

**PERSONAL EXPENSE
TRACKER APPLICATION**

IBM-3891-1662628702

NALAIYA THIRAN PROJECTBASED
LEARNING ONPROFESSIONAL
READLINESS FORINNOVATION,
EMPLOYNMENT AND
ENTERPRENEURSHIP

PROJECT REPORT
BY

VISHNU SURIYAN K T(922519104181)
VIGNESH K(922519104178)
THARUN PRASANTH S(922519104173)
THARUN KUMAR G(922519104171)

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE & ENGINEERING

V.S.B ENGINEERING COLLEGE,KARUR 639111.

INDEX

1. INTRODUCTION

1. Project Overview
2. Purpose

2. LITERATURE SURVEY

1. Existing problem
2. References
3. Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

1. Empathy Map Canvas
2. Ideation & Brainstorming
3. Proposed Solution
4. Problem Solution fit

4. REQUIREMENT ANALYSIS

1. Functional requirement
2. Non-Functional requirements

5. PROJECT DESIGN

1. Data Flow Diagrams
2. Solution & Technical Architecture
3. User Stories

6.PROJECT PLANNING & SCHEDULING

1. Sprint Planning & Estimation

2. Sprint Delivery Schedule

3. Reports from JIRA

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

1. Feature 1

2. Feature 2

3. Database Schema (if Applicable)

8. TESTING

1. Test Cases

1. User Acceptance Testing

9.RESULTS

1. Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

INTRODUCTION

a. Project Overview

TEAM ID:PNT2022TMID33459

Industry Mentor(s) Name : Kusboo

Faculty Mentor(s) Name : Manikandan.t

Skills Required:

IBM Cloud,
Registry

1. INTRODUCTION

a. Project Overview

This project is based on expense tracking.This project aims to create an easy,faster and smooth cloud application .For better expense tracking we developed our project that will help the

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

b. Purpose

Personal finance management is an important part of people'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. Before you decide to go in for a money manager, it is important to decide the type you want.

2. LITERATURE SURVEY

a. Existing problem

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

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

- i. Manual entry and routing of expense reports (62%)
- ii. Lack of visibility into spend data (42%)
- iii. Inability to enforce travel policies (29%)
- iv. Lost expense reports (24%)
- v. Lengthy expense approval system and reimbursement cycles(23%)

b. References

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

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

3. Problem Statement Definition

I am	Mr. Kumar is a 30 years old man.
I'm trying to	He wants to save the daily expenses of those who cannot manage his expenses.
But,	He cannot successfully lead a household and fulfill his goals.
Because	In the current world, Mobile phones and laptops have become a part of living, such an app would be handy to deal with all our expenses.
Which makes me feel	Mint offers you customized solutions, personalizing your experience of spending and expense tracking. All your due bill payments and monthly expenses can be integrated into Mint along with other apps. It allows you not to miss any amount and track where you spend your money.

Customer Problem Statement :

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

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


1.IDEATION & PROPOSED SOLUTION

a. Empathy Map Canvas

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Many organizations have their own system to record their income and expenses, which They feel it is the main key point of their business progress. It is good habit for a person to record daily expenses and earning but due to unawareness and lack of proper applications to suit their privacy, lacking decision making capacity people are using traditional note keeping methods to do so. Due to lack of a complete tracking system, there is a 2 constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.
2.	Idea / Solution description	We are building an android application named as "Expense Tracker". As the name suggests, this project is an android app which is used to track the daily expenses of the user. It is like digital record keeping which keeps the records of expenses done by a user. The application keeps track of the Income and Expenses of both users on a day-to-day basis. This application takes the
		income of a user and manages its daily expenses so that the user can save money. If you exceed the daily expense allowed amount it will give you a warning, so that you don't spend much and that specific day. If you spend less money than the daily expense allowed amount, the money left after spending is added into the user's savings. The application generates reports of the expenses of each end of the month. The amount saved can be used for celebrating festivals, Birthdays or Anniversary.

3.	Novelty / Uniqueness	It will have various options to keep record (for example Food, Traveling Fuel, Salary etc.). Automatically it will keep on sending notifications for our daily expenditure. In today's busy and expensive life, we are in a great rush to make moneys, but at the end of the month We broke off. As we are unknowingly spending money on title and unwanted things. So, we have come over with the plan to follow our profit. Here user can define their own categories for expense type like food, clothing, rent and bills where they have to enter the money that has been spend and likewise can add some data in extra data to indicate the expense.
4.	Social Impact / Customer Satisfaction	Customers who use our web application can really save more money. It is easy to use because it has a see and understand type and also a nice GUI. It is handy to use anywhere and anytime. Customers do not need to use spreadsheets anymore.
5.	Business Model (Revenue Model)	It attracts more people because it looks tiny but more powerful. It is open source so anyone can incorporate our model and add their own features. One more important point need to remember is that it's completely free to use, no need to pay for using.
6.	Scalability of the Solution	Monitoring your everyday expenses can set aside your cash, yet it can likewise help you set your monetary objectives for what's to come. On the off chance that you know precisely where your sum is going, much of a stretch to see where a few reductions and bargains can be made. Expense Tracker project is for keeping our day-to-day expenditures and will help us to keep record of our money daily. The project what we have created is work more proficient than the other income and expense tracker. The project effectively keeps away from the manual figuring for trying not to ascertain the pay and cost each month. It's a user-friendly application.

B. Ideation & Brainstorming



Brainstorm & idea prioritization

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

- 10 minutes to prepare
- 1 hour to collaborate
- 2-8 people recommended

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.

10 minutes

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

[Open article](#)

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

We are not able to track down the expense of what we did and also control loss of money. We are not able to do calculation correctly.

Key rules of brainstorming

To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgement.
- Listen to others.
- Go for volume.
- If possible, be visual.

[Share template feedback](#)

2

Brainstorm

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

10 minutes

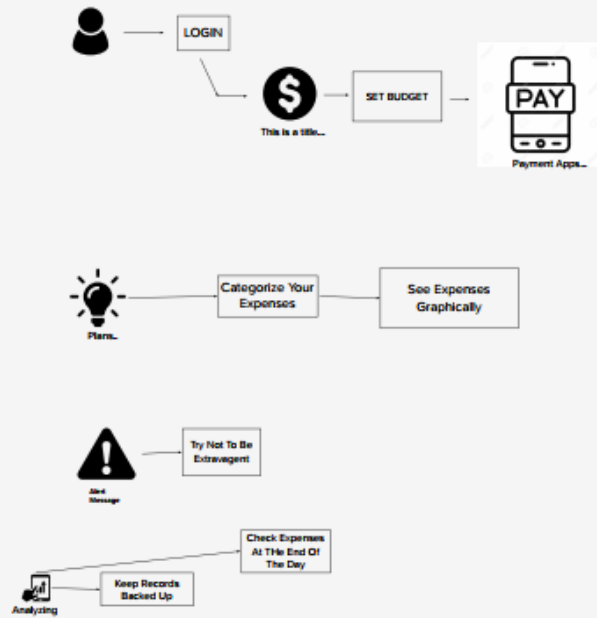
Vishnu Suriyan**Vignesh****Tharun Kumar****Tharun Prasanth**

3

Group Ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes





c.Proposed Solution

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

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

Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) CS people who wants reduce extravagant usage of money	6. CUSTOMER CONSTRAINTS CC Maximum usage of money per day is 150	5. AVAILABLE SOLUTIONS AS shop where discounts are provided; usage of google pay since via physical payment the vendor may give any product instead of giving change; buy the groceries in a small store where the products will be cheap and fresh, whereas in high end supermarkets the same quantity of products is very high rated.	Explore AS, differentiate
	2. JOBS-TO-BE-DONE / PROBLEMS J&P use our app to keep track on your expenses reduce the unwanted usage of money learn to be simple	9. PROBLEM ROOT CAUSE RC He or She likes to live a luxurious life despite of their low salary Trying to buy a new things even if it is available already	7. BEHAVIOUR BE spend money in necessary things, Avoiding extravagant usage of money	
Identify strong TR & EM	3. TRIGGERS TR He or She wasn't able to save money for emergency use whereas others are able to overcome the problem	10. YOUR SOLUTION SL set the default limit of usage of money to 150 and it can be adjusted upto 350 trying ways to reduce usage of money	8. CHANNELS of BEHAVIOUR CH ONLINE Use of google pay should have a limit of how much we used	Extract online & offline CH of BE
	4. EMOTIONS: BEFORE / AFTER EM Before using this because of this problem the money i had kept was drained in a drastic way After using this, my money usage is reduced and i saved money		OFFLINE set daily usage of money to 150	

4. REQUIREMENT ANALYSIS

a, Functional requirements

FR No.	Functional Requirement	Description
FR-1	Register	Registration is the process of the user to complete the application's form. Certain details must be submitted such as e-mail address, password, and password confirmation. The user is identified using these details.
FR-2	Login	The login screen is used to verify the identity of the user. The account can be accessed using the user's registered email address and password.

FR-3	Categories	On the main page, we can see overall revenue and spending, as well as the balance remaining after expenditure, as well as the user's entire categories namely Entertainment, Cloth, Food and Drinks, Health and Fitness and so on.
FR-4	Update Daily Expensive	The user can upload the daily expensive details what they are spending on each day. The details such as cloth, entertainment, food, health etc.,
FR-5	View Expensive Chart	This module used to see a pictorial depiction of all details in the form of a pie chart, where each slice of the pie chart represents that the viewer to gain an <u>approximate</u> notion of which category has the highest expenses.
NFR-6	Set Alert	When a user attempts to spend more than the <u>pre-defined</u> amount limit, the app will automatically send an alert if the threshold amount they selected for an alert is exceeded.

b. Non-Functional requirements

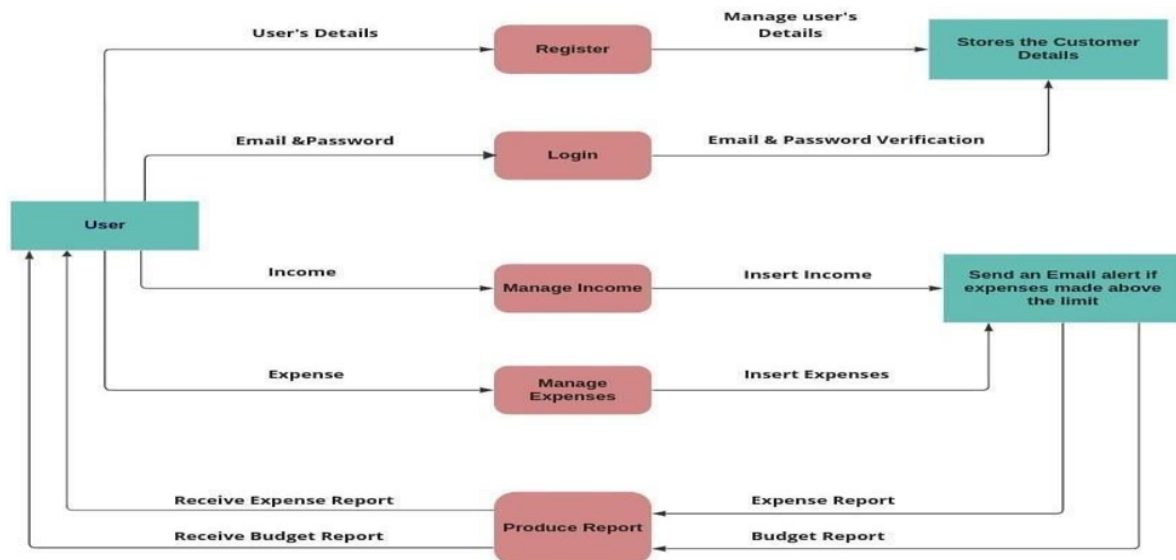
NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	The system shall allow the users to access the system with pc using web application. The system uses a web application as an interface. The system is user friendly which makes the system easy.
NFR-2	Security	A security requirement is a statement of needed security functionality that ensures one of many different security properties of software is being satisfied.
NFR-3	Reliability	he system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week. 24 hours a day.
NFR-4	Performance	The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the request <u>submittal</u> . The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 seconds to appear on the screen.

NFR-5	Availability	The system is available 100% for the user and <u>is used</u> 24 hrs a day and 365 days a year. The system <u>shall be</u> operational 24 hours a day and 7 days a week.
NFR-6	Scalability	Scalability is the measure of a system's ability <u>to increase</u> or decrease in performance and cost in response to changes in application and system processing demands.

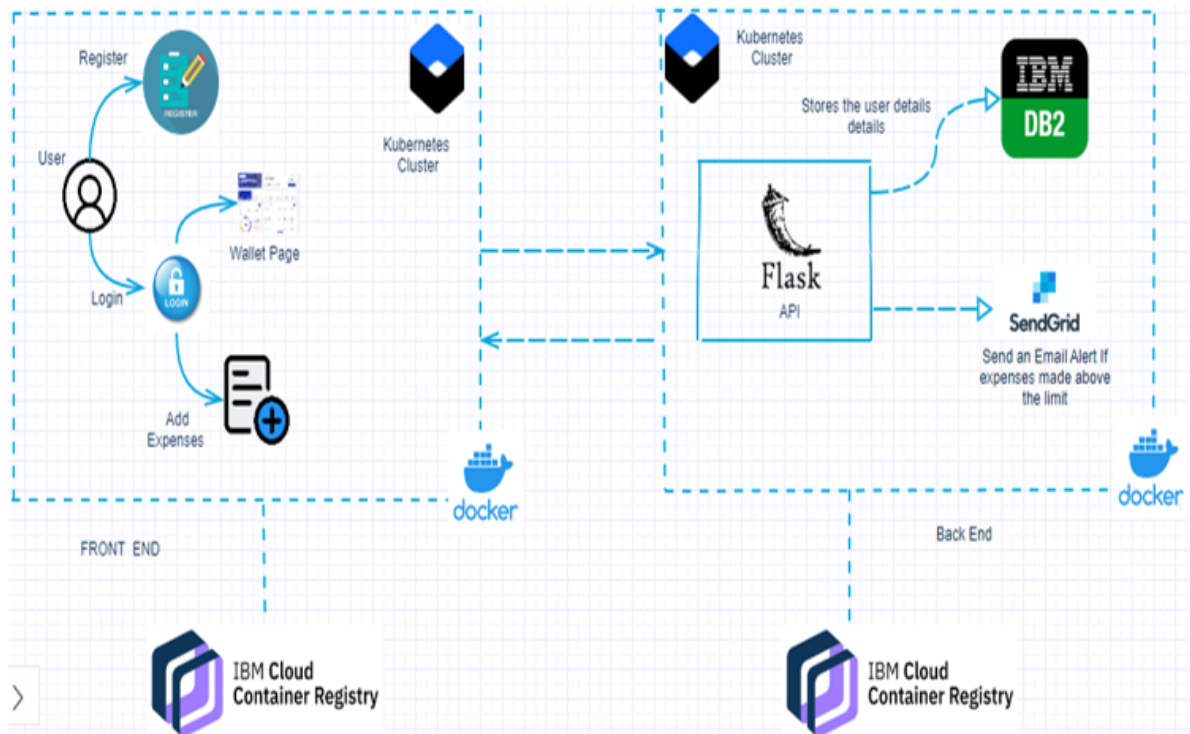
5. PROJECT DESIGN

a. 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 Diagrams:



B .Solution & Technical Architecture



C.User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority
Customer (web user)	Registration	USN-1	As a user, I can register for the application by entering mail id and password	I can access my account/ dashboard	High
		USN-2	As a user,I will receive a confirmation email once I have registered for the email and click application	I can receive a confirmation email	High
		USN-3	As a user, I can access using mail	I can register through mail	Low
	Login	USN-4	As a user, I can login application by entering application using email and password	I can access the application	High
	Dashboard	USN-5	As a user,I can view my income and expenditure details	I can view my daily expenses	High
Customer care executive		USN-6	As a customer care executive, I can solve the login issue and other issues of the solution at any application	I can provide support	Medium
Administrator	Application	USN-7	As an administrator,I can upgrade or update the application	I can fix the bug	Medium

6. PROJECT PLANNING & SCHEDULING

a. Sprint planning and estimation

Sprint	Functional Requirement (Epic)	User Story Number
Sprint-1	Registration	USN-1
Sprint-1		USN-2
Sprint-2		USN-3
Sprint-1		USN-4
Sprint-1	Login	USN-5
Sprint -2	Dashboard	USN-6
Sprint-2		USN-7
Sprint-3	Backend	USN-8

Sprint	Functional Requirement (Epic)	User Story Number
Sprint-3		USN-9
Sprint-4	Containerization & Testing	UNS-10
Sprint-4		USN-11

User Story / Task
As a user, I can register for the application by entering my email, password, and confirming my password.
As a user, I will receive a confirmation email once I have registered for the application
As a user, I can register for the application through Facebook
As a user, I can register for the application through Gmail
As a user, I can log into the application by entering email & password
As a user after logging in, I wished to see my wallet page.
As a user, I can add expenses under the expense page.
As a developer, I need to create a back-end database for storing information.

User Story / Task
As a developer, automate the mail to send alerts when expense reaches the limit.
As a developer, I Need to container the project in a professional way to work everywhere.
As a developer, test the project to check whether the project correctly works or not.

Story Points	Priority	Team Members
2	High	Vignesh, Tharun Kumar
1	High	Vishnu Suriyan
2	Low	Tharun Prasanth
2	Medium	Vishnu Suriyan
1	High	Tharun Prasanth
1	Low	Vignesh, Tharun Kumar
2	High	Vishnu Suriyan
1	High	Vignesh, Tharun Kumar

Story Points	Priority	Team Members
1	Medium	Tharun Prasanth
2	High	Vishnu Suriyan
2	High	Tharun Kumar

b. Sprint Delivery Schedule

S.NO	MILESTONES	ACTIVITIES	DATE
1.	Preparation Phase	Pre-requisites	24 Aug 2022
		Prior Knowledge	25 Aug 2022
		Project Structure	23 Aug 2022
		Project Flow	23 Aug 2022
		Project Objectives	22 Aug 2022
		Registrations	26 Aug 2022
		Environment Set-up	27 Aug 2022
2.	Ideation Phase	Literature Survey	29 Aug 2022 – 03 Sept 2022
		Empathy Map	5 Sept 2022 - 7 Sept 2022
		Problem Statement	8 Sept 2022 - 10 Sept 2022
		Ideation	12 Sept 2022 – 16 Sept 2022
3.	Project Design Phase - 1	Proposed Solution	19 Sept 2022 – 23 Sept 2022
		Problem Solution Fit	24 Sept 2022 – 26 Sept 2022
		Solution Architecture	27 Sept 2022 – 30 Sept 2022

4.	Project Design Phase - 2	Customer Journey Map	03 Oct 2022 – 08 Oct 2022
		Requirement Analysis	09 Oct 2022 – 11 Oct 2022
		Data Flow Diagrams	11 Oct 2022 – 14 Oct 2022
		Technology Architecture	15 Oct 2022 - 16 Oct 2022
5.	Project Planning Phase	Milestones & Tasks	17 Oct 2022 – 18 Oct 2022
		Sprint Schedules	19 Oct 2022 – 22 Oct 2022
6.	Project Development Phase	Sprint - 1	24 Oct 2022 – 29 Oct 2022
		Sprint – 2	31 Oct 2022 – 05 Nov 2022
		Sprint – 3	07 Nov 2022 – 12 Nov 2022
		Sprint – 4	14 Nov 2022 – 19 Nov 2022

Reports from JIRA

The image displays two screenshots of the Jira Software interface for the 'PERSONAL EXPENSE TRACKER' project.

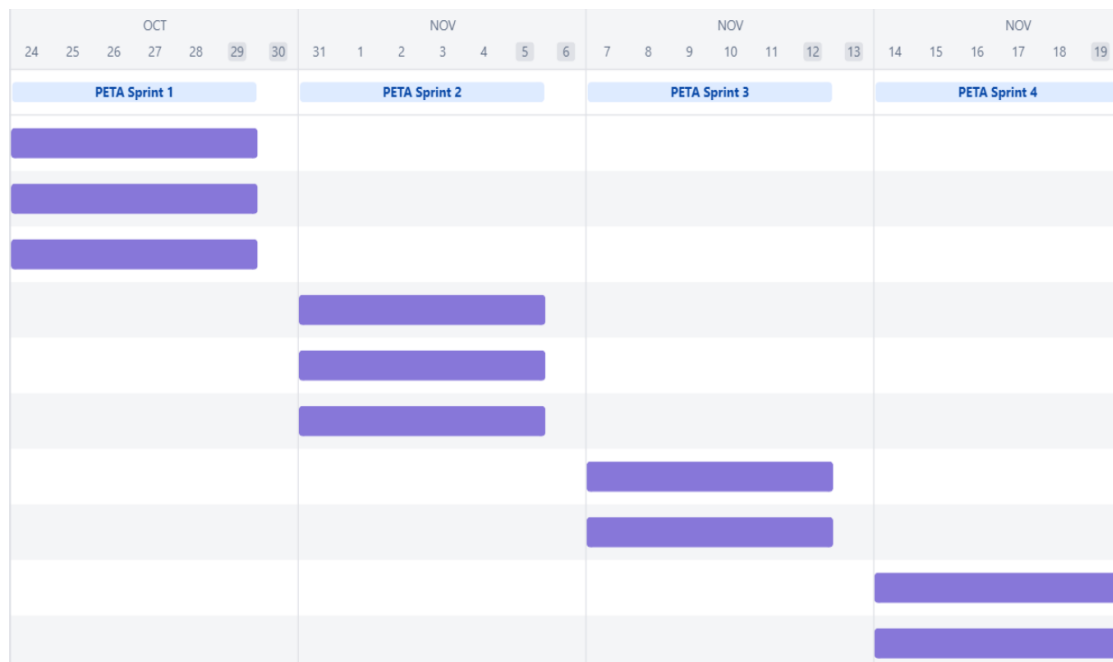
Top Screenshot: Backlog View

- Project:** PERSONAL EXPENSE TRACKER
- View:** Backlog
- Sprint:** PET Sprint 2 (29 Oct - 5 Nov, 3 issues)
- Issues:**
 - PET-1: providing my email, password, and confirming my password. (REGISTRATION) - TO DO
 - PET-5: create user account (REGISTRATION) - TO DO
 - PET-6: add expense (REGISTRATION) - TO DO
- Backlog (4 issues):**
 - PET-7: set budget - TO DO
 - PET-8: add wallet page - TO DO
 - PET-9: show expense graphically - TO DO
 - PET-10: Alert user through mail - TO DO

Bottom Screenshot: PET Sprint 2 View

- Project:** PERSONAL EXPENSE TRACKER
- View:** PET Sprint 2
- Sprint:** PET Sprint 2 (5 days remaining, Complete sprint)
- Columns:**
 - TO DO:** Empty
 - IN PROGRESS:** Empty
 - REVIEW 2 ISSUES:**
 - PET-1: providing my email, password, and confirming my password. (REGISTRATION)
 - PET-6: add expense (REGISTRATION)
 - DONE 1 ISSUE:**
 - PET-5: create user account (REGISTRATION)

PNT2022TMID33459



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_mysqlldb import MySQL
```

```
# import MySQLdb.cursors
```

```
import re
```

```
from flask_db2 import DB2
```

```
import ibm_db
```

```
import ibm_db_dbi
```

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

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

```
import os
```

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

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

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

PNT2022TMID33459

```
# app.config['MYSQL_USER'] = 'D2DxDUPBii'
# app.config['MYSQL_PASSWORD'] = 'r8XBO4GsMz'
# app.config['MYSQL_DB'] = 'D2DxDUPBii'
''''

dsn_hostname = "3883e7e4-18f5-4afe-be8c□
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"
dsn_uid = "sbb93800"
dsn_pwd = "wobsVLm6ccFxcNLe"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "31498"
dsn_protocol = "tcpip"
dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
    "PORT={3};"
    "PROTOCOL={4};"
    "UID={5};"
    "PWD={6};"
).format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol, dsn_uid,
dsn_pwd)
''''

# app.config['DB2_DRIVER'] = '{IBM DB2 ODBC DRIVER}'
app.config['database'] = 'bludb'
app.config['hostname'] = '3883e7e4-18f5-4afe-be8c□
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'
app.config['port'] = '31498'
app.config['protocol'] = 'tcpip'
app.config['uid'] = 'sbb93800'
app.config['pwd'] = 'wobsVLm6ccFxcNLe'
app.config['security'] = 'SSL'
try:
    mysql = DB2(app)
    conn_str='database=bludb;hostname=3883e7e4-18f5-4afe-be8c□
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=31498;protocol=tcpip;\.
uid=sbb93800;pwd=wobsVLm6ccFxcNLe;security=SSL'
    ibm_db_conn = ibm_db.connect(conn_str,"")
    print("Database connected without any error !!!")
except:
```

PNT2022TMID33459

```
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")

```

PNT2022TMID33459

```
@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():
```


PNT2022TMID33459

```
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"])
```

PNT2022TMID33459

```
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"])
```

PNT2022TMID33459

```
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)
```

PNT2022TMID33459

```
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()
```

PNT2022TMID33459

```
# 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 = []
```

PNT2022TMID33459

```
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]
```

PNT2022TMID33459

```
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).
```

PNT2022TMID33459

```
# 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":
```


PNT2022TMID33459

```
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().
```

PNT2022TMID33459

```
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)
```

PNT2022TMID33459

```
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:
```

PNT2022TMID33459

- 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: {}'.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")

PNT2022TMID33459

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

b. User Acceptance Testing:

				Date	3-Nov-22		
				Team ID	PNT2022TMD33459		
				Project Name	Personal Expenses Tracker		
				Maximum Marks	4 marks		
Test case ID	Feature Type	Component	Test Scenario	Pre-Requsite	Steps To Execute	Test Data	Expected Result
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button		1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Signup popup displayed or not	http://127.0.0.1:5000	Login/Signup popup should display
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Signup popup with below UI elements: a.email text box b.password text box c.Login button d.New customer? Create account link e.Last password? Recovery password link	http://127.0.0.1:5000	Application should show below UI elements a.email text box b.password text box c.Login button with orange colour d.New customer? Create account link e.Last password? Recovery password link
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: gipopev566@shopulit.com password: Testing123	User should navigate to user account homepage
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter Invalid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: wasanex395@shopulit.com password: Testing123	Application should show 'Incorrect email or password' validation message.
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter invalid password in password text box 5.Click on login button	Username: cotegef720@lance7.com password: Testing123	Application should show 'Incorrect email or password' validation message.
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter Invalid username/email in Email text box 4.Enter invalid password in password text	Username: jibeta621@shopulit.com password: Testing123	Application should show 'Incorrect email or password' validation message.

c. Defect Analysis:

2. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	6	4	2	5	17
Duplicate	2	0	3	0	5
External	5	2	0	1	8
Fixed	12	2	4	20	31
Not Reproduced	0	0	1	0	1
Skipped	0	1	1	1	2
Won't Fix	1	3	2	1	8
Totals	26	12	13	25	76

9.RESULTS

a. Performance Metrics

- i. Tracking income and expenses: Monitoring the income and tracking all expenditures (through bank accounts, mobile wallets, and credit & debit cards).
- ii. Transaction Receipts: Capture and organize your payment receipts to keep track of your expenditure.
- iii. Organizing Taxes: Import your documents to the expense tracking app, and it will streamline your income and expenses under the appropriate tax categories.
- iv. Payments & Invoices: Accept and pay from credit cards, debit cards, net banking, mobile wallets, and bank transfers, and track the status of your invoices and bills in the mobile app itself. Also, the trackingapp sendsremindersfor payments an d automatically matches the payments with invoices.
- v. Reports: The expense tracking app generates and sends reports to give a detailed insight about profits, losses, budgets, income, balance sheets, etc.,
- vi. Ecommerce integration: Integrateyour expense trackingapp wit h your eCommerce store and track your sales through payments received via multiple payment methods.
- vii. Vendors and Contractors: Manage and track all the payments to the vendors and contractors added to the mobile app.

- viii. Access control: Increase your team productivity by providing access control to particular users through custom permissions. ix. Track Projects: Determine project profitability by tracking labor costs, payroll, expenses, etc., of your ongoing project. x. Inventory tracking: An expense tracking app can do it all. Right from tracking products or the cost of goods, sending alert notifications when the product is running out of stock or the product is not selling, to purchase orders. xi. In-depth insights and analytics: Provides in-built tools to generate reports with easy-to-understand visuals and graphics to gain insights about the performance of 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.
- ix. Track Projects: Determine project profitability by tracking labor costs, payroll, expenses, etc., of your ongoing project. Inventory tracking: An expense tracking app can do it all.
- x. 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

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-38391-1660379694>

Project Demo Link:[PNT2022IBM33459-PET-DEMO-VEDIO](#)