PNT2022TMID09631

PERSONAL EXPENSE TRACKER

APPLICATION IBM-Project-25691-1659970868

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

A PROJECT REPORT

BY

Kavinaya N(310619205046)

Afra Thahseen J(310619205004)

Abdul Waseem Nihaal KW(310619205002)

Jayasri PS(310619205041)

BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY

EASWARI ENGINEERING COLLEGE, CHENNAI - 600089

INDEX

1. **INTRODUCTION**
    a. Project Overview
    b. Purpose
2. **LITERATURE SURVEY**
    a. Existing problem
    b. References
    c. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
    a. Empathy Map Canvas
    b. Ideation & Brainstorming
    c. Proposed Solution
    d. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
    a. Functional requirement
    b. Non-Functional requirements
5. **PROJECT DESIGN**
    a. Data Flow Diagrams
    b. Solution & Technical Architecture
    c. User Stories
6. **PROJECT PLANNING & SCHEDULING**
    a. Sprint Planning & Estimation
    b. Sprint Delivery Schedule
    c. Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
    a. Feature 1

b. Feature 2

c. Database Schema (if Applicable)

8. **TESTING**

a. Test Cases

b. User Acceptance Testing

9. **RESULTS**

a. Performance Metrics

10. **ADVANTAGES & DISADVANTAGES**

11. **CONCLUSION**

12. **FUTURE SCOPE**

13. **APPENDIX**

Source Code

GitHub & Project Demo Link

1. INTRODUCTION

TEAM ID : PNT2022TMID09631

INDUSTRY MENTOR : Kusboo

FACULTY MENTOR : Dr.K.Johny Elma

Skills Required:

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

a) Project Overview

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

b) Purpose

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

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

2. LITERATURE SURVEY

a. Existing problem

In a study conducted by Forrestrerin 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:

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

b. References

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

S.No	TITLE	PROPOSED WORK	TOOLS USED/ ALGORITHM	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.

c. Problem Statement Definition :

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

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

3. IDEATION & PROPOSED SOLUTION

a) Empathy Map Canvas

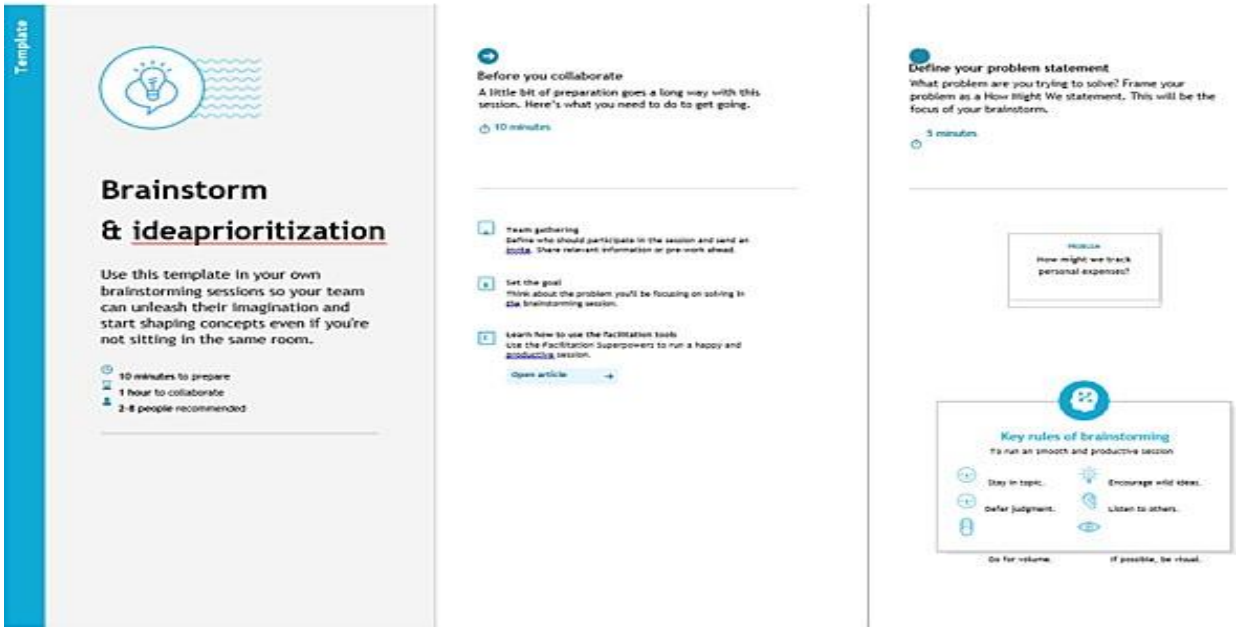
An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

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

b) Ideation & Brainstorming

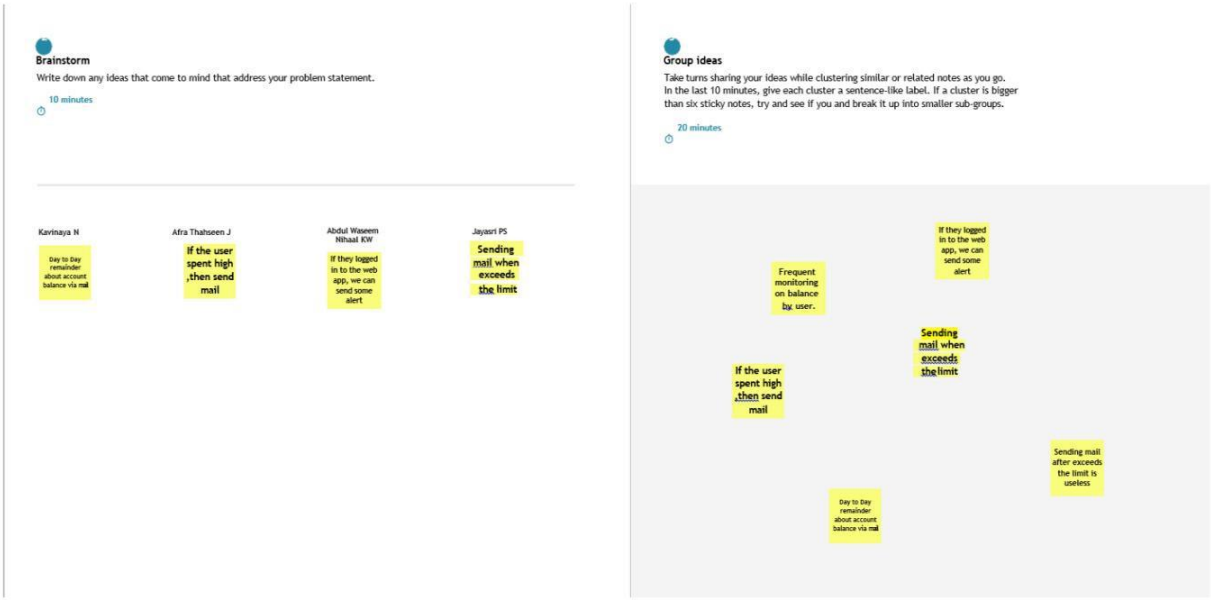
Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate,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

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

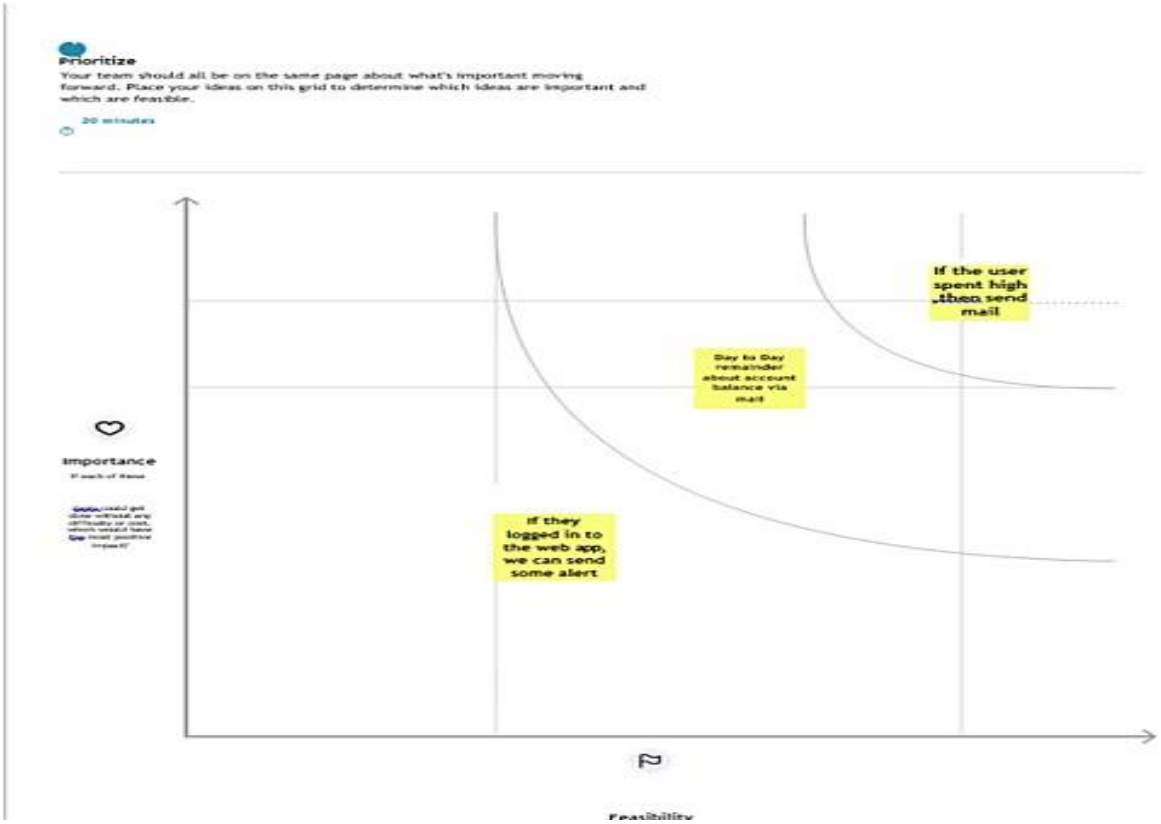


PNT2022TMID09631

Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization

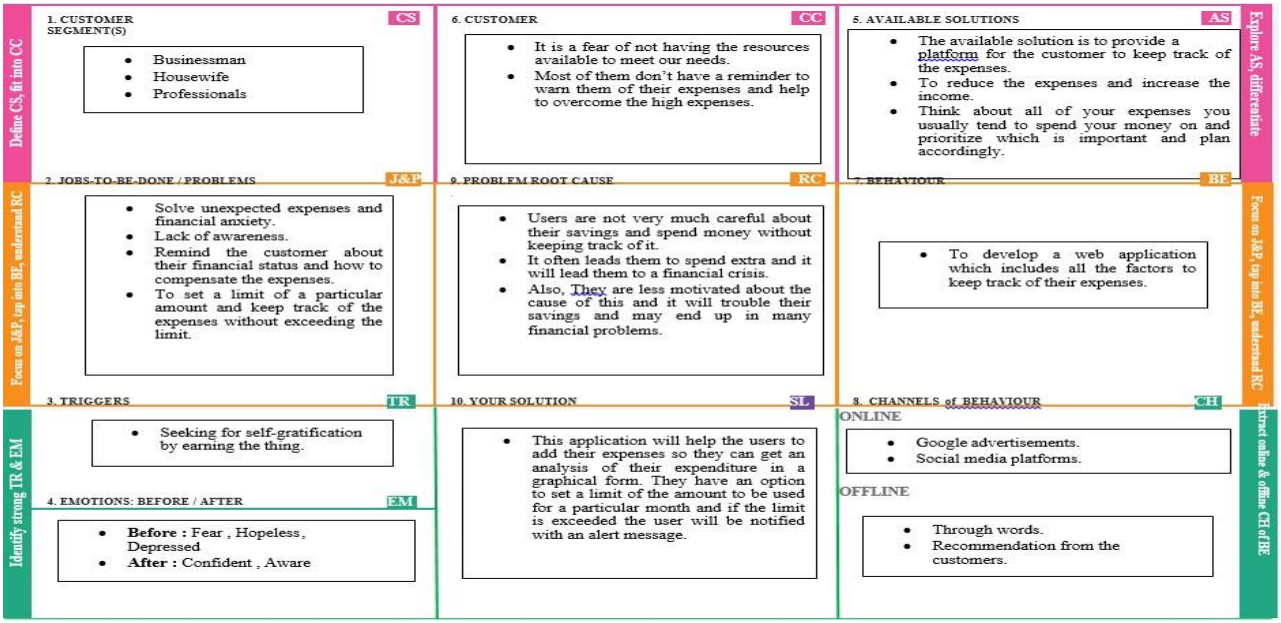


c) Proposed Solution

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

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

d) Problem Solution Fit



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 approximatenotion 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 alertif the threshod amount they selected for an aert is exceeded

b) Non-Functional requirements

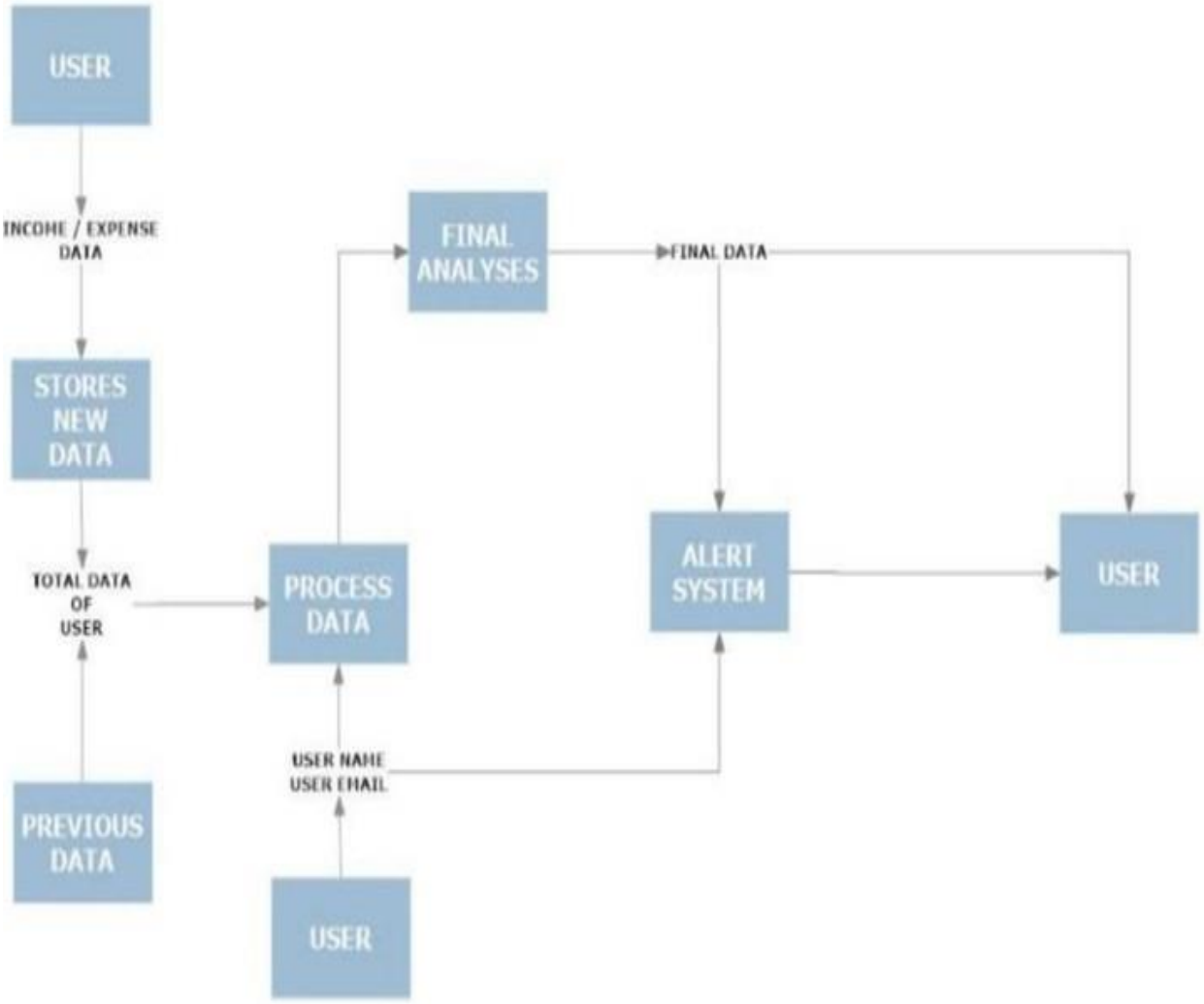
NFR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The system shall allow the users to access the system with pc using web application. The system uses a web application as an interface. The system is user friendly which makes the system easy.
NFR-2	**Security**	A security requirement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

NFR-4	**Performance**	The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 seconds to appear on the screen.
NFR-5	**Availability**	The system is available 100% for the user and isused 24 hrs a day and 365 days a year. The system shallbe operational 24 hours a day and 7 days a week.
NFR-6	**Scalability**	Scalability is the measure of a system's ability toincrease or decrease in performance and cost in response to changes in application and system processing demands.

5. PROJECT DESIGN

a) Data Flow Diagrams

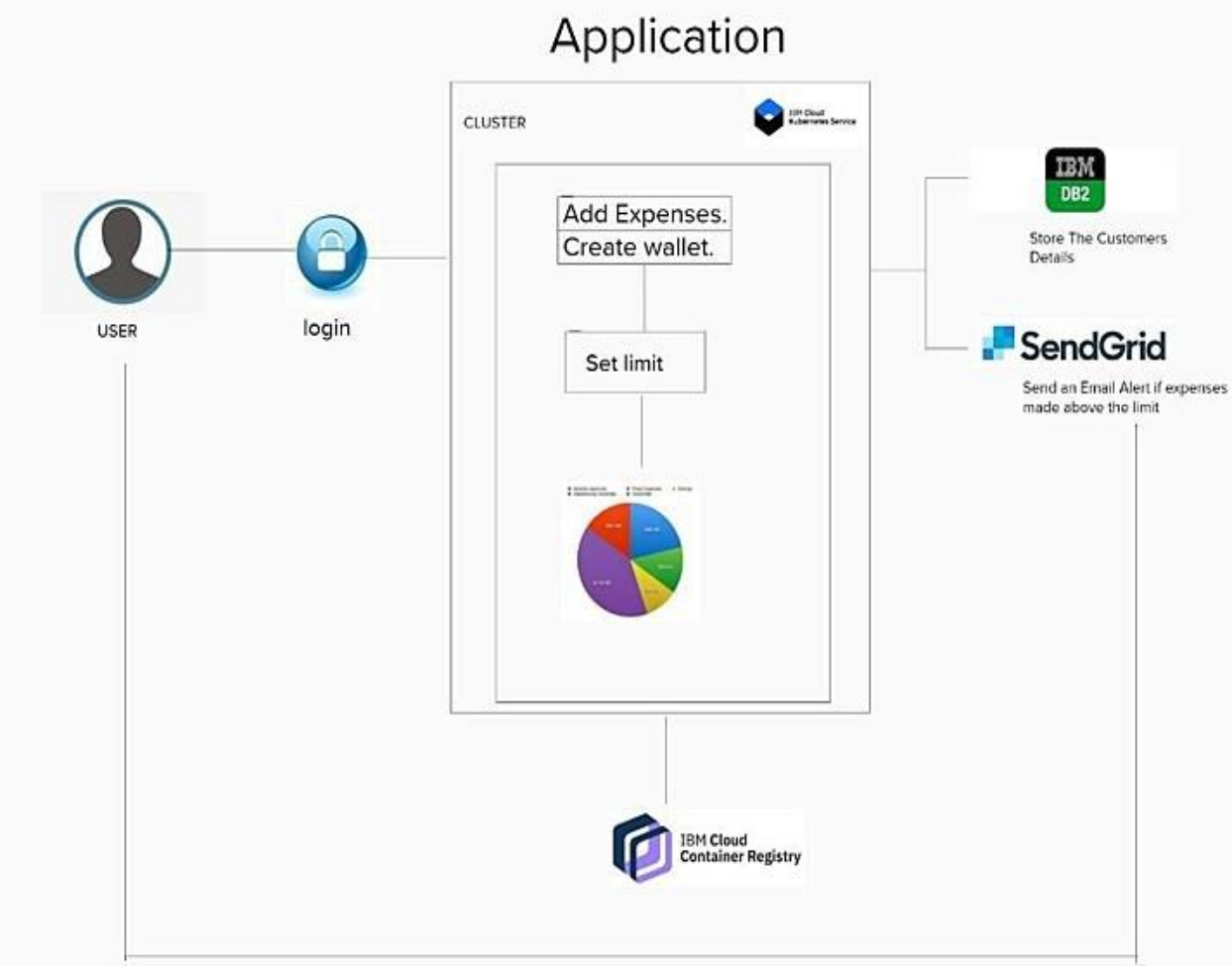
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



b) Solution & Technical Architecture

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

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



c) User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority
Customer (web user)	Registration	USN-1	As a user, I can register for the application by entering mail id and password	I can access my account/ dashboard	High
		USN-2	As a user,I will receive a confirmation email once I have registered for the email and click application	I can receive a confirmation email	High
		USN-3	As a user, I can access using mail	I can register through mail	Low

	Login	USN-4	As a user, I can login application by entering application using email and password	I can access the application	High
	Dashboard	USN-5	As a user,I can view my income and expenditure details	I can view my daily expenses	High
Customer care executive		USN-6	As a customer care executive, I can solve the login issue and other issues of the solution at any application	I can provide support	Medium
Administrator	Application	USN-7	As an administrator,I can upgrade or update the application	I can fix the bug	Medium

6. PROJECT PLANNING & SCHEDULING

a) Sprint planning and estimation

Sprint planning and Estimation is done by the entire team during Sprint Planning Meeting. The objective of the Estimation would be to consider the User Stories for the Sprint by Priority and by the Ability of the team to deliver during the Time Box of the Sprint.

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Jayasri
		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Afra Thahseen
	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Abdul Waseem Nihaal
	Dashboard	USN-4	Logging in takes to the dashboard for the logged user.	`2	High	Kavinaya
*Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only*						
Sprint 2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Afra Thahseen
	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Abdul Waseem Nihaal
	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Kavinaya
		USN-4	Making dashboard interactive with JS	2	High	Jayasri
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Abdul Waseem Nihaal
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user's query	1	Medium	Kavinaya
	SendGrid	USN-3	Using SendGrid to send mail to the user about their expenses	1	Low	Jayasri
		USN-4	Integrating both frontend and backend	2		Afra Thahseen
*Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only*						
Sprint-4	Docker	USN-1	Creating image of website using docker/	2	High	Kavinaya
	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Jayasri
	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Afra Thahseen
	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Abdul Waseem Nihaal

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

b) Sprint Delivery Schedule

Sprint delivery schedule is a schedule prepare with timelines Within which a particular task should be completed.

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
		Literature Survey	29 Aug 2022 – 03 Sept 2022

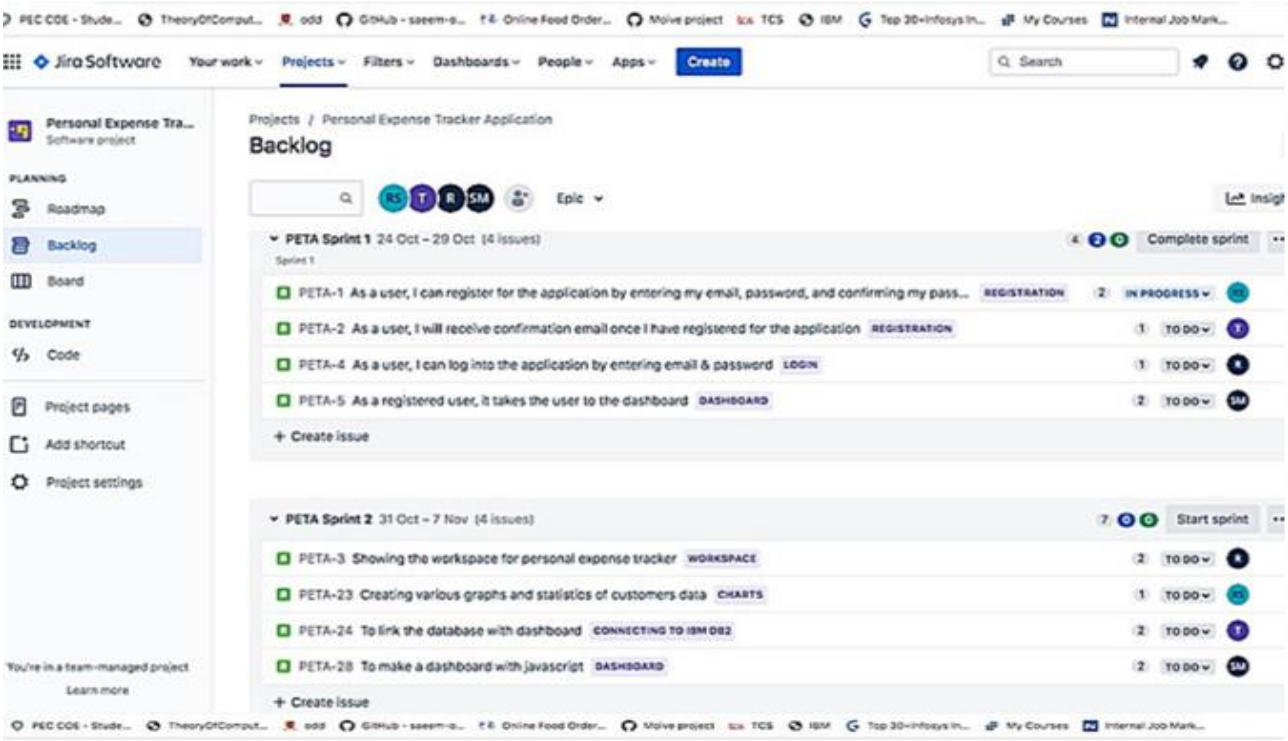
2.	**Ideation Phase**	Empathy Map	5 Sept 2022 - 7 Sept 2022
		Problem Statement	8 Sept 2022 - 10 Sept 2022
		Ideation	12 Sept 2022 – 16 Sept 2022
3.	**Project Design Phase - 1**	Proposed Solution	19 Sept 2022 – 23 Sept 2022
		Problem Solution Fit	24 Sept 2022 – 26 Sept 2022
		Solution Architecture	27 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 –

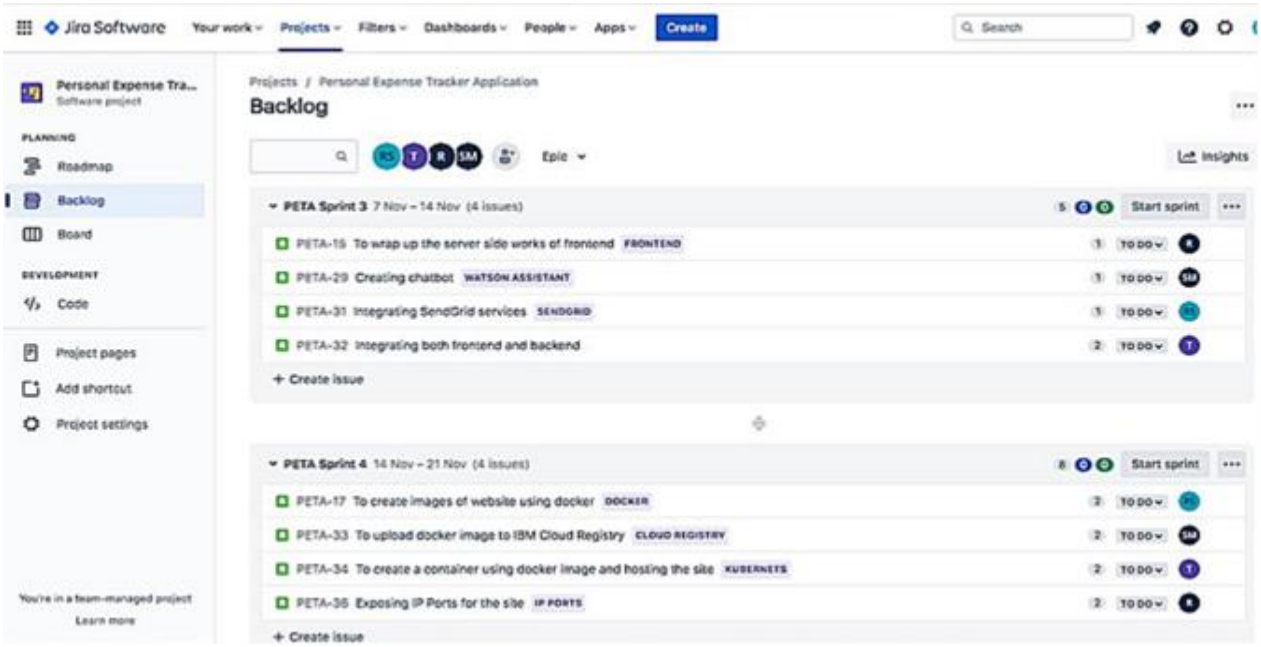
			29 Oct 2022
		Sprint – 2	31 Oct 2022 – 05 Nov 2022
		Sprint – 3	07 Nov 2022 – 12 Nov 2022
		Sprint – 4	14 Nov 2022 – 19 Nov 2022

c) Reports from JIRA

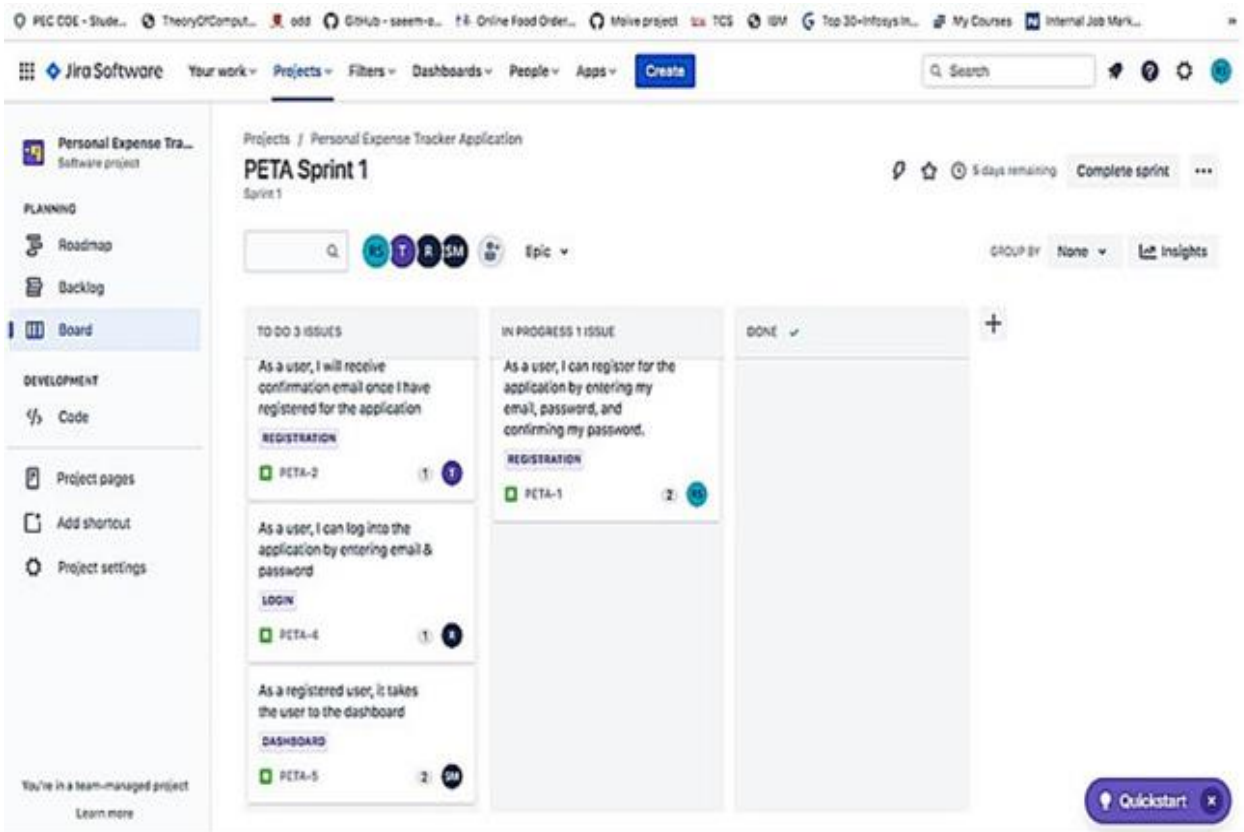
Jira is a software project management tool that helps teams plan, assign track, report, and manage work and brings teams together for everything from agile software development and customer support to start-ups and enterprises. Software teams build better with Jira Software, the #1 tool for agile teams.

i. Backlog




Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Personal Expense Tra...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / Personal Expense Tracker Application
Backlog
Epic
Insights
PETA Sprint 3 7 Nov – 14 Nov (4 issues)
Start sprint
PETA-15 To wrap up the server side works of frontend FRONTEND
PETA-29 Creating chatbot WATSON ASSISTANT
PETA-31 Integrating SendGrid services SENDGRID
PETA-32 Integrating both frontend and backend
Create issue
PETA Sprint 4 14 Nov – 21 Nov (4 issues)
Start sprint
PETA-17 To create images of website using docker DOCKER
PETA-33 To upload docker image to IBM Cloud Registry CLOUD REGISTRY
PETA-34 To create a container using docker image and hosting the site KUBERNETS
PETA-36 Exposing IP Ports for the site IP PORTS
Create issue

ii. Board


Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Personal Expense Tra...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / Personal Expense Tracker Application
PETA Sprint 1
Sprint 1
5 days remaining
Complete sprint
Epic
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
As a user, I can log into the application by entering email & password
LOGIN
PETA-4
As a registered user, it takes the user to the dashboard
DASHBOARD
PETA-5
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
DONE
Quickstart

iii. Road Map


T
NOV
DEC
JAN '23
Sprints
PETA...
PETA...
PETA...
PETA Spr...
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

i. Python App:

app.py:

```
# -*- coding: utf-8 -*-
"""
Spyder Editor
This is a temporary script file.
"""
from flask import Flask, render_template, request, redirect, session
# from flask_mysqldb import MySQL
# import MySQLdb.cursors
import re
from flask_db2 import DB2
import ibm_db
import ibm_db_dbi
from sendemail import sendgridmail,sendmail
# from gevent.pywsgi import WSGIServer
import os
app = Flask(__name__)
app.secret_key = 'a'
# app.config['MYSQL_HOST'] = 'remotemysql.com'
# app.config['MYSQL_USER'] = 'D2DxDUPBii'
# app.config['MYSQL_PASSWORD'] = 'r8XBO4GsMz'
# app.config['MYSQL_DB'] = 'D2DxDUPBii'
"""
dsn_hostname = "3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"
dsn_uid = "sbb93800"
```

```
dsn_pwd = "wobsVLm6ccFxcNLe"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "31498"
dsn_protocol = "tcpip"
dsn = (
   "DRIVER={0};"
   "DATABASE={1};"
   "HOSTNAME={2};"
   "PORT={3};"
   "PROTOCOL={4};"
   "UID={5};"
   "PWD={6};"
).format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol, dsn_uid, dsn_pwd)
"""
# app.config['DB2_DRIVER'] = '{IBM DB2 ODBC DRIVER}'
app.config['database'] = 'bludb'
app.config['hostname'] = '3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'
app.config['port'] = '31498'
app.config['protocol'] = 'tcpip'
app.config['uid'] = 'sbb93800'
app.config['pwd'] = 'wobsVLm6ccFxcNLe'
app.config['security'] = 'SSL'
try:
   mysql = DB2(app)
   conn_str='database=bludb;hostname=3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=31498;protocol=tcpip;\
         uid=sbb93800;pwd=wobsVLm6ccFxcNLe;security=SSL'
```

```
    ibm_db_conn = ibm_db.connect(conn_str,'','')
    print("Database connected without any error !!")
except:
    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'])))
    #   texpense = cursor.fetchall()
    #   print(texpense)
      param1 = "SELECT TIME(date) as tn, amount FROM expenses WHERE userid = " + str(session['id']) + " AND DATE(date) = DATE(current timestamp) ORDER BY date DESC"
```

```
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpense = []
    while dictionary1 != False:
        temp = []
        temp.append(dictionary1["TN"])
        temp.append(dictionary1["AMOUNT"])
        texpense.append(temp)
        print(temp)
        dictionary1 = ibm_db.fetch_assoc(res1)
  #  cursor = mysql.connection.cursor()
  #  cursor.execute('SELECT * FROM expenses WHERE userid = % s AND DATE(date) = DATE(NOW()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
  #  expense = cursor.fetchall()
    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND DATE(date) = DATE(current timestamp) ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
```

```
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)
    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0
    for x in expense:
        total += x[4]
        if x[6] == "food":
            t_food += x[4]
        elif x[6] == "entertainment":
            t_entertainment  += x[4]
        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]
        elif x[6] == "EMI":
            t_EMI  += x[4]
        elif x[6] == "other":
            t_other  += x[4]
    print(total)
    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
```

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


@app.route("/month")
def month():
  #   cursor = mysql.connection.cursor()
  #   cursor.execute('SELECT DATE(date), SUM(amount) FROM expenses WHERE userid= %s AND MONTH(DATE(date))= MONTH(now()) GROUP BY DATE(date) ORDER BY DATE(date) ',(str(session['id'])))
  #   texpense = cursor.fetchall()
  #   print(texpense)
    param1 = "SELECT DATE(date) as dt, SUM(amount) as tot FROM expenses WHERE userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp) GROUP BY DATE(date) ORDER BY DATE(date)"
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpense = []
    while dictionary1 != False:
      temp = []
      temp.append(dictionary1["DT"])
      temp.append(dictionary1["TOT"])
      texpense.append(temp)
      print(temp)
      dictionary1 = ibm_db.fetch_assoc(res1)
  #   cursor = mysql.connection.cursor()
  #   cursor.execute('SELECT * FROM expenses WHERE userid = % s AND MONTH(DATE(date))= MONTH(now()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
  #   expense = cursor.fetchall()
```

```
    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)
    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0
    for x in expense:
        total += x[4]
        if x[6] == "food":
            t_food += x[4]
```

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


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

```
    #   print(texpense)

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

      res1 = ibm_db.exec_immediate(ibm_db_conn, param1)

      dictionary1 = ibm_db.fetch_assoc(res1)

      texpense = []

      while dictionary1 != False:

         temp = []

         temp.append(dictionary1["MN"])

         temp.append(dictionary1["TOT"])

         texpense.append(temp)

         print(temp)

         dictionary1 = ibm_db.fetch_assoc(res1)

    #   cursor = mysql.connection.cursor()

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

    #   expense = cursor.fetchall()

      param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " 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", 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: v1
kind: Service
metadata:
  name: flask-app-service
spec:
  selector:
    app: flask-app
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 5000
  type: LoadBalancer
```

manifest.yml:

```
applications:
 - name: Python Flask App IBCMR 2022-10-19
   random-route: true
   memory: 512M
   disk_quota: 1.5G
```

sendemail.py:

```
import smtplib
import sendgrid as sg
import os
from sendgrid.helpers.mail import Mail, Email, To, Content
SUBJECT = "expense tracker"
s = smtplib.SMTP('smtp.gmail.com', 587)

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

def sendgridmail(user,TEXT):
    # from_email = Email("shridhartp24@gmail.com")
    from_email = Email("tproduct8080@gmail.com")
```

```
to_email = To(user)
subject = "Sending with SendGrid is Fun"
content = Content("text/plain",TEXT)
mail = Mail(from_email, to_email, subject, content)
# Get a JSON-ready representation of the Mail object
mail_json = mail.get()
# Send an HTTP POST request to /mail/send
response = sg.client.mail.send.post(request_body=mail_json)
print(response.status_code)
print(response.headers)
```

iii. 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:

A test case is a set of actions performed on a system to determine if it satisfies software requirements and functions correctly. The purpose of a test case is to determine if different features within a system are performing as expected and to confirm that the system satisfies all related standards, guidelines and customer requirements. The process of writing a test case can also help reveal errors or defects within the system.

a.

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

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done.

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

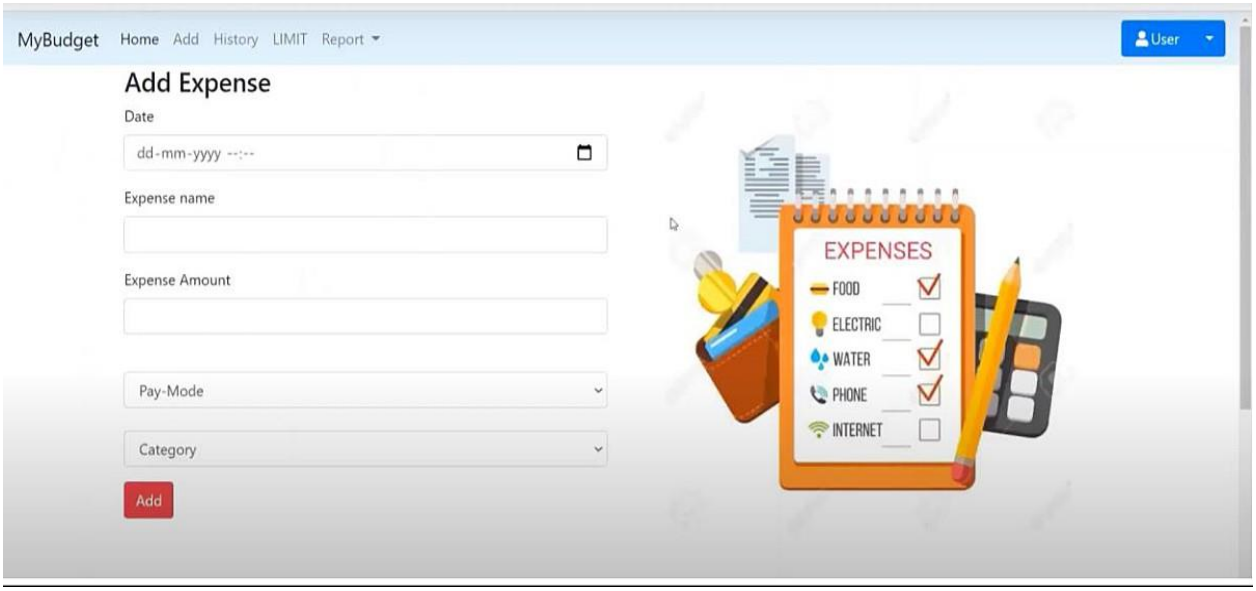
2. Test Case Analysis

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

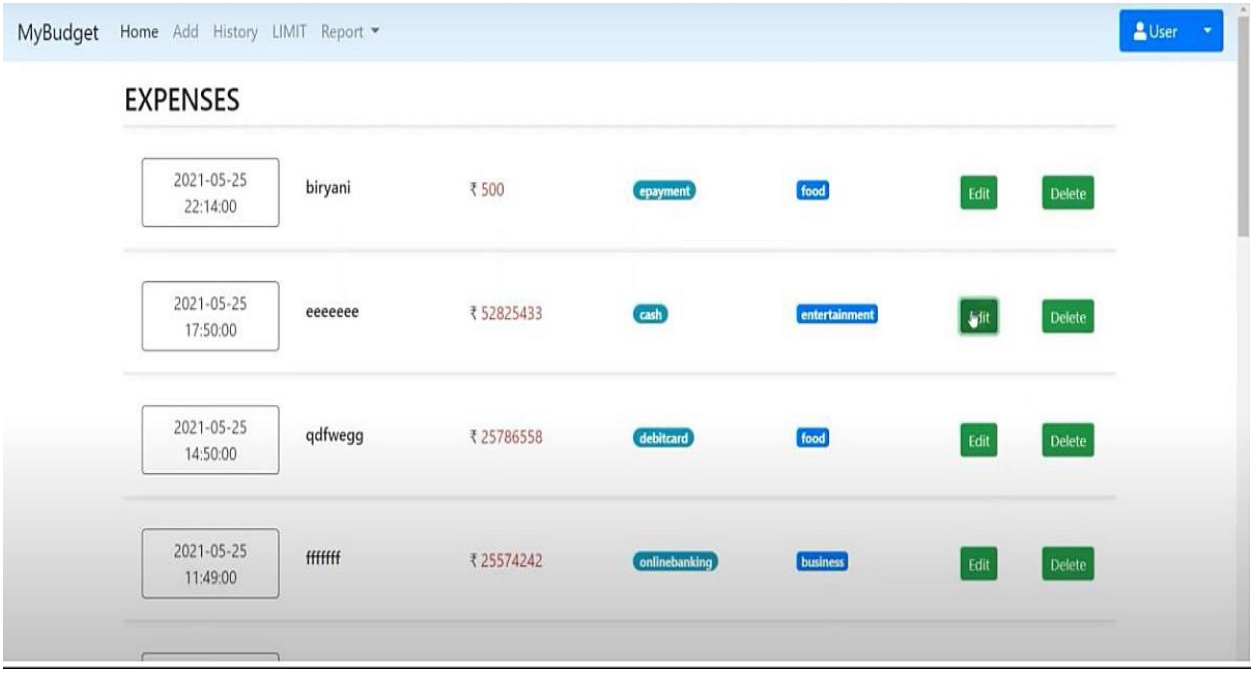
Section	Total Cases	Not Tested	Fail	Pass
Interface	7	0	0	7
Login	20	0	0	20
Logout	2	0	0	2
Limit	3	0	0	3
Signup	8	0	0	8
Final Report Output	4	0	0	4

9. RESULTS

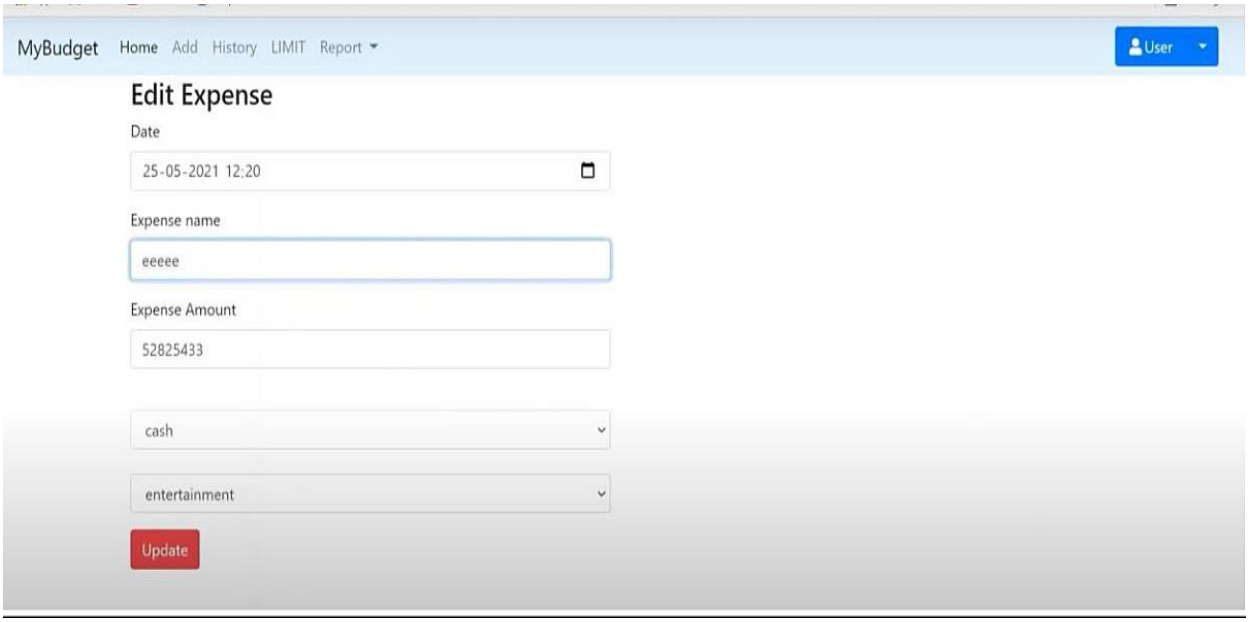
ADDING EXPENSES:



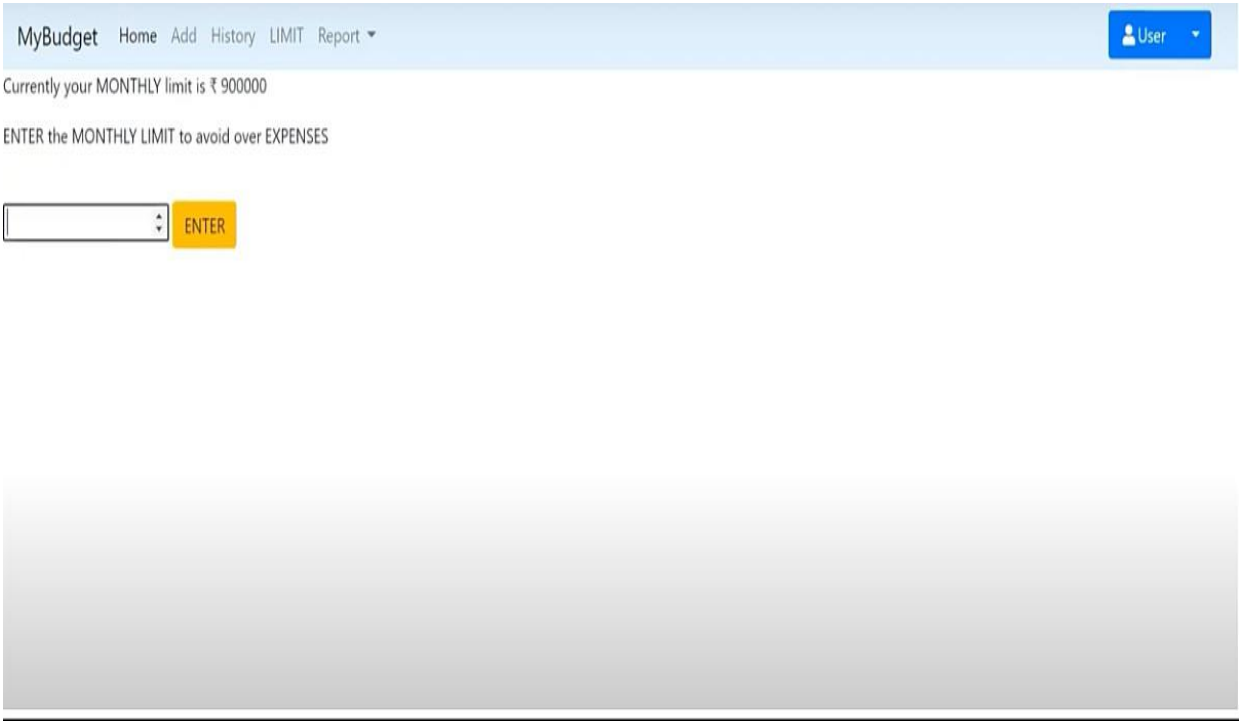
EXPENSE HISTORY:



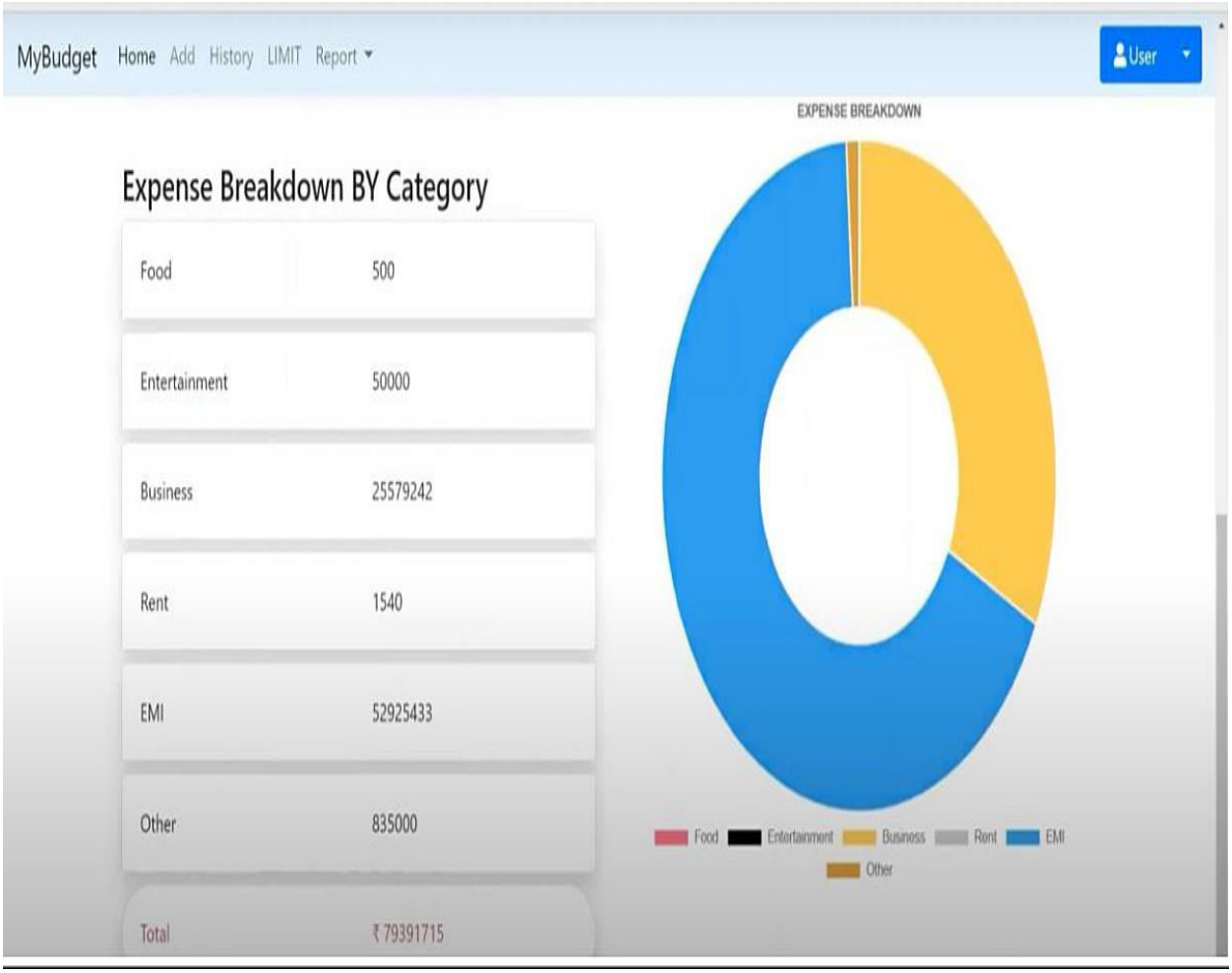
EDITING EXPENSE:



SETTING LIMIT:



EXPENDITURE REPORT - PIECHART



i. 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 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. **E-commerce integration:** Integrateyour expense trackingapp with your eCommerce store and track your sales through payments received via multiple payment methods.

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

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

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

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

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

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

10. ADVANTAGES & DISADVANTAGES

1. **Achieve your business goals** with a tailored mobile app that perfectly fits your business.

2. **Scale-up** at the pace your business is growing.

3. Deliver an **outstanding** customer experience through additional control over the app.

4. Control the **security** of your business and customer data

5. Open **direct marketing channels** with no extra costs

   with methods such as push notifications.

6. **Boost the productivity** of all the processes within 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:

- Multiple language interface.
- Provide backup and recovery of data.
- Provide better user interface for user.
- Mobile apps advantage.

13. APPENDIX

Source Code Github Link : https://github.com/IBM-EPBL/IBM-Project-25691-1659970868

Project Demo Link: https://youtu.be/ee-CZB4IFQA