



SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution. Affiliated to Anna University, Chennai)
Kuniamuthur, Coimbatore - 641 008



PERSONAL EXPENSE TRACKER

A PROJECT REPORT

Submitted by

KARTHIK A (19EUEC064)
KARTHIK R (19EUEC065)
KAVEENA K (19EUEC066)
KAVIN PRASATH V (19EUEC067)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

NOVEMBER 2022



SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution. Affiliated to Anna University, Chennai)
Kuniamuthur, Coimbatore - 641 008



BONAFIDE CERTIFICATE

Certified that this project report titled “**PERSONAL EXPENSE TRACKER**” is the bonafide work of **Mr. KARTHIK A (19EUEC064), Mr. KARTHIK R(19EUEC065), Miss.KAVEENA K (19EUEC066), Mr.KAVIN PRASATH V (19EUEC067)** who carried out the project work under my supervision.

SIGNATURE

Dr S. SASI PRIYA

HEAD OF THE DEPARTMENT

SIGNATURE

Ms. DEVI D

SUPERVISOR

Department of Computer Science and Engineering
Sri Krishna College of Engineering and Technology
Kuniamuthur, Coimbatore

Submitted for the Project viva-voce examination held on_____

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENTS

CHAPTER NO.	TITLE
	ABSTRACT
	LIST OF FIGURES
1	INTRODUCTION <ul style="list-style-type: none">1.1 Project Overview1.2 Purpose
2	LITERATURE SURVEY <ul style="list-style-type: none">2.1 Existing problem2.2 References2.3 Problem Statement Definition
3	IDEATION & PROPOSED SOLUTION <ul style="list-style-type: none">3.1 Empathy Map Canvas3.2 Ideation & Brainstorming3.3 Proposed Solution3.4 Problem Solution fit
4	REQUIREMENT ANALYSIS <ul style="list-style-type: none">4.1 Functional requirement4.2 Non-Functional requirements
5	PROJECT DESIGN <ul style="list-style-type: none">5.1 Data Flow Diagrams5.2 Solution & Technical Architecture5.3 User Stories
6	PROJECT PLANNING & SCHEDULING <ul style="list-style-type: none">6.1 Sprint Planning & Estimation6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

7 CODING & SOLUTIONING

7.1 Feature 1

7.2 Feature 2

8 TESTING

8.1 Test Cases

8.2 User Acceptance Testing

9 RESULTS

9.1 Performance Metrics

10 ADVANTAGES & DISADVANTAGES

11 CONCLUSION

12 FUTURE SCOPE

13 APPENDIX

Source Code

GitHub & Project Demo Link

LIST OF FIGURES

FIGURE NO.	TITLE
1.1	Empathy Map
1.2	Problem Statement & Brainstorm
1.3	Group Ideas & Prioritize
1.4	Problem Solution fit
2.1	Data Flow Diagram
2.2	Architecture diagram
3.1	Sprint Delivery Schedule
3.2	Sprint Report
4.1	Performance Metrics

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 Levvel 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

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

I am	Describe customer with 3-4 key characteristics - who are they?	Describe the customer and their attributes here
I'm trying to	List their outcome or "job" the case about - what are they trying to achieve?	List the thing they are trying to achieve here
but	Describe what problems or barriers stand in the way - what bothers them most?	Describe the problems or barriers that get in the way here
because	Enter the "root cause" of why the problem or barrier exists - what needs to be solved?	Describe the reason the problems or barriers exist
which makes me feel	Describe the emotions from the customer's point of view - how does it impact them emotionally?	Describe the emotions the result from experiencing the problems or barriers

Personal Expense Tracker Application:



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 help 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.



3.2 Ideation & Brainstorming

Brainstorm & Idea Prioritization Template:


Brainstorming provides a free and open environment that encourages everyone within a team to 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 are encouraged 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


Template



Brainstorm & ideaprioritization


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 pre-work 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](#) →


**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


How might we track personal expenses?





Key rules of brainstorming


To run a smooth and productive session


Stay in topic.

Encourage wild ideas.

Defer judgment.

Listen to others.

Go for volume.

If possible, be visual.

Step-2: Brainstorm, Idea Listing and Grouping

Brainstorm

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

10 minutes

Kavinaya N

Day to Day reminder about account balance via mail

Afra Thahseen J

If the user spent high, then send mail

Abdul Waseem Nihhaal KW

If they logged in to the web app, we can send some alert

Jayasri PS

Sending mail when exceeds the limit

Group ideas

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

20 minutes

Frequent monitoring on balance by user.

If they logged in to the web app, we can send some alert

If the user spent high, then send mail

Sending mail when exceeds the limit

Day to Day reminder about account balance via mail

Sending mail after exceeds the limit is useless

Step-3: Idea Prioritization

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

Importance

If each of these

could not get done without any difficulty or cost, which would have the most positive impact?

Feasibility

If the user spent high, then send mail

Day to Day reminder about account balance via mail

If they logged in to the web app, we can send some alert

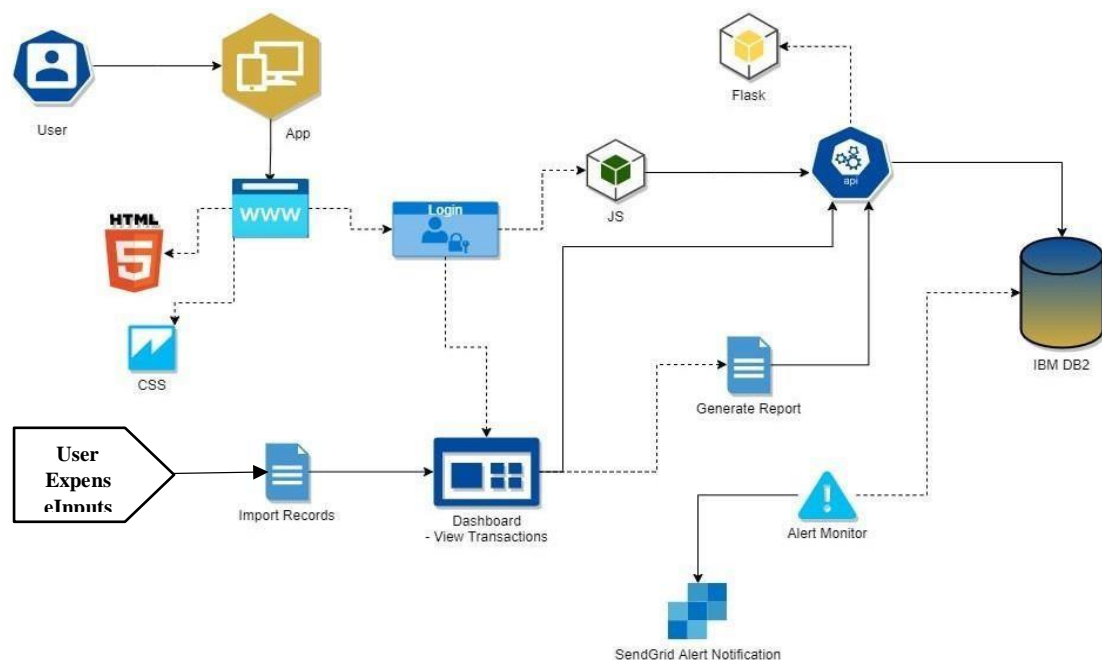
3.3 Proposed Solution

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

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	<p>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.</p> <p>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</p> <p>So in order to optimize their work and make peoples life easier our expense tracker application will be much helpful for financialmanagement</p> <p>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.</p> <p>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.</p>

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



3.4 Problem Solution fit

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) <small>CS</small> Who is your customer? Person who try to manage their expenses in an efficient way	6. CUSTOMER CONSTRAINTS <small>CC</small> What constraints prevent your customers from taking action or limit their choices? No Internet Connection	5. AVAILABLE SOLUTIONS <small>AS</small> Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? Note & pen system, Mind Calculation
--	---	--

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS <small>J&P</small> Which jobs-to-be-done (or problems) do you address for your customers? <ul style="list-style-type: none">Large manual calculations are need to calculate the monthly expensesHuman error may be happen	9. PROBLEM ROOT CAUSE <small>RC</small> What is the real reason that this problem exists? What is the back story behind the need to do this job? <ul style="list-style-type: none">Lack of tracking their expensesNo graphical representation to track their expenses	7. BEHAVIOUR <small>BE</small> What does your customer do to address the problem and get the job done? Easier for the user to follow the process of tracking expenses
---	---	--

Focus on BE, bottom into CS, understand CC

3. TRIGGERS

TR

What triggers customers to act?

It reduces time rather than writing it manually in the notebook

4. EMOTIONS: BEFORE / AFTER

How do customers feel when they face a problem or a job and afterward?

Before: Difficult to track the expense and struggle to analyze their expenses after spending it

After: Easy to track the expenses and graphical representation allow them to understand the expenses they made before easily

8. CHANNELS of BEHAVIOUR

8.1 ONLINE

What kind of actions do customers take online?

- Tracking their expenses
- Getting detailed reviews from email notification

8.2 OFFLINE

What kind of actions do customers take offline?

Export the data when online and use it in offline

10. YOUR SOLUTION



- Create a web application to track their expenses anywhere at any time
- Get real-time graphical representation of their expenses
- Alert notification through email

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 approximatention 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.

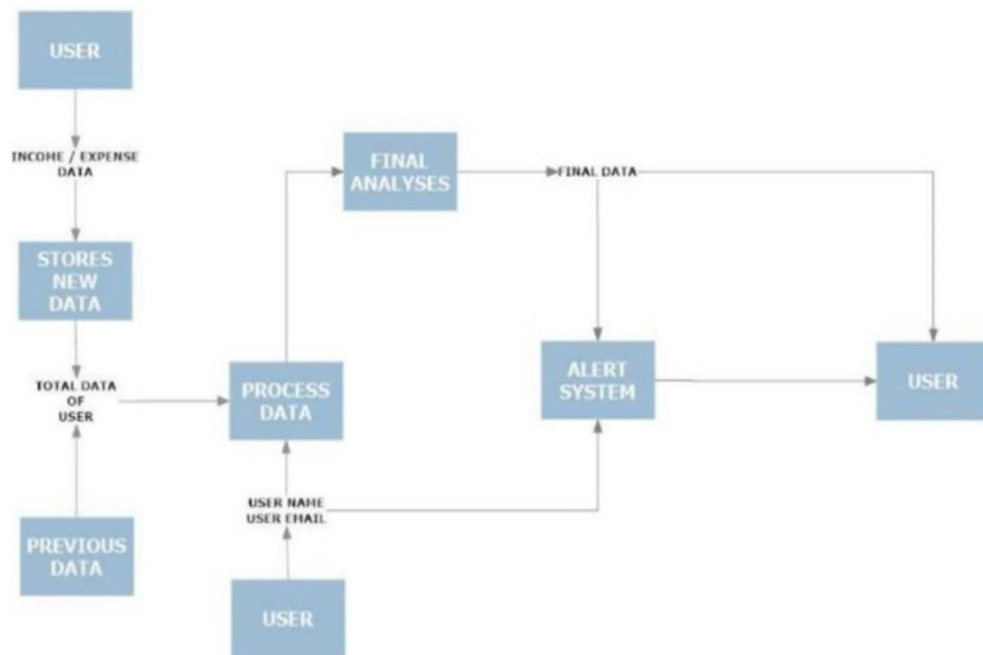
NFR-3	Reliability	<p>he system has to be 100% reliable</p> <p>due to the importance of data and the damages that can be caused by incorrect or incomplete data.</p> <p>The system will run 7 days a week.</p> <p>24 hours a day.</p>
NFR-4	Performance	<p>The information is refreshed depending upon whether some updates have occurred or not in the application.</p> <p>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.</p> <p>Responses to view information shall take no longer than 5 seconds to appear on the screen.</p>
NFR-5	Availability	<p>The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.</p>
NFR-6	Scalability	<p>Scalability is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands.</p>

CHAPTER 5

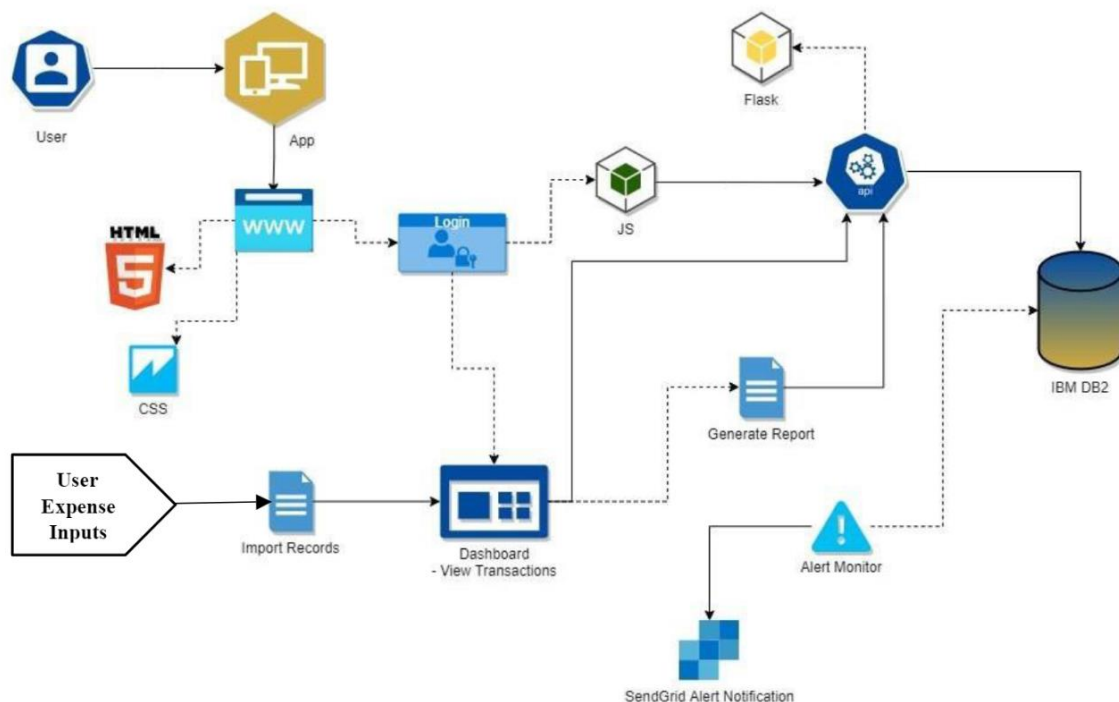
PROJECT DESIGN

5.1 Data Flow Diagrams

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



5.2 Solution & Technical Architecture



5.3 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 myaccount/ 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 mydaily 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

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

S. N O	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

6.2 Sprint Delivery Schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Registration	US N-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Kaveena
		US N-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Kavin Prasath
	Login	US N-3	As a user, I can log into the application by entering email & password	1	High	Karthik A
	Dashboard	US N-4	Logging in takes to the dashboard for the logged user.	2	High	Karthik R
<i>Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only</i>						
Sprint 2	Workspace	US N-1	Workspace for personal expense tracking	2	High	Kavin Prasath
	Charts	US N-2	Creating various graphs and statistics of customer's data	1	Medium	Karthik A
	Connecting to IBM DB2	US N-3	Linking database with dashboard	2	High	Karthik R
		US N-4	Making dashboard interactive with JS	2	High	Kaveena

Sprint-3		US N-1	Wrapping up the server side works of frontend	1	Medium	Karthik R
	Watson Assistant	US N-2	Creating Chatbot for expense tracking and for clarifying user's query	1	Medium	Kavin Prasath
	SendGrid	US N-3	Using SendGrid to send mail to the user about their expenses	1	Low	Kaveena
		US N-4	Integrating both frontend and backend	2		Karthik A
<i>Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only</i>						
Sprint-4	Docker	US N-1	Creating image of website using docker/	2	High	Kaveena
	Cloud Registry	US N-2	Uploading docker image to IBM Cloud registry	2	High	Karthik R
	Kubernetes	US N-3	Create container using the docker image and hosting the site	2	High	Karthik A
	Exposing	US N-4	Exposing IP/Ports for the site	2	High	Kavin Prasath

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

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).

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

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, session# from
flask_mysqlldb import
MySQL
# import
MySQLdb.cursorsimport
re

from flask_db2 import
DB2import ibm_db
import ibm_db_dbi
from sendmail 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.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=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 ")

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'])))# texpanse = cursor.fetchall()

    # print(texpanse)
    param1 = "SELECT TIME(date) as tn, amount FROM expenses WHERE userid = " +
str(session['id']) + " AND DATE(date) = DATE(current timestamp) ORDER BY date
DESC"

    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)

    dictionary1 = ibm_db.fetch_assoc(res1)

    texpanse = []

```

```

while dictionary1 !=
    False:temp = []
    temp.append(dictionary1["TN"])
    temp.append(dictionary1["AMOUNT"])
    texpanse.append(temp)
    print(temp)
    dictionary1 = ibm_db.fetch_assoc(res1)
# cursor = mysql.connection.cursor()
# cursor.execute('SELECT * FROM expenses WHERE userid = % s AND DATE(date) =
DATE(NOW()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
# expense = cursor.fetchall()
param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + "
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", texpanse = texpanse, expense = expense, total =
total ,

        t_food = t_food,t_entertainment = t_entertainment,

        t_business = t_business, t_rent = t_rent,

        t_EMI = t_EMI, t_other = t_other )

@app.route("/month"
)def month():

    # cursor = mysql.connection.cursor()

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

',(str(session['id'])))# texpanse =

cursor.fetchall()

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

    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)

    dictionary1 = ibm_db.fetch_assoc(res1)

    texpanse = []
    while dictionary1 !=

        False:temp = []

        temp.append(dictionary1["DT"])

        temp.append(dictionary1["TOT"])

    )texpanse.append(temp)

    print(temp)

    dictionary1 = ibm_db.fetch_assoc(res1)

```



```

# cursor = mysql.connection.cursor()

# cursor.execute('SELECT * FROM expenses WHERE userid = % s AND
MONTH(Date(date))= MONTH(now()) AND date ORDER BY
`expenses`.`date`DESC',(str(session['id'])))

# expense = cursor.fetchall()
param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND
MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current
timestamp)ORDER BY date DESC"

res = ibm_db.exec_immediate(ibm_db_conn, param)

dictionary = ibm_db.fetch_assoc(res)

expense = []

while dictionary !=
    False:temp = []
    temp.append(dictionary["ID"])
    temp.append(dictionary["USERID"])
    temp.append(dictionary["DATE"])
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    expense.append(temp)
    print(temp)
    dictionary = ibm_db.fetch_assoc(res)

total=0

t_food=0

t_entertainment=0

t_business=0

t_rent=0

t_EMI=0

t_other=0

```

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

    elif x[6] ==
        "entertainment":
            t_entertainment += x[4]

    elif x[6] ==
        "business":
            t_business += x[4]
    elif x[6] ==
        "rent":t_rent
        += x[4]

    elif x[6] ==
        "EMI":t_EMI
        += x[4]

    elif x[6] ==
        "other":t_other
        += x[4]

print(total)
print(t_food)
print(t_entertainment)
print(t_business)
print(t_rent)
print(t_EMI)
print(t_other)
return
render_template("today.html", texpanse =
texpanse, expense =
expense, total =total ,
```

```

t_food = t_food,t_entertainment = t_entertainment,
t_business = t_business, t_rent = t_rent,
t_EMI = t_EMI, t_other = t_other )

```

```
@app.route("/year")
```

```
def year():
```

```

    # cursor = mysql.connection.cursor()

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

```

```
   ',(str(session['id'])))# texpanse
```

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

```
    print(texpanse)
```

```

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

```

```
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
```

```
    dictionary1 = ibm_db.fetch_assoc(res1)
```

```
    texpanse = []
```

```
    while dictionary1 !=
```

```
        False:temp = []
```

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

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

```
        )texpanse.append(temp)
```

```
        print(temp)
```

```
        dictionary1 = ibm_db.fetch_assoc(res1)
```

```
    # cursor = mysql.connection.cursor()
```

```

    # cursor.execute('SELECT * FROM expenses WHERE userid = % s AND
YEAR(DATE(date))= YEAR(now()) AND date ORDER BY
`expenses`.`date` DESC',(str(session['id'])))

```

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

```
param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + "
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("today.html", texpense =
texpense, expense =
expense, total =total ,

        t_food = t_food,t_entertainment = t_entertainment,
        t_business = t_business, t_rent = t_rent,
        t_EMI = t_EMI, t_other = t_other )

#log-out

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

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

deployment.yaml:

apiVersion: apps/v1

kind: Deployment

metadata:

name: sakthi-flask-node-deployment

spec:

replicas: 1

selector:

matchLabels:

app: flasknode

template:

metadata:

labels:

app: flasknode

spec:

containers:

- name: flasknode

image: icr.io/sakthi_expense_tracker2/flask-template2

imagePullPolicy: Always

ports:

- containerPort: 5000

flask-service.yaml:

apiVersion:

v1kind:

Service

metadata:

name: flask-app-service

spec:

selector:

app: flask-app

ports:

- 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",
"lxiqbmpnxbkiemh")message = 'Subject:
{ }\n{n{ }' .format(SUBJECT, TEXT)
# s.sendmail("il.tproduct8080@gmail.com", email, message)
s.sendmail("il.tproduct8080@gmail.com", email, message)
s.quit()
def sendgridmail(user,TEXT):
    # from_email =
    Email("shridhartp24@gmail.com")from_email =
    Email("tproduct8080@gmail.com") to_email =
    To(user)
    subject = "Sending with SendGrid is
Fun"content =
    Content("text/plain",TEXT)
    mail = Mail(from_email, to_email, subject, content)
    # Get a JSON-ready representation of the Mail object
    mail_json = mail.get()
    # Send an HTTP POST request to /mail/send
    response = sg.client.mail.send.post(request_body=mail_json)
    print(response.status_code)
    print(response.headers)

```

Database Schema

Tables :

1.Admin

:

```

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

```

2.Expense:

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

3.LIMIT

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

8. TESTING:

a.TestCases:

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	BUG ID	Executed By
LoginPage_TC_001	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	-		Kavinaya
LoginPage_TC_002	Functional	Home Page	Verify that the error message is displayed when the user enters the wrong credentials	1.Go to website 2. Enter Invalid username and password	Username: XXXX Password: 12345	Error message should displayed	Working as expected	Pass	-		Afra
LoginPage_TC_002	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	-		Abdul Waseem
LoginPage_TC_003	Functional	Home page	Verify user is able to log into application with Valid credentials	1. Go to website 2. Enter details and click login	Username: Kavi password: 123456	User should navigate to user account homepage	Working as expected	Pass	-		Jayasri
LoginPage_TC_004	Functional	Login page	Verify user is able to log into application with Invalid credentials	1. Go to website 2. Enter details and click login	Username: Kavi password: 123456	Application should show 'Incorrect email or password' validation message.	Working as expected	Pass	-		Afra
LoginPage_TC_004	Functional	Login page	Verify user is able to log into application with Invalid credentials	1. Go to website 2. Enter details and click login	Username: Kavi password: 123456	Application should show 'Incorrect email or password' validation message.	Working as expected	Pass	-		Kavinaya
LoginPage_TC_005	Functional	Login page	Verify user is able to log into application with Invalid credentials	1. Go to website 2. Enter details and click login	Username: Kavi password: 123456	Application should show 'Incorrect email or password' validation message.	Working as expected	Pass	-		Abdul Waseem
AddExpensePage_TC_006	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	-		Jayasri

b. User Acceptance Testing

1. Defect Analysis

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

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

2. Test Case Analysis

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

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

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 and automatically matches the payments with invoices.
- v. Reports: The expense tracking app generates and sends reports to give a detailed insight about profits, losses, budgets, income, balance sheets, etc.,
- vi. Ecommerce integration: Integrate your expense tracking app with your eCommerce store and track your sales through payments received via multiple payment methods.
- vii. Vendors and Contractors: Manage and track all the payments to the vendors and contractors added to the mobile app.
- viii. Access control: Increase your team productivity by providing access control to particular users through custom permissions.
- ix. Track Projects: Determine project profitability by tracking labor costs, payroll, expenses, etc., of your ongoing project.
- x. Inventory tracking: An expense tracking app can do it all. Right from tracking products or the cost of goods, sending alert notifications when the product is running out of stock or the product is not selling, to purchase orders.
- xi. In-depth insights and analytics: Provides in-built tools to generate reports with easy-to-understand visuals and graphics to gain insights about the performance of your business.
- xii. Recurrent Expenses: Rely on your budgeting app to track, streamline, and automate all the recurrent expenses and remind you on a timely basis.

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.

CHAPTER 10 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-15933-1659606190>

Project Demo Link:

https://drive.google.com/file/d/1dapdiNPHZanndmuksmrGXvQ3xHdJfP0W/view?usp=share_link

