PERSONAL EXPENSE TRACKER APPLICATION

A PROJECT REPORT

Submitted by

TEAM LEADER : KAVIYA J
TEAM MEMBER : DURKADEVI C
TEAM MEMBER : PHOOVIHA MK
TEAM MEMBER : SAKTHIPRIYA S

TEAM ID - PNT2022TMID47312

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

GOVERNMENT COLLEGE OF ENGINEERING
SRIRANGAM

ABSTRACT

In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management.

Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert.

TABLE OF CONTENT

INTRODUCTION

1.1 Project Overview

Management about pay up then debts has been a true problem because a long time. People are much less in all likelihood in accordance with preserve song about theirs spending, stand it, in checkbook yet even spreadsheets. Despite that, because of the current decades, it has acquired a latter perspective with the advent on modern technologies yet the internet as is turning into extra yet more accessible. Daily Expense Tracker is a path to that amount execute assist us after hold above along our spending. Not solely that, such perform help us pinpoint areas up to expectation we bear been conclusion or music upcoming bill payments.

1.2 Purpose

Daily Expense Tracker helps in accordance with maintain the document regarding daily costs yet month-to-month income because someone person or additionally generates a month-to- month file over the expenses. The Daily Expense Tracker software tracks whole the expenses then help the person in imitation of boss his/ her personal prices consequently so much the consumer desire reach a proper pecuniary stability. The tracking regarding prices is classified daily, hebdomadal yet monthly, such helps in imitation of advice more charges made. To utilize the Daily Expense Tracker the person has after signal over consonant name, smartphone no, address, electronic mail address, username, and password yet assure password over the user. The consumer executes find enlisted just an odd time, care of user be able simply some record. The remainder is put in because future expenses.

LITERATURE SURVEY

2.1 Existing problem

The Expense tracker existing system does not provide the user portable device management level, existing system only used on desktop software so unable to update anywhere expenses done and unable to update the location of the expense details disruptive that the proposed system provides. In existing, we need to maintain the Excel sheets, CSV files for the user daily, weekly and monthly expenses. In existing, there is no as such complete solution to keep a track of its daily expenses easily. To do so a person as to keep a log in a diary or in a computer system, also all the calculations need to be done by the user which may sometimes results in mistakes leading to losses. The existing system is not user friendly because data is not maintained perfectly. But this project will not have any reminder to remain a person in a specific date, so that is the only drawback in which the remainder is not present. This project will be an unpopulated information because it has some disadvantages by not remind a person for each and every month. But it can used to perform calculation on income and expenses to overcome this problem we propose the new project.

2.2 References

S. NO	PAPER NAME	AUTHOR	DESCRIPTION
1	Expense manager application	Velmurugan , Albert Mayan , Niranjana Richard Francis	In this paper, we develop a mobile application developed for the android platform that keeps record of user personal expenses, his/her contribution in group expenditures, top investment options, view of the current stock market, read authenticated financial news and grab the best ongoing offers in the market in popular categories. With our application, one can manage their expenses and decide on their budget more effectively.
2	A Novel Expense	Muskaan Sharma, Ayush Bansal,	statistical analysis has to be done to be able to give users correct

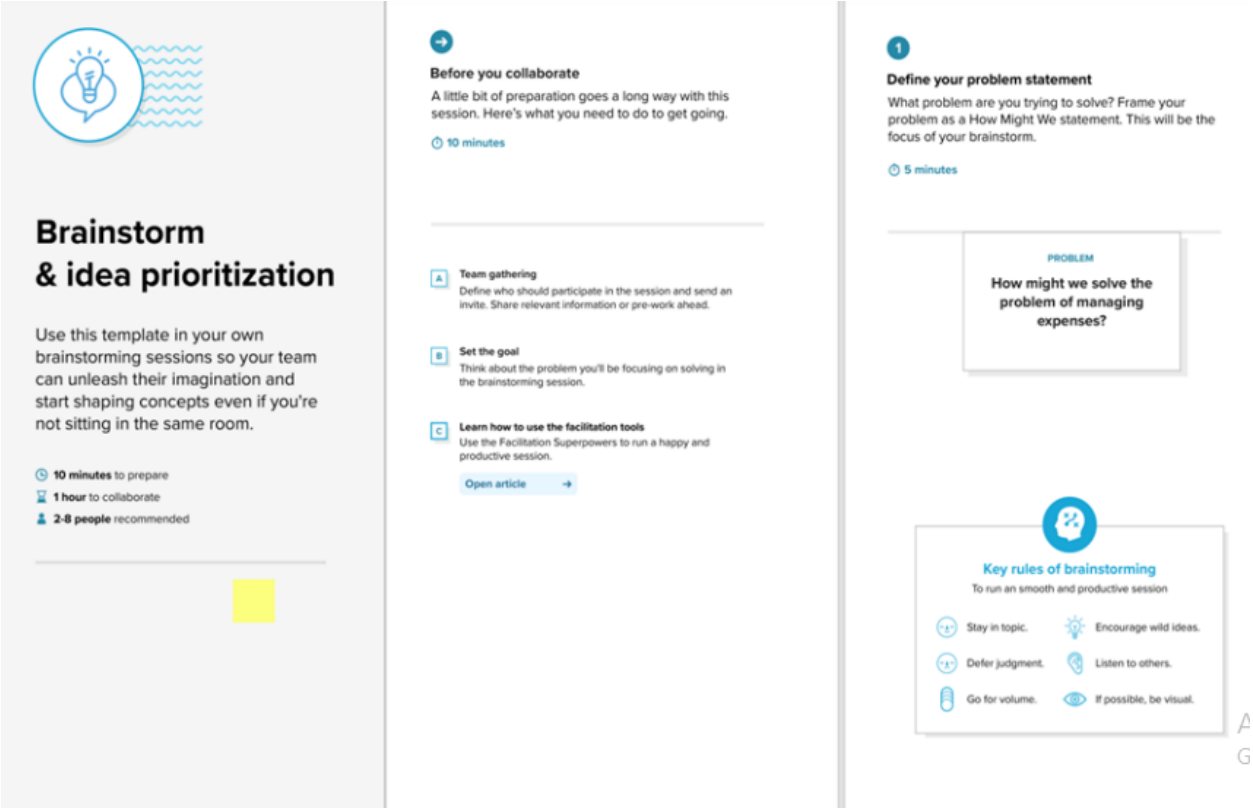
	Tracker using Statistical Analysis	Shivam Sethi	information on their expenses and help them spend better. This helps the society in issues like bankruptcy and save time from manual calculations. For using such application, a user needs to provide his/her total income or the amount he/she is spending per day and each user details or information are going to be stored in a unique way.
3	Online Income And Expense Tracker	S. Chandini, T. Poojitha, D. Ranjith, V.J. Mohammed Akram, M.S. Vani, V. Rajyalakshmi	Our application acts as an indicator or reminder example in the fastest world which we can't able to remember what are the things we have to do for the end of month and what are the payments we have to pay for the particular month. Due to some conflicts or some other stress we forget some times that what are the income or where the money has to come from or what the payments we have to pay. This application will help you to make a note for what or the things we have to do for the end of month.

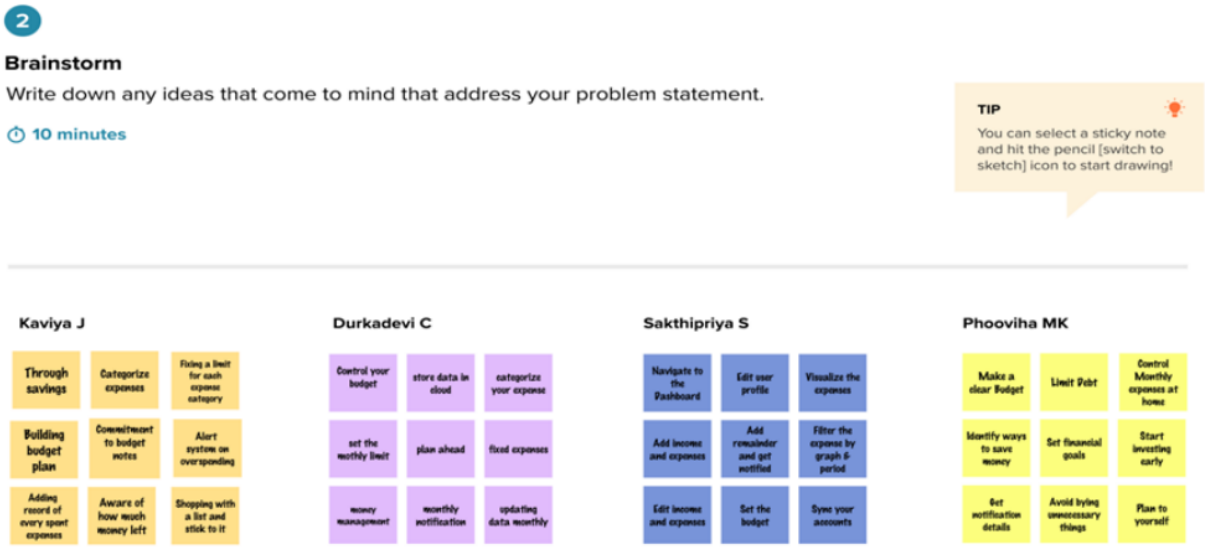
2.3 Problem Statement Definition

IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming





3

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

By means of savings, we can avoid unnecessary expenses.

Kinds of expenses should be categorized and a separate fixed budget limit should be assigned to each.

While go for purchasing with a pre-planned list, it can prevent us from buying things that are not actually needed.

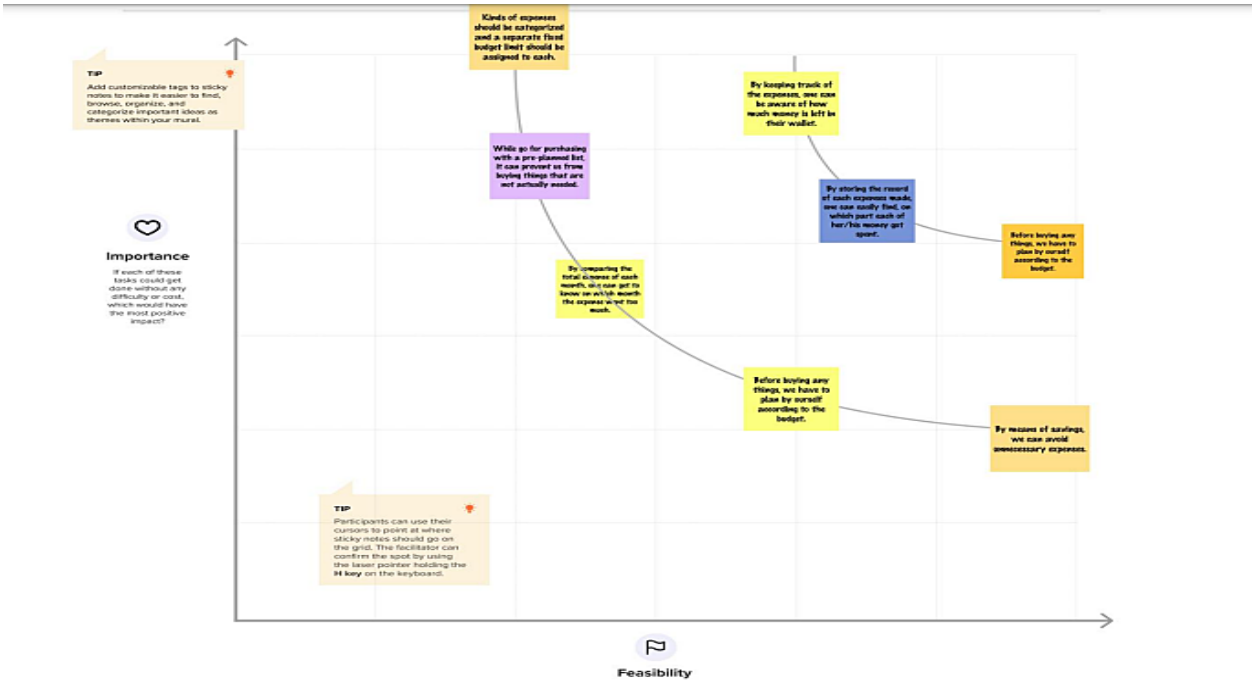
By storing the record of each expenses made, one can easily find, on which part each of her/his money got spent.

By keeping track of the expenses, one can be aware of how much money is left in their wallet.

Before buying any things, we have to plan by ourself according to the budget.

By comparing the total expense of each month, one can get to know on which month the expense went too much.

If we spend money more than the budget plan, the notification setting will be helpful by sending an alert message.



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement	People find it difficult to manage the moneytheyearn and they often end up in spending their money unnecessarily without a proper budget plan
2	Idea / Solution description	By using an application which is handy andgivesyou a clear vision to segregate your moneytobespent on each necessary expenses, one candefinitely save money apart fromthe expenses which are planned.
3.	Novelty / Uniqueness	Once the user set the budget limit on eachcategory of expenses, the application wouldalert when they overspend on something and alsoonecan get to know on which part they spendmostlyand depending on that Money can be spent wisely.
4.	Social Impact / Customer Satisfaction	The application acts as a supermind for the userswhich leads them, alerts themand guides themby making them spend their money wiselyonlyonnecessary stuffs and also it can store everyexpenses which were made and could showtheavailable balance which makes themaware of howmuch money they've left. Hence this couldbethebest way to manage money and that is what theneed of every users.
5.	Business Model	The application provides advertisements whichmight be useful for the users and in returnhelpful in increasing the revenue. The users can alsoopt for memberships which let themrelish extraspecial features
6.	Scalability of the Solution	It would work perfectly even while storingahugeamount of expense datasets and giving multiplecommands doesn't affect its performance or scalability.

3.4 Problem Solution fit

Define CS, fit into

1. CUSTOMER SEGMENT(S) CS

- People who are struggling to track their expenses are our customers.
- One who wants to manage the money they earn.

6. CUSTOMER CONSTRAINTS

- Accessing of particular features needs premium account.
- Internet connection.

5. AVAILABLE SOLUTIONS

- Usage of manual notes by noting down every spent expenses.
- Keeping the bills.

Explore AS, differ

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

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

- There might be manual error in the expenses calculation process and lack of expense history maintenance.
- In paper-based expense tracker it is difficult to track our monthly expenses manually.
- This records may get lost in case of any obstacles.

9. PROBLEM ROOT CAUSE RC

- Less focus on career and development.
- Being unconscious while spending money.
- Not having much knowledge on financial stability.
- Real-time tracking is difficult for physical mode of payment.

7. BEHAVIOUR BE

- Customers get unlimited access to their calculation or they may create notes on their mobile.
- This approach makes it very simple and really beneficial to estimate their expenditure and needs.

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

3. TRIGGERS TR

- It can create awareness among common people about their income and expenses.
- It reduces time rather than entering details manually.
- Seeing others using their money wisely.

4. EMOTIONS: BEFORE / AFTER EM

- Difficulty in managing their money and being unaware of their monthly expenses that they have spent daily-Before
- Efficient way to tackle and manage their important expenses with more security and can also have a proper budget to spend their expenses-After

10. YOUR SOLUTION SL

- Creating an application that will help in accounting, budgeting, providing them with useful insight about money management.
- This app brings high security and real time tracking of expenses done by the user.

8. CHANNELS OF BEHAVIOUR CH

8.1 ONLINE

- Real-time notification for un-tracked expenses is not available.
- Researched on available application features.

8.2 OFFLINE

- Got ideas from people to get to know their actual needs.

REQUIREMENT ANALYSIS

4.1 Functional requirement

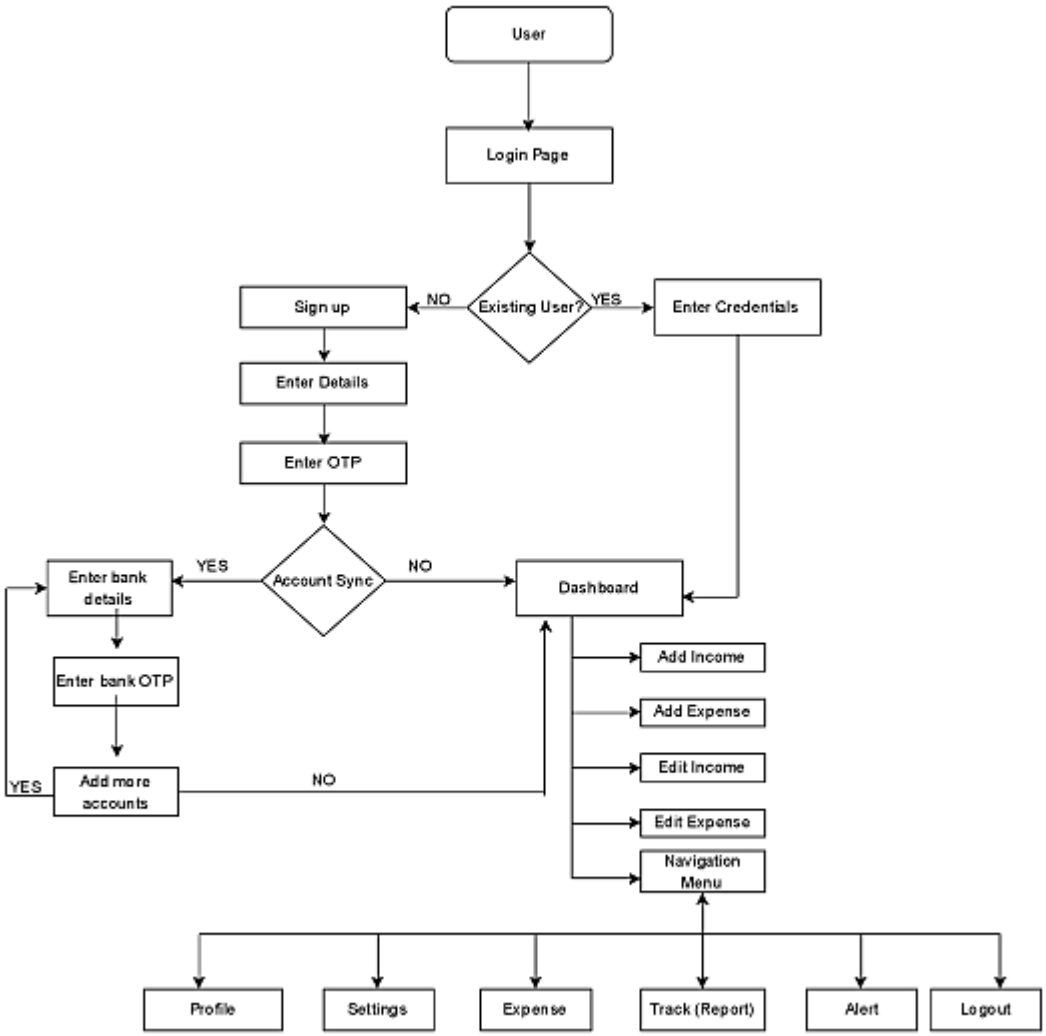
FR No.	Functional Requirement	Sub Requirement
FR-1	User Registration	Registration through application for collecting user details.
FR-2	User Category	Allowance of categorization of different expenses.
FR-3	Forgot Password	Resetting the password or username by sending an OTP to user's Phone SMS.
FR-4	User Login	User Login to the dashboard through mail account or by entering username and password.
FR-5	Dashboard	User can add the expense and can evaluate them using the provided options.
FR-6	Alert User	Alert the user through Notification system.
FR-7	Result Page	Shows the user result of the tracked expense.
FR-8	Visual Representation	Expense report should be generated in a graphical format.

4.2 Non-Functional requirements

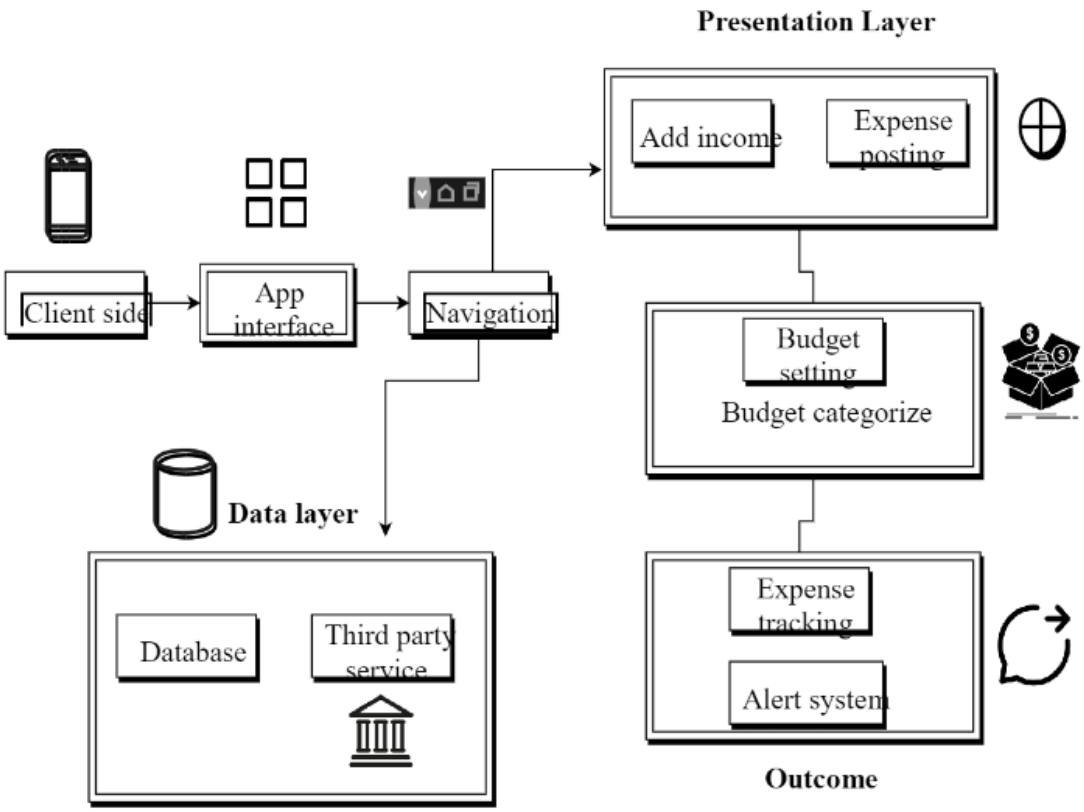
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Used to keep a record of how your expense got spent.
NFR-2	Security	Customer data's are protected with strong password.
NFR-3	Reliability	If the customer face any problem, then they can sent query messages.
NFR-4	Performance	Makes trustworthy calculations and performance consistently well.
NFR-5	Availability	Application is made available 24/7 for the user.
NFR-6	Scalability	It would work perfectly even while storing huge amount of datasets and giving multiple commands doesn't affect its performance.

PRJECT DESIGN

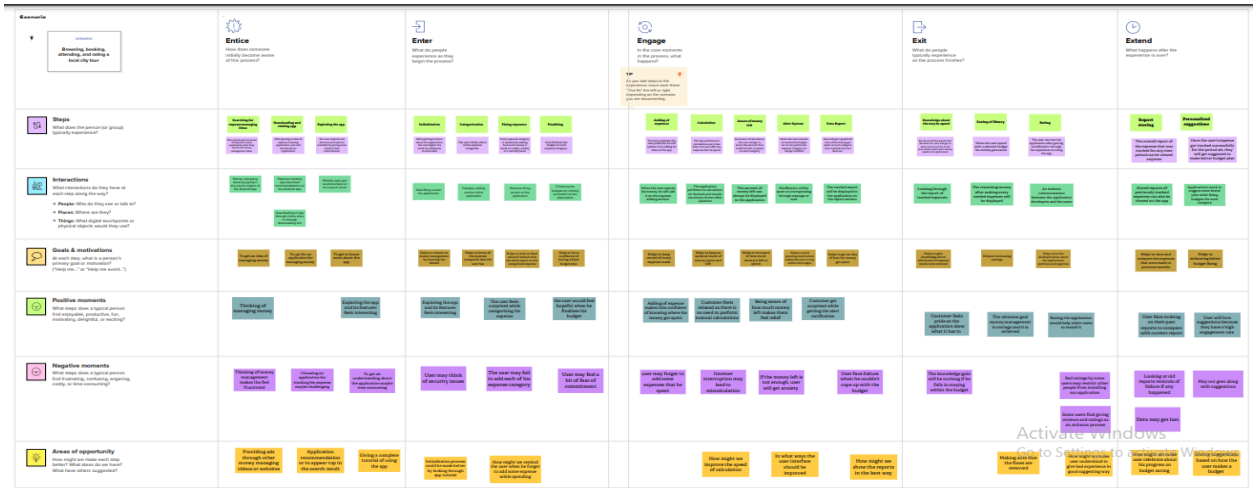
5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories



PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

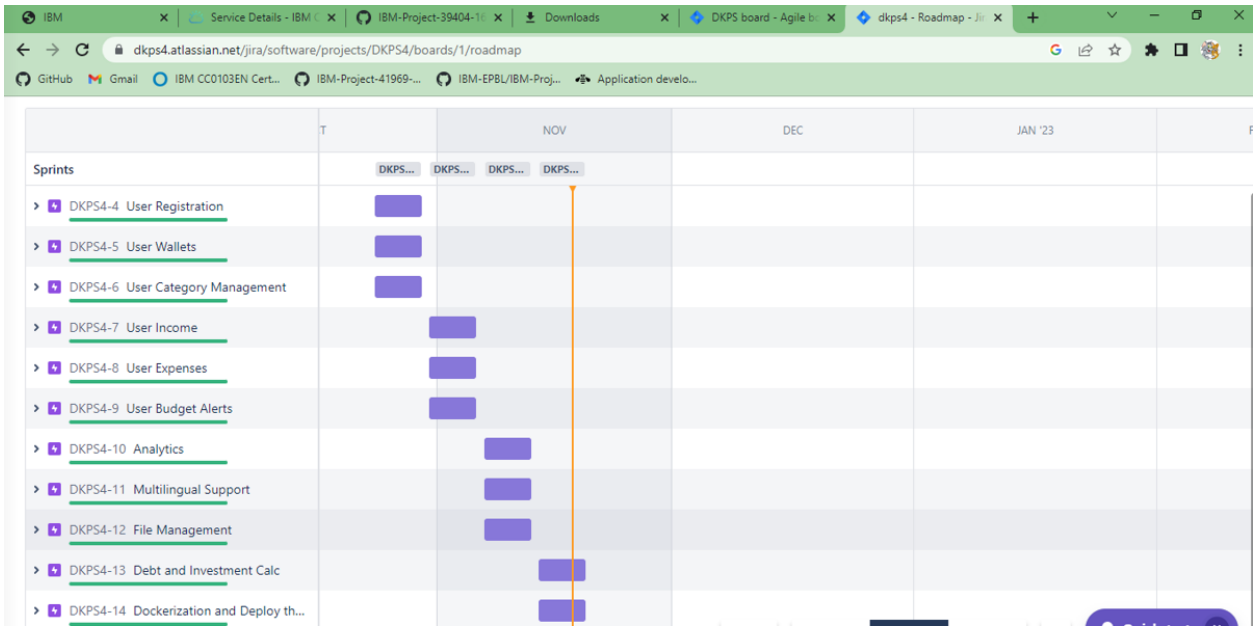
SPRINTS	FUNCTIONAL REQUIREMENT	USER STORY NO	USER STORY	STORY POINTS	PRIORITY	TEAM MEMBERS
ST – 1	User Registration	USN-1	Create an ccount for the users to get access to all the features	10	High	Sakthipriya S, Kaviya J
ST – 1	User Wallets	USN-2	Wallets hold the users money	5	High	Durkadevi C
ST – 1	User Category Management	USN-3	Users can customize their income and expense categories	5	Medium	Phooviha MK
ST – 2	User Income	USN-4	Users can attach their income to the wallets	7	High	Kaviya J
ST – 2	User Expenses	USN-5	Users can deduct their amount from the wallets	7	High	Sakthipriya S
ST – 2	User Budget Alerts	USN-6	Users can set alerts and limit their expenses	6	Medium	Phooviha MK
ST – 3	Analytics	USN-7	Users can get a visualization on their income and expenses	6	High	Kaviya J, Durkadevi C
ST – 3	Multilingual Support	USN-8	Users should be able to use the application in their languages	4	Low	Phooviha MK, Sakthipriya S
ST – 3	File Management	USN-9	Users should be able to attach files to their income or expenses	6	Medium	Kaviya J, Durkadevi C

ST – 4	Debt and Investment Calc	USN-11	Users can calculate their returns and risks	7	Medium	Phooviha MK, Kaviya J
ST – 4	Dockerization and Deploy the application	USN-13	Container the application and deploy to the kubernetes cluster	13	High	Kaviya J, Sakthipriya S, Phooviha MK, Durkadevi C

6.2 Sprint Delivery Schedule

SPRINTS	TOTAL STORY POINTS	DURATION	SPRINT START DATE	SPRINT END DATE	STORY POINTS COMPLETED	SPRINT RELEASE DATE
SPRINT – 1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
SPRINT – 2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
SPRINT – 3	20	6 Days	07 Oct 2022	12 Nov 2022	20	12 Nov 2022
SPRINT – 4	20	6 Days	14 Nov 2022	19 Oct 2022	20	19 Nov 2022

6.3 Reports from JIRA



CODING AND SOLUTIONING

7.1 Feature 1

- Multiuser authentication
- Wallets
- Income/Expense Transactions
- Budget Alerts

7.2 Feature 2

- News Board with Search
- Multilingual Support

7.3 Database Schema

```
CREATE TABLE users (
    username VARCHAR(32) UNIQUE NOT NULL,
    email VARCHAR(32) NOT NULL,
    password LONG VARCHAR
);

CREATE TABLE wallets (
    wid VARCHAR(32) UNIQUE NOT NULL,
    wname VARCHAR(30) NOT NULL,
    wamount INTEGER NOT NULL,
    username VARCHAR(30) NOT NULL
)

CREATE TABLE transactions (
    tid VARCHAR(32) UNIQUE NOT NULL,
    tdate VARCHAR(12) NOT NULL,
    descp VARCHAR(1000) NOT NULL,
    wallet VARCHAR(32) NOT NULL,
    ttype VARCHAR(7) NOT NULL,
```

```
    category VARCHAR(20) NOT NULL,
    amount INTEGER NOT NULL,
    attachment VARCHAR(100),
    username VARCHAR(30) NOT NULL
);

CREATE TABLE budgets (
    bid VARCHAR(32) UNIQUE NOT NULL,
    bname VARCHAR(30) NOT NULL,
    bdate VARCHAR(12) NOT NULL,
    bamount INTEGER NOT NULL,
    bwallet VARCHAR(32) NOT NULL,
    username VARCHAR(30) NOT NULL
);
```

TESTING

8.1 Test Cases

S.No	Module	Test Cases
1	User	- Form Input Validation - User Creation - User Retrieval - Forgot Password - Session Validation
2	Wallet	- CRUD Wallets - Input Validation
3	Transactions	- CRUD Income - CRUD Expenses - Image Append - Wallet Upgradation - Budget Validation
4	Budget	- CRUD Budget - Budget Alert Scheduling
5	External APIs	- News Fetch - Multilingual Translation Validation

8.2 User Acceptance Testing

1. Purpose Document

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

2. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	6	3	1	2	12
Duplicate	1	1	0	0	2
External	2	1	0	0	3
Fixed	4	0	1	0	5
Not Reproduced	0	2	0	0	2
Skipped	1	1	1	1	3
Won't Fix	0	3	0	2	5
Totals	14	11	3	5	33

---

3. Test Case Analysis

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	5	0	0	5
Client Application	35	0	0	35
Security	8	0	0	8
Outsource Shipping	10	0	0	10
Exception Reporting	3	0	0	3
Final Report Output	15	0	0	15
Version Control	4	0	0	4

RESULTS

9.1 Performance Metrics

Latency:

Personal Expense Tracker application deployed on the IBM Kubernetes Cluster offers up to 150 - 200ms in the average traffic and 200 -250ms in the stressed loaded environment with Ngnix container acting as the service in the cluster.

Rate:

Personal Expense Tracker Application can handle up to 300K to 400K requests at a time and the over the handle can wear down but satisfies the request upon delayed manner.

No. of failures:

Personal Expenses Tracker Application is designed in such a way that it can handle different input and respond accordingly. The application is aggressively tested with different test cases so that it reduces the rate of failure up to 0.75%.

ADVANTAGES AND DISADVANTAGES

Advantages:

1.Management and control
2.Evaluation of policies
3.Capital reinforcement
4.Systematic and organized
5.Constructive

Disadvantages:

1.Inaccurate and unrealistic
2.Inflexible
3.Finance oriented
4.Time-consuming
5.Conflicts

CONCLUSION

Daily Expense Tracker is a gadget that being developed to help customers in budget planning.This new system has overcome most of the limitations of the existing system.The project what we have developed is work more efficient than the other income and expense tracker. This project successfully avoids the manual calculation for avoiding calculating the income and expense per month. The modules are developed with efficient and also in an attractive manner. The developed systems dispense the problem and meet the needs of by providing reliable and comprehensive information. All the requirements projected by the user have been met by the system. The newly developed system consumes less processing time and all the details are updated and processed immediately.In short, this application will help its customers to overcome the wastages of money.

FUTURE SCOPE

The Future Enhancements of the application can be allowed to support in all the upcoming android versions. History can be set to view all the details in the app even if the particular data is deleted from the database. Statistics could be prepared based on the Income, Expense details of the user. Printing the details of the particular income or expense details can be made. Some of the extra components are like enabling users to register to the application using existing email or social network account, it will synchronize the users profile data to the application.

APPENDIX

SOURCE CODE

```
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'
"""
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;prot
ocol=tcpip;\
            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"])
        ex11pense.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.c


@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)
```

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-39404-1660411573

DEMO VIDEO LINK:

https://drive.google.com/drive/folders/1KBqd73DI3-2kFhTOIrfCfE3ZJNvtmnMV?usp=sharing