PERSONAL EXPENSE TRACKER APPLICATION

PROJECT IBM REPORT

Submitted by

TEAM ID:PNT2022TMID17019

ARUN M	**191041010**
HRITHIK S	**191041028**
LOGESHWARAN P	**191041049**
MOHAMMED FHAYAZ M	**191041057**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

In

ELECTRONICS AND COMMUNICATION ENGINEERING

MAHENDRA ENGINEERING COLLEGE

(Autonomous)

Mahendhirapuri, Mallasamudram

Namakkal Dt.-637 503

INDEX

1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

2. LITERATURE SURVEY

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

a . Empathy Map Canvas

b. Ideation & Brainstorming

c. Proposed Solution

d. Problem Solution fit

4. REQUIREMENT ANALYSIS

a. Functional requirement

b. Non-Functional requirements

2. PROJECT DESIGN

a. Data Flow Diagrams

b. Solution & Technical Architecture

c . User Stories

2. PROJECT PLANNING & SCHEDULING

a. Sprint Planning & Estimation

b. Sprint Delivery Schedule

c. Reports from JIRA

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

a. Feature 1

b. Feature 2

c. Database Schema (if Applicable)

8. TESTING

a. Test Cases

b. User Acceptance Testing

2. RESULTS

a. Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

1. INTRODUCTION

a. Project Overview

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

b. Purpose

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

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

2. LITERATURE SURVEY

a. Existing problem

In a study conducted by Forrester in 2016 surveying small and medium businesses (SMBs) across the world, 56% companies reported expense management as being the biggest challenge for their finance departments. In another survey conducted by Level Research in 2018 in North America, respondents reported the following pain points in expense management before adopting automation:

i. Manual entry and routing of expense reports (62%)

ii. Lack of visibility into spend data (42%)

iii. Inability to enforce travel policies (29%)

iv. Lost expense reports (24%)

v. Lengthy expense approval system and reimbursement cycles (23%)

b. References

[1] Title: Security and privacy challenges in mobile cloud computing: survey and way ahead

Author: Muhammad Baqer Mollah et al

Journal: Journal of Network and Computer Applications

Year:2017

Methodology:

Computational offloading, Virtualization

Scope: primary security and privacy issues facing cloud computing are highlighted in this survey in order to raise awareness within the academic and scientific communities. While there are many difficulties, comparable security solutions have been suggested and found in the literature by numerous researchers to address the difficulties. The recent works are also briefly presented in this work.

[2] Title: Exploring infrastructure support for app based services on cloud platforms

Author: Hai Nguyen et al

Journal: Journal of Cloud Computing Advances Systems and Applications

Year:2017

Methodology: Virtualization, Introspection and Security

Scope: In this paper, a rich model's design and implementation are discussed, allowing third-party cloud apps to access a client's virtual machines (VMs) and carry out privileged operations. The infrastructure support required to support cloud apps was discussed.Different design approaches to deploy cloud apps were also addressed.

[3] Title : Mobile Financial Management Application using Google Cloud Vision API

Author: Kurniawan Dwi Saputra et al

Journal: International Conference on Computer Science and Computational Intelligence

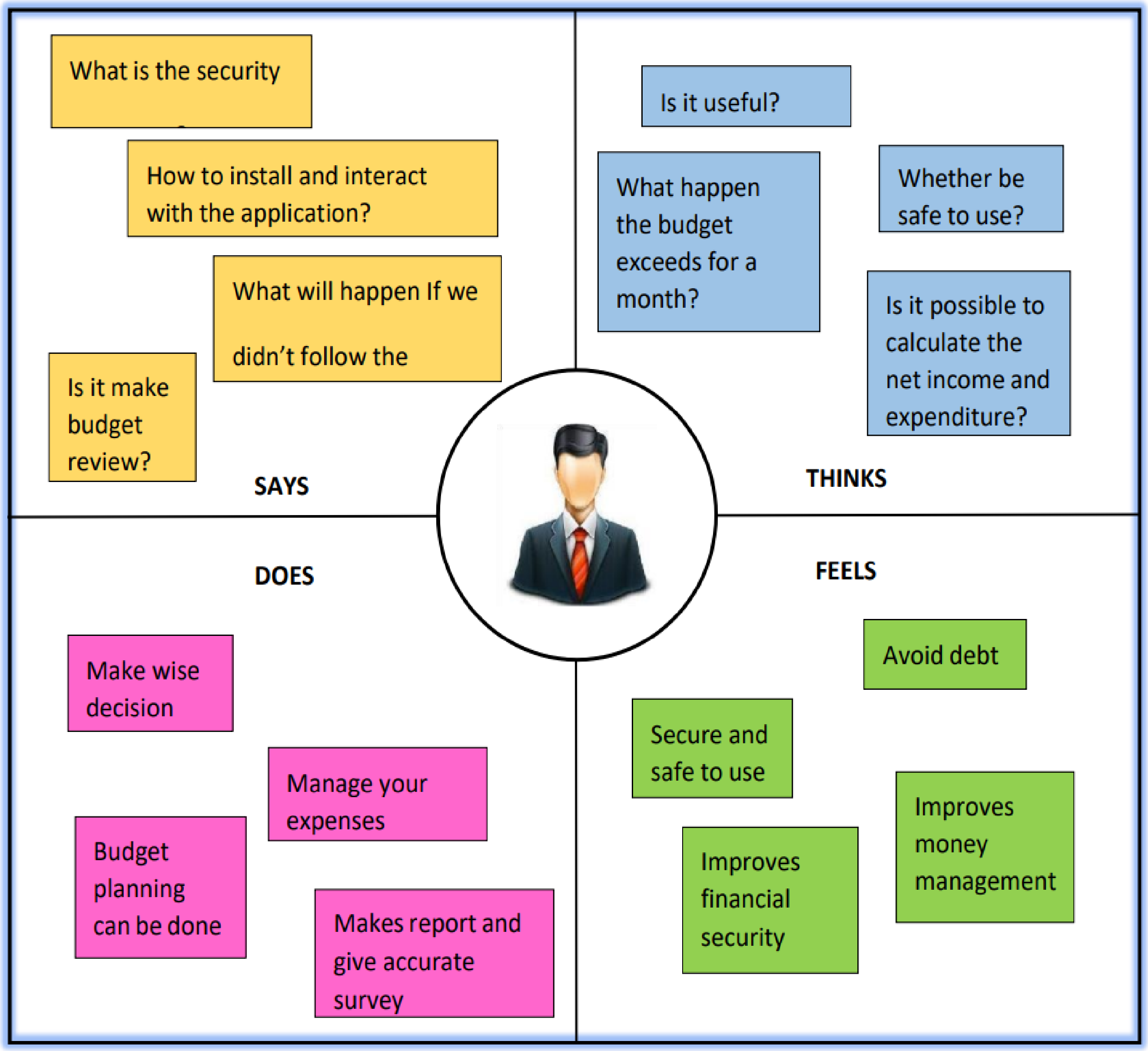
Year:2019

Methodology: Google Vision Cloud API, Optical Character Recognition

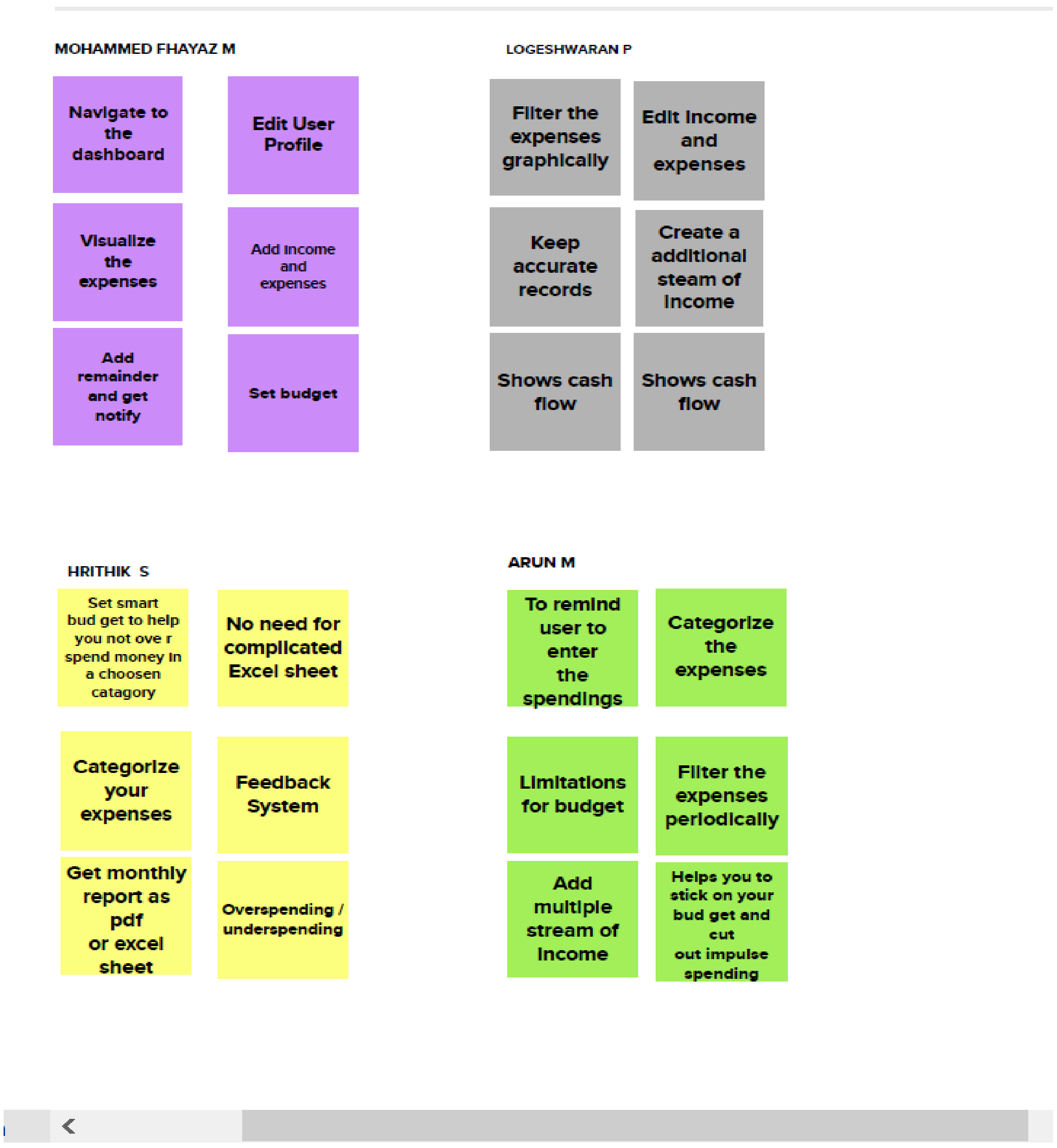
Scope: In order to address the primary financial issues , this study looked at the potential usefulness of the mobile application "Manage on Money (MoM)".OCR technology was created using Google Cloud Vision API. This technique works well for locating a single precise keyword on a receipt printed in black ink. MoM enables users to arrange their recurring expenses and sends a push reminder prior to the due date. One Signal API serves as the foundation for this notification.

1. IDEATION & PROPOSED SOLUTION

a. Empathy Map Canvas


What is the security
How to install and interact with the application?
What will happen If we didn't follow the
Is it make budget review?
SAYS
Is it useful?
What happen the budget exceeds for a month?
Whether be safe to use?
Is it possible to calculate the net income and expenditure?
THINKS
DOES
Make wise decision
Manage your expenses
Budget planning can be done
Makes report and give accurate survey
FEELS
Avoid debt
Secure and safe to use
Improves financial security
Improves money management

B. Ideation & Brainstorming



MOHAMMED FHAYAZ M

- Navigate to the dashboard
- Edit User Profile
- Visualize the expenses
- Add income and expenses
- Add remainder and get notify
- Set budget

LOGESHWARAN P

- Filter the expenses graphically
- Edit income and expenses
- Keep accurate records
- Create a additional steam of income
- Shows cash flow
- Shows cash flow

HRITHIK S

- Set smart bud get to help you not ove r spend money in a choosen catagory
- No need for complicated Excel sheet
- Categorize your expenses
- Feedback System
- Get monthly report as pdf or excel sheet
- Overspending / underspending

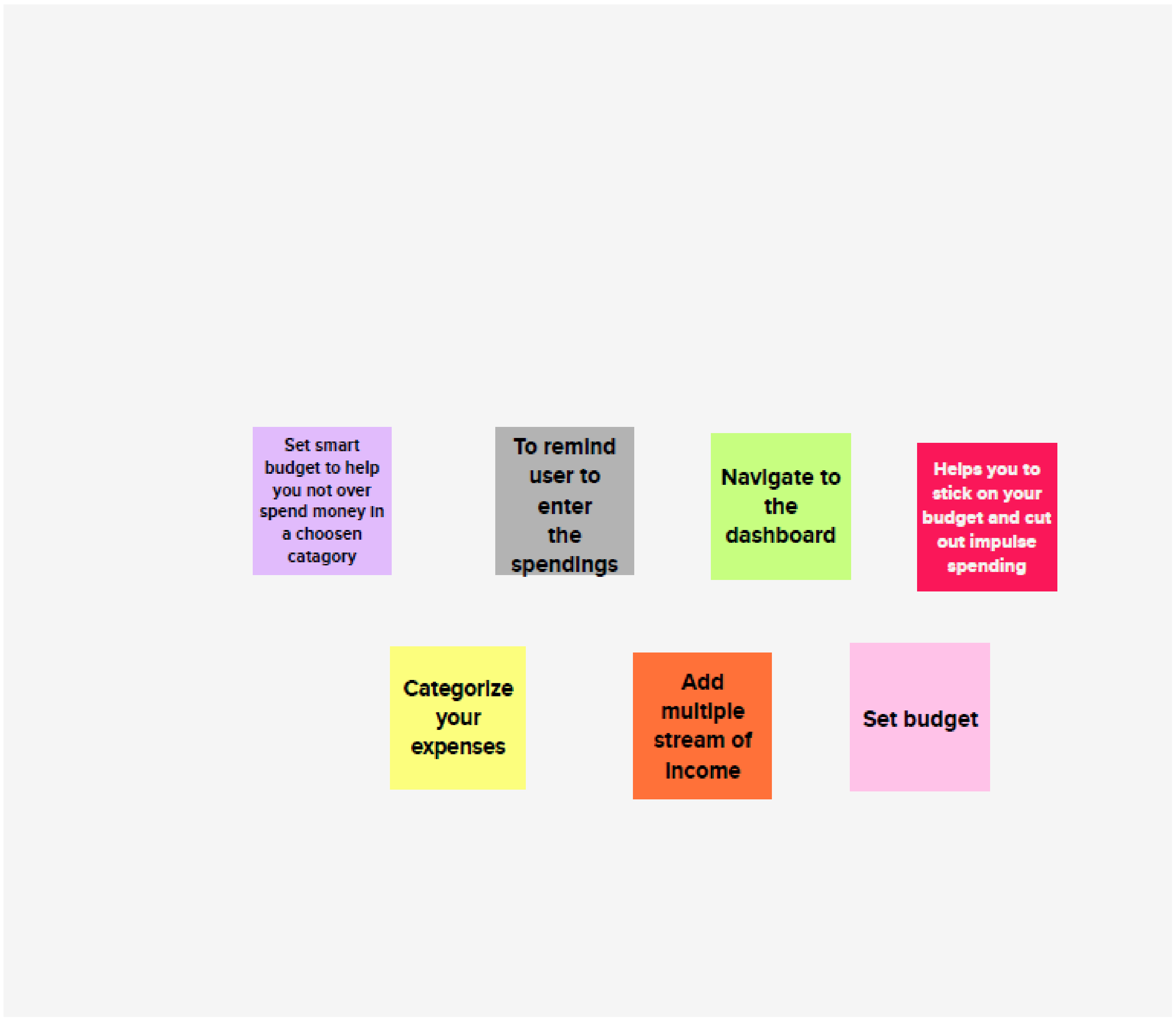
ARUN M

- To remind user to enter the spendings
- Categorize the expenses
- Limitations for budget
- Filter the expenses periodically
- Add multiple stream of income
- Helps you to stick on your bud get and cut out impulse spending

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



Set smart budget to help you not over spend money in a choosen catagory

To remind user to enter the spendings

Navigate to the dashboard

Helps you to stick on your budget and cut out impulse spending

Categorize your expenses

Add multiple stream of income

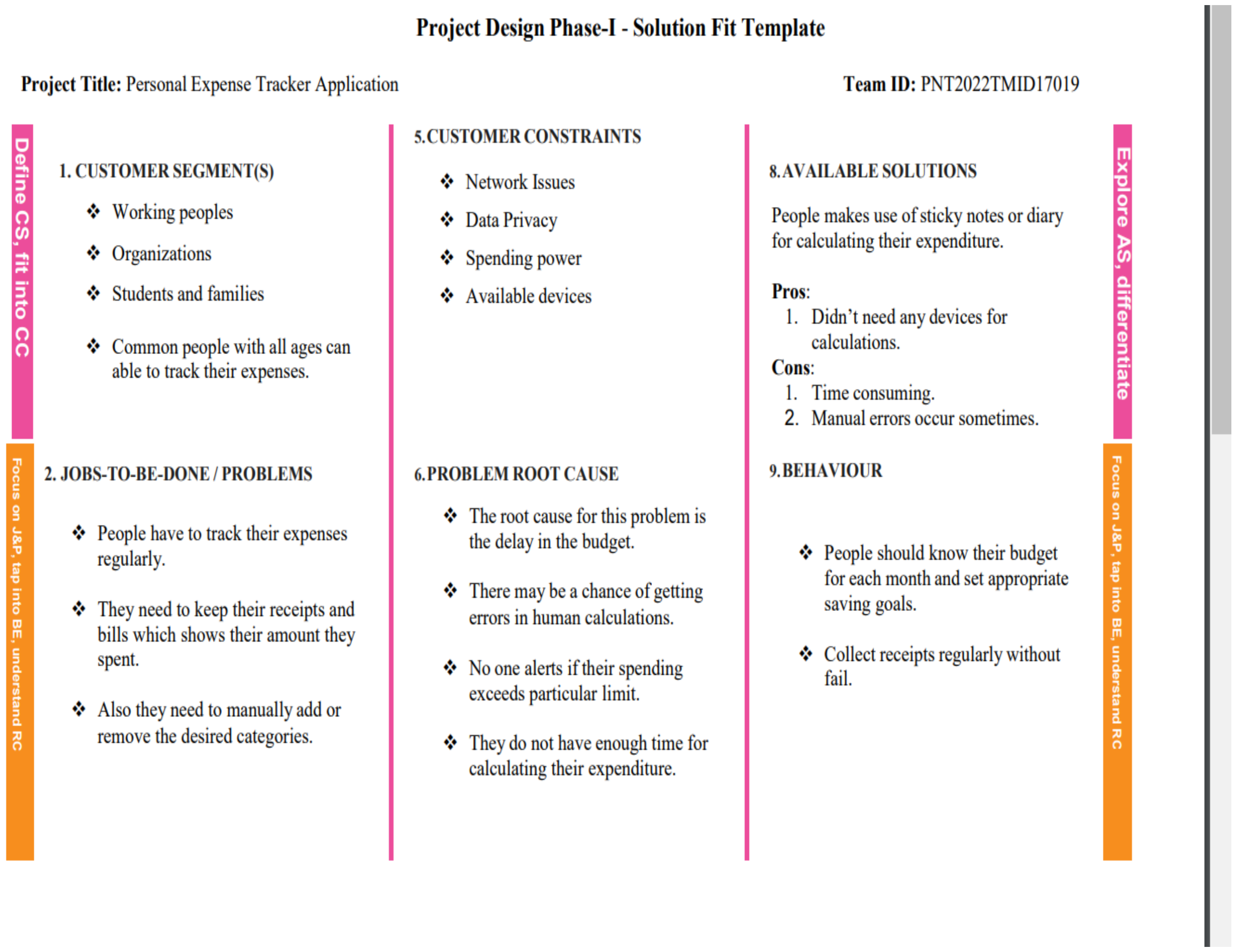
Set budget

C.Proposed Solution

S.No	Parameters	Description
1	Problem Statement (Problem to be solved)	Tracking expenses is one of the key factors in making budget work. At the instant, there is no as such complete solution present easily which enables a person to keep a track of its daily expenditure easily. To do so a person has to keep a log in a diary or in a computer, also all the calculations need to be done by the user which may sometimes results in errors leading to losses. Due to lack of a complete tracking system, there is a constant overload to rely on the daily entry of the expenditure.
2	Idea / Solution description	In this project, we developed a cloud-based web application which keeps track of user's personal expenses. This system attempts to free the user with as much as possible the burden of manual calculation and to keep the track of the expenditure. This system also eliminates sticky notes, bills.
3	Novelty / Uniqueness	This personal expense tracker Application has features that enables the user to have an option to set a limit for the amount to be used. If the limit is exceeded the user will be notified with an Email and SMS alert. This tracker doesn't have annoying ads.

4	Social Impact / Customer Satisfaction	The user will be able to Stick to their Spending Limits. They can able to scan their bills anytime thus data loss is avoided. Users can also able to get an analysis of their expenditure in graphical forms.
5	Business Model (Revenue Model)	This application will generate revenue by offering premium features to the user. Advertising through app is the easy way to earn money. Users may pay to remove the app advertisements. Through subscription the users can able to connect their bank account.
6	Scalability of the Solution	Since this application is deployed on IBM Cloud, it can handle multiple users at a time. With our application, the users can be able to manage their expenses more effectively and know about their budget Vs income.

d. Problem Solution Fit



Project Design Phase-I - Solution Fit Template

Project Title: Personal Expense Tracker Application

Team ID: PNT2022TMID17019

Define CS, fit into CC

1. CUSTOMER SEGMENT(S)

- ❖ Working peoples
- ❖ Organizations
- ❖ Students and families
- ❖ Common people with all ages can able to track their expenses.

5. CUSTOMER CONSTRAINTS

- ❖ Network Issues
- ❖ Data Privacy
- ❖ Spending power
- ❖ Available devices

8. AVAILABLE SOLUTIONS

People makes use of sticky notes or diary for calculating their expenditure.

Pros:

1. Didn't need any devices for calculations.

Cons:

1. Time consuming.
2. Manual errors occur sometimes.

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS

- ❖ People have to track their expenses regularly.
- ❖ They need to keep their receipts and bills which shows their amount they spent.
- ❖ Also they need to manually add or remove the desired categories.

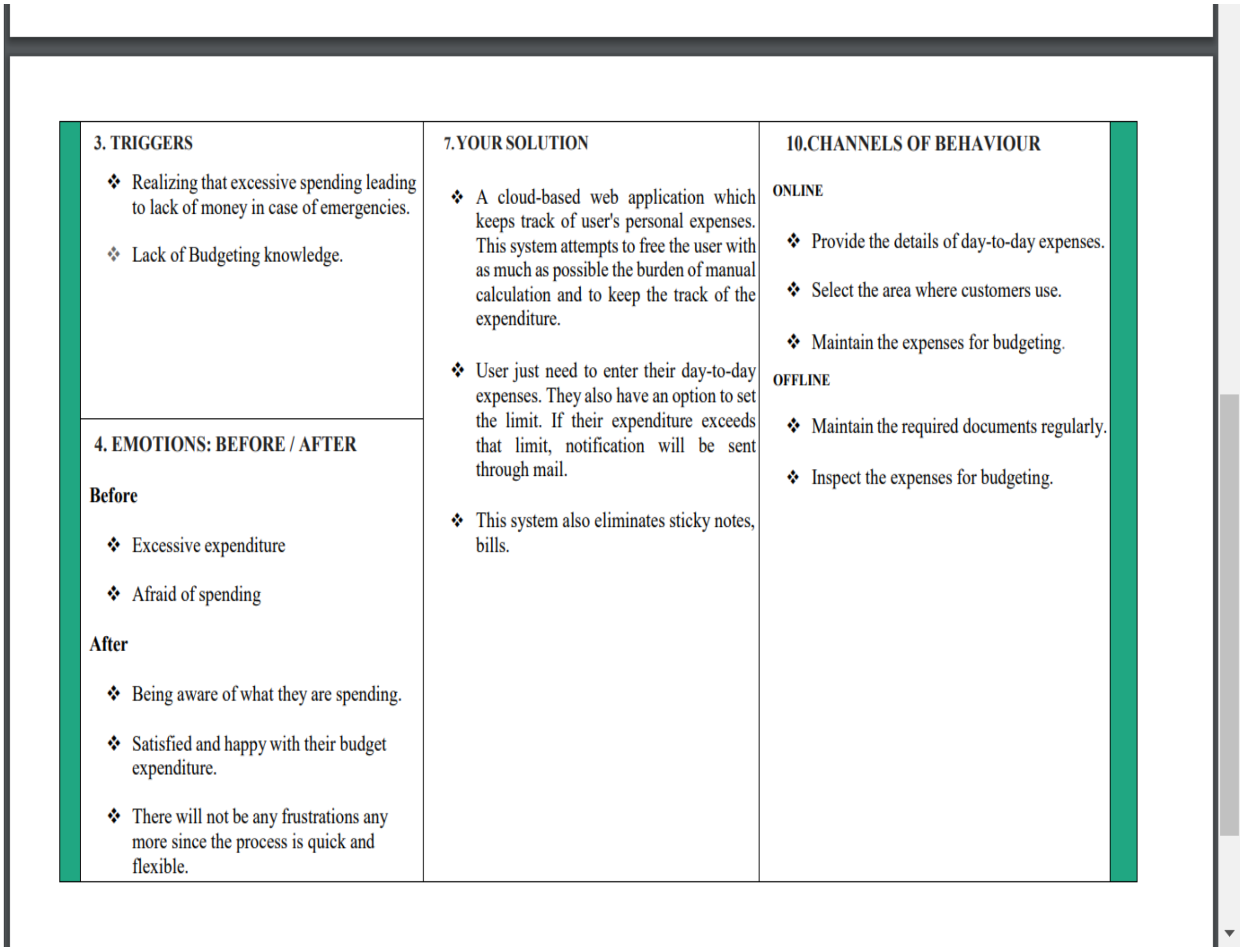
6. PROBLEM ROOT CAUSE

- ❖ The root cause for this problem is the delay in the budget.
- ❖ There may be a chance of getting errors in human calculations.
- ❖ No one alerts if their spending exceeds particular limit.
- ❖ They do not have enough time for calculating their expenditure.

9. BEHAVIOUR

- ❖ People should know their budget for each month and set appropriate saving goals.
- ❖ Collect receipts regularly without fail.

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



3. TRIGGERS

- ❖ Realizing that excessive spending leading to lack of money in case of emergencies.
- ❖ Lack of Budgeting knowledge.

4. EMOTIONS: BEFORE / AFTER

Before

- ❖ Excessive expenditure
- ❖ Afraid of spending

After

- ❖ Being aware of what they are spending.
- ❖ Satisfied and happy with their budget expenditure.
- ❖ There will not be any frustrations any more since the process is quick and flexible.

7.YOUR SOLUTION

- ❖ A cloud-based web application which keeps track of user's personal expenses. This system attempts to free the user with as much as possible the burden of manual calculation and to keep the track of the expenditure.
- ❖ User just need to enter their day-to-day expenses. They also have an option to set the limit. If their expenditure exceeds that limit, notification will be sent through mail.
- ❖ This system also eliminates sticky notes, bills.

10.CHANNELS OF BEHAVIOUR

ONLINE

- ❖ Provide the details of day-to-day expenses.
- ❖ Select the area where customers use.
- ❖ Maintain the expenses for budgeting.

OFFLINE

- ❖ Maintain the required documents regularly.
- ❖ Inspect the expenses for budgeting.

4. Requirement Analysis

a. Functional Requirements

FR No.	Functional Requirement	Description
FR-1	**Register**	Registration is the process of the user to complete the application's form. Certain details must be submitted such as e-mail address, password, and password confirmation. The user is identified using these details.
FR-2	**Login**	The login screen is used to verify the identity of the user. The account can be accessed using the user's registered email address and password.
FR-3	**Categories**	On the main page, we can see overall revenue and spending, as well as the balance remaining after expenditure, as well as the user's entire categories namely Entertainment, Cloth, Food and Drinks, Health and Fitness and so on.
FR-4	**Update Daily Expensive**	The user can upload the daily expensive details what they are spending on each day. The details such as cloth, entertainment, food, health etc.,
FR-5	**View Expensive Chart**	This module used to see a pictorial depiction of all details in the form of a pie chart, where each slice of the pie chart represents that the viewer to gain an approximate notion of which category has the highest expenses.
FR-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.

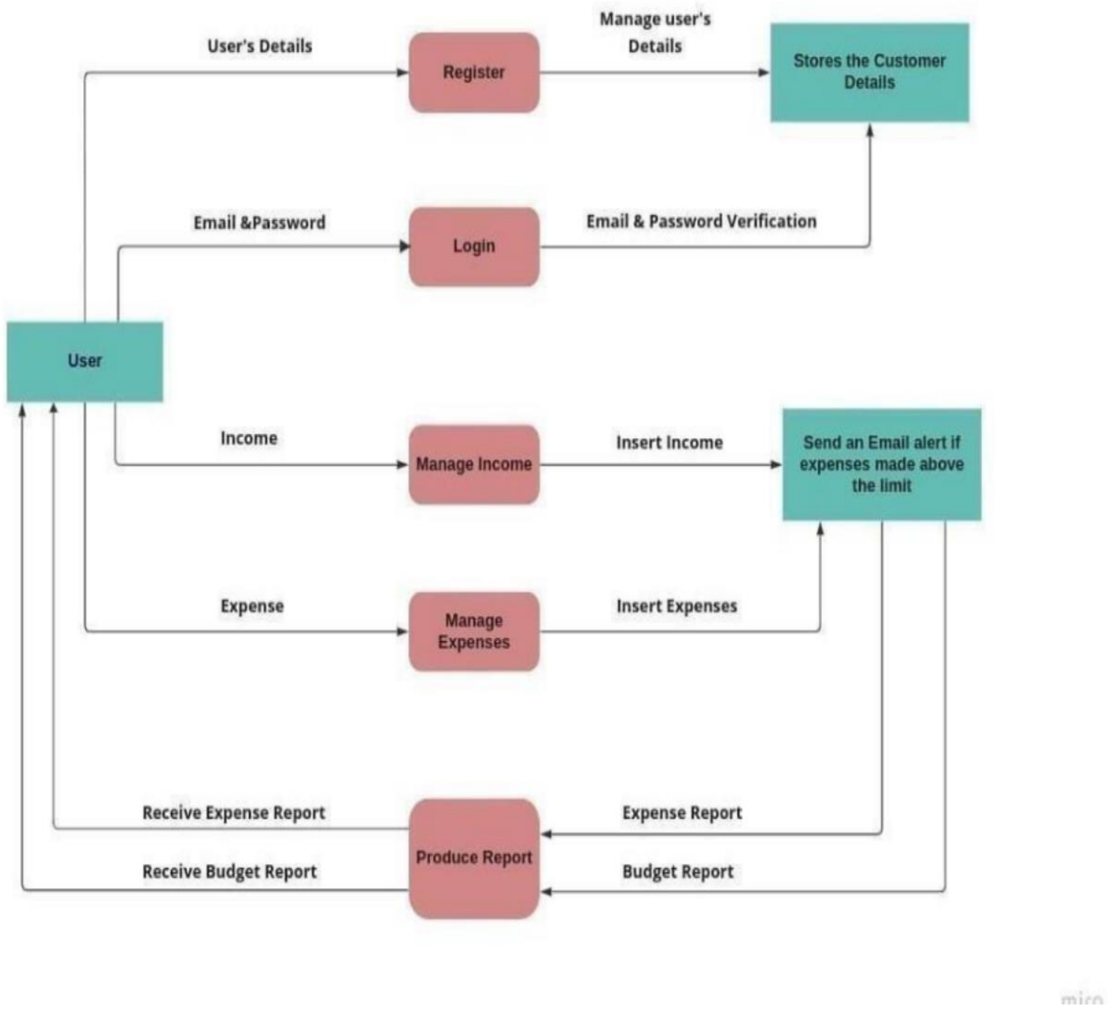
b. Non-Functional requirements

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	The system shall allow the users to access the system with pc using web application. The system uses a web application as an interface. The system is user friendly which makes the system easy.
NFR-2	Security	A security requirement is a statement of needed security functionality that ensures one of many different security properties of software is being satisfied.
NFR-3	Reliability	he system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week. 24 hours a day. NFR-
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 issued 24 hours a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.
NFR-6	Scalability	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.

5. PROJECT DESIGN

a. Data Flow Diagrams



b.Solution & Technical Architecture

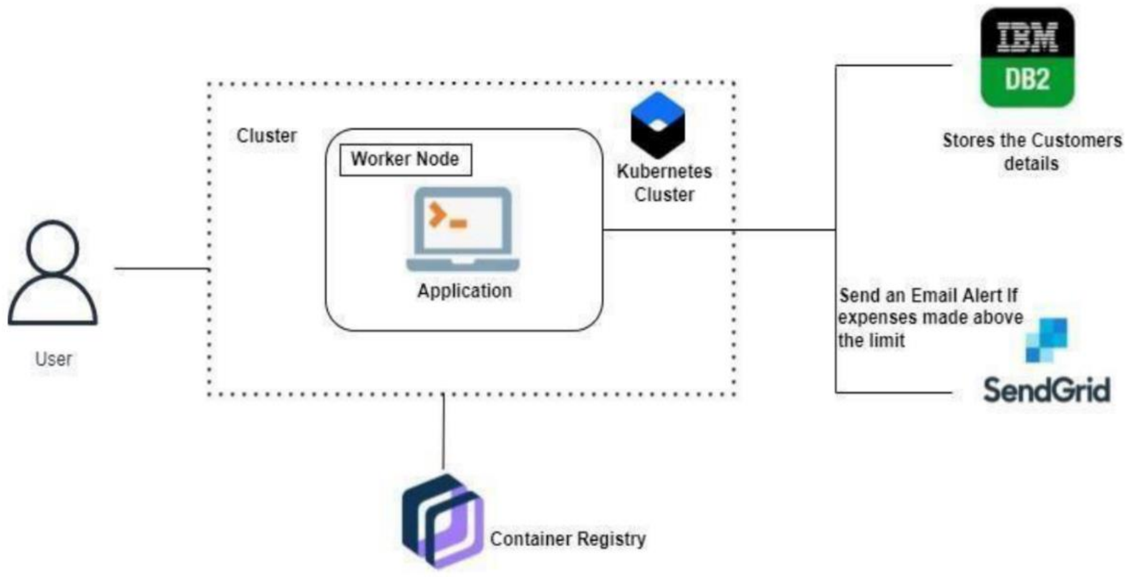


Table-1: Components & Technologies:

S.No.	Component	Description	Technology
1.	User Interface	The user can Interact with the application with use of Chatbot	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	The application contains the sign in/sign up where the user will login into the main dashboard	Java / Python
3.	Application Logic-2	Dashboard contains the fields like Add income, Add Expenses, Save Money	IBM Watson STT service
4.	Application Logic-3	The user will get the expense report in the graph form and also get alerts if the expense limit exceeds	IBM Watson Assistant,SendGrid
5.	Database	The Income and Expense data are stored in the MySQL database	MySQL, NoSQL, etc.
6.	Cloud Database	With use of Database Service on Cloud, the Jser data are stored in a well secured Manner	IBM DB2, IBM Cloudant etc.
7.	File Storage	IBM Block Storage used to store the Financial data of the user	IBM Block Storage or Other Storage Service or Local Filesystem

Table-2: Application Characteristics:

S.No.	Characteristics	Description	Technology
1.	Open-Source Frameworks	Flask Framework in Python is used to implement this Application	Python-Flask
2.	Security Implementations	This Application Provides high security to the user Financial data. It can be done by using the Container Registry in IBM cloud	Container Registry, Kubernetes Cluster
3.	Scalable Architecture	Expense Tracker is a life time access supplication. It's demand will increase when the user's income are high	Container Registry, Kubernetes Cluster
4.	Availability	This application will be available to the user at any part of time	Container Registry, Kubernetes Cluster
5.	Performance	The performance will be high because there will be no network traffics in the application	Kubernetes Cluster

C.User Stories

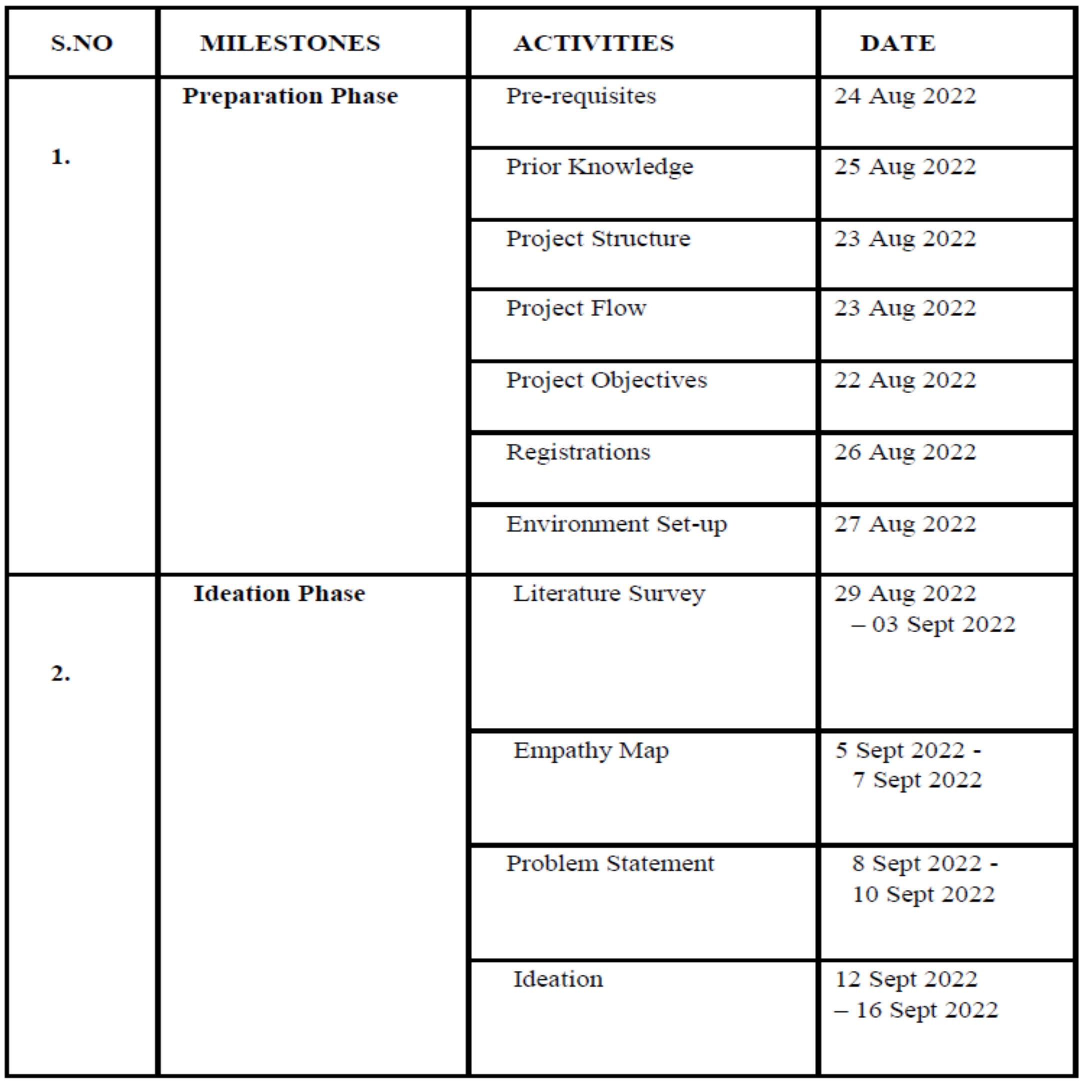
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account /dashboard	High	
	Login	USN-2	As a user, I can log into the application by entering email & password	I can access theapplication	High	
	Dashboard	USN-3	As a user I can enter my income andexpenditure details.	I can view my dailyexpenses	High	
Customer Care Executive		USN-4	As a customer care executive, I can solvethe log in issues and other issues of the application.	I can provide support or solution at any time 24*7	Medium	
Administrator	Application	USN-5	As an administrator I can upgrade or update the application.	I can fix the bug which arises for the customersand users of the application	Medium	

6. PROJECT PLANNING & SCHEDULING

a. Sprint planning and estimation

Sprit 4	20	6 days	14 NOV 2022	20 NOV 2022	20	19 NOV 2022

a. Sprint Delivery Schedule



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

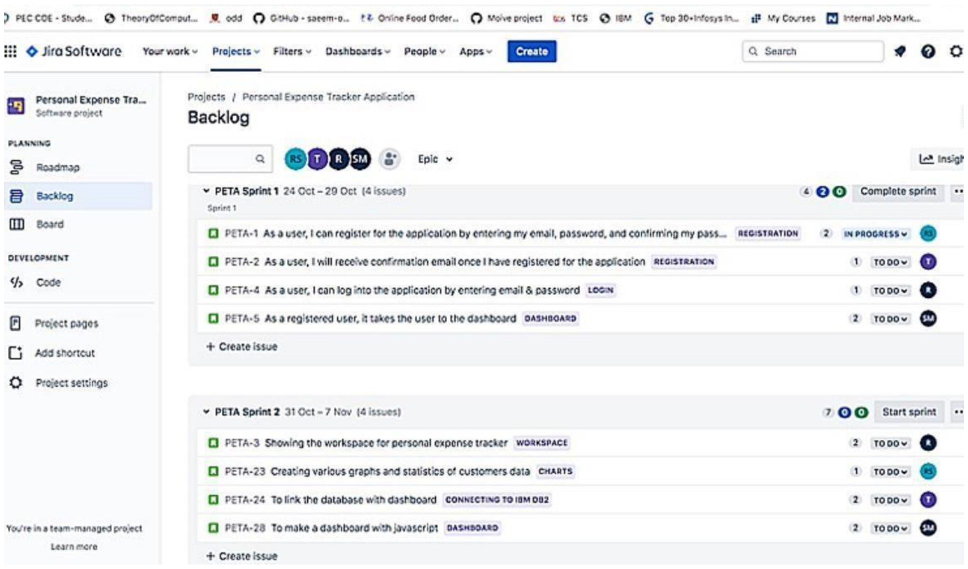
3.	**Project Design Phase - 1**	Proposed Solution	19 Sept 2022 – 23 Sept 2022
		Problem Solution Fit	24 Sept 2022 – 26 Sept 2022

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

a. Reports from JIRA

i. Backlog



Projects / Personal Expense Tracker Application

Backlog

PETA Sprint 1 24 Oct – 29 Oct (4 issues) 4 2 0 Complete sprint

Sprint 1

PETA-1 As a user, I can register for the application by entering my email, password, and confirming my pass... REGISTRATION 2 IN PROGRESS

PETA-2 As a user, I will receive confirmation email once I have registered for the application REGISTRATION 1 TO DO

PETA-4 As a user, I can log into the application by entering email & password LOGIN 1 TO DO

PETA-5 As a registered user, it takes the user to the dashboard DASHBOARD 2 TO DO

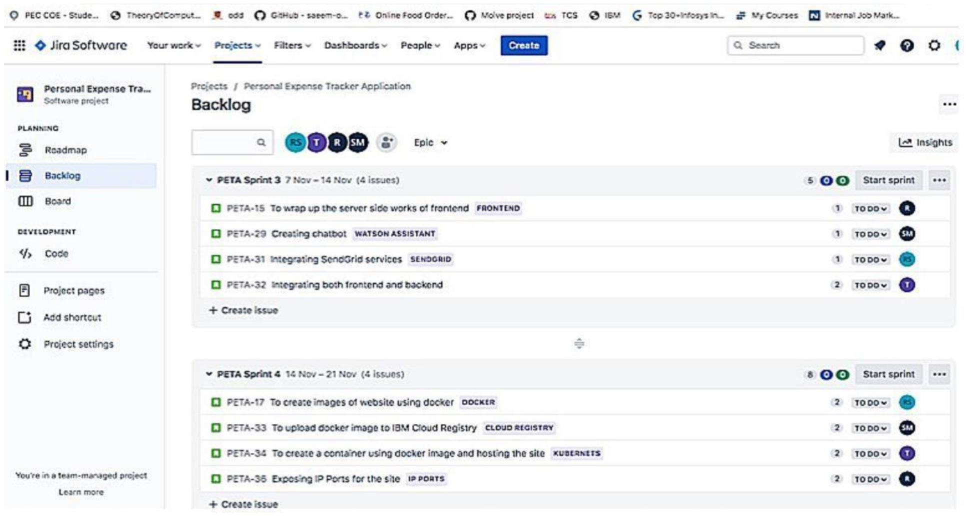
PETA Sprint 2 31 Oct – 7 Nov (4 issues) 7 0 0 Start sprint

PETA-3 Showing the workspace for personal expense tracker WORKSPACE 2 TO DO

PETA-23 Creating various graphs and statistics of customers data CHARTS 1 TO DO

PETA-24 To link the database with dashboard CONNECTING TO IBM DB2 2 TO DO

PETA-28 To make a dashboard with javascript DASHBOARD 2 TO DO



Projects / Personal Expense Tracker Application

Backlog

PETA Sprint 3 7 Nov – 14 Nov (4 issues) 5 0 0 Start sprint

PETA-15 To wrap up the server side works of frontend FRONTEND 1 TO DO

PETA-29 Creating chatbot WATSON ASSISTANT 1 TO DO

PETA-31 Integrating SendGrid services SENDGRID 1 TO DO

PETA-32 Integrating both frontend and backend 2 TO DO

PETA Sprint 4 14 Nov – 21 Nov (4 issues) 8 0 0 Start sprint

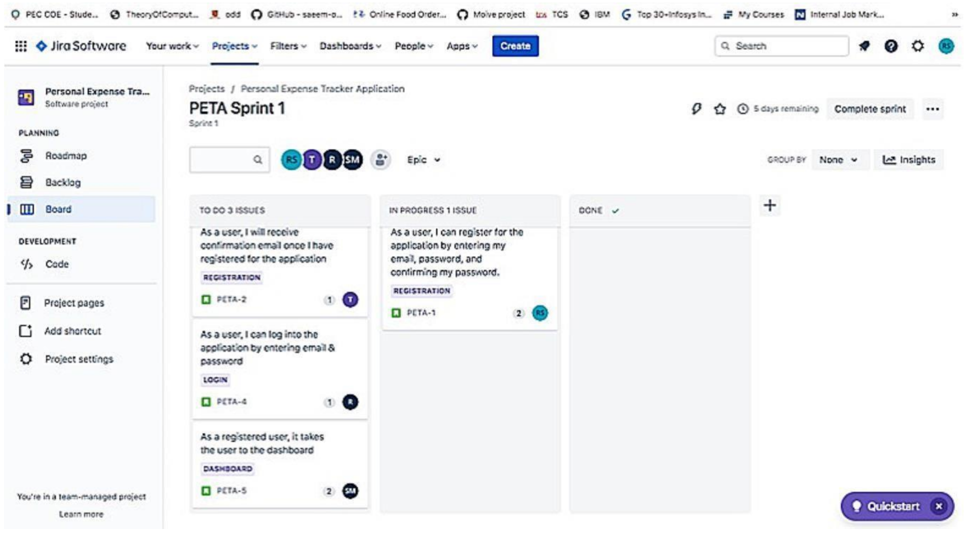
PETA-17 To create images of website using docker DOCKER 2 TO DO

PETA-33 To upload docker image to IBM Cloud Registry CLOUD REGISTRY 2 TO DO

PETA-34 To create a container using docker image and hosting the site KUBERNETS 2 TO DO

PETA-36 Exposing IP Ports for the site IP PORTS 2 TO DO

ii. Board



Personal Expense Tra...
Software project

Projects / Personal Expense Tracker Application

PETA Sprint 1
Sprint 1

5 days remaining Complete sprint

GROUP BY None Insights

TO DO 3 ISSUES

As a user, I will receive confirmation email once I have registered for the application
REGISTRATION
PETA-2 1 T

As a user, I can log into the application by entering email & password
LOGIN
PETA-4 1 R

As a registered user, it takes the user to the dashboard
DASHBOARD
PETA-5 2 SM

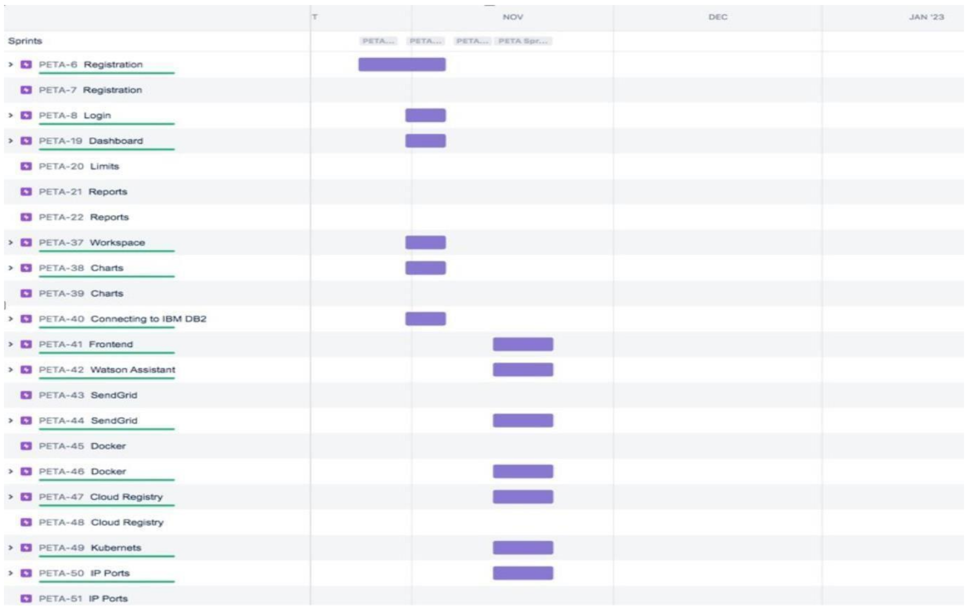
IN PROGRESS 1 ISSUE

As a user, I can register for the application by entering my email, password, and confirming my password.
REGISTRATION
PETA-1 2 RS

DONE

You're in a team-managed project
Learn more

iii. Road Map



Sprints

PETA-6 Registration
PETA-7 Registration
PETA-8 Login
PETA-19 Dashboard
PETA-20 Limits
PETA-21 Reports
PETA-22 Reports
PETA-37 Workspace
PETA-38 Charts
PETA-39 Charts
PETA-40 Connecting to IBM DB2
PETA-41 Frontend
PETA-42 Watson Assistant
PETA-43 SendGrid
PETA-44 SendGrid
PETA-45 Docker
PETA-46 Docker
PETA-47 Cloud Registry
PETA-48 Cloud Registry
PETA-49 Kubernets
PETA-50 IP Ports
PETA-51 IP Ports

7. CODING & SOLUTIONING

app.py:

```
# -*- coding: utf-8 -*-
"""
Spyder Editor
This is a temporary script file.
"""
from flask import Flask, render_template, request, redirect, session
# from flask_mysqldb import MySQL
# import MySQLdb.cursors import re
from flask_db2 import DB2
import ibm_db import
ibm_db_dbi
from sendemail import sendgridmail,sendmail
# from gevent.pywsgi import WSGIServer import
os
app = Flask( name )
app.secret_key = 'a'
# app.config['MYSQL_HOST'] = 'remotemysql.com'
# app.config['MYSQL_USER'] = 'D2DxDUPBii'
# app.config['MYSQL_PASSWORD'] = 'r8XBO4GsMz'

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

"""

dsn_hostname = "3883e7e4-18f5-4afe-be8cfa31c41761d2.

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 = (
"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-be8cfa31c41761d2.

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-be8cfa31c41761d2.

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"])
```

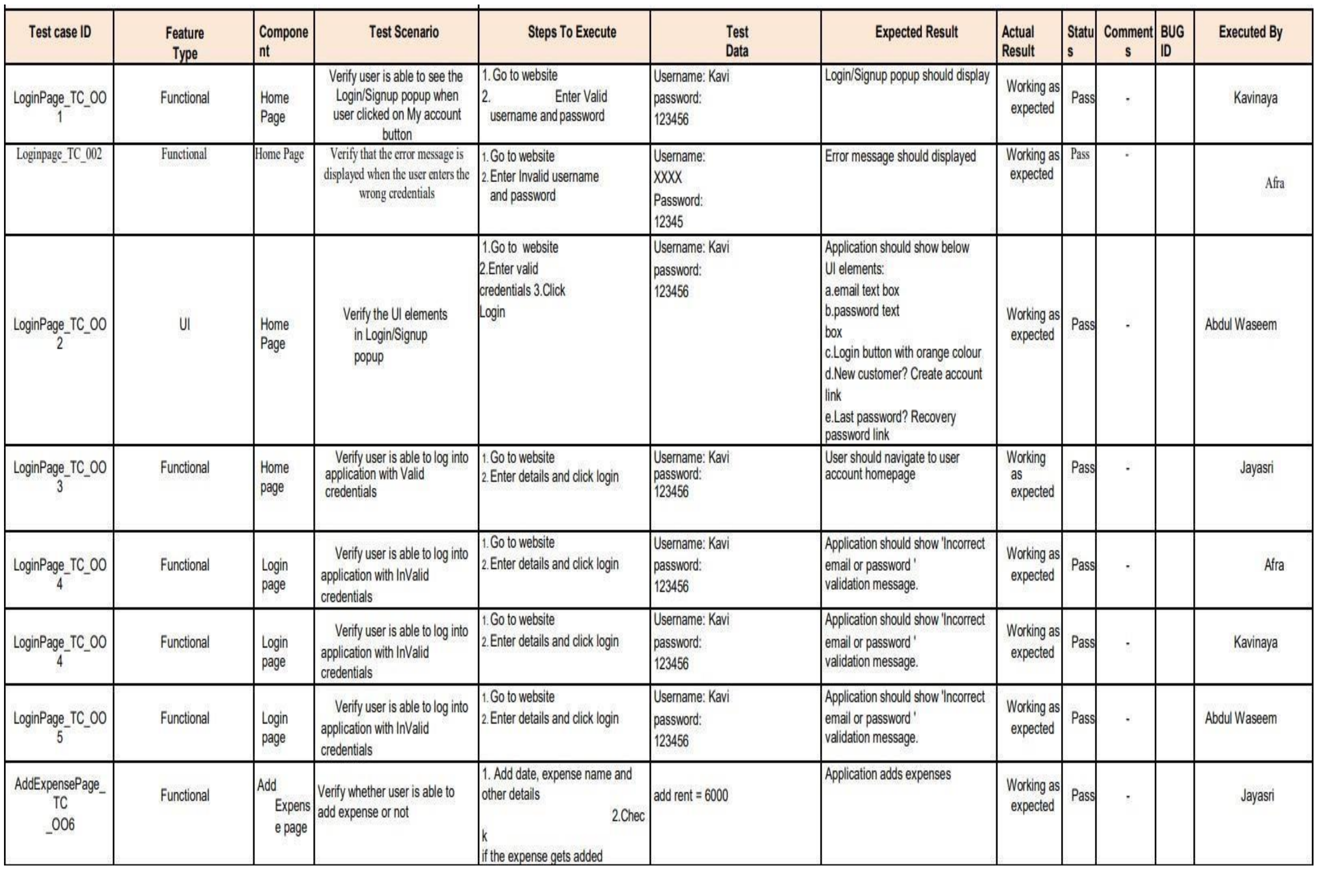
```
temp.append(dictionary["USERID"])


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)
```

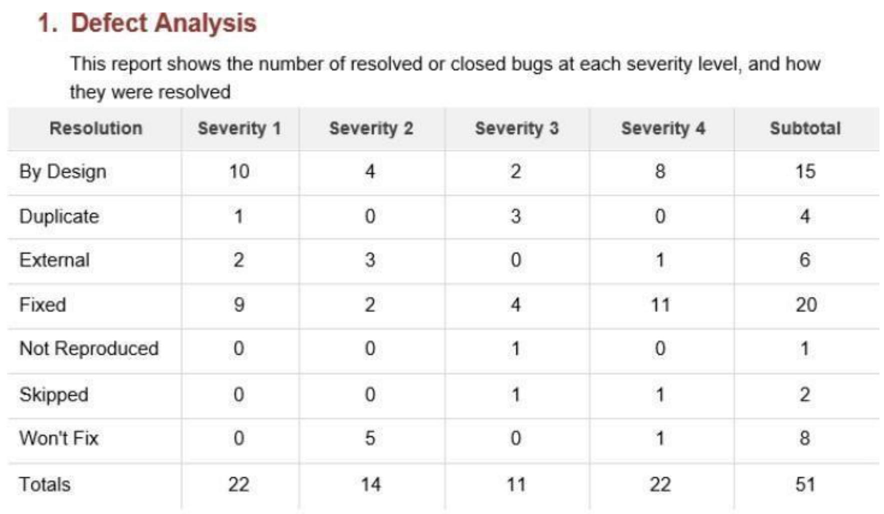
8. TESTING:

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_OO 1	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_OO 2	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_OO 3	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_OO 4	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_OO 4	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_OO 5	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 _OO6	Functional	Add Expense page	Verify whether user is able to add expense or not	1. Add date, expense name and other details 2.Check if the expense gets added	add rent = 6000	Application adds expenses	Working as expected	Pass	-		Jayasri

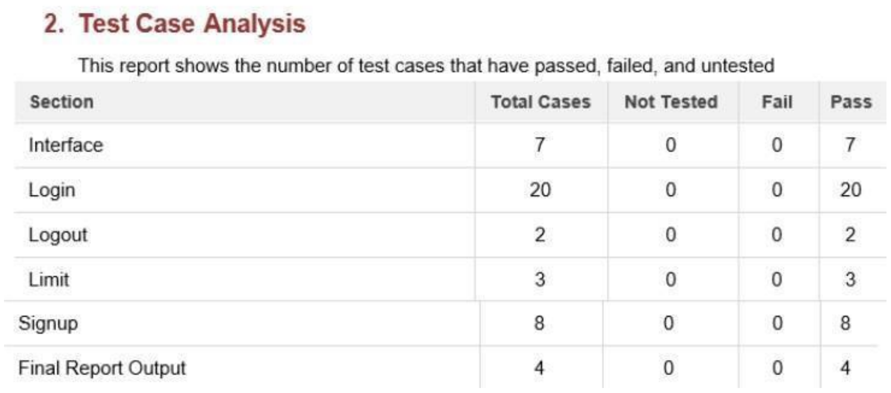
b.User Acceptance Testing



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

9. RESULTS

a. Performance Metrics

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

ii. Transaction Receipts: Capture and organize your payment

Receipts to keep track of your expenditure.

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

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

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

vi. Ecommerce integration: Integrateyour expense trackingapp wit h your eCommerce store and track your sales through payments received via multiple payment methods.

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

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

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

x. Inventory tracking: An expense tracking app can do it all.

Right from tracking products or the cost of goods, sending alert

notifications when the product is running out of stock or the

product is not selling, to purchase orders.

xi. In-depth insights and analytics: Provides in-built tools to

generate reports with easy-to- understand visuals and graphics

to gain insights about the performance of yourbusiness.

xii. Recurrent Expenses: Rely on your budgeting app to track,

streamline, and automate all the recurrent expenses and remind

you on a timely basis.

10. ADVANTAGES & DISADVANTAGES

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

10. Optimize sales processes to generate more revenue through enhanced data collection.

11. CONCLUSION

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

12. FUTURE

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

1. The application is unable to maintain the backup of data once it is uninstalled.

2. This application does not provide higher decision capability.

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

3. Multiple language interface.

4. Provide backup and recovery of data.

5. Provide better user interface for user.

6. Mobile apps advantage.