directory but you cannot Add/Remove Users in the directory.	Ease of use.	Medium	SUDHARSAN N

PNT2022TMID33421

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	30 Oct 2022	04 Nov 2022	20	4 Nov 2022
Sprint-3	20	6 Days	05 Nov 2022	10 Nov 2022	20	10 Nov 2022
Sprint-4	20	6 Days	11 Nov 2022	16 Nov 2022	20	16 Nov 2022

Velocity:

Imagine we have a 10-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)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

b. Sprint Delivery Schedule

S.NO	ACTIVITY TITLE	ACTIVITY DESCRIPTION	DURATION
1	**Understanding the project**	Assign the team members after that create repository in the GitHub and then assign task to each member and guide them how to access the GitHub while submitting the assignments	1 week
2	**Staring The Project**	Team Members to Assign All the Tasks Based on Sprints and Work on It Accordingly.	1 week
3	**Completing Every Task**	Team Leader should ensure that whether every team member have completed the assigned task or not	1 week
4	**Stand Up Meetings**	Team Lead Must Have a Stand-Up Meeting with The Team and Work on The Updates and Requirement Session	1 week
5	**Deadline**	Ensure that team members are completing every task within the deadline	1 week

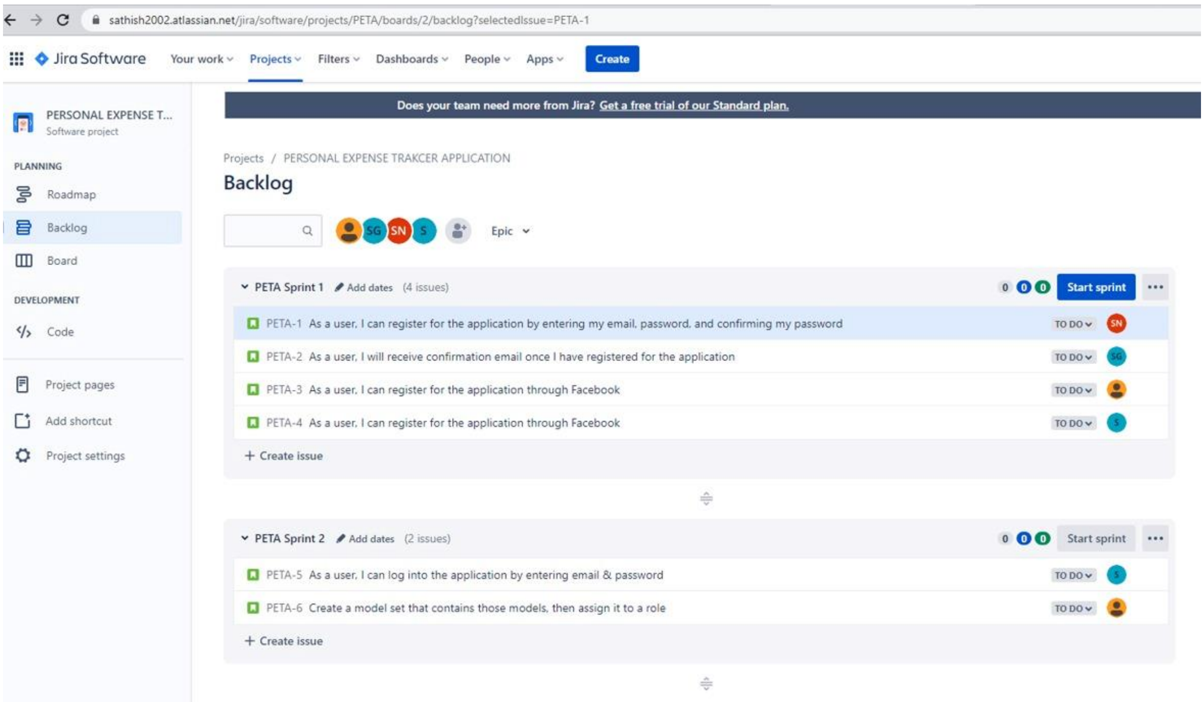
		Solution Architecture	27 Sept 2022 – 30 Sept 2022

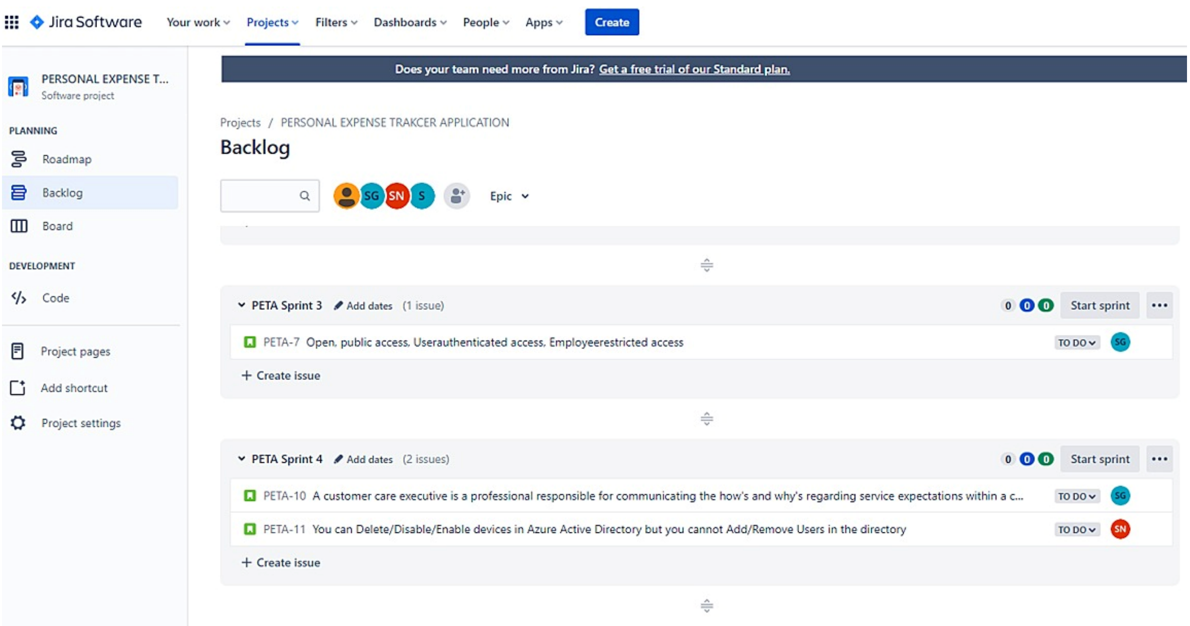
6	**Budget and Scope of project**	Analyze the overall budget which must be within certain limit it should be favorable to every person	1 week

PNT2022TMID33421

a. **Reports from JIRA**

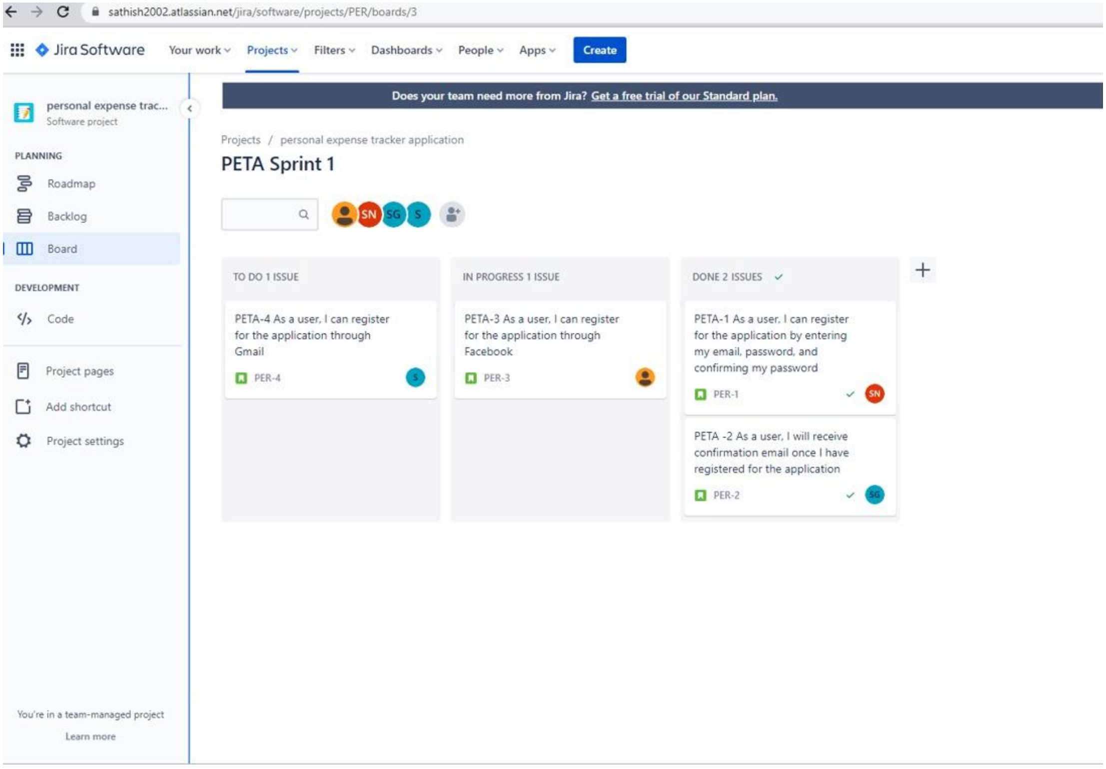
i. Backlog



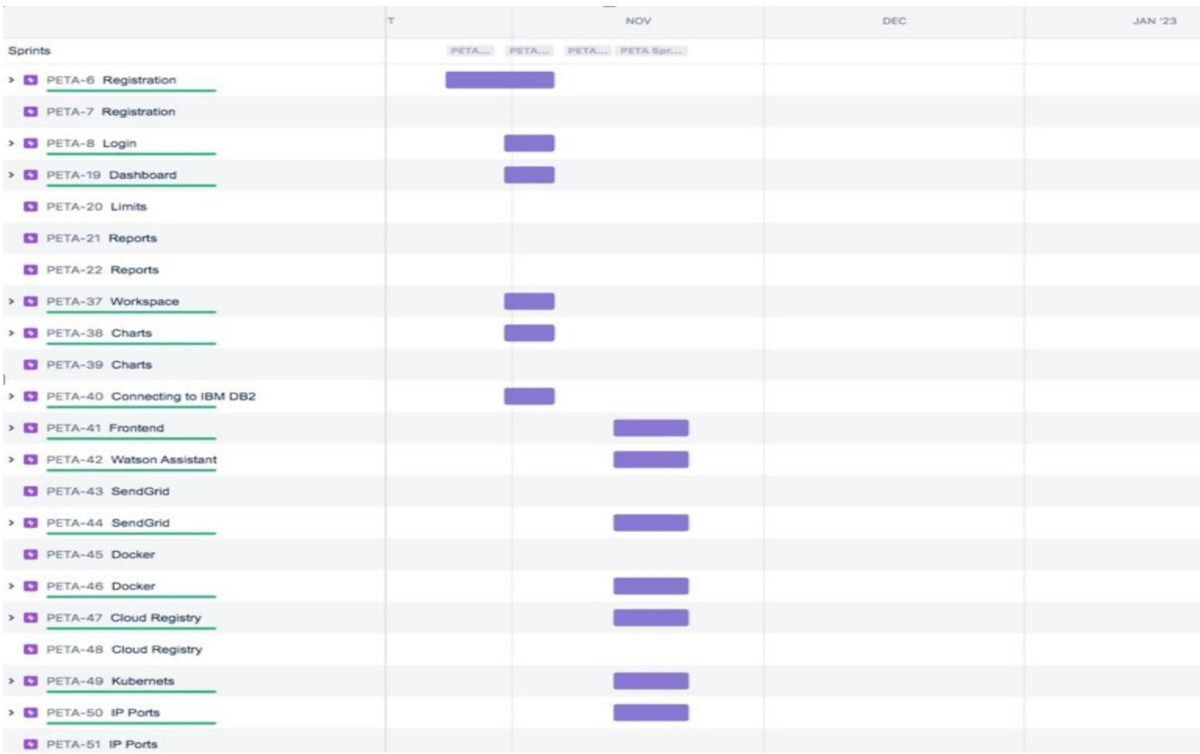


PNT2022TMID33421

ii. Board



iii. Road Map



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.appdomain.cloud;port=31498;protocol=tcpi
p;\
```

```
            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()
```

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

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 and tracking all expenditures (through bank accounts, mobile wallets, and credit & debit cards).

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

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

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

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

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

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

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

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

x. Inventory tracking: An expense tracking app can do it all. Right from tracking products or the cost of goods, sending alert notifications when the product is running out of stock or the product is not selling, to purchase orders.

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

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

10. ADVANTAGES & DISADVANTAGES

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

9. Ability to provide **valuable insights**.

10. Optimize sales processes to generate **more revenue** through enhanced data collection.

11. CONCLUSION

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

12. FUTURE

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

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

2. This application does not provide higher decision capability.

To further enhance the capability of this application, we recommend the following features to be incorporated into the system:

3. Multiple language interface.

4. Provide backup and recovery of data.

5. Provide better user interface for user.

6. Mobile apps advantage.

13. APPENDIX

Source Code Github Link : https://github.com/IBM-EPBL/IBM-Project-40097-1660623339

Project Demo Link: https://drive.google.com/file/d/1K9h4hygUDftZxJKq6bgAk5dMn8-XoNvQ/view?usp=drivesdk