PERSONAL EXPENSE TRACKER APPLICATION
TEAM ID: PNT2022TMID03112

A PROJECT REPORT

Submitted by

KAVINAYA V (19EUEC068)
KAVYA T (19EUEC069)
KEERAN M (19EUEC070)
KEERTHANA S (19EUEC071)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution, Affliliated to Anna University Chennai - 600 025)

NOVEMBER 2022

ABSTRACT

Modern life offers a plethora of options of services and goods for consumers. As a result, people's expenses have gone up dramatically, e.g., compared to a decade ago, and the cost of living has been increasing day by day. Thus it becomes essential to keep a check on expenses in order to live a good life with a proper budget set up. The application "Personal Expense Tracker" is developed to manage the daily expenses in a more efficient and manageable way. By using this application we can reduce the manual calculations of the daily expenses and keep track of the expenditure. In this application, user can provide his income to calculate his total expenses per day and these results will be stored for each user. The application has the provision to predict the income and expense for the manager using data mining. This application will also have a feature which will help you stay on budget because you know your expenses. Expense tracker application will generate report at the end of month to show Expense via a graphical representation. We also have added a special feature which will distributes your expenses in different categories suitable for the user. An expense history will also be provided in application.The application allows users to track their expenses daily, weekly, monthly, and yearly in terms of summary, bar graphs, and pie-charts. This mobile application is a full detailed expense tracker tool that will not only help users keep a check on their expenses, but also cut down the unrequired expenses, and thus will help provide a responsible lifestyle.

1. INTRODUCTION

1.1 PROJECT OVERVIEW

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.

1.2 PURPOSE

Personal finance management is an important part of people's lives. However, everyone does not have the knowledge or time to manage their finances in a proper manner. And, even if a person has time and knowledge, they do not bother with tracking their expenses as they find it tedious and 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 manager and money manager, an expense tracker is a software or application that helps to keep an accurate record of your money inflow and outflow. Many people in India live on a fixed income, and they find that towards the end of the month they don't have sufficient money to meet their needs. While this problem can arise due to low salary, invariably it is due to poor money management skills.

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

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

The existng problem is tracking the expenses of the people money. It cannot be done just by uploading the dataset but realtime money calculation is not done efficiently.

- Data might get stolen or easily lost
- Susceptible to human errors while calculating
- More time consuming due to manual process
- Data is not accurate and it is difficult to update the data in real time

2.2 REFERENCES

[1] Rajaprabha, M. N. (2017). Family Expense Manager Application in Android. MS&E, 263(4), 042050

[2] N. ZahiraJahan MCA., M. Phil, K. I. Vinodhini, “Personalized Expense Managing Assistant Using Android”, International Journals of Computer Techniques (IJCT), Volume: 3 Issue: 2, ISSN: 2394-2231 (March-April 2016).

[3] Sabab, S. A., Islam, S. S., Rana, M. J., & Hossain, M. (2018, September). eExpense: A smart approach to track everyday expense. In 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT) (pp. 136-141). IEEE.

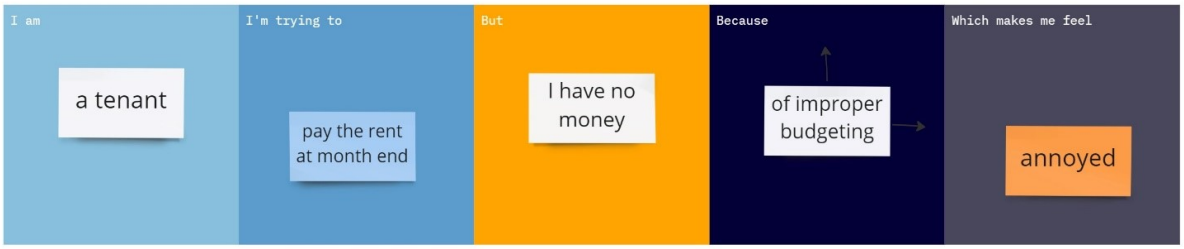
[4] Kan, C., Lynch, J., & Fernbach, P. (2015). How budgeting helps consumers achieve financial goals. ACR North American Advances

[5] Sharma, R., 2020. Case Study Of Expense Tracking App: Get Daily Alerts Of Your Expense. [online] Medium

[6] Underwood, D. (2011). A Case Study of Tracking Expenses by Commodity at Widget Farmers’ Cooperative.

2.3 PROBLEM STATEMENT DEFINITION

In today's financial world, money management is the necessary part of life. A person must keep track of their daily expenses in order to spend within the budget. The proper balance between income and expense is a must for a comfortable livelihood. But in the absence of proper management of money, we left with no savings at all. Generally paper based expense tracking system is used to track the expenses. But the problem is that data might get lost due to flood, accidents etc.. Additionally since all computations must be done manually, there is a chance for errors and consequent losses. In order to overcome all these drawbacks, a personal expense tracking app is developed.



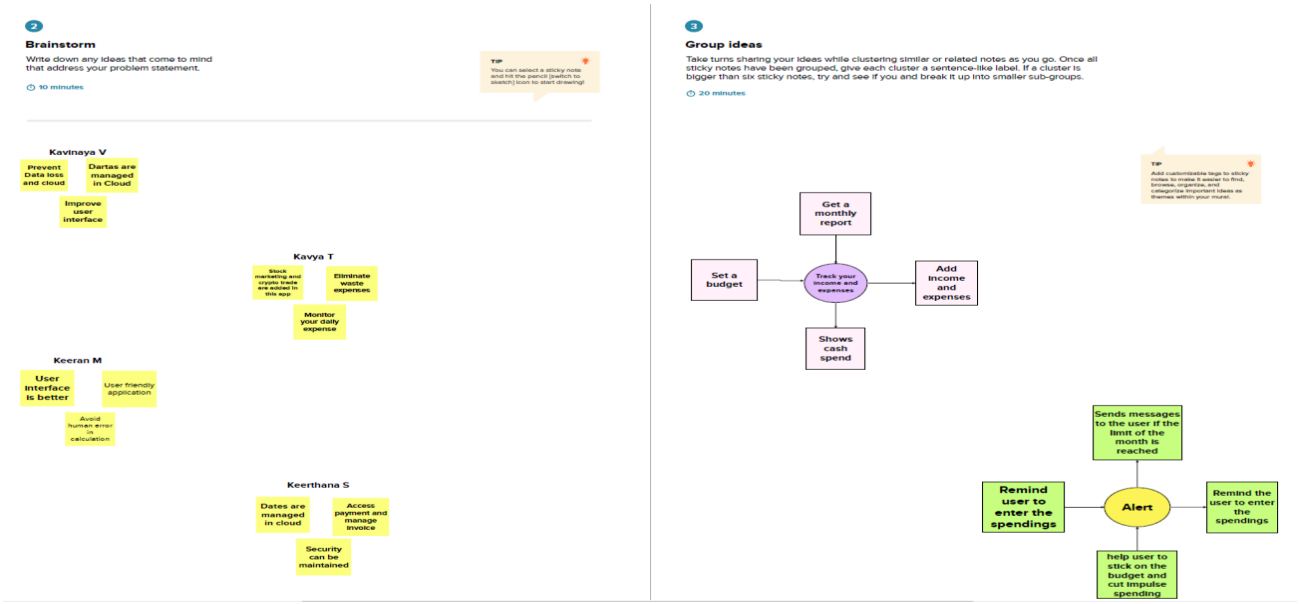
Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Software engineer	Make a monthly budget	It is difficult to analyse	Improper categorizing of expenses	Frustrated
PS-2	Business organization	Keep track of daily expenses	Calculating the monthly cost requires more time	It is hard to update the budget details using a manual system	Confused and irritated

3.IDEATION AND PROPOSED SOLUTION

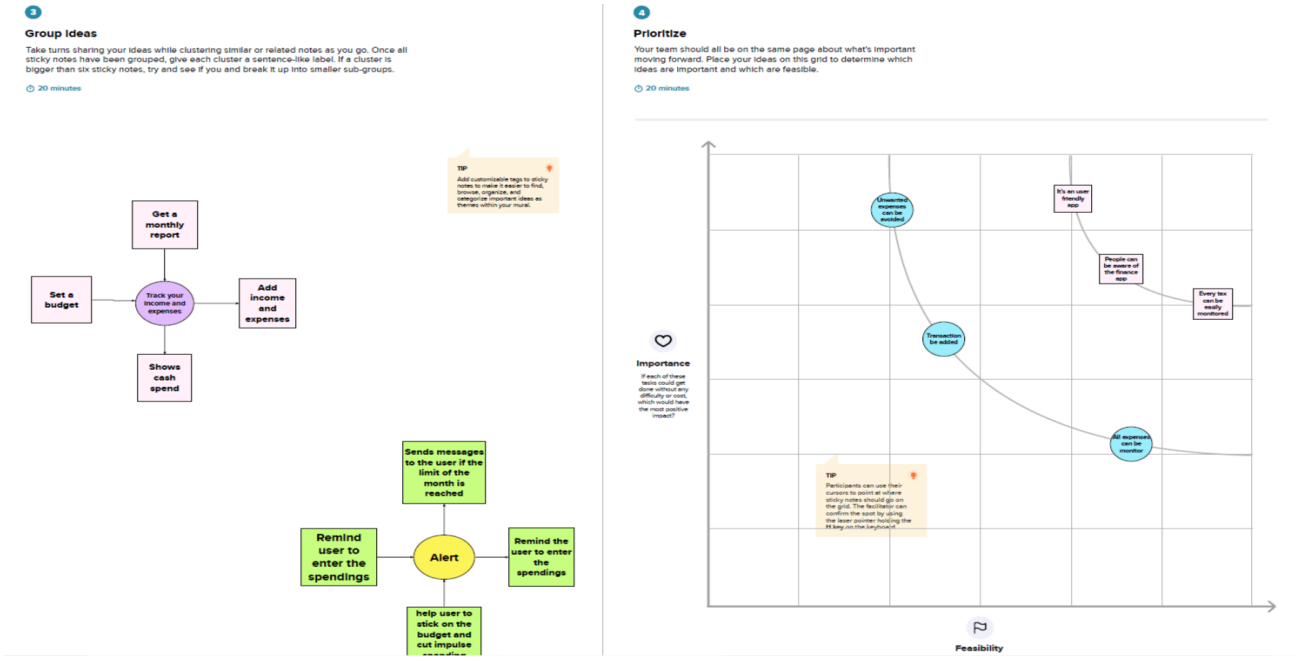
3.1 EMPATHY MAP CANVAS

Empathy Map

3.2 IDEATION AND BRAINSTORMING



Problem Statement & Brainstorm



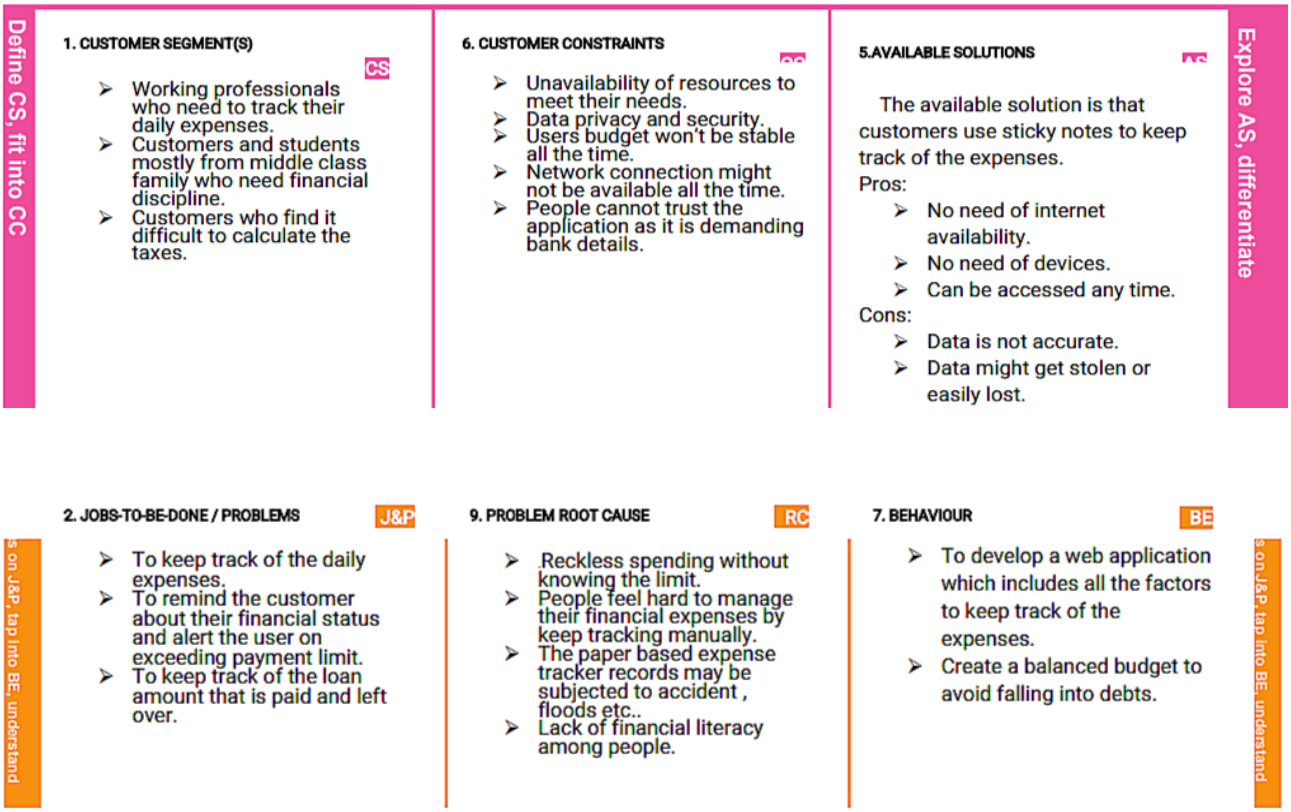
Group Ideas & Prioritize

3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	In today's financial world, money management is the necessary part of life. The proper balance between income and expense is a must for a comfortable livelihood. But in the absence of proper management of money, we left with no savings at all. Generally paper based expense tracking system is used to track the expenses. But the problem is that data might get lost due to flood, accidents etc..
2.	Idea / Solution description	Personal expense tracker app allow users to track their daily expenses so that they do not exceed their budget. This app will not only keep a check on user's expenses but also cut down the unnecessary expenses. This app helps to maintain the financial stability of users.
3.	Novelty / Uniqueness	It displays the expenditure on certain categories in a graphical format. It keeps track of the user's savings and expenditure and notifies to limit the expenditure. It saves time.
4.	Social Impact / Customer Satisfaction	It is used to reduce the unnecessary expenses made by the user. The user will be able to prioritize the spending. It lets you to regulate spending impulses and eliminate worthless spending, thereby avoiding debt. At every point, you will be aware about how much money you are left with.

5.	Business Model (Revenue Model)	It is a good business model. The user can be suggested the right product to buy based on their budget. It provides finance related advertisements. This app is provided for free of cost.
6.	Scalability of the Solution	It provides backup and recovery of data. It can handle large number of users with high performance and data security. Users can track this application anywhere and anytime. This app provides better user interface. Multiple users can access this application and it reduces human errors.

3.4 PROBLEM SOLUTION FIT



3. TRIGGERS TR

- It triggers the customers when they face a financial crisis due to excessive expenditure.
- Lack of awareness about existing solution to track the expenses.

4. EMOTIONS: BEFORE / AFTER EM

Before:

- Unable to track the expenses.
- Fear about the future as they would need relatively more money to provide for a family.
- Feels hopeless in emergency situations.

After:

- Customers are satisfied because they know how much money they spend on a particular category.
- No tensions as they could limit the expenditure by cutting down the unwanted expenses.

10. YOUR SOLUTION

The proposed solution is personal expense tracker application. It helps to track the daily expenses of users and it will alert the users when the expenditure exceeds the budget by sending notifications.

It eliminates the manual tracking which is done by sticky notes, diary etc.. The users can be able to prioritize the spending so that they can able to cut down the unnecessary expenses. It reduces manual errors while calculating expenses. It provides high performance for a large number of users.

8. CHANNELS of BEHAVIOUR CH

Online:

- Users can download the reports of savings and expenditures.
- Use cloud based software for tracking the expenses.
- Social media can be used to influence other people about this app.

Offline:

- Maintain sticky notes to note down the expenses.
- Users find it difficult to track their money during bustle works.

Problem Solution fit

4.REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

FR No	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login through valid username and password
FR-4	Dashboard	User can add the expense and evaluate them by using the provided options. It displays the summary of expenses.
FR-5	Budget Limit	User can be able to set the budget limit for a week or a month. By setting the budget limit, the user can increase the savings by reducing the unwanted expenses.
FR-6	Expense Tracker	It helps to track the expenses by setting the budget limit. It makes the user to categorize the expenses.
FR-7	Report generation	It displays the graphical representation of users expense in the form of pie-chart ,bar graph etc..
FR-8	Alert	Alert the user through mail when the expense exceeds the budget. It gives the notification of every transactions.

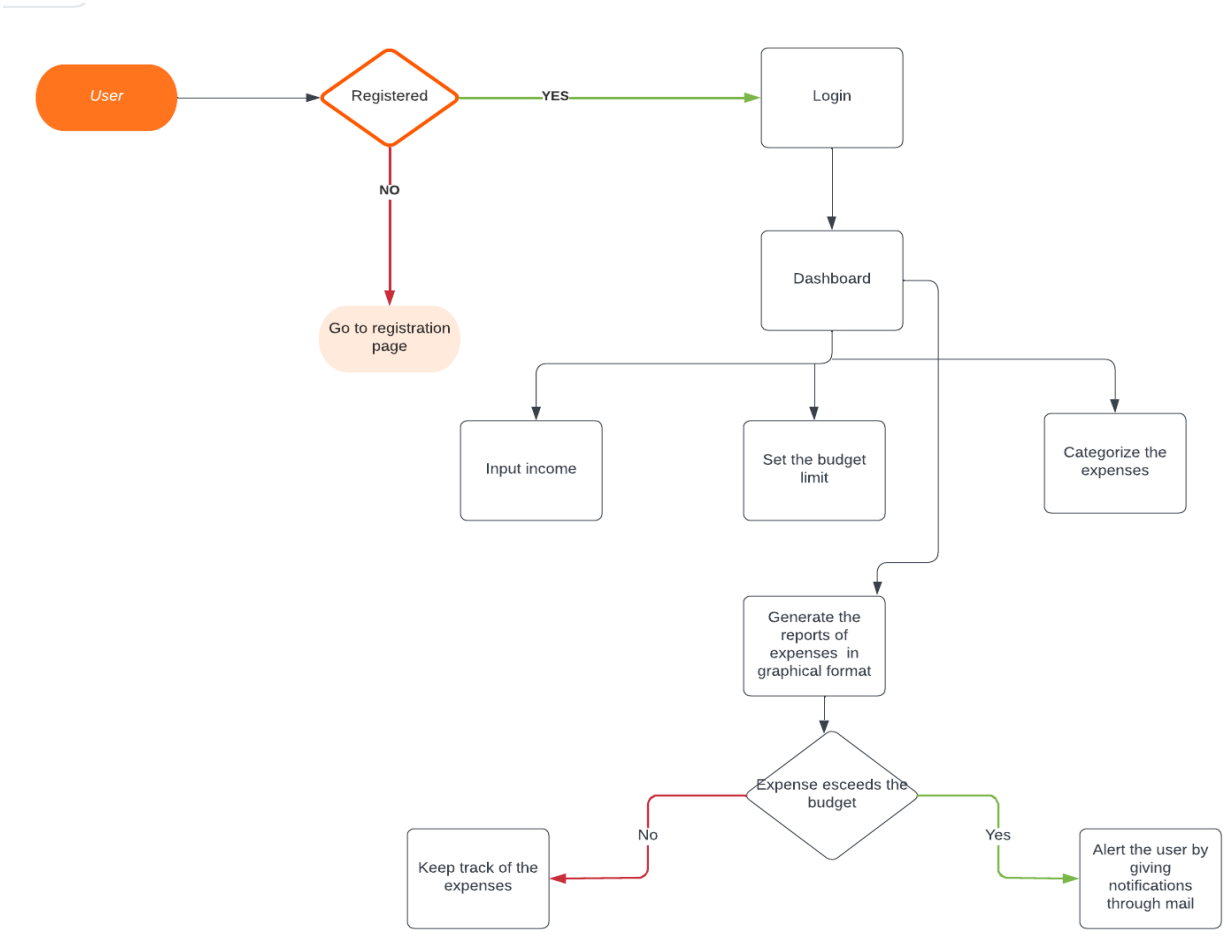
4.2 NON FUNCTIONAL REQUIREMENTS

FR No	Non-Functional Requirement	Description
NFR-1	**Usability**	It helps the users to maintain the accurate track of expenses and savings. It is user friendly. It helps the users to cut down the unwanted expenses.
NFR-2	**Security**	Customers set their own account in this app using their email which is secured by a password. It might prevent from cyber crime activities.
NFR-3	**Reliability**	Each data record is stored in a effective database scheme with high security. There is no possibility of data loss.
NFR-4	**Performance**	It gives users the option to add or delete the categories of expenses and to track the money flow. The performance of this application is considered to be very good because of its lightweight database design.
NFR-5	**Availability**	This application is available for all the time. Users can access this app from any where and any time. It provides portability for the users.
NFR-6	**Scalability**	It has the ability to handle large number of users with high performance and data security.

5.PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

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

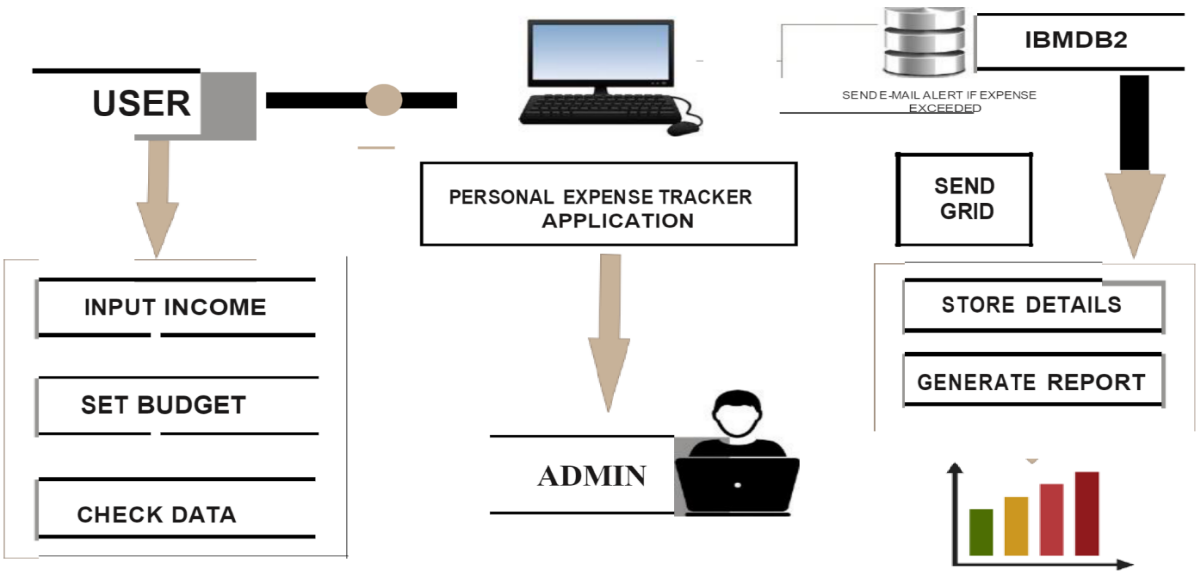


Data Flow Diagram

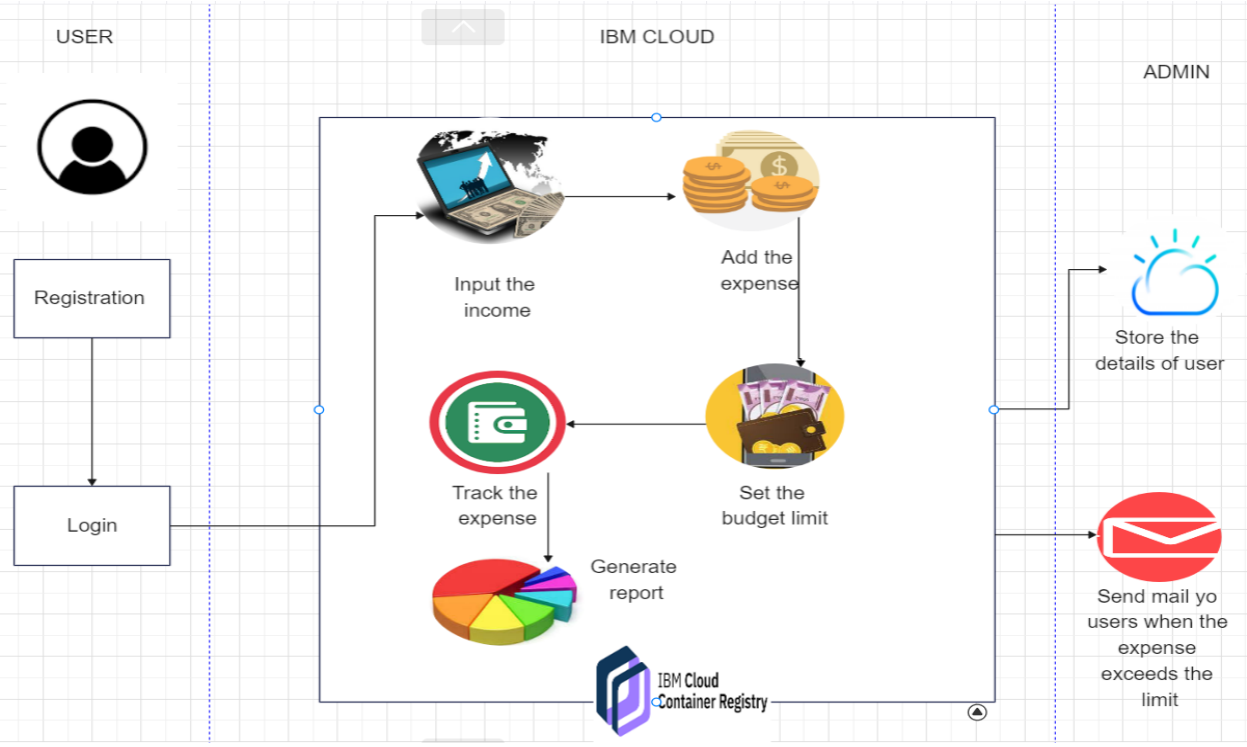
5.2 SOLUTION AND TECHNICAL ARCHITECTURE

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspectsof the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.



Architecture diagram



Technology stack

5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have	I can receive confirmation email & click confirm	High	Sprint-1

			registered for the application			
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register for the application and access the dashboard using gmail	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can login into the application and access it	High	Sprint-1
	Dashboard	USN-6	As a user, I can view my profile and my daily expenses.	I can view the daily expenses and expenditure details	High	Sprint-1
		USN-7	As a user, I can set the monthly budget limit	I can set the limit and reduce the unwanted expenses	Low	Sprint-4
Customer (Web user)		USN-8	As a user, I will get alert message when the expense exceeds the budget	I can track the expenses and increase the savings	High	Sprint-1
Customer Care Executive		USN-9	As a customer care executive, I can easily solve the issues faced by the customer.	I can provide support to the customers any time	Medium	Sprint-4
Administrator	Application	USN-10	As an administrator, I can update the application and provide new features to the user.	I can solve any problems raised by the customer	Medium	Sprint-3

6. PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Kavinaya V, Kavya T
		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Keeran, Keerthana S
	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Kavya T, Keerthana S
	Dashboard	USN-4	After login, it redirects the user to the dashboard	`2	High	Kavinaya V, Keeran
Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only						

Sprint 2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Kavya T, Keeran
	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Kavinaya V, Keerthana S
	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Keeran , Keerthana S
		USN-4	Making dashboard interactive with JS	2	High	Kavinaya V, Kavya T
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Kavinaya V, Keeran
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user's query	1	Medium	Kavya T, Keerthana S
	SendGrid	USN-3	Using SendGrid to send mail to the user about their expenses	1	Low	Kavya T, Keerthana S
		USN-4	Integrating both frontend and backend	2	High	Kavinaya V, Keeran

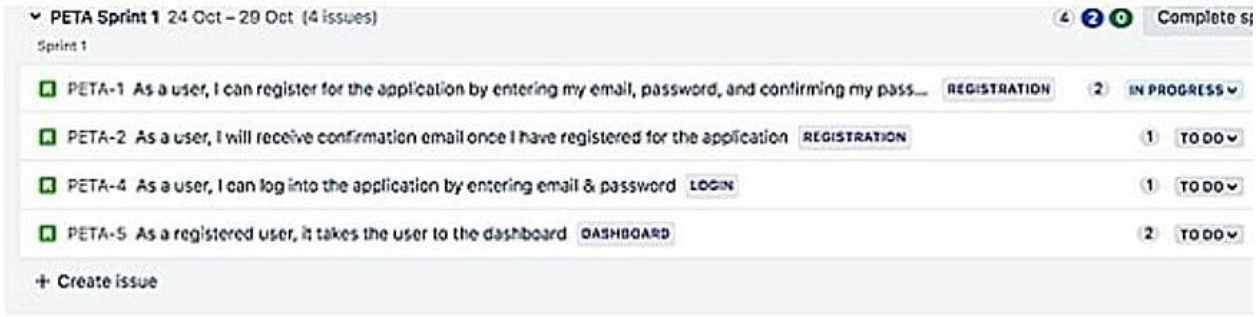
	Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only					
Sprint-4	Docker	USN-1	Creating image of website using docker/	2	High	Kavinaya V, Kavya T, Keeran, Keerthana S
	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Kavinaya V, Kavya T, Keeran, Keerthana S
	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Kavinaya V, Kavya T, Keeran, Keerthana S
	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Kavinaya V, Kavya T, Keeran, Keerthana S

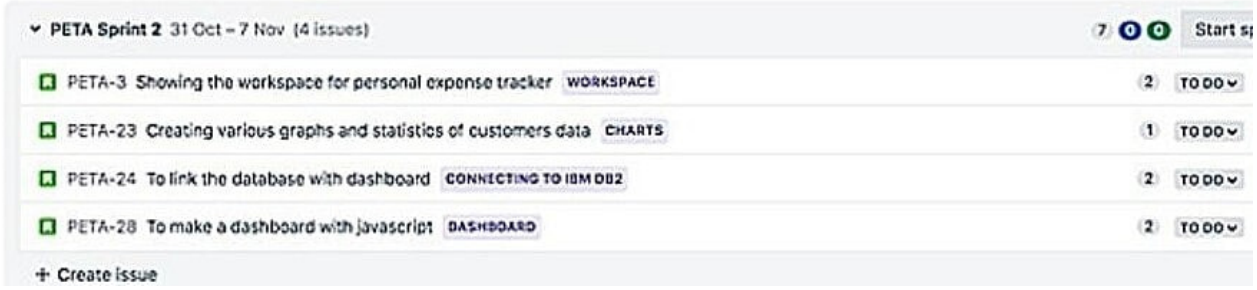
6.2 SPRINT DELIVERY SCHEDULE

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

6.3 REPORTS FROM JIRA

I)BACKLOG

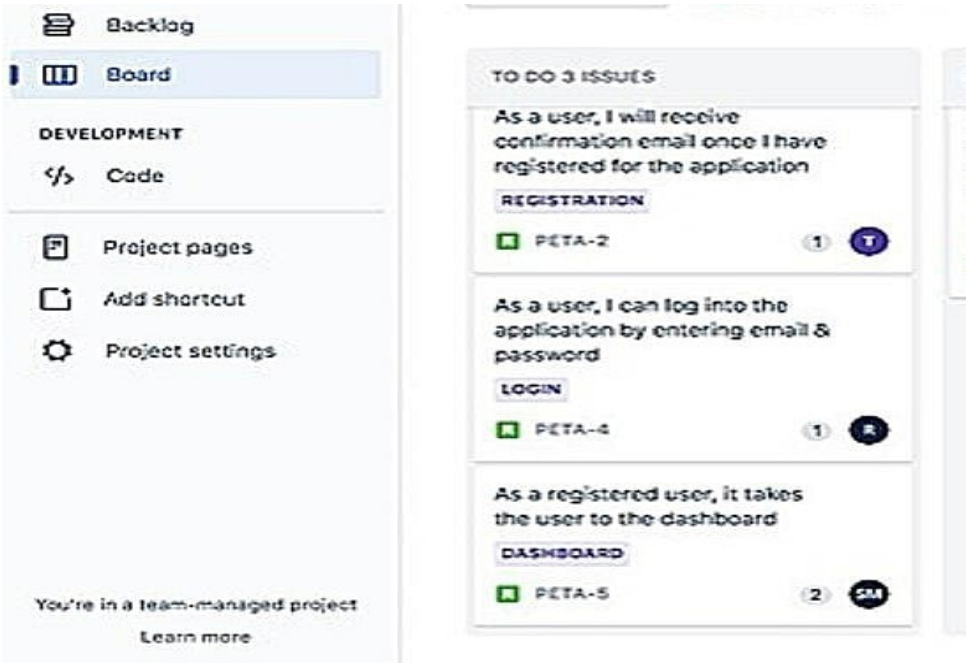

PETA Sprint 1 24 Oct – 29 Oct (4 issues)
Sprint 1
PETA-1 As a user, I can register for the application by entering my email, password, and confirming my pass... REGISTRATION IN PROGRESS
PETA-2 As a user, I will receive confirmation email once I have registered for the application REGISTRATION TO DO
PETA-4 As a user, I can log into the application by entering email & password LOGIN TO DO
PETA-5 As a registered user, it takes the user to the dashboard DASHBOARD TO DO
+ Create issue


PETA Sprint 2 31 Oct – 7 Nov (4 issues)
PETA-3 Showing the workspace for personal expense tracker WORKSPACE TO DO
PETA-23 Creating various graphs and statistics of customers data CHARTS TO DO
PETA-24 To link the database with dashboard CONNECTING TO IBM DB2 TO DO
PETA-28 To make a dashboard with javascript DASHBOARD TO DO
+ Create issue

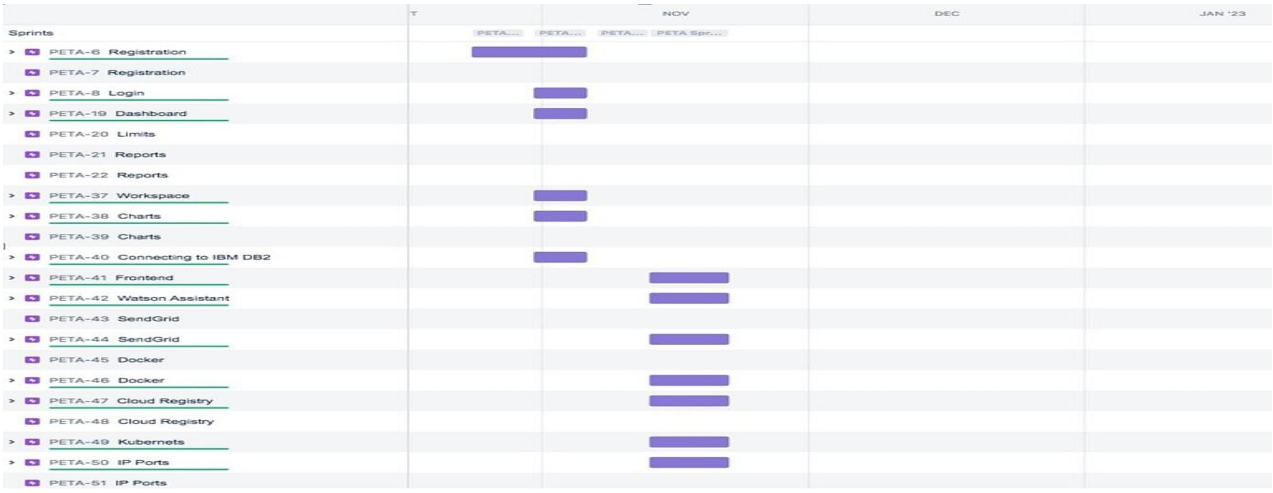

PETA Sprint 3 7 Nov – 14 Nov (4 issues)
PETA-15 To wrap up the server side works of frontend FRONTEND TO DO
PETA-29 Creating chatbot WATSON ASSISTANT TO DO
PETA-31 Integrating SendGrid services SENDGRID TO DO
PETA-32 Integrating both frontend and backend TO DO
+ Create issue


PETA Sprint 4 14 Nov – 21 Nov (4 issues)
PETA-17 To create images of website using docker DOCKER TO DO
PETA-33 To upload docker image to IBM Cloud Registry CLOUD REGISTRY TO DO
PETA-34 To create a container using docker image and hosting the site KUBERNETS TO DO
PETA-36 Exposing IP Ports for the site IP PORTS TO DO
+ Create issue

II)BOARD



III)ROAD MAP



7.CODING AND SOLUTIONING

7.1 FEATURES

Feature 1: Add Expense
Feature 2: Update expense
Feature 3: Delete Expense
Feature 4: Set Limit
Feature 5: Send Alert Emails to users

Other Features: Track your expenses anywhere, anytime. Seamlessly manage your money and budget without any financial paperwork. Just click and submit your invoices and expenditures. Access, submit, and approve invoices irrespective of time and location. Avoid data loss by scanning your tickets and bills and saving in the app. Approval of bills and expenditures in real-time and get notified instantly. Quick settlement of claims and reduced human errors with an automated and streamlined billing process

7.2 CODE

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;protoco
l=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"])
        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)
```

7.3 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

8.1 TEST CASES

S.NO	**Test Cases**	**Result**
1	Verify the user is able to see the Sign up page when the user clicks the signup buttonin navigation bar	Positive
2	Verify the UI elements in the Sign up page	Positive
3	Verify the user is ableto register intothe application by providing valid details	Positive
4	Verify the user is able to see the sign in page when the user clicks the sign in button innavigation bar	Positive
5	Verify the UI elementsin the Sign in page	Positive

6	Verify the user is able to login intothe application by providing valid details	Positive
7	Verify the user is able to see theadd expenses page when the user clicksthe add expenses navigation bar	Positive
8	Verify the UI elements in the add expenses page	Positive
9	Verify the user is ableto add expenses by providing valid details	Positive
10	Verify the user is able to see the home button in the add expensespage	Positive
11	Verify whether the expenses added can be deleted from the cloud	Positive
12	Verify whether the expenses added can be edited	Positive
13	Verify whether the date, month and year are valid while user entering the expenses	Positive
14	Verify whether the datatypes entered by the userin the addexpenses page are floator integers	Positive
15	Verify whether expenses addedpreviously still exist	Positive
16	Verify whether the expenses amountentered fits between in the rangeof integers or float data types	Positive

8.2 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

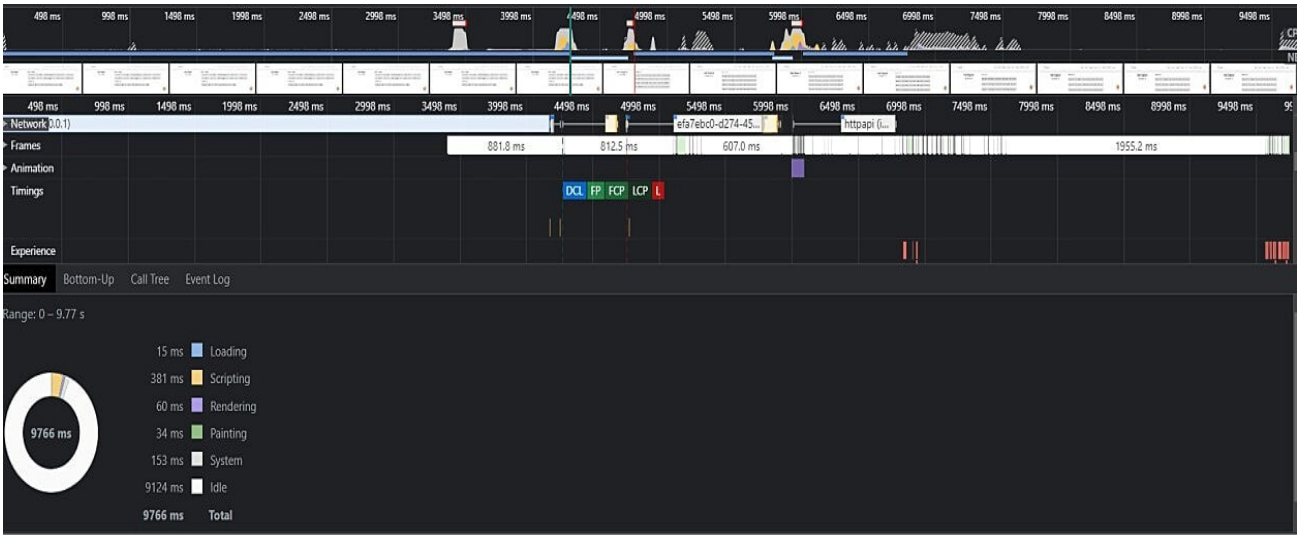
This report shows the number of test cases that have passed, failed, and untested

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.RESULT

9.1 PERFORMANCE METRICES

The performance metrics include page speed, time to title, bounce rate, time to start render, time to interact, DNS lookup speed, Requests per second, error rate, time to first byte/last byte and conversion rate. The performance metrics of our application are efficient and are given below for reference.



Performance metrices

10.ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- It is a user-friendly application.
- Simple user interface
- The best organizations have a way of tracking and handling these reimbursements. This ideal practice guarantees that the expenses tracked are

accurately and in a timely manner. From a company perspective, timely settlements of these expenses when tracked well will certainly boost employees' morale

- Effective expense tracking and reporting will avoid conflict
- It reduces manual errors while calculating
- The users can be able to prioritize the spending so that they can cut down the unnecessary expenses.
- Attracts more, number of users as it is available in the form of Mobile application instead of What‟s app group.
- Usage of this application will greatly reduce time in selecting the right donor.

DISADVANTAGES

- It requires an active internet connection.
- It relays on the details provided by the user.

11.CONCLUSION

Tracking your expenses daily can not only save your amount, but it can also assist you set financial goals for the longer term. If you know exactly where your amount goes every month, you will easily see where some cutbacks and compromises can be made. The project that we have developed is more efficient than the other income and expense trackers. The project successfully avoids the manual calculation which is performed usually in the absence of an expense tracker. The modules are developed efficiently and also in an attractive manner. The application will eliminate sticky notes, spreadsheets, and ledgers that cause confusion, data inconsistency problems while recording and splitting expenses. With our application users can manage their expenses more effectively and they will be better at managing the expenses.Expense Tracker project is for keeping our day-to-day expenditures will helps us to keep record of our money daily.This application would work as a charm for tracking expenses and provides the user, the flexibility to see the reports in any formatThe 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.

12.FUTURE SCOPE

In future the Expense Tracker can be enhanced by adding the feature to “incorporate a shared expense group” apart from keeping a personal expense log.The application could have been more user friendly. For example – if the user keeps track of the daily expenses and spends money at Starbucks everyday, he has to enter all the information (merchant, date, amount etc.) himself all over again. A PDF feature would be implemented so that the user can see the total expenses/incomes in a much simpler PDF format in one file.

13.APPENDIX

Github and Project demo link

https://github.com/IBM-EPBL/IBM-Project-17215-1659630829

https://drive.google.com/drive/folders/1RsRJEV6Q824An4bIoGoUiiMkKbpKp55w