

**PERSONAL EXPENSE  
TRACKER APPLICATION  
IBM-Project-38339-1660378735**

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

**A PROJECT REPORT  
BY**

Sreenidhi B(922519104156)

Shangamithra V(922519104146)

Sudeepa R(922519104163)

Yogetha S(922519104183)

**BACHELOR OF ENGINEERING  
(COMPUTER SCIENCE AND ENGINEERING)**

**VSB ENGINEERING COLLEGE,KARUR -639111**

## **1. INTRODUCTION**

- a. Project Overview
- b. Purpose

## **2. LITERATURE SURVEY**

- a. Existing problem
- b. References
- c. Problem Statement Definition

## **3. IDEATION & PROPOSED SOLUTION**

- a. Empathy Map Canvas
- b. Ideation & Brainstorming
- c. Proposed Solution
- d. Problem Solution fit

## **4. REQUIREMENT ANALYSIS**

- a. Functional requirement
- b. Non-Functional requirements

## **5. PROJECT DESIGN**

- a. Data Flow Diagrams
- b. Solution & Technical Architecture
- c. User Stories

## **6. PROJECT PLANNING & SCHEDULING**

- a. Sprint Planning & Estimation
- b. Sprint Delivery Schedule
- c. Reports from JIRA

## **7. CODING & SOLUTIONING (Explain the features added in the project along with code)**

- a. Feature 1
- b. Feature 2
- c. Database Schema (if Applicable)

## **8. TESTING**

- a. Test Cases
- b. User Acceptance Testing

## **9. RESULTS**

- a. Performance Metrics

## **10. ADVANTAGES AND DISADVANTAGES**

## **11. CONCLUSION**

## **12. FUTURE SCOPE**

## **13. APPENDIX**

Source Code

GitHub & Project Demo Link

TEAM ID : PNT2022TMID33435

INDUSTRY MENTOR : Kusboo

FACULTY MENTOR : Manikandan.T

### **Skills Required:**

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

## **1. INTRODUCTION**

### **a. Project Overview**

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

### **b. Purpose**

The “Expense Tracker” is developed using Angular 8 for front end and SQL lite for back end. Expense Tracker helps to maintain the record of daily expenses and monthly income of an users from anywhere and also generates a monthly report of the expenses in pdf format. The Expense Tracker app tracks all the expenses and helps the user to manage his/her expenses so that the user is the path of financial stability. The Tracking of expenses is categorised by week, month

and year, it helps to see the more expenses made. To use the Expense Tracker the user has to sign up into such as name, phone no., address, email address, username, password and confirm password of the user. The user can get enlisted just a single time, per user can just one record. The remainder is set if the type future expense. The whole subtleties of the income or expense can be seen or refreshed or can be erased by long pressing the specific rundown thing. The things in the rundown can be separated by month, year and date. When the month's end is arrived at the complete pay, all out past expense and all-out future expense are determined and shown for the user.

## 2. LITERATURE

### a. Existing problem

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 iPhone device, designed and marketed by Apple Inc., is one of the top-selling smartphones in the USA, and with the launch of the new iPhone5 on September 21, 2012, whose sales have already surpassed the previous iPhone handsets (iPhone 4S, iPhone4) sales, it is apparent that people have been using smartphones as an organizational tool. XpensTrak, the Expense Tracker Mobile Application was developed for iPhone users to keep track of their expenses and determine whether they are spending as per their set budget. Potential users need to input the required data such as the expense amount, merchant, category, and date when the expense was made. Optional data such as sub-category and extra notes about the expense can be entered as well. 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. An analysis comparing existing expense tracking software with the one being introduced is provided..

<b>I am</b>	Mr.xyz, who needsto track all hisfinancial transactions.
<b>I'm trying to</b>	Reduce my dailyexpenses as I am spending a lot withoutany limitations.
<b>But,</b>	I could not control and monitor my daily expenses.

<b>Because</b>	I am trying to have a report on all financial actions on a daily, weekly, monthly, and annual basis.
<b>Which makes me feel</b>	Relaxed and helps me to access all information at any time and from any location and saved in offline mode.

## b. References

## Personal Expense Tracker Application:

<b>I am</b>	<b>I'm Trying to</b>	<b>But</b>	<b>Because</b>	<b>Which Makes Me Feel</b>
Mr. xyz, who needs to track all his financial transactions.	Reduce my daily expenses as I am spending a lot without any limitations	I could not control and monitor my daily expenses.	I am trying to have a report on all financial actions on a daily, weekly, monthly, and annual basis.	Relaxed and helps me to access all information at any time and from any location

### c. Problem Statement Definition

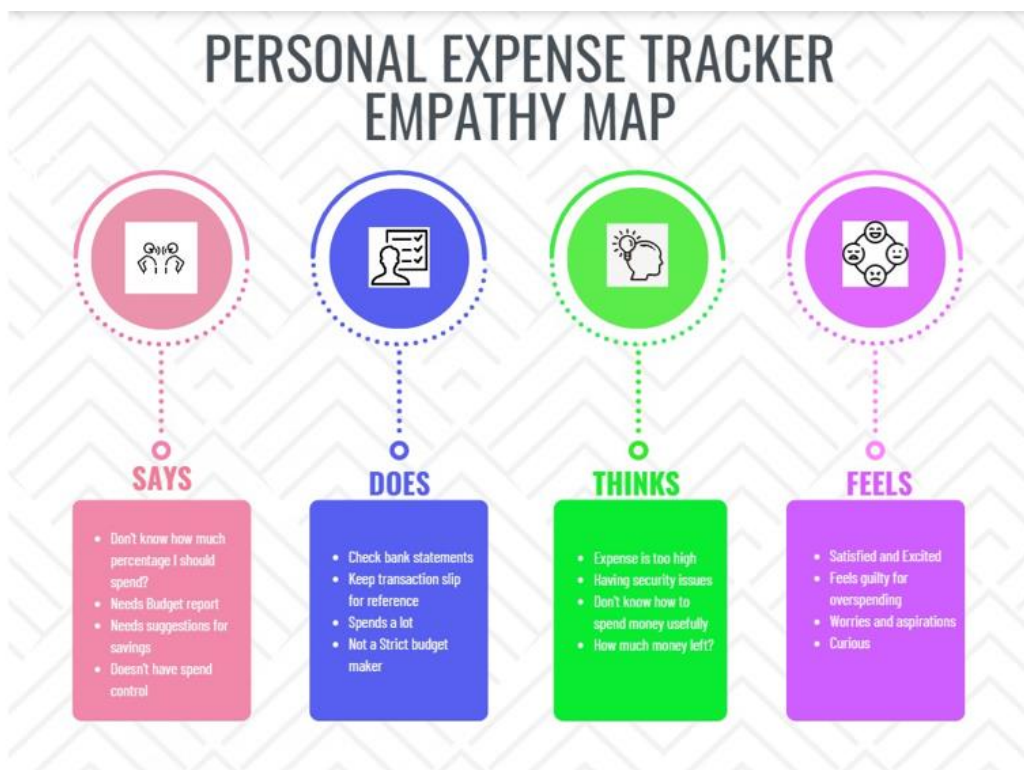
Mr.xyz can use a personal expense tracker, which is a particular type of digital diary that assists in keeping track of all of our financial transactions and offers a report on all financial actions on a daily, weekly, monthly, and annual basis. Mr.xyz will be reminded to enter expenses and incomes so that the application's tracking system can use them. Mr.xyz can simply access all information at any time and from any location because it is saved in offline mode. The Daily Expense Tracker's user interface is very simple and elegant, making it simple to grasp and the finest approach to record our financial data.

<b>Problem Statement (PS)</b>	<b>I am</b>	<b>I'm trying to</b>	<b>But</b>	<b>Because</b>	<b>Which makes me feel</b>
PS-1	Busy Employee	Keep track of daily expenses.	It is uncomfortable for me to make calculations manually.	I don't feel that I am good at doing calculations.	Doubtful and insecure.
PS-2	Busy Employee	Have an update on spending to reduce the expense.	It is not possible by paper and pen calculation.	I don't have enough time to calculate it and I may miss some expenses.	To me and around me have difficulty in tracking expenses manually.

### 3. IDEATION & PROPOSED SOLUTION

#### a. Empathy Map Canvas

At the instant, there is no such complete solution present easily or we should say free of cost Which enables a person to keep a track of its daily expenditure easily. To do so a person has to keep a log in a diary or in a computer, also all the calculations needs to be done by the user which may sometimes result in errors leading to losses. Due to lack of a complete tracking system, there is a constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.






## b. Ideation &amp; Brainstorming

**Step 1: Team Gathering, Collaboration and Select the Problem Statement**

**Brainstorm**



## PERSONAL EXPENSE TRACKER -IDEATION

At the instant, there is no such complete solution present easily or we should say free of cost. Which enables a person to keep a track of its daily expenditure easily. To do so a person has to keep a log in a diary or in a computer, also all the calculations needs to be done by the user which may sometimes result in errors leading to losses. Due to lack of a complete tracking system, there is a constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.

**Before you collaborate**

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

- A** Team gathering  
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.
- B** Set the goal  
Think about the problem you'll be focusing on solving in the brainstorming session.
- C** Learn how to use the facilitation tools  
Use the Facilitation Superpowers to run a happy and productive session.

**1 Define your problem statement**

Tracking expenses involves identifying expenditures throughout the month. Another reason you must identify your expenditures throughout the month is to become more aware of your spending habits.

PROBLEM

We should have a habit of tracking our expenses. If you don't know where your money is going, you won't be able to recognize negative spending behaviors that you can easily change to make your money work for you.

Key rules of brainstorming

To run a smooth and productive session

Stay in topic.

Defer judgment.

Go for volume.

Encourage wild ideas.

Listen to others.

If possible, be visual.

**Step 2 : Brainstorm, Idea listing and Grouping**

**Brainstorm**  
Write down any ideas that come to mind that address your problem statement.

**SREENIDHI B**

Login to the App

Generate Daily Report

Connect to Payment Apps

Virtualize the expenses

**SHANGAMITHRA V**

Set Budget

Add Income

Categorize Expenses

Curated and focused

**SUDEEPA R**

See Expense Graphically

Edit Expenses per day

Show income and expense separately.

Improved on time performance

**YOGETHA S**

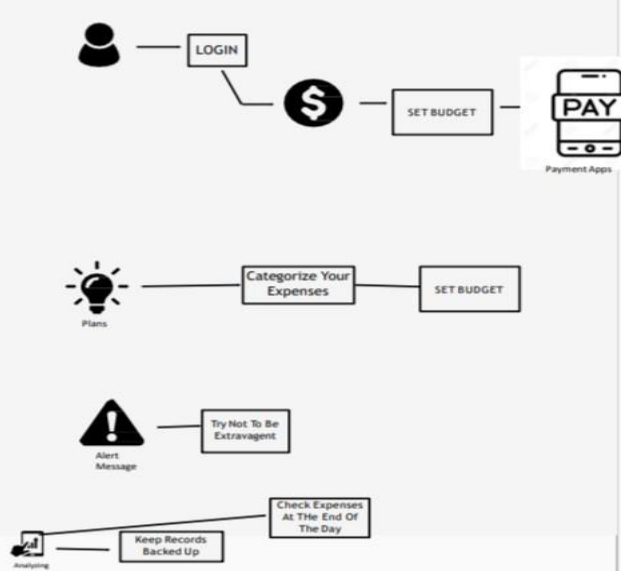
Alert Message

Different language options

Analyze Your Expense At The End Of The Day

Save some amount Of Salary For Exceptional Cases

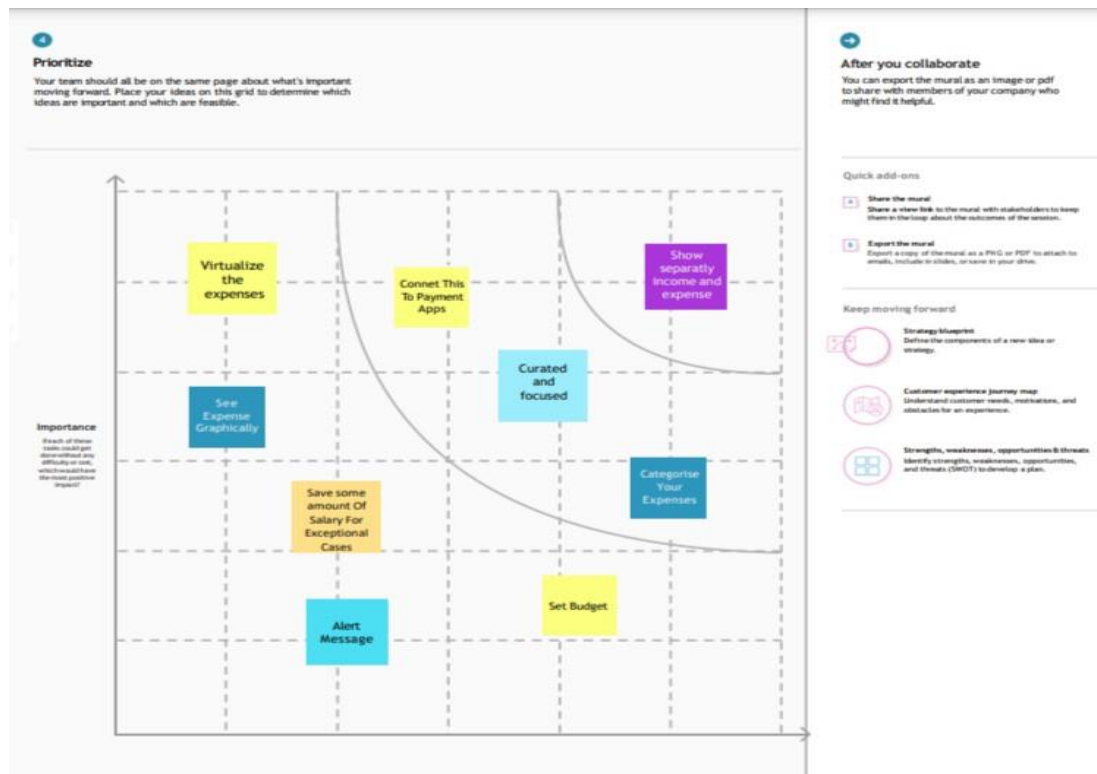
**3 Group ideas**  
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.



```

graph LR
    User((User)) --> LOGIN[LOGIN]
    LOGIN --> Budget((Budget))
    Budget --> SETBUDGET[SET BUDGET]
    SETBUDGET --> PAY[PAY]
    PAY --- PaymentApps[Payment Apps]
    
    Plans((Plans)) --> Categorize[Categorize Your Expenses]
    Categorize --> SETBUDGET
    
    Alert((Alert Message)) --> TryNot[Try Not To Be Extravagant]
    
    Analyzing((Analyzing)) --> Backup[Keep Records Backed Up]
    Backup --> Check[Check Expenses At The End Of The Day]
  
```

### Step 3 : Idea Prioritization



### c. Proposed Solution



S. No.	Parameter	Description
1.	PROBLEM STATEMENT (PROBLEM TO BE SOLVED)	The Expenses Tracker System was created to monitor users' expenditures and forecast their monthly budget. Users should be notified if they have gone over their budget by the system, which should be able to provide reports of their expenditure. Since prediction was made using the least square method, the system is intended to be dynamic. Users' personal information, income, and expenses are also provided via the system. The Expenses Tracker System is a method for consumers to better manage their spending in more effective manners.
2.	IDEA / SOLUTION DESCRIPTION	The goal is to move from, at most, a spreadsheet to, at worst, some sort of collection and

		categorization of money spent. That is something that apps are capable of. The experience, though, differs greatly. These are the most widely used apps for iOS and Android, but you'll probably need to test out a few before you discover one you enjoy. While the majority of the apps on our list are free, you'll probably need to upgrade for a small cost to access all the features.
3.	NOVELTY / UNIQUENESS	It's time to quit maintaining records of your cash payments and online transactions on paper and in Excel spreadsheets! Papers are more hazardous to the environment and difficult to handle. Excel sheets, on the other hand, might provide an online solution but aren't very helpful for handling money. Therefore, it is preferable to create money management software that gathers information from the data and aids in corporate decision-making. With regards to keeping track of spending, you are never able to locate the cash or digital payments that you have made. Therefore, you should use a business spending tracker app if you want to keep track of your financial assets. By simply clicking on the photos of the receipts in your costs management app, you may store every receipt.
4.	SOCIAL IMPACT / CUSTOMER SATISFACTION	Tracking your spending is often the first step in getting your finances in order.. You can see exactly where your money is going and where you may make savings by analysing method. With the help of mobile spending tracker applications, you can easily include this into your daily routine. These apps do overlap with budgeting tools, but cost tracker apps focus more on your spending while the latter offers a broad overview of your finances. These apps typically classify your expenses and give you a clear picture of your purchasing behaviour. Whether the app to track your expenses that can quickly record every transaction.
5.	BUSINESS MODEL (FINANCIAL BENEFITS)	One of the main purposes of an expense tracker is to assist you in keeping an orderly and comprehensive record of various expenses. You

		must take pictures and screenshots of every receipt and save them through the tracker app in order to manage transaction records and arrange them nearly there. The receipts and records that are maintained in the cloud when you create an expenditure tracking app enable you to access and review them whenever necessary.
6.	SCALABILITY OF SOLUTION	An expense tracker, often known as an expense manager or money manager, is a piece of software or an application that assists in maintaining accurate records of your money coming in and going out.

### d. Problem Solution Fit

Problem-Solution fit canvas 2.0		Purpose / Vision of Personal Expense Tracker	
Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span> Who is your customer? The one who spends more money without any limitations.	<b>6. CUSTOMER CONSTRAINTS</b> <span>CC</span> What constraints prevent your customers from taking action or limit their choices of solutions? Budget friendly, Saves amount, available saving options.	<b>5. AVAILABLE SOLUTIONS</b> <span>AS</span> Which solutions are available to the customers when they face the problem or need to get the job done? Pen and paper is an alternative to digital expense tracker.
	Focus on J&P, tap into BE, understand RC	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <span>J&amp;P</span> Which jobs-to-be-done (or problems) do you address for your customers? To keep their transaction slip for record.	<b>9. PROBLEM ROOT CAUSE</b> <span>RC</span> What is the real reason that this problem exists? Customers have to do it because they are spending more money without any limitations.
<b>3. TRIGGERS</b> <span>TR</span> What triggers customers to act? Their expense is too high.		<b>10. YOUR SOLUTION</b> <span>SL</span> Expense tracker solutions will automate the entire traditional expense workflow and integrate with multiple digital solutions to improve the budget planning.	<b>8. CHANNELS OF BEHAVIOUR</b> <span>CH</span> <b>8.1 ONLINE</b> What kind of actions do customers take online? Extract online receipts <b>8.2 OFFLINE</b> What kind of actions do customers take offline? Extract offline receipts from #7 and use it for making budget.
Identify strong TR & EM	<b>4. EMOTIONS</b> <span>EM</span> BEFORE-> Feels guilty and Worried AFTER-> Satisfied and Excited		Extract online & offline CH of BE

## 4. REQUIREMENT ANALYSIS

### a. Functional Requirements:

FR No.	Functional Requirements(Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through FormRegistration through Gmail
FR-2	User Confirmation	Enter the valid username and password
FR-3	Login	Enter the valid username and password
FR-4	Calender	Personal expense tracker application shall allow user to add the data to their expenses.
FR-5	Expense Tracker	This application should graphically represent the expense in the form of report.
FR-6	Report generation	Report generation through Message Report generation through Gmail
FR-7	Category	This application shall allow users to add categories of their expenses.

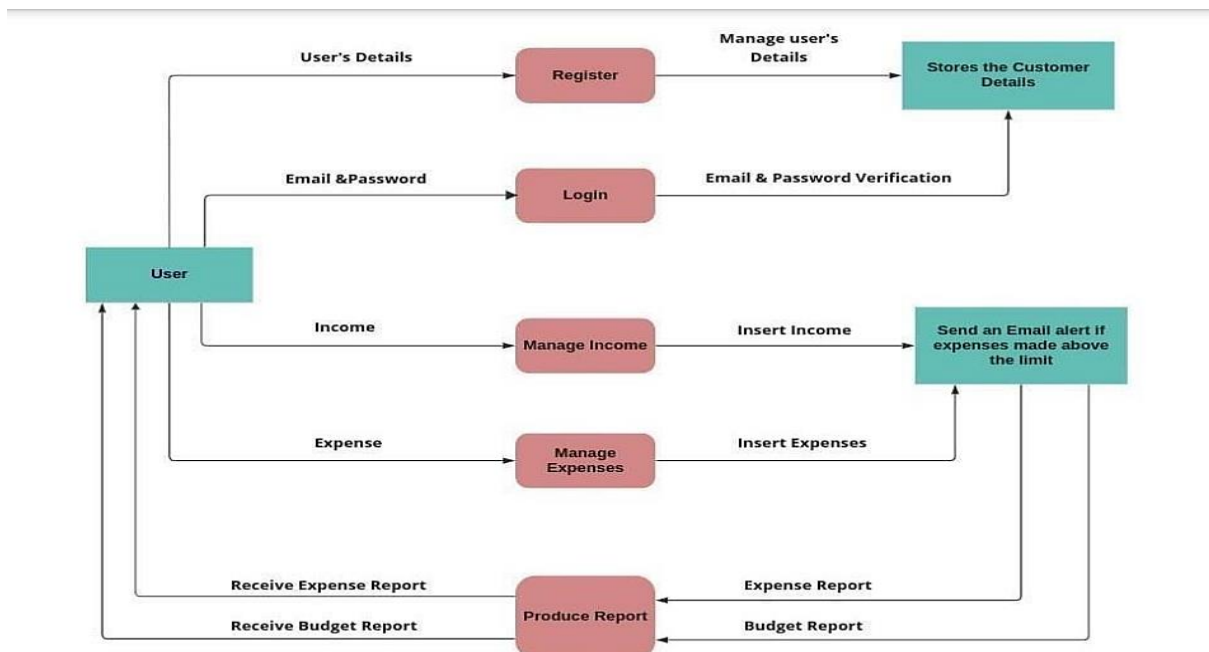
### b. Non-functional Requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Helps to keep an accurate record of your moneyinflow and outflow.

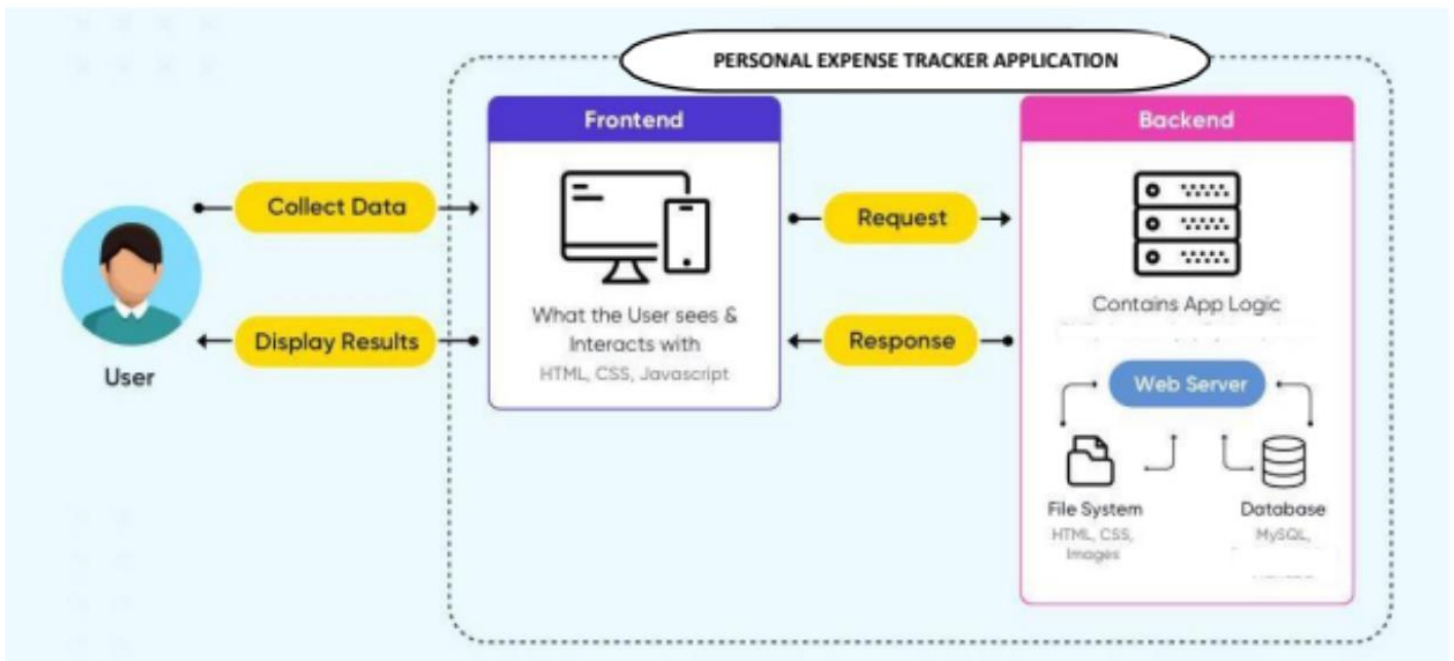
NFR-2	Security	Budget tracking apps are considered very safe from cybercriminals.
NFR-3	Reliability	Each data record is stored on a well built efficient database schema. There is no risk of data loss
NFR-4	Performance	The types of expense are categories along with an option. Throughput of the system is increased due to light weight database support.
NFR-5	Availability	It is available all the time. No time constraint
NFR-6	Scalability	The ability to appropriately handle increasing demands.

## 5. PROJECT DESIGN

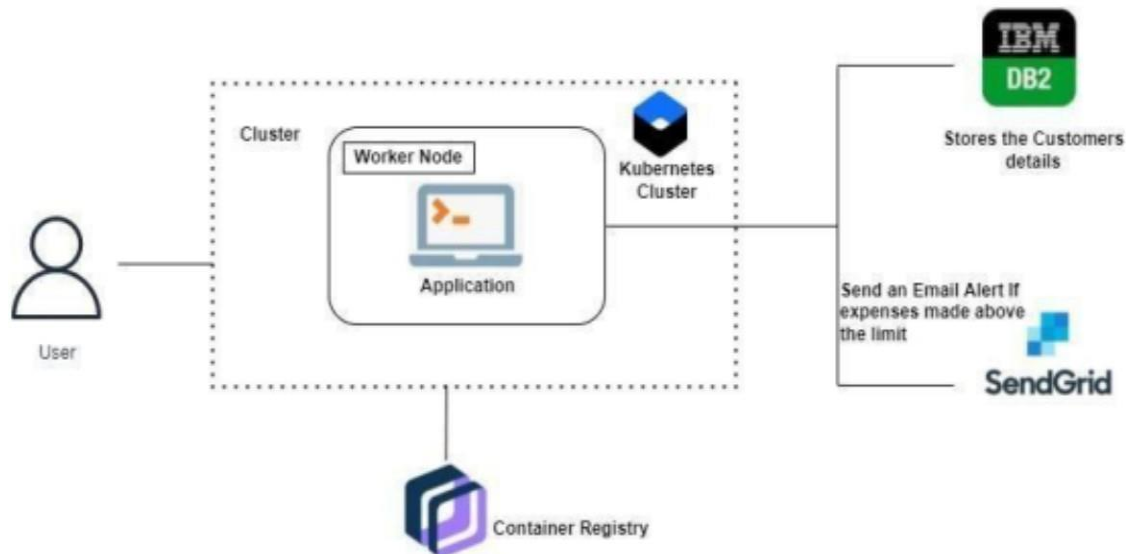
### a. Data Flow Diagram



## b. Solution and Technical Architecture



### Technical Architecture:



### c. User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority
Customer (Mobile user)	Registration	USN-1	As a user, I can register to the application by providing my email, password, and confirming my password.	I can access my account / dashboard	High
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low
	Login	USN-4	As a user, I can log into the application by entering email & password	I can access the application	High
	Dashboard	USN-5	As a user I can enter my income and expenditure details.	I can view my daily expenses	High
Customer Care Executive		USN – 6	As a customer care executive, I can solve the log in issues and other issues of the application.	I can provide support or solution at any time 24*7	Medium
Administrator	Application	USN – 7	As an administrator I can upgrade or update the application.	I can fix the bug which arises for the customers and users of the application	Medium

## 6. PROJECT PLANNING & SCHEDULING

### a. 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	Sreenidhi
		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Shangamithra
	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Sudeepa
	Dashboard	USN-4	Logging in takes to the dashboard for the logged user.	2	High	Yogetha
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	Sreenidhi
	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Shangamithra
	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Sudeepa
		USN-4	Making dashboard interactive with JS	2	High	Yogetha
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Sreenidhi
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user's query	1	Medium	Shangamithra
	SendGrid	USN-3	Using SendGrid to send mail to the user about their expenses	1	Low	Sudeepa
		USN-4	Integrating both frontend and backend	2	High	Yogetha
Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only						
	Docker	USN-1	Creating image of website using docker/	2	High	Sreenidhi
	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Shangamithra
Sprint-4	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Sudeepa
	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Yogetha

## b. Sprint Delivery Schedule

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

## c. Reports from JIRA

### i) Board

The screenshot shows the Jira Board view for the 'Personal Expense Tracker Application'. The board is titled 'PETA Sprint 1' and has a 'Complete sprint' button. The board is divided into four columns: 'TO DO 3 ISSUES', 'IN PROGRESS 1 ISSUE', 'IN REVIEW 1 ISSUE', and 'DONE 1 ISSUE'. The 'TO DO' column contains three issues: 'PETA-5' (REGISTRATION), 'PETA-7' (REGISTRATION), and 'PETA-10' (LOGIN). The 'IN PROGRESS' column contains one issue: 'PETA-2' (REGISTRATION). The 'IN REVIEW' column contains one issue: 'PETA-1' (REGISTRATION). The 'DONE' column contains one issue: 'PETA-3' (REGISTRATION). The left sidebar shows the project navigation menu with 'Board' selected. The top navigation bar includes 'Jira Software', 'Your work', 'Projects', 'Filters', 'Dashboards', 'People', 'Apps', and 'Create'. The bottom status bar shows the date and time as 16-11-2022, 14:26.

### ii) Roadmap

The screenshot shows the Jira Roadmap view for the 'Personal Expense Tracker Application'. The roadmap is titled 'Roadmap' and has a 'View settings' button. The roadmap is divided into four columns: 'OCT', 'NOV', 'DEC', and 'JAN '23'. The 'OCT' column contains three sprints: 'PETA-8 Registration', 'PETA-9 Login', and 'PETA-11 Comment'. The 'NOV' column contains one sprint: 'PETA-10 Login Page'. The 'DEC' column contains one sprint: 'PETA-10 Login Page'. The 'JAN '23' column contains one sprint: 'PETA-10 Login Page'. The left sidebar shows the project navigation menu with 'Roadmap' selected. The top navigation bar includes 'Jira Software', 'Your work', 'Projects', 'Filters', 'Dashboards', 'People', 'Apps', and 'Create'. The bottom status bar shows the date and time as 16-11-2022, 14:25.

## iii) Backlog

Personal Expense Tracker Application  
Software project

PLANNING  
Roadmap  
Backlog  
Board  
DEVELOPMENT  
Code  
Project pages  
Add shortcut  
Project settings

Projects / Personal Expense Tracker Application

## Backlog

Search filters: SB, S, YS, SV, Epic

Does your team need more from Jira? Get a free trial of our Standard plan.

▼ PETA Sprint 1 23 Oct – 30 Oct (6 issues) 0 5 5 Complete sprint

- PETA-3 As a user, I can register for the application through Facebook. REGISTRATION 5 DONE
- PETA-1 As a user, I can register to the application by providing my email, password, and confirming my password. REGISTRATION 3 IN REVIEW
- PETA-2 As a user, I will receive confirmation email once I have registered for the application. REGISTRATION 2 IN PROGRESS
- PETA-5 As a user I can enter my income and expenditure details. REGISTRATION TO DO
- PETA-7 As an administrator I can upgrade or update the application. REGISTRATION TO DO
- PETA-10 Login Page LOGIN TO DO

+ Create issue

▼ Backlog (2 issues)

- PETA-4 As a user, I can log into the application by entering email & password. TO DO

Quickstart

31°C Cloudy  
14:26 16-11-2022

## 7. CODING & SOLUTIONING

**app.py:**

```
# -*- coding: utf-8 -*-
```

```
"""
```

Spyder Editor

This is a temporary script file.

```
"""
```

```
from flask import Flask, render_template,request, redirect,session
```

```
# from flask_mysqldb import MySQL
```

```
# import MySQLdb.cursors
```

```
import re
```

```
from flask_db2 import DB2
```

```
import ibm_db
```

```
import ibm_db_dbi
```

```
from sendemail import sendgridmail,sendmail
```

```
# from gevent.pywsgi import WSGIServer
```

```
import os
```

```
app = Flask(__name__)
```

```
app.secret_key = 'a'
```

```
#app.config['MYSQL_HOST'] = 'remotemysql.com'
```

```
# app.config['MYSQL_USER'] = 'D2DxDUPBii'
```

```
#app.config['MYSQL_PASSWORD'] = 'r8XBO4GsMz'
```

```
# app.config['MYSQL_DB'] = 'D2DxDUPBii'
```

```
''''''
```

```
dsn_hostname = "3883e7e4-18f5-4afe-be8c-  
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"
```

```
dsn_uid = "sbb93800"
```

```
dsn_pwd = "wobsVLm6ccFxcNLe"
```

```
dsn_driver = "{IBM DB2 ODBC DRIVER}"
```

```
dsn_database = "bludb"
```

```
dsn_port = "31498"
```

```
dsn_protocol = "tcpip"
```

```
dsn = (
```

```
"DRIVER={0};"
```

```
"DATABASE={1};"
```

```
"HOSTNAME={2};"
```

```
"PORT={3};"
```

```
"PROTOCOL={4};"
```

```
"UID={5};"
```

```
"PWD={6};"
```

```
).format(dsn_driver, dsn_database, dsn_hostname, dsn_port,  
dsn_protocol, dsn_uid, dsn_pwd)
```

```

"""

```

```

# app.config['DB2_DRIVER'] = '{IBM DB2 ODBC DRIVER}'

app.config['database'] = 'bludb'

app.config['hostname'] = '3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'

app.config['port'] = '31498'

app.config['protocol'] = 'tcpip'

app.config['uid'] = 'sbb93800'

app.config['pwd'] = 'wobsVLm6ccFxcNL'

app.config['security'] = 'SSL'

try:

    mysql = DB2(app)

```

```

    conn_str='database=bludb;hostname=3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=31498;protocol=tcpip;\

```

```

uid=sbb93800;pwd=wobsVLm6ccFxcNL;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)
PNT2022TMID09631

    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)', (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_co
nn, 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_con
```

```
n, 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'])))

    # texpanse = cursor.fetchall()

    # print(texpanse)

    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)
```

```
texpanse = []
```

```
while dictionary1 != False:
```

```
temp = []
```

```
temp.append(dictionary1["TN"])
```

```
temp.append(dictionary1["AMOUNT"])
```

```
texpanse.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", texpanse = texpanse, 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'])))
```

```
    # texpanse = cursor.fetchall()
```

```
    # print(texpanse)
```

```
    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)
```

```
    texpanse = []
```

```
    while dictionary1 != False:
```

```
        temp = []
```

```
        temp.append(dictionary1["DT"])
```

```
        temp.append(dictionary1["TOT"])
```

```
        texpanse.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", texpanse = texpanse, 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'])))
```

```

# texpanse = cursor.fetchall()

# print(texpanse)

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)

texpanse = []

while dictionary1 != False:

    temp = []

    temp.append(dictionary1["MN"])

    temp.append(dictionary1["TOT"])

    texpanse.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", texpanse = texpanse, expense  
= expense, total = total ,
```

```
t_food = t_food,t_entertainment = t_entertainment,
```

```
t_business = t_business, t_rent = t_rent,
```

```
t_EMI = t_EMI, t_other = t_other )
```

```
#log-out
```

```
@app.route('/logout')
```

```
def logout():
```

```
    session.pop('loggedin', None)
```

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

```
    session.pop('username', None)
```

```
    session.pop('email', None)
```

```
    return render_template("home.html")
```

```
port = os.getenv('VCAP_APP_PORT', '8080')
```

```
if __name__ == "__main__":
```

```
    app.secret_key = os.urandom(12)
```

```
    app.run(debug=True, host='0.0.0.0', port=port)
```

**deployment.yaml:**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sakthi-flask-node-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: flasknode
  template:
    metadata:
      labels:
        app: flasknode
    spec:
      containers:
        - name: flasknode
          image: icr.io/sakthi_expense_tracker2/flask-template2
          imagePullPolicy: Always

      ports:
        - containerPort: 5000
```

**flask-service.yaml:**

```
apiVersion: v1
kind: Service
metadata:
  name: flask-app-service
```

spec:

selector:

app: flask-app

ports:

- name: http

protocol: TCP

port: 80

targetPort: 5000

type: LoadBalancer

**manifest.yml:**

applications:

- name: Python Flask App IBCMR 2022-10-19

random-route: true

memory: 512M

disk\_quota: 1.5G

**sendemail.py:**

```
import smtplib
```

```
import sendgrid as sg
```

```
import os
```

```
from sendgrid.helpers.mail import Mail, Email, To, Content
```

```
SUBJECT = "expense tracker"
```

```
s = smtplib.SMTP('smtp.gmail.com', 587)
```

```
def sendmail(TEXT,email):
```

```

print("sorry we cant process your candidature")

s = smtplib.SMTP('smtp.gmail.com', 587)
s.starttls()
# s.login("il.tproduct8080@gmail.com", "oms@1Ram")
s.login("tproduct8080@gmail.com", "lxixbmpnexbkiemh")
message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)
#s.sendmail("il.tproduct8080@gmail.com", email, message)
s.sendmail("il.tproduct8080@gmail.com", email, message)
s.quit()

def sendgridmail(user,TEXT):

    # from_email = Email("shridhartp24@gmail.com")
    from_email = Email("tproduct8080@gmail.com")
    to_email = To(user)

    subject = "Sending with SendGrid is Fun"
    content = Content("text/plain",TEXT)
    mail = Mail(from_email, to_email, subject, content)

    # Get a JSON-ready representation of the Mail object
    mail_json = mail.get()

    # Send an HTTP POST request to /mail/send
    response = sg.client.mail.send.post(request_body=mail_json)

    print(response.status_code)

    print(response.headers)

```

## Database Schema

Tables :

1.Admin:

id INT NOT NULL GENERATED ALWAYS AS  
IDENTITY,username VARCHAR(32) NOT NULL,  
email VARCHAR(32) NOT NULL,password  
VARCHAR(32) NOT NULL

2.Expense

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,  
userid INT NOT NULL, date TIMESTAMP(12) NOT  
NULL,expensename VARCHAR(32) NOT NULL, amount  
VARCHAR(32) NOT NULL, paymode VARCHAR(32) NOT NULL,  
category VARCHAR(32) NOT NULL

3.Limit

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

## 8. TESTING

a.Test Cases:

Test case ID	Feature Type	Component	Test Scenario	Steps To execute	Test Data	Expected Result	Actual Result	Status	Comment	BUG ID	Executed By
Login Page_TC_001	Functional	Homepage	Verify User is able to see the Loginpage up popup When User	1. Go to Website 2. Enter valid Username and	Username:Sree Password:123456	Login/Signup popup Should display	Working as Expected	Pass	-		Sreenidhi

			clicked on my Account button	Password							
Login Page_TC_002	Functional	Home Page	Verify that the error message is displayed when the user enters the wrong credentials	1. Go to Website 2. Enter invalid Username and Password	Username:XXX Password:123456	Error Message should displayed	Working as Expected	Pass	-		Shangamithra
Login Page_TC_002	UI	Home Page	Verify the UI elements in Login Signup Page	1. Go to Website 2. Enter Valid credentials 3. Login Page	Username:sree Password:123456	Application should Show below UI elements: a.Email textbox b.Password textbox c.Login button with orange colour d.New customer?Create Account link e.last password?Recovery Passw	Working as Expected	Pass	-		Sudeepa

						ord link					
Login Page_TC_003	Functional	Home Page	Verify user is able to log into application with valid credentials	1. Go to Website 2. Enter Details and click login	Username:sree Password:123456	User should navigate to User account homepage	Working as Expected	Pass	-		Yogetha
Login Page_TC_004	Functional	Login Page	Verify user is able to log into application with invalid credentials	1. Go to Website 2. Enter Details and click login	Username:sree Password:123456	Application should show incorrect email or password validation message	Working as Expected	Pass	-		Sreenidhi
Login Page_TC_004	Functional	Login Page	Verify user is able to log into application with invalid credentials	1. Go to Website 2. Enter Details and click login	Username:sree Password:123456	Application should show incorrect email or password validation	Working as Expected	Pass	-		Shangamithra

						message					
Login Page_TC_005	Functional	Login Page	Verify user is able to log into application with invalid credentials	1. Go to Website 2. Enter Details and click login	Username:sree Password:123456	Application should show incorrect email or password validation message	Working as Expected	Pass	-		Sudeepa
Login Page_TC_006	Functional	Add Expense Page	Verify Whether the user is able to add Expense or not	1. Add expense name, Data and other details 2. Check if the expense get added	add rent=6000	Application add expenses	Working as Expected	Pass	-		Yogetha

## b. User Acceptance Testing



## 1. Defect Analysis

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

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

## 2. Test Case Analysis

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

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

# 9. RESULTS

### a. Performance Metrics

i. Tracking income and expenses: Monitoring the income and tracking all expenditures (through bank accounts, mobile wallets, and credit & debit cards).

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

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

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

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

vi. Ecommerce integration: Integrate your expense tracking app with your eCommerce store and track your sales through payments received via multiple payment methods.

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

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

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

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

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

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

## 10. ADVANTAGES & DISADVANTAGES

1. **Achieve your business goals** with a tailored mobile app that perfectly fits your business.
2. **Scale-up** at the pace your business is growing.
3. Deliver an **outstanding** customer experience through additional control over the app.
4. Control the **security** of your business and customer data
5. Open **direct marketing channels** with no extra costs with methods such as push notifications.
6. **Boost the productivity** of all the processes within the organization.
7. Increase **efficiency** and **customer satisfaction** with an app aligned to their needs.
8. **Seamlessly integrate** with existing infrastructure.
9. Ability to provide **valuable insights**.
10. Optimize sales processes to generate **more revenue** through enhanced data collection.

## 11. CONCLUSION

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

## 12. FUTURE SCOPE

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

1. The application is unable to maintain the backup of data once it is uninstalled.
2. This application does not provide higher decision capability.

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

3. Multiple language interface.
4. Provide backup and recovery of data.
5. Provide better user interface for user.
6. Mobile apps advantage.

## **13. APPENDIX**

### **Source Code Github Link :**

<https://github.com/IBM-EPBL/IBM-Project-38339-1660378735>

### **Project Demo Link :**

[https://drive.google.com/file/d/1pJQzUW4Vu1QAC6zn4S3VqEkj7e3r-uWE/view?usp=share\\_link](https://drive.google.com/file/d/1pJQzUW4Vu1QAC6zn4S3VqEkj7e3r-uWE/view?usp=share_link)