SACS MAVMM ENGINEERING COLLEGE
Kidaripatty (PO), Alagarkoil (via), Madurai – 625 301.

PERSONAL EXPENSE TRACKER

A PROJECT REPORT

Submitted by

SIVABALAN R (912319104035)
THAYANITHI B (912319104042)
SIVASANKAR P (901231904036)
SUFGANRAJ S (912319104041)

In partial fulfillment for the award of the

degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

NOVEMBER 2022

TABLE OF CONTENTS

CHAPTER NO. **TITLE**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 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.

1.2 Purpose

Personal finance management is an important part of people's lives. However, everyone does not have the knowledge or time to manage their finances in a proper manner. And, even if a person has time and knowledge, they do not bother with tracking their expenses as they find it tedious and time-consuming. Now, you don't have to worry about managing your expenses, as you can get access to an expense tracker that will help in the active management of your finances.

Also known as expense manager and money manager, an expense tracker is a software or application that helps to keep an accurate record of your money inflow and outflow. Many people in India live on a fixed income, and they find that towards the end of the month they don't have sufficient money to meet their needs. While this problem can arise due to low salary, invariably it is due to poor money management skills.

People tend to overspend without realizing and this can prove to be disastrous. Using a daily expense manager can help you keep track of how much you spend every day and on what. At the end of the month, you will have a clear picture where your money is going. This is one of the best ways to get your expenses under control and bring some semblance of order to your finances.

CHAPTER 2

LITERATURE SURVEY

2.1 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. Inability to enforce travel policies (29%)
vi. Lost expense reports (24%)

2.2 References

1. https://ijariie.com/AdminUploadPdf/XTRA_STUDENT_EXPENSE_TRACKING_APPLICATION_ijariie16372.pdf

2. https://www.researchgate.net/publication/347972162_Expense_Manager_Application

3. https://ijarsct.co.in/Paper391.pdf

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

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

1.3 Problem Statement Definition

- The person needs way to analysis the daily expense.
- Daily expense in a more efficient and manageable way.
- So that he/ she manage their budget need pictorial representation for easy understanding.

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.

Personal Expense Tracker Application:

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	an employee.	Make a monthly budget.	There are no facilities to set a budget.	I need to save money for my future plans.	Frustrated.
PS-2	A manager.	Keep track of my expenses.	Can't categorize the various types of expenses.	There is no option to organize the various expenses.	Uncomfortable.

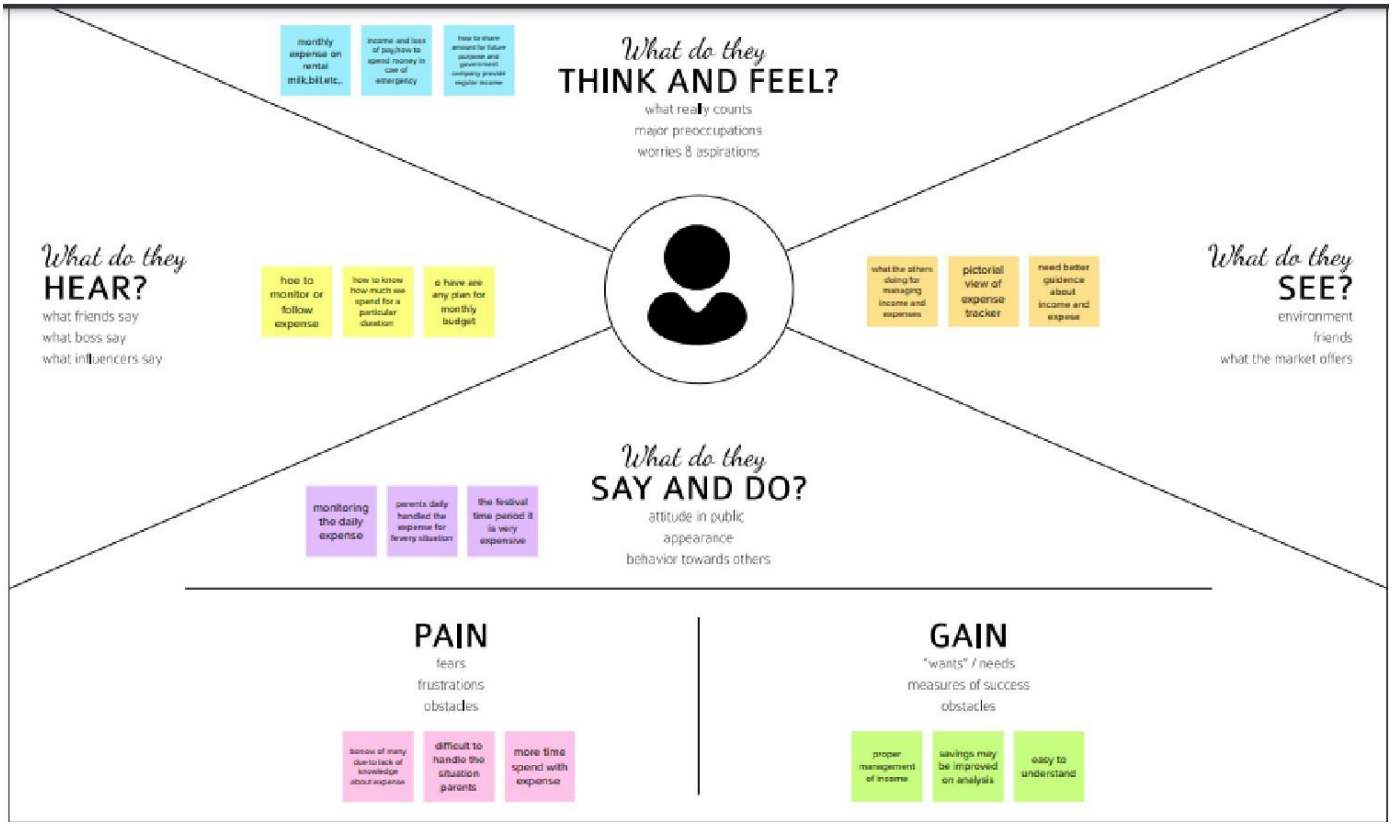
CHAPTER 3 IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

It is a useful tool to helps teams better understand their users.
Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



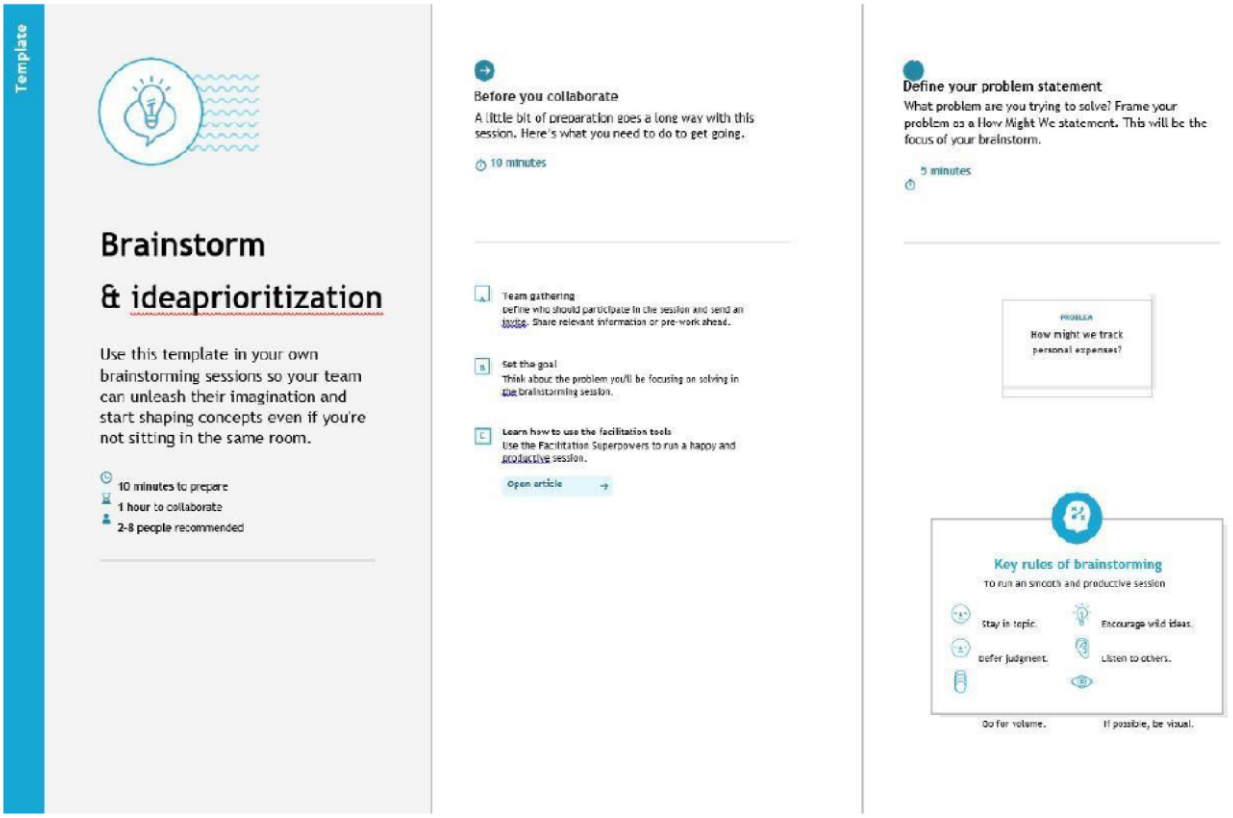
Brainstorm & Idea Prioritization Template:

Brainstorming provides a free and open environment that encourages everyone within a teamto participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants areencouraged to collaborate, helping each other develop a rich amount of creative solutions.

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.

Reference: https://www.mural.co/templates/empathy-map-canvas

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

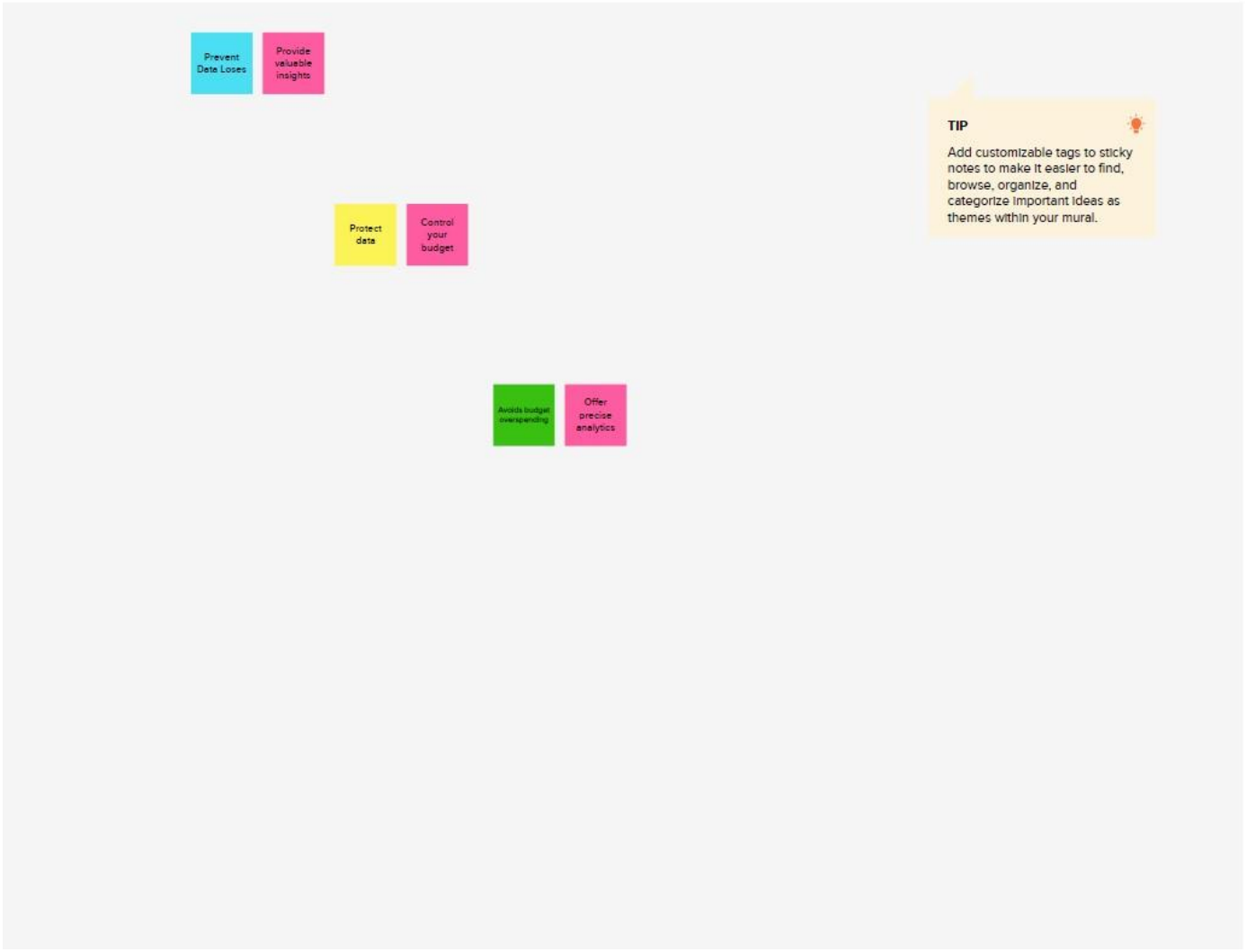


3

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 and break it up into smaller sub-groups.

20 minutes



2

Brainstorm

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

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

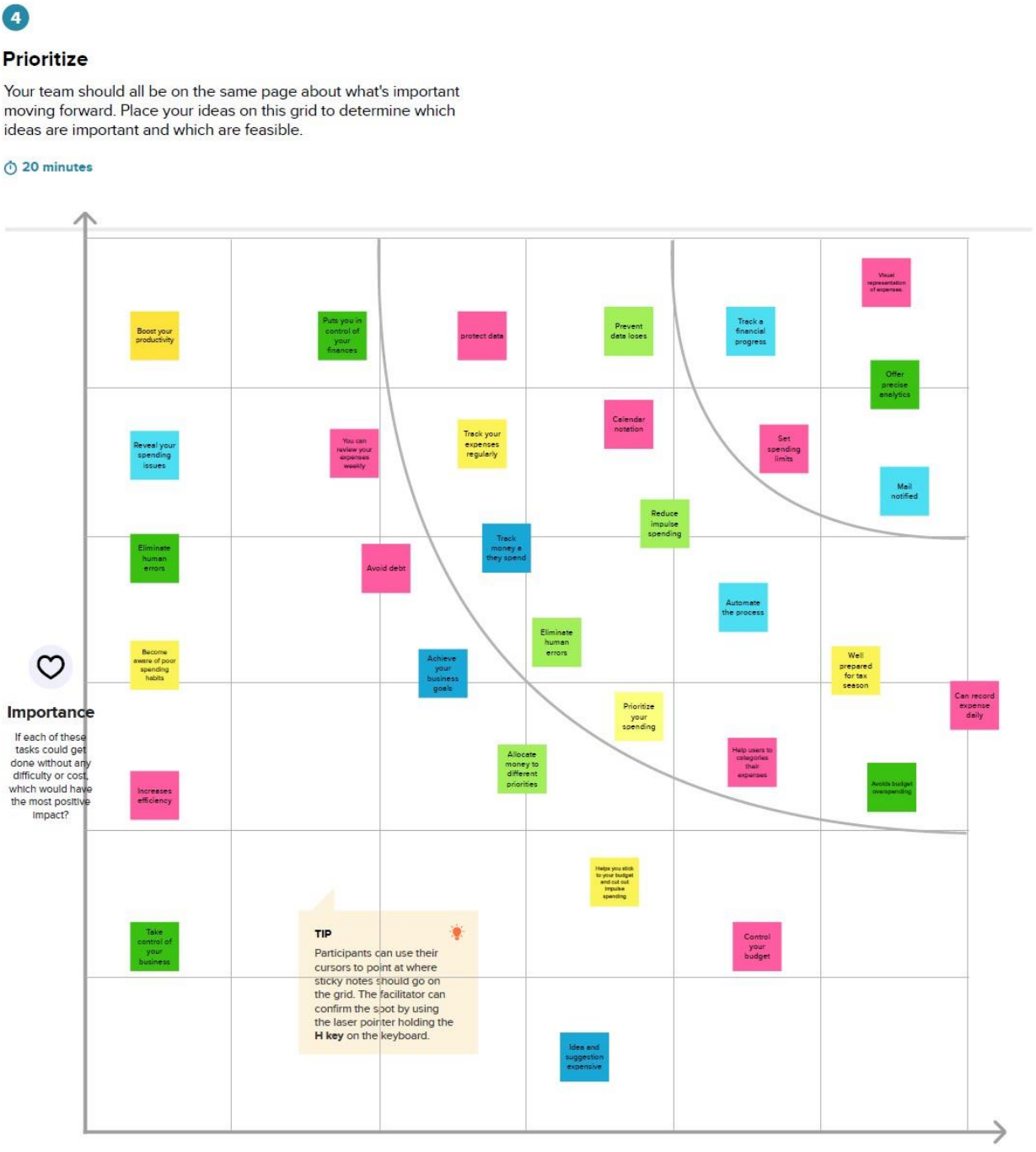
Kaviya B

Priya S

Theerthana K

Priyadharshini K

Step-3: Idea & Prioritization



After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A **Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B **Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template →

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template →

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template →

3.3 Proposed Solution

Project team shall fill the following information in proposed solution template.

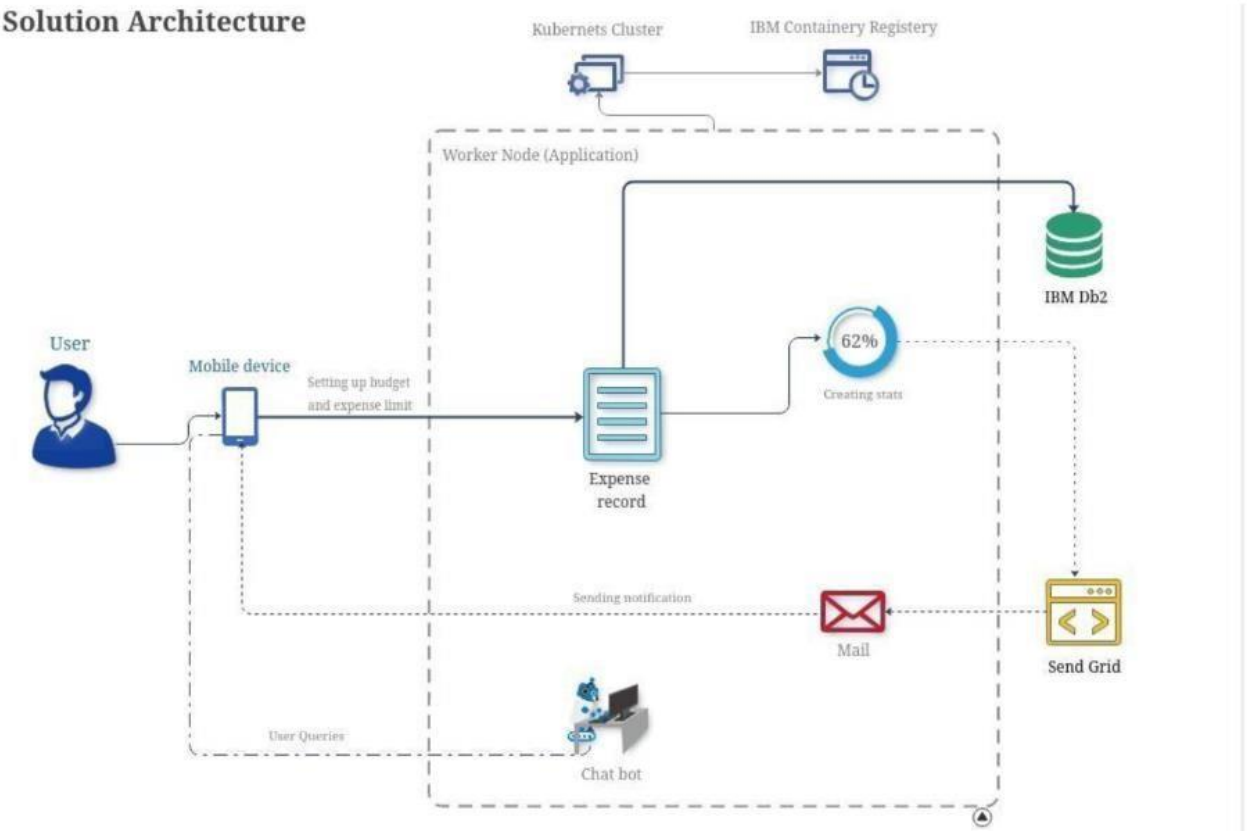
S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Earlier, our parents use to track all their expenses by writing down in a small notebook and calculating it on their own Even still many of them follow the same to maintain their financial expenses even some of them don't care of their expenses and spendings. Not only in our homes ,Expenses are need to be tracked in many large scale and small scale sectors such as in many schools, colleges, marketing companies , departmental stores , etc So in order to optimize their work and make peoples life easier our expense tracker application will be much helpful for financialmanagement The outcome of the application will be much useful for them to acknowledge the daily expenses and track the monthly expenses from their income with a limit to spend. They can easily track and view their expenses with a statistical data. In short, tracking our financial expenses is a great deal especially in this scenario so making those tracking easier is the job of this application.

2.	Idea / Solution description	Due to the busy and hectic lifestyle people tend to overlook their budget and end up spending an excessive amount of money since they usually didn't plan their budget wisely. user cannot predict future expenses. While they can write down their expenses in a excel spreadsheet, their lack of knowledge in managing finances will be a problem
3.	Novelty / Uniqueness	This application tracks your every expenses anywhere and anytime without using the paper work. Just click and enter your expenditure. to avoid data loss, quick settlements and reduce human error. To provide the pie chart or graph lines in this application.
4.	Social Impact / Customer Satisfaction	Using this application one can track their personal expenses and frame a monthly/annual budget. If your expense exceeded than specified limit, the application will show you an alert message in form of a pie chart.
5.	Business Model (Revenue Model)	Business people can use subscription/premium feature of this application to gain revenue.
6.	Scalability of the Solution	IBM cloud will automatically allocate the storage for the users.

Solution Architecture

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

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



3.4 Problem Solution fit

PNT2022TMID45814

Project Title: Personal Expense Tracker Application **Project Design Phase-I - Solution Fit Template** **Team ID:** PNT2022TMID45814

CHAPTER 4
REQUIREMENT ANALYSIS

4.1 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 approximatenotion of which category has the highest expenses.

NFR-6	**Set Alert**	When a user attempts to spend more than the pre-defined amount limit, the app will automatically send an alert if the threshold amount they selected for an alert is exceeded.

4.2 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.

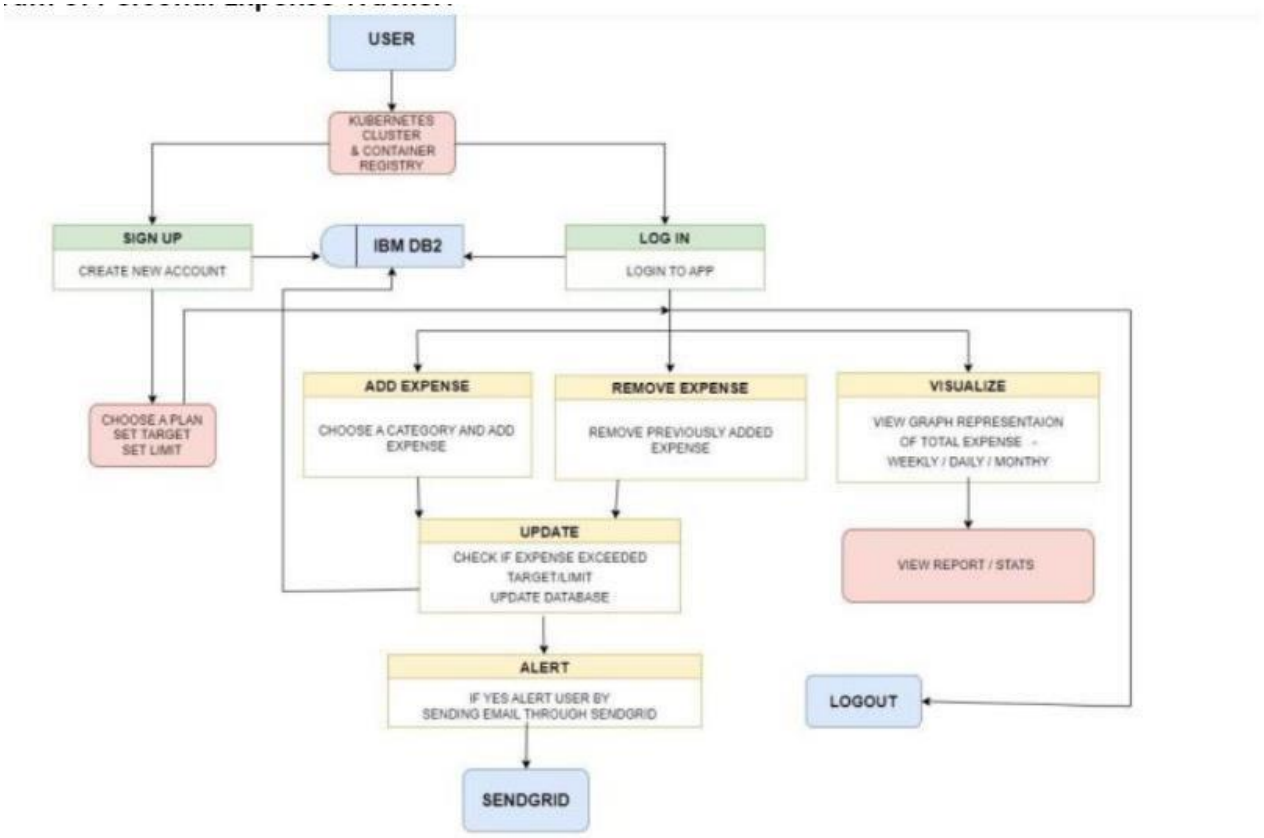
NFR-4	**Performance**	The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing 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 isused 24 hrs a day and 365 days a year. The system shallbe operational 24 hours a day and 7 days a week.
NFR-6	**Scalability**	Scalability is the measure of a system's ability toincrease or decrease in performance and cost in response to changes in application and system processing demands.

CHAPTER 5

PROJECT DESIGN

5.1 Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows with in 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.

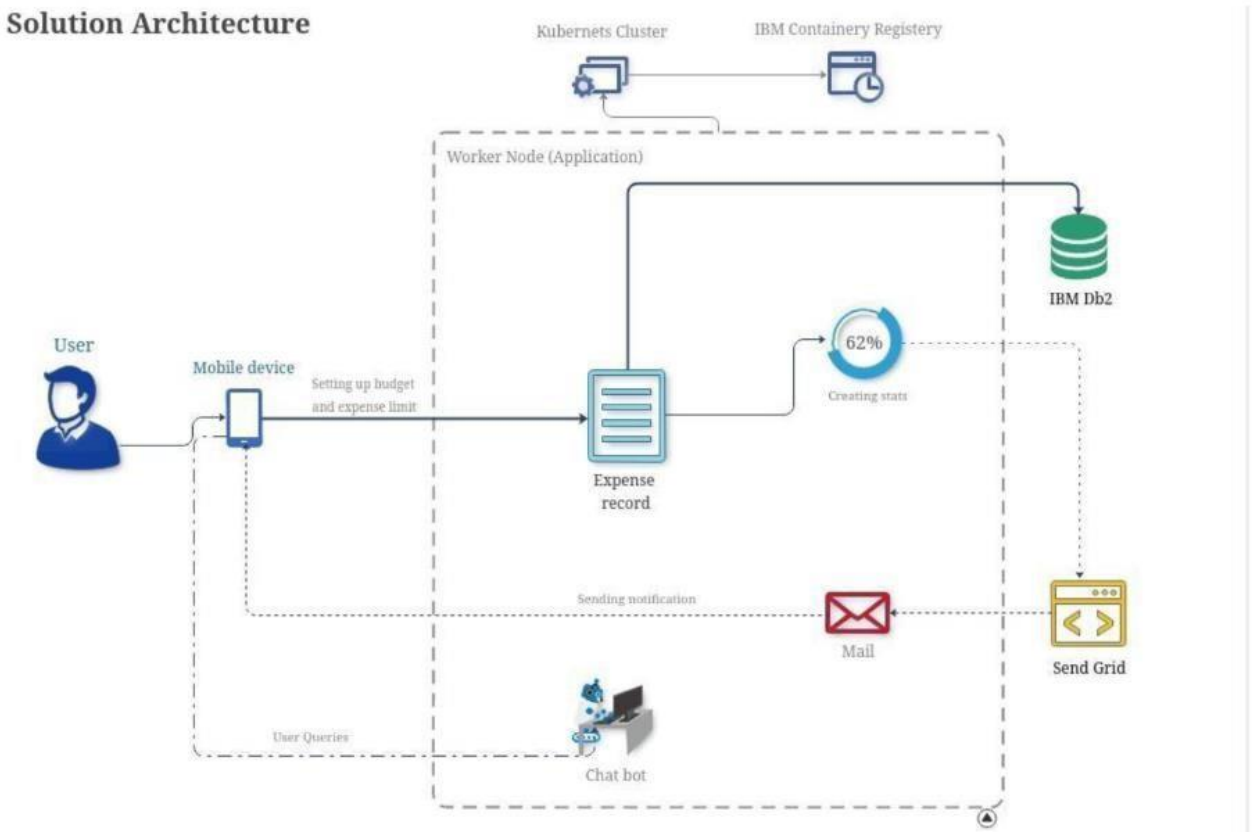


5.2 Solution & Technical Architecture

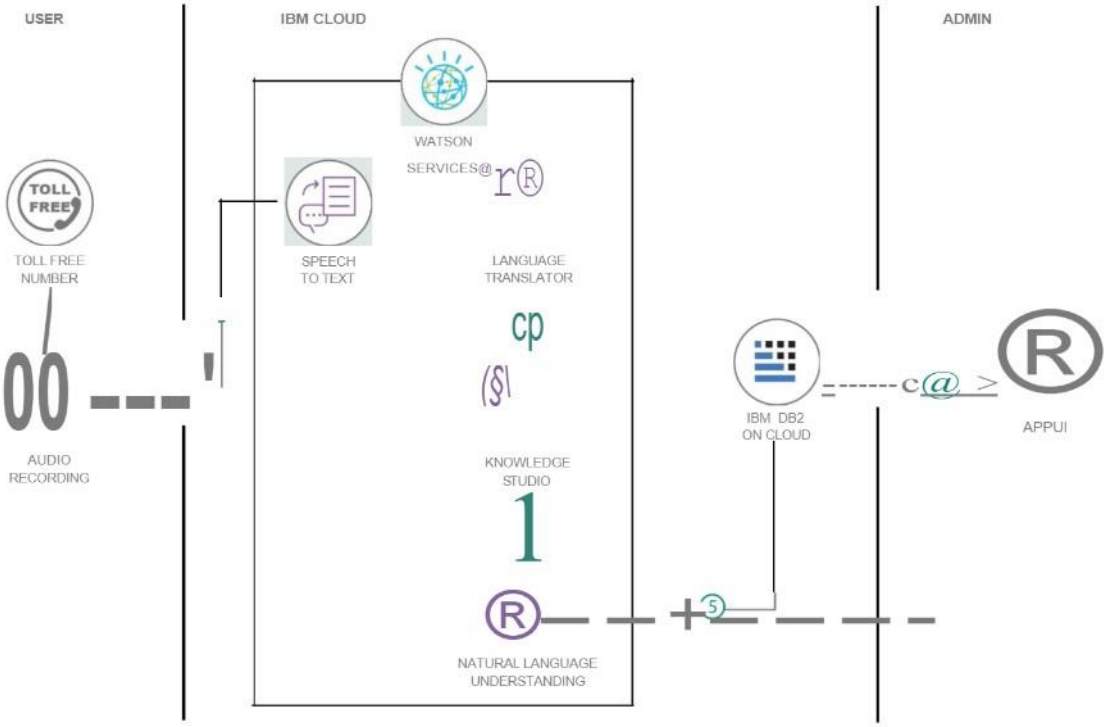
Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals a

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

PNT2022TMID45814



Technical Architecture



5.3 User Stories

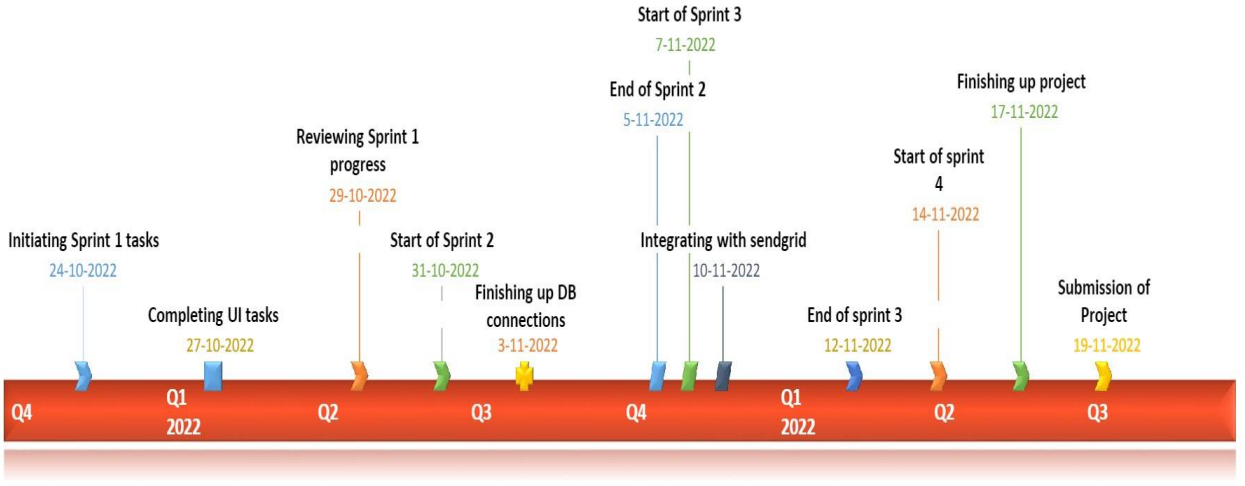
PNT2022TMID45814

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

PNT2022TMID45814

CHAPTER 6 PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation



PNT2022TMID45814

6.2 Sprint Delivery Schedule

Product Backlog, Sprint Schedule and Estimation

Use the below template to create product backlog and sprint schedule

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	Kaviya B
Sprint-1	Login	USN-2	As a user, I can log into the application by entering email & password	2	High	Kaviya B
Sprint-2	Add Expense	USN-3	As a user, I can add the day-to-day expense to the application	1	Medium	Priya S
Sprint-2	Edit and Delete Expense	USN-4	As a user, I can edit and delete the previously created expense	1	Medium	Priya S
Sprint-3	Creating time-based filters in history.	USN-5	As a user, I can see the time-based history of expenses.	2	High	Theerthana K
Sprint-3	Integrating with pie-charts for analysis	USN-6	As a user, I can view diagrammatic representation of expenses	1	Medium	Theerthana K
Sprint-4	Enabling limit feature	USN-7	As a user, I can set monthly limit to expenses	1	Medium	Priyadharshini K
Sprint-4	Sending Email Alerts	USN-8	As a user, I will receive a mail if I cross a limit	2	High	Priyadharshini K

Project Tracker,Velocity &Burndown Chart

PNT2022TMID45814

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	16	6 Days	24 Oct 2022	29 Oct 2022	16	29 Oct 2022
Sprint-2	12	6 Days	31 Oct 2022	05 Nov 2022	10	05 Nov 2022
Sprint-3	14	6 Days	07 Nov 2022	12 Nov 2022	13	12 Nov 2022
Sprint-4	14	6 Days	14 Nov 2022	19 Nov 2022	13	19 Nov 2022

PNT2022TMID45814

Velocity :

We have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). Calculating the team's average velocity (AV).

$$AV = \text{sprint duration} = 20 / 6 = 3.33 \text{ velocity}$$

Burndown Chart

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

https://www.visual-paradigm.com/scrum/scrum-burndown-chart/

https://www.atlassian.com/agile/tutorials/burndown-charts

Reference:

https://www.atlassian.com/agile/project-management

https://www.atlassian.com/agile/tutorials/how-to-do-scrum-with-jira-software

https://www.atlassian.com/agile/tutorials/epics

https://www.atlassian.com/agile/tutorials/sprints

https://www.atlassian.com/agile/project-management/estimation

https://www.atlassian.com/agile/tutorials/burndown-charts

CHAPTER 7
CODING & SOLUTIONING

app.py:

```
# -*- coding: utf-8 -
*-"""
Spyder Editor
This is a temporary script
file."""
from flask import
Flask, render_template,
request, redirect,
```

PNT2022TMID45814

```
session#                from
flask_mysqldb import
MySQL
# import
MySQLdb.cursorsimport
re    from    flask_db2
import      DB2import
ibm_db          import
ibm_db_dbi
from sendemail import sendgridmail,sendmail
# from gevent.pywsgi
import WSGIServerimport
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.clou
d'  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=tcp i p;\
          uid=sbb93800;pwd=wobsVLm6ccFxcNLe;security=SSL'
    ibm_db_conn = ibm_db.connect(conn_str,",")


    print("Database connected without any error !!")except:
    print("IBM DB Connection error : " + DB2.conn_errormsg())

# app.config["]
# mysql = MySQL(app) #HOME--PAGE
@app.route("/home") def
home():
    return render_template("homepage.html")
@app.route("/" )def add(): return
render_template("home.html")

#SIGN--UP--OR--REGISTER

@app.route("/signup") def
signup():
    return              render_template("signup.html")
  @app.route('/register',      methods      =['GET',
  'POST'] def register:
    msg =”
       print("Break point1")
```

PNT2022TMID45814

```
 if request.method == 'POST' : username =
    request.form['username']email =
    request.form['email'] password =
    request.form['password']
print("Break point2" + "name: " + username + "------" + email + "     " + password)

            try:
       print("Break      point3")      connectionID      =
       ibm_db_dbi.connect(conn_str, ",
       ")cursor = connectionID.cursor() print("Break
    point4")except:
       print("No connection Established")


    # cursor = mysql.connection.cursor() #
    with app.app_context():
    #     print("Break point3")
    #     cursor = ibm_db_conn.cursor()
    # print("Break   point4")   print("Break

      point5")

    sql = "SELECT * FROM register WHERE username =
    ?"stmt  =  ibm_db.prepare(ibm_db_conn,  sql)  ibm_db.bind_param(stmt,  1,
    username)  ibm_db.execute(stmt)  result  =  ibm_db.execute(stmt)  print(result)
    account = ibm_db.fetch_row(stmt) print(account) param = "SELECT * FROM
    register WHERE username = " + "\'" + username +
    "\'"res = ibm_db.exec_immediate(ibm_db_conn, param)
    print(" ---") dictionary = ibm_db.fetch_assoc(res) while
    dictionary != False: print("The ID is : ",
       dictionary["USERNAME"])dictionary                              =
       ibm_db.fetch_assoc(res)

    # dictionary = ibm_db.fetch_assoc(result)
    # cursor.execute(stmt)


    # account = cursor.fetchone()
```

```
# print(account)

# while ibm_db.fetch_row(result) != False:
#     # account = ibm_db.result(stmt)
#       print(ibm_db.result(result, "username"))
#
print(dictionary["username"]
) print("break point 6") if
account:
    msg = 'Username already exists !'
elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):msg
    = 'Invalid email address !'
elif not re.match(r'[A-Za-z0-9]+', username):
    msg = 'name must contain only characters and numbers !'
else:
   sql2 = "INSERT INTO register (username, email,password) VALUES (?, ?,
    ?)"stmt2 = ibm_db.prepare(ibm_db_conn, sql2)
    ibm_db.bind_param(stmt2, 1, username)
    ibm_db.bind_param(stmt2, 2, email)
    ibm_db.bind_param(stmt2, 3, password)
```

```
        ibm_db.execute(stmt2)
        # cursor.execute('INSERT INTO register VALUES (NULL, % s, % s, % s)',(username, email,password)) #
        mysql.connection.commit() msg = 'You
        have successfully registered !'
    return render_template('signup.html', msg = msg)
#LOGIN--PAGE
@app.route("/signin" )def
signin():
  return render_template("login.html")
@app.route('/login',methods =['GET', 'POST'])def
login():
  global userid msg
  = "
  if request.method == 'POST' : username
    = request.form['username']password
    = request.form['password'] # cursor =
    mysql.connection.cursor()
    # cursor.execute('SELECT * FROM register WHERE username = % s AND password = % s', (username, password ),)
    # account = cursor.fetchone()
    # print (account) sql = "SELECT * FROM register WHERE username = ?
    and password = ?" stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.bind_param(stmt, 2, password)
    result = ibm_db.execute(stmt)
    print(result) account =
    ibm_db.fetch_row(stmt) print(account)
    param = "SELECT * FROM register WHERE username = " + "\'" + username + "\'" + "and password = " + "\'" + password + "\'" res =
    ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    # sendmail("hello sakthi","sivasakthisairam@gmail.com")
if account:
```

```
        session['loggedin'] = True session['id'] =
        dictionary["ID"]userid = dictionary["ID"]
        session['username'] =
        dictionary["USERNAME"]session['email'] =
        dictionary["EMAIL"] return
        redirect('/home')
    else:
        msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)
    #ADDING-- DATA
@app.route("/add") def
adding():
    return render_template('add.html')
@app.route('/addexpense',methods=['GET', 'POST']) def
addexpense():
    date = request.form['date'] expensename =
    request.form['expensename'] amount =
    request.form['amount'] paymode =
    request.form['paymode'] category =
    request.form['category']
    print(date) p1 = date[0:10] p2 =
    date[11:13] p3 = date[14:] p4 = p1
    + "-" + p2 + "." + p3 + ".00"
    print(p4)
    # cursor = mysql.connection.cursor()
    # cursor.execute('INSERT INTO expenses VALUES (NULL, % s, % s, % s, % s, % s, %
s)', (session['id'] ,date, expensename, amount, paymode, category))
    # mysql.connection.commit()
    # print(date + " " + expensename + " " + amount + " " + paymode + " " + category)
    sql = "INSERT INTO expenses (userid, date, expensename, amount, paymode, category)
VALUES (?, ?, ?, ?, ?, ?)" stmt =
    ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, session['id'])
    ibm_db.bind_param(stmt, 2, p4)
```

```
ibm_db.bind_param(stmt, 3,
expensename) ibm_db.bind_param(stmt,
4, amount) ibm_db.bind_param(stmt, 5,
paymode) ibm_db.bind_param(stmt, 6,
category) ibm_db.execute(stmt)

print("Expenses added")
# email part
param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp)ORDER BY date DESC"
res = ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res) expense =
[] while dictionary != False:temp = []
temp.append(dictionary["ID"])
temp.append(dictionary["USERID"])
temp.append(dictionary["DATE"])
temp.append(dictionary["EXPENSENAME"])
temp.append(dictionary["AMOUNT"])
temp.append(dictionary["PAYMODE"])
temp.append(dictionary["CATEGORY"])
expense.append(temp) print(temp)
   dictionary = ibm_db.fetch_assoc(res)
total=0 for x in
expense: total
+= x[4] param =
"SELECT id,
limitss FROM
limits WHERE
userid = " +
str(session['id'])
+ "
ORDER BY id DESC LIMIT 1" res =
ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res)
row =
[]s = 0
```

```
    while dictionary != False:temp = []
        temp.append(dictionary["LIMITSS"]
        ) row.append(temp) dictionary =
        ibm_db.fetch_assoc(res) s = temp[0]

    if total > int(s):
        msg = "Hello " + session['username'] + " , " + "you have crossed the monthly limit of Rs." + s + "/- !!!" + "\n" + "Thank you, " + "\n" + "Team Personal Expense Tracker."
        sendmail(msg,session['email'])
    return redirect("/display")

#DISPLAY---graph
@app.route("/display"
)def display():
    print(session["username"],session['id'])


    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT * FROM expenses WHERE userid = % s AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
    # expense = cursor.fetchall()
    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res) expense =
    [] while dictionary != False:temp = []
    temp.append(dictionary["ID"])
    temp.append(dictionary["USERID"])
    temp.append(dictionary["DATE"])
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    expense.append(temp) print(temp) dictionary =
    ibm_db.fetch_assoc(res)
    return render_template('display.html' ,expense = expense)


#delete---the--data
@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])def
delete(id):
```

```
    # cursor = mysql.connection.cursor()
    # cursor.execute('DELETE FROM expenses WHERE id = {0}'.format(id))#
    mysql.connection.commit()

    param = "DELETE FROM expenses WHERE id = " + id res
    = ibm_db.exec_immediate(ibm_db_conn, param)

    print('deleted successfully')
  return redirect("/display")
#UPDATE---DATA
@app.route('/edit/<id>', methods = ['POST', 'GET' ])def
edit(id):
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT * FROM expenses WHERE id = %s', (id,))
    # row = cursor.fetchall() param = "SELECT * FROM
    expenses WHERE id = " + id res =
    ibm_db.exec_immediate(ibm_db_conn, param) dictionary
    = ibm_db.fetch_assoc(res) row = [] while dictionary !=
    False:temp = [] temp.append(dictionary["ID"])
    temp.append(dictionary["USERID"])
    temp.append(dictionary["DATE"])
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    row.append(temp) print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    print(row[0]) return render_template('edit.html',
    expenses = row[0]) @app.route('/update/<id>',
    methods = ['POST']) def update(id):
  if request.method == 'POST' :
      date = request.form['date'] expensename =
      request.form['expensename'] amount =
      request.form['amount'] paymode =
```

```
request.form['paymode']  category  =
request.form['category']

# cursor = mysql.connection.cursor()
# cursor.execute("UPDATE `expenses` SET `date` = % s , `expensename` = % s , `amount` = % s, `paymode` = % s, `category` = % s WHERE `expenses`.`id` = % s ",(date, expensename, amount, str(paymode), str(category),id))
# mysql.connection.commit()
p1 = date[0:10] p2 = date[11:13] p3 = date[14:] p4
= p1 + "-" + p2 + "." + p3 + ".00"
sql = "UPDATE expenses SET date = ? , expensename = ? , amount = ?, paymode = ?, category = ? WHERE id = ?" stmt =
ibm_db.prepare(ibm_db_conn, sql)
ibm_db.bind_param(stmt, 1, p4)
ibm_db.bind_param(stmt, 2, expensename)
ibm_db.bind_param(stmt, 3, amount)
ibm_db.bind_param(stmt, 4, paymode)
ibm_db.bind_param(stmt, 5, category)
ibm_db.bind_param(stmt, 6, id)
ibm_db.execute(stmt) print('successfully
updated') return redirect("/display"

#limit
@app.route("/limit" )def
limit():
return redirect('/limitn')
@app.route("/limitnum" , methods = ['POST' ])def
limitnum():
if request.method == "POST":
number= request.form['number']
# cursor = mysql.connection.cursor()
# cursor.execute('INSERT INTO limits VALUES (NULL, % s, % s) ',(session['id'], number))
# mysql.connection.commit()

sql = "INSERT INTO limits (userid, limitss) VALUES (?,
```

```
        ?)"stmt = ibm_db.prepare(ibm_db_conn,
        sql) ibm_db.bind_param(stmt, 1,
        session['id']) ibm_db.bind_param(stmt, 2,
        number) ibm_db.execute(stmt) return
        redirect('/limitn')
        @app.route("/limitn") def limitn():
      # cursor = mysql.connection.cursor()
   # cursor.execute('SELECT limitss FROM `limits` ORDER BY `limits`.`id` DESC LIMIT
   1')# x= cursor.fetchone()
   # s = x[0]
   param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + "
ORDER BY id DESC LIMIT 1" res =
   ibm_db.exec_immediate(ibm_db_conn, param)
   dictionary = ibm_db.fetch_assoc(res)
   row =
   []s = ''
   /-"
   while dictionary != False:temp = []
      temp.append(dictionary["LIMITSS"]
      ) row.append(temp)
      dictionary = ibm_db.fetch_assoc(res) s
      = temp[0]

   return render_template("limit.html" , y= s)
#REPORT
@app.route("/today" )def
today():
    # cursor = mysql.connection.cursor()
   # cursor.execute('SELECT TIME(date) , amount FROM expenses WHERE userid = %s
AND DATE(date) = DATE(NOW())
   ',(str(session['id'])))# texpense = cursor.fetchall()
   # print(texpense) param1 = "SELECT TIME(date) as tn, amount FROM expenses
     WHERE userid = " +
str(session['id']) + " AND DATE(date) = DATE(current timestamp) ORDER BY date
DESC" res1 = ibm_db.exec_immediate(ibm_db_conn,
     param1) dictionary1 = ibm_db.fetch_assoc(res1)
     texpense = []
```

```
    while dictionary1 != False:temp = []
        temp.append(dictionary1["TN"])
        temp.append(dictionary1["AMOUNT"]
        ) texpense.append(temp) print(temp)
        dictionary1 = ibm_db.fetch_assoc(res1)
  # cursor = mysql.connection.cursor()
  # cursor.execute('SELECT * FROM expenses WHERE userid = % s AND DATE(date) = DATE(NOW()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
  # expense = cursor.fetchall() param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " ANDDATE(date) = DATE(current timestamp) ORDER BY date DESC" res = ibm_db.exec_immediate(ibm_db_conn, param) dictionary = ibm_db.fetch_assoc(res) expense = [] while dictionary != False:temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"]
        ) temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp) print(temp)
        dictionary = ibm_db.fetch_assoc(res) total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0    t_EMI=0
    t_other=0  for  x  in
    expense:   total   +=
    x[4]   if   x[6]   ==
    "food":
        t_food += x[4] elif x[6]
      ==          "entertainment":
        t_entertainment   +=   x[4]
```

```
        elif   x[6]   ==   "business":

        t_business += x[4]

        elif x[6] ==

           "rent":t_rent +=

           x[4]


        elif x[6] ==

           "EMI":t_EMI +=

           x[4]

elif x[6] ==

           "other":t_other +=

           x[4]

print(total) print(t_food)

         print(t_entertainment

         ) print(t_business)

         print(t_rent)

         print(t_EMI) print(t_other

         )


        return render_template("today.html", texpense = texpense, expense = expense, total =
      total ,

                        t_food = t_food,t_entertainment = t_entertainment,

      t_business = t_business, t_rent = t_rent, t_EMI = t_EMI, t_other =

      t_other ) @app.route("/month"

      )def month():

         # cursor = mysql.connection.cursor()

        # cursor.execute('SELECT DATE(date), SUM(amount) FROM expenses WHERE userid=
      %s AND MONTH(DATE(date))= MONTH(now()) GROUP BY DATE(date)
      ORDER
      BY DATE(date)

         ',(str(session['id'])))# texpense = cursor.fetchall()

         # print(texpense) param1 = "SELECT DATE(date) as dt, SUM(amount) as tot FROM
           expenses WHERE
      userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current timestamp) AND
      YEAR(date) = YEAR(current timestamp) GROUP BY DATE(date) ORDER BY
```

```
DATE(date)"                    res1                    =
    ibm_db.exec_immediate(ibm_db_conn,          param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpense = [] while
    dictionary1 !=
    False:temp = []
       temp.append(dictionary1["DT"])
       temp.append(dictionary1["TOT"]
       )texpense.append(temp) print(temp)
       dictionary1 =
       ibm_db.fetch_assoc(res1) # cursor =
       mysql.connection.cursor()

  # cursor.execute('SELECT * FROM expenses WHERE userid = % s AND
MONTH(DATE(date))= MONTH(now()) AND date ORDER BY
`expenses`.`date`DESC',(str(session['id'])))
  # expense = cursor.fetchall() param = "SELECT * FROM expenses WHERE userid = "
+ str(session['id']) + " AND MONTH(date) = MONTH(current timestamp) AND
YEAR(date) = YEAR(current timestamp)ORDER BY date DESC" res =
ibm_db.exec_immediate(ibm_db_conn, param) dictionary = ibm_db.fetch_assoc(res)
expense = [] while dictionary != False:temp = [] temp.append(dictionary["ID"])
temp.append(dictionary["USERID"])                temp.append(dictionary["DATE"])
temp.append(dictionary["EXPENSENAME"])    temp.append(dictionary["AMOUNT"])
temp.append(dictionary["PAYMODE"])          temp.append(dictionary["CATEGORY"])
expense.append(temp) print(temp)
       dictionary = ibm_db.fetch_assoc(res)
    total=0 t_food=0
    t_entertainment=
    0 t_business=0
    t_rent=0 t_EMI=0
    t_other=0

    for x in expense:
       total += x[4] if
       x[6] == "food":
```

```
            t_food += x[4] elif x[6]
        ==          "entertainment":
        t_entertainment  +=  x[4]
        elif  x[6]  ==  "business":
        t_business += x[4]
        elif x[6] ==
          "rent":t_rent
          += x[4]
        elif x[6] ==
          "EMI":t_EMI +=
          x[4]
elif x[6] ==
          "other":t_other +=
          x[4]
print(total)              print(t_food)
            print(t_entertainment)
            print(t_business)
            print(t_rent)
            print(t_EMI)
            print(t_other)   return
            render_template("toda
            y.html",  texpense  =
            texpense,  expense  =
            expense, total =total ,
                        t_food = t_food,t_entertainment = t_entertainment,
                        t_business = t_business, t_rent = t_rent, t_EMI =
                        t_EMI, t_other = t_other )

    @app.route("/year") def
    year():
      # cursor = mysql.connection.cursor()
      # cursor.execute('SELECT MONTH(date), SUM(amount) FROM expenses WHERE
    userid= %s AND YEAR(DATE(date))= YEAR(now()) GROUP BY MONTH(date) ORDER
    BY
    MONTH(date)
```

```
',(str(session['id'])))# texpense
= cursor.fetchall() # print(texpense)

param1 = "SELECT MONTH(date) as mn, SUM(amount) as tot FROM expenses WHERE userid = " + str(session['id']) + " AND YEAR(date) = YEAR(current timestamp)GROUP BY MONTH(date) ORDER BY MONTH(date)" res1 = ibm_db.exec_immediate(ibm_db_conn, param1) dictionary1 = ibm_db.fetch_assoc(res1) texpense = []
while dictionary1 != False:temp = []
temp.append(dictionary1["MN"])
temp.append(dictionary1["TOT"]
)texpense.append(temp)
print(temp)
dictionary1 = ibm_db.fetch_assoc(res1)
# cursor = mysql.connection.cursor()
# cursor.execute('SELECT * FROM expenses WHERE userid = % s AND YEAR(DATE(date))= YEAR(now()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
# expense = cursor.fetchall()
param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " ANDYEAR(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("toda
    y.html",  texpense  =
    texpense,  expense  =
    expense, total =total ,
                t_food = t_food,t_entertainment = t_entertainment, t_business
                = t_business, t_rent = t_rent,
                t_EMI = t_EMI, t_other = t_other )
  #log-out

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

```
def logout():
   session.pop('loggedin',          None)
session.pop('id',                   None)
session.pop('username',             None)
session.pop('email',     None)     return
render_template('home.html')   port   =
os.getenv('VCAP_APP_PORT',
'8080')if_name ==_" " main ": _
    app.secret_key      =      os.urandom(12)      app.run(debug=True,
    host='0.0.0.0', port=port)
```

deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment metadata: name:
sakthi-flask-node-deployment
spec:
   replicas: 1 selector:
     matchLabels: app:
       flasknode
   template:
     metadata: labels:
       app: flasknode
     spec: containers:
 - name:          flasknode          image:
   icr.io/sakthi_expense_tracker2/flask-
   template2
        imagePullPolicy: Always
        ports:
 - containerPort:  5000  flask-service.yaml:
   apiVersion:  v1kind:  Service  metadata:
   name: flask-app-service
spec:
  selector:   app:
    flask-app
  ports:
```

PNT2022TMID45814

```
 - name: http protocol:
    TCPport: 80
    targetPort:  5000
type: LoadBalancer
```

manifest.yml:

```
applications:
 - name: Python Flask App IBCMR 2022-
   10-
   19random-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,
ContentSUBJECT = "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,
emailVARCHAR(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,
amountVARCHAR(32) NOT NULL, paymode
VARCHAR(32) NOT NULL,
category VARCHAR(32) NOT NULL
```

3. LIMIT

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

8. TESTING:

a.TestCases:

PNT2022TMID45814

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status
IndexPage_TC_OO1	UI	Index Page	Verify user is able to navigate through the page	Get to know about the URL of the application.	1.Enter URL and click go	http://127.0.0.1:5000	Index page of the application need to be viewed by the	Working as expected	Pass
IndexPage_TC_002	Functional	Index Page	Verify user is able to see the login page by clicking on the get started button.	Get to know about the application.	1.Enter URL and click go 2.Click on get started button 3.Verify login/Singup popup displayed or not	http://127.0.0.1:5000/signin	Login/Signup popup should display	Working as expected	Pass
LoginPage_TC_OO3	UI	Signup Page	Verify the UI elements in Login/Signup popup	Filling the appropriate details for creating an account.	1.Enter URL and click go 2.Click onget started button. 3.Verify login/Singup 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/signin	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_OO4	Functional	Login page	Verify user is able to log into application with Valid credentials	By giving valid username and password	1.Enter URL 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	http://127.0.0.1:5000/home	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	By giving invalid username and password	1.Enter URL 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	http://127.0.0.1:5000/signin	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass
HomePage_TC_005	HomePage_TC_006	Home page	Verify user is able to view the features of the application.	User Login to the page to access the features of the application.	1.Enter URL 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 6.Access the features of the application.	http://127.0.0.1:5000/home	User need to view the features of the dashboard.	Working as expected	Pass
HomePage_TC_OO5	Functional	Home page	Verify user is able to access the feature.	Based on the requirements user will choose the feature of the application and click the feature.	1.Enter URL 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 box 5.Click on login button 6.Navigate through the home page. 7.Choose the necessary feature.	http://127.0.0.1:5000/home	User is able to access the features of the application.	Working as expected	Pass
HomePage_TC_006	Functional	Home page	Verify user is able to access the profile page	Based on the requirements user will choose the feature of the application and click on the profile feature.	1.Enter URL 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 box 5.Click on login button 6.Navigate through the home page. 7.Choose the necessary feature 8.Access the feature.	http://127.0.0.1:5000/profile	User is able to access the profile page.	Working as expected	Pass
ProfilePage_TC_007	UI	Profile Page	Verify user is able to view the features of the profile page.	User will click the icon to access the feature.	1.Enter URL 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 box 5.Click on login button 6.Navigate through the home page. 7.Choose the necessary feature 8.Access the feature.	http://127.0.0.1:5000/profile	User is able to view the features of the profile page.	Working as expected	Pass

PNT2022TMID45814

b.User Acceptance Testing

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [PERSONAL EXPENSE TRACKER APPLICATION] project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	3	1	2	16
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	13	12	25	74

3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	50	0	0	50
Security	1	0	0	1
Outsource Shipping	3	0	0	3

Exception Reporting	8	0	0	8
Final Report Output	4	0	0	4
Version Control	2	0	0	2

CHAPTER 8 RESULTS

8. 1 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 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: Integrate your expense tracking app with your e-Commerce 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.

CHAPTER 9 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.

CHAPTER10

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.

CHAPTER 11

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.

CHAPTER 12

APPENDIX

Source Code Github Link : https://github.com/IBM-EPBL/IBM-Project-41969-1660646664

PNT2022TMID45814

Project Demo Link:

https://drive.google.com/drive/folders/1NP5on-kHKpS15ij9cP3Sn_QaZisHGA2k