Team ID: PNT2022TMID34004

PERSONAL EXPENSE TRACKER APPLICATION

IBM-Project-40544-1660631023

NALAIYA THIRAN PROJECT

BASED LEARNING ON

PROFESSIONAL READLINESS FOR INNOVATION,

EMPLOYNMENT AND ENTERPRENEURSHIP

A PROJECT REPORT

BY

Iswarya J (960219104046)

Reyma Jerusha M (9602191083)

Ashika Jebi J P (960219104026)

Anusha M (960219104019)

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE AND ENGINEERING

ARUNACHALA COLLEGE OF ENGINEERING FOR WOMEN

-

VELLICHANTHAI - 629203

PROJECT REPORT FORMAT

1.INTRODUCTION

1.1Project Overview

1.2 Purpose

2.LITERATURE SURVEY

2.1 Existing Problem

2.2 References

2.3 Problem Statement Definition

3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation &Brainstorming

3.3 Proposed Solution

3.4 Problem Solution fit

4.REQUIREMENT ANALYSIS

4.1 Functional Requirement

4.2 Non-Functional Requirements

5.PROJECT DESIGN

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 Users Stories

6.PROJECT PLANNING &SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Report from JIRA

1.INTRODUCTION

In this project we propose an application known as "Income and Expense Tracker" which is helpful to manage out income and expense as an daily or periodically or else whenever we want to remaind. It also act as indicator or remainder example in the fastest world which we can't able to remember what are the things we have to do for the end of month and what are the payments we have to pay for the particular month.Due to some conflict or some other stress we forget some times that what are the income or where the money has to come from or what the payments we have to pay.

This application will help you to make a note for what or the things we have to do for the end of month.For example,like how much it expense for monthly and what are the expenses for a month .Some of the expense features like food expenses billing expenses like phone ,electricity, taxation and some other personal expenses. In income features if we are a businessman having some multi business he doesn't know from which part income has come and how much income it has come for him ,but with the help of this application he can divide and store all the income and he can set a reminder for specific date to remain so that he can manage and finalize the income for erach month . In this fast moving world this application will be very useful for a people who was a family and especially for a business people

1.1PROJECT OVERVIEW

TEAM ID : PNT2022TMID3004

INDUSTRY MENTOR : Kusboo

FACULTY MENTOR : A . Annie Portia

Skills Required:

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

Project expense tracking is how businesses monitor the costs of project.For different initiative,like changing processes, developing new products and creating a new marketing campaign each might have a project manager to oversee the schedules and budgets.With expense tracking,managers can see how much they spend on item.

1.2 Purpose

Ability to monitor costs incrementally : Tracking expenses throughout a project provide you the ability to view various expense categories and time periods . This can help you understand how much money you can spend for the rest of the project while staying within your budget .

Allows for budgeting on future project : If you record your expenses for a project,you can use that information to budget for similar future project. For example, if you license software to complete a project , you can include that budject if you need to use that software for other initiatives.

Improved decision-making: By tracking your expenses , you can make more informed decision that if you didn't know how much you spent. You might spend additional money on resources to keep a project on schedule, but if you knew these expense might make the your project over budject, You could explore other options .

2.Literature Survey

Title: Expense Tracker : A Smart Approach to Track Everyday Expense.

Author Name: H Gupta, AP Singh, N Kumar, JA Blessy.

Year of publishing: 2020.

Description:

Since the beginning of human civilization, people have exchanged their destiny for one another to buy or sell goods. Since then, it has become an important and irreplaceable part of our daily lives. Most of us have a fixed income and we get it on time (i.e., daily, monthly, annual, etc.). In addition, everyone follows a strict budget of spending. Generally, the budget is assembled according to category. Categories vary, for example, food, entertainment, transportation, education, health, clothing, and so on. However, spending is limited to budget revenue. For this reason, we need to keep track of our expenses so that they do not exceed our budget. In the old days, people would track their expenses manually, which meant that using a penand paper system would be very laborious and time consuming. These days the availability of electronic devices like smartphones and computers has made our lives much easier and faster. We can use computers to track your daily expenses using the available online and offline software. There are some apps that can track daily expenses . These apps use a manual input system from the keyboard, which is laborious and time consuming. To meet the challenge of avoiding manual input, we propose the best way to do the same things in an automated and efficient way that takes less time. Under the proposed approach, users can spend, fill and monitor data.

Title: Online Income and Expense Tracker.

Author Name: S. Chandini, T. Poojitha, D. Ranjith, V.J. Mohammed Akram, M.S. Vani.

Year of publishing: 2019.

Description:

Income and Expense Tracker will maintain data of daily, weekly, monthly, yearly expenses, Manages your expenses and earnings in a simple and intuitive way. User can select category of expense, enter other information like user can capture photo, add location, select amount of expense etc. And this will save to the local database.

User can view and sort expense as per weekly, monthly, yearly. By using this, we can reduce the manual calculations for their expenses and keep the track of the expenditure. In this, user can provide his income to calculate his total expenses per day and these results will stored for unique user. People when usually go for trips or movies with friends they can use this traker to maintain their expense. It will be easy for them to share the bill in this tracker.This will display graph as per selected view. And user can enter his monthly income or limit of monthly Expense in this tracker. This tracker system provides an integrated set of features to help you to manage your expenses and cash flow.

Title: Personal Expense Assistant Management: An Android Based Application.

Author Name: Sali, Moussa; Abbo, Abdel Salam.

Year of publishing: 2016.

Description:

Personal Expense Assistant Management is an application aiming to manage our daily expenses in a more efficient and manageable way. The application attempts to free the user with as much as possible the burden of manual calculation and to keep the track of his expenditure. Instead of keeping a dairy or a log of the expenses on the smartphones or laptops, it enables the user to not just keep the tab on the expenses but also to plan ahead keeping the past budget in mind. With the help of this application, a user may be able to add, delete or change the current entered bill entry efficiently. The graphical representation of the budget is the lucrative part of the system as it appeals the user more and is easy to understand and incorporate for future planning. The user interface of the system ticks the boxes of consistency, easy readable dialogue boxes, easy exit and easy to get used to requirements for any ideal user interface.

2.1 Existing Problem

If we want to balance a income and expense for each month we have to do it manually but we can't do this for each and every month those who have a lot of income and expenses, so to reduce the stress for the person and make easy to calculate the income and expense, this android application has been so much helpful for a per-son to avoid the manual way calculating his income and expenses. In this application we have features like add expenses categories add income so that we can add what are the income and expenses has been done for a month, but this application will not have any reminder to remain a person in a specific date, so that is the only drawback in which the remainder is not present. This application will be a unpopulated data because it has some disadvantages by not alerting a person for each and every month. But it can used to perform calculation on income and expenses to overcome this problem we propose the new application.

2.2 Reference

1.Accreditation and Quality Assurance Committee (AQAC) in Palestine General

2. Report of Information Technology and Engineering Higher Education in. Palestine. Accreditation and Quality Assurance Commission (AQAC). Ramallah, Palestine: Palestinian Ministry of Education and Higher Education; 2007 Apr.2. Engineering Association of Palestine. Current Engineering Statistics Book. Ramallah; 2005.

3. Prados J, Peterson G, Lattuca L. Quality Assurance of Engineering Education Through Accreditation: The Impact Of Engineering Criteria 2000 and Its Global Influence. Journal of Engineering Education. 2005 Jan; 94(1):165–84.

4. Chen JW, Yen M. Engineering Accreditation: A Foundation For Continuing Quality Improvement. 2005 Mar 1–5; Tainan. Exploring Innovation In Education and Research,

5. Homma H. Accreditation System in Indonesia. JSME news. 2004 May; 15(1)

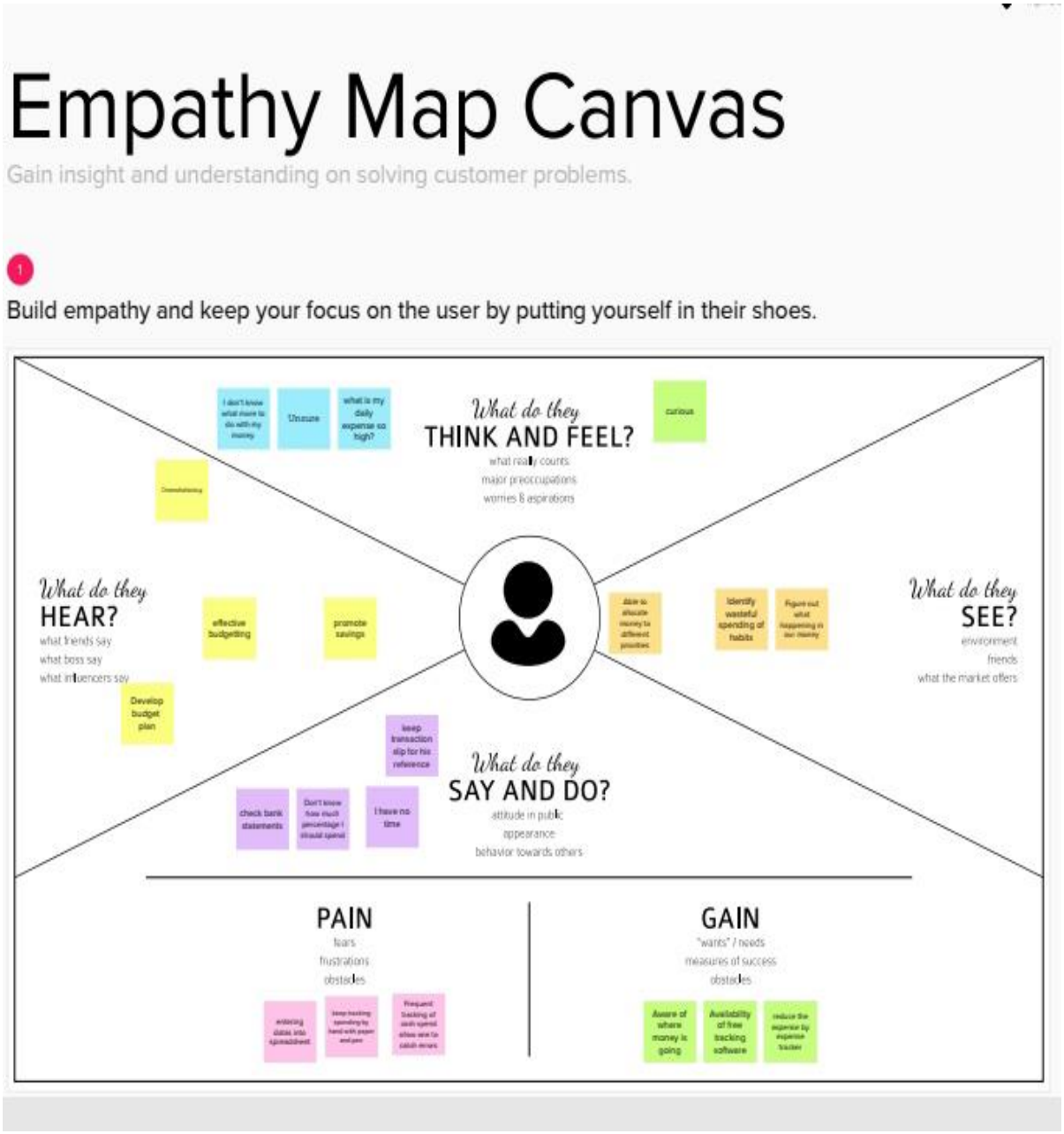
2.3 Problem Statement Definition

Customer Problem Statement Template:

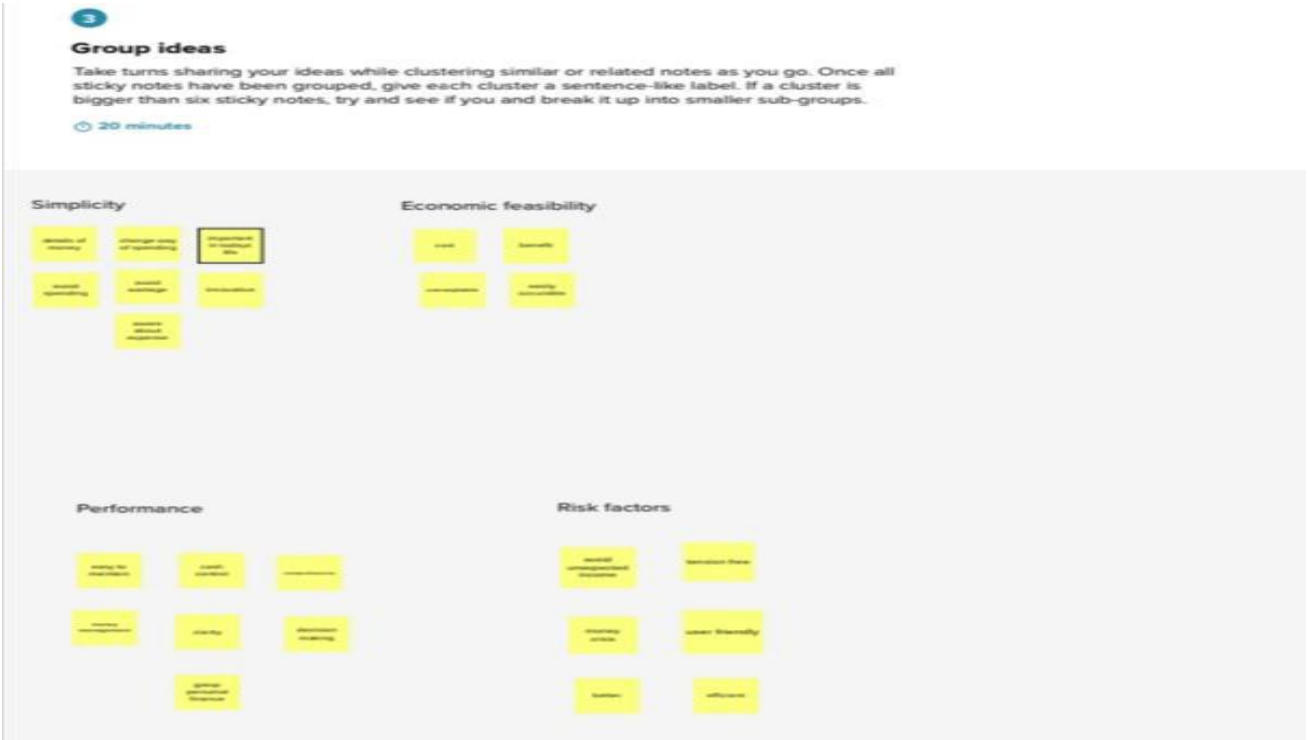
Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A person who wants to manage finances	Develop budget plan	I have no time	Keeping track of money spent in paper makes it a tedious process	Unsure about my monthly expenses
PS-2	A person who wants to manage finances	Promote savings	Don't know how much percentage I can save	Entering data's into spreadsheet is time consuming	Unsure about what to do with my money

3.Ideation and Proposed Solution

3.1 Empathy Map Canva



3.2 Ideation & Brainstorming



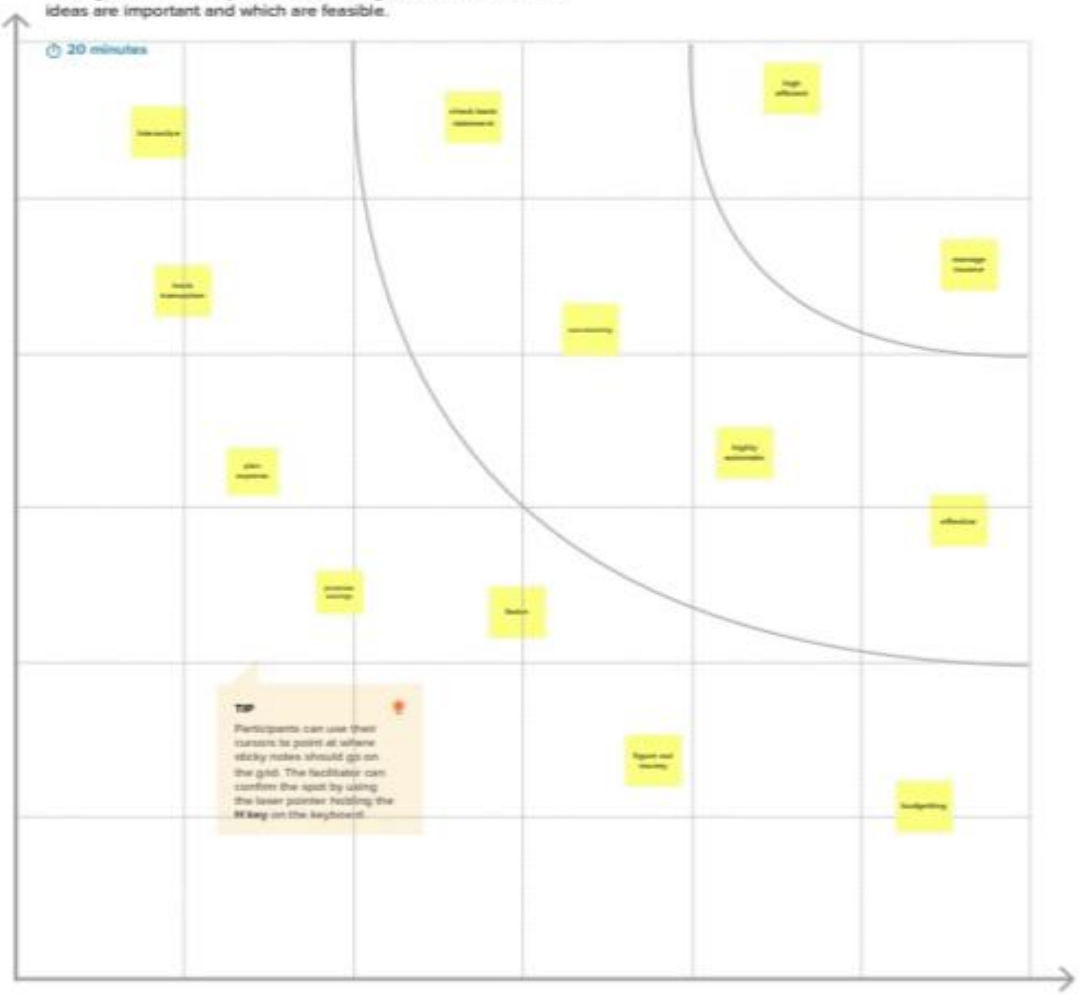
customizable tags to sticky
; make it easier to find,
organize, and
ize important ideas as
within your mural.

4

Prioritize

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

20 minutes



Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **M key** on the keyboard.

Feasibility

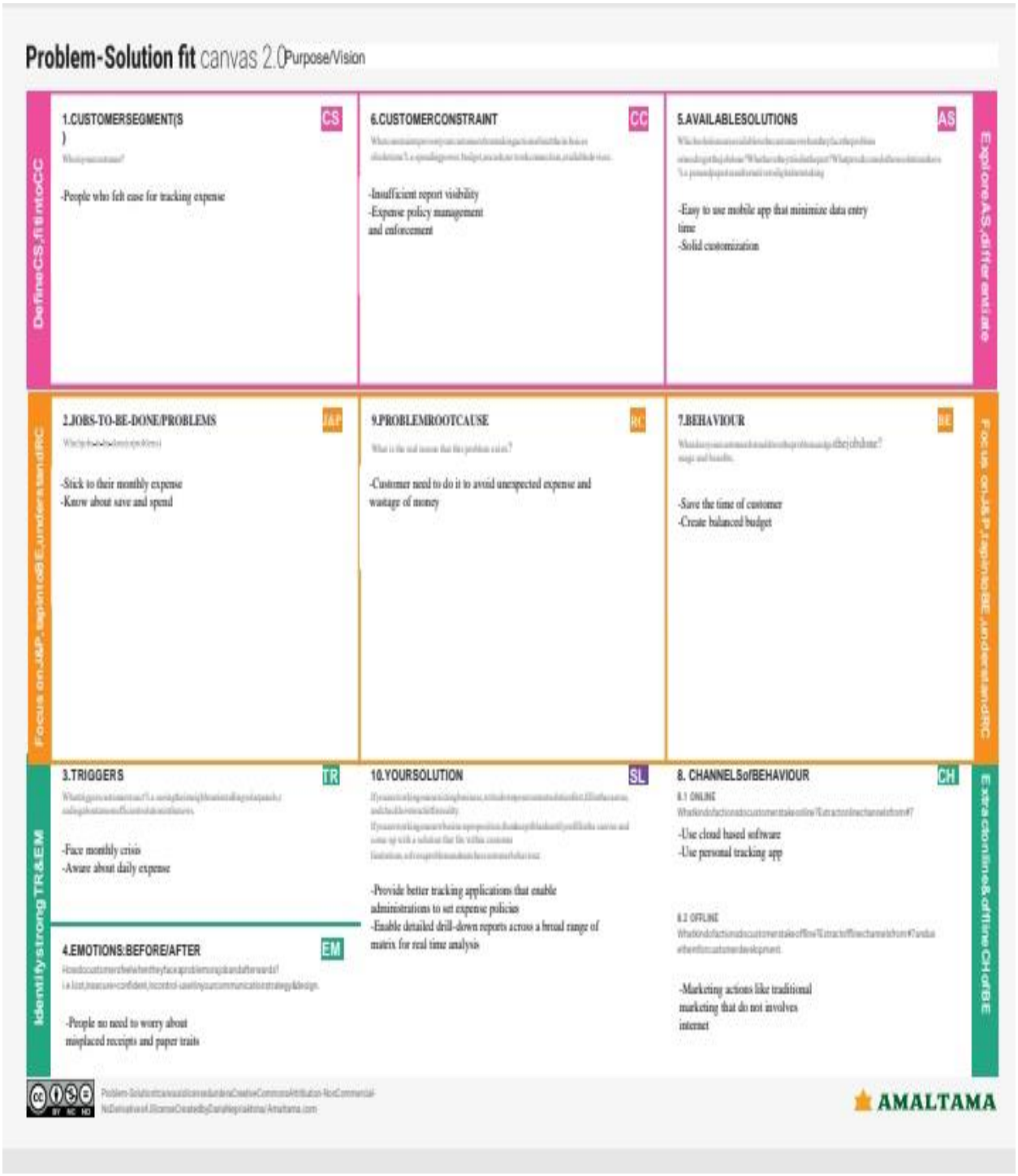
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

3.3 Proposed Solution

Proposed Solution Template

Parameters	Description
Problem Statement	This project is based on an expense and income tracking system.Its aim is to create an easy ,faster and tracking system between the expense and the income.This way experience tracked help people to track income expense day to day and making life tension free.
Solution Description	Personal Expense Tracker (PET) is a daily Expense Management System .This software can be really used by end user who have Android running devices with them .
Uniqueness	We found various similar product that have already been developed in the market. Unlike all those products Personal Expense Tracker provides security and graphical results.
Customer Satisfaction	Provides security and graphical results .We provide the users to enter their wish-list before any purchase .It generates notifications to notify user about their timely entry.
Scalability Of Solution	To further enhance the capability of their application, we recommend these features * Nepali language interface * Provides backup and recovery of data * Provide better user interface for user.
Financial Benefits	PET is easiest and most user friendly personal finance android application. The system attempts to free the user with as much as possible the burden of manual calculation and to keep the track of the expenditure.

3.4 Problem Solution – fit



4.Requirement Analysis

4.1 Functional Requirements

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	User Confirmation	Confirmation via Email
FR-3	Tracking Expense	Helpful insights about money management
FR-4	Alert Message	Give alert mail if the amount exceeds the budget limit

4.2 Non-Functional Requirements

Non-functional Requirements:

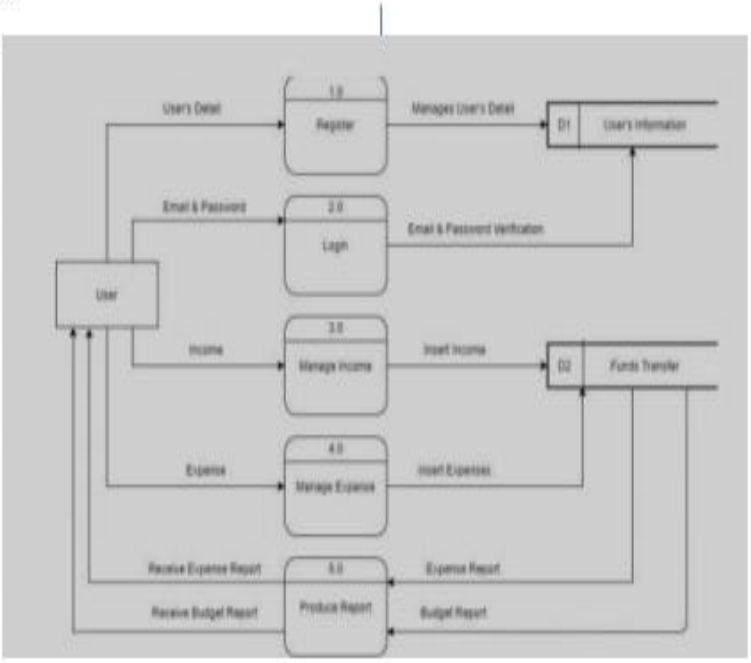
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	You will able to allocate money to different priorities and also help you to cut down on unnecessary spending
NFR-2	**Security**	It employs the latest security and technology measures to keep customers personal and financial information safe
NFR-3	**Reliability**	• Used to manage his/her expense so that the user is the path of financial stability. • It is categorized by week, month, and year and also helps to see more expenses made. • Helps to define their own categories.
NFR-4	**Performance**	Help to gain control of your finance, pay down debt, grow your net worth, help to upload receipts, track mileage
NFR-5	**Availability**	Able to track business expense and monitor important for maintaining healthy cash flow but also qualifying for deductions that could reduce your taxable income
NFR-6	**Scalability**	To know where money goes and you can ensure that money is used widely

5. Project Design

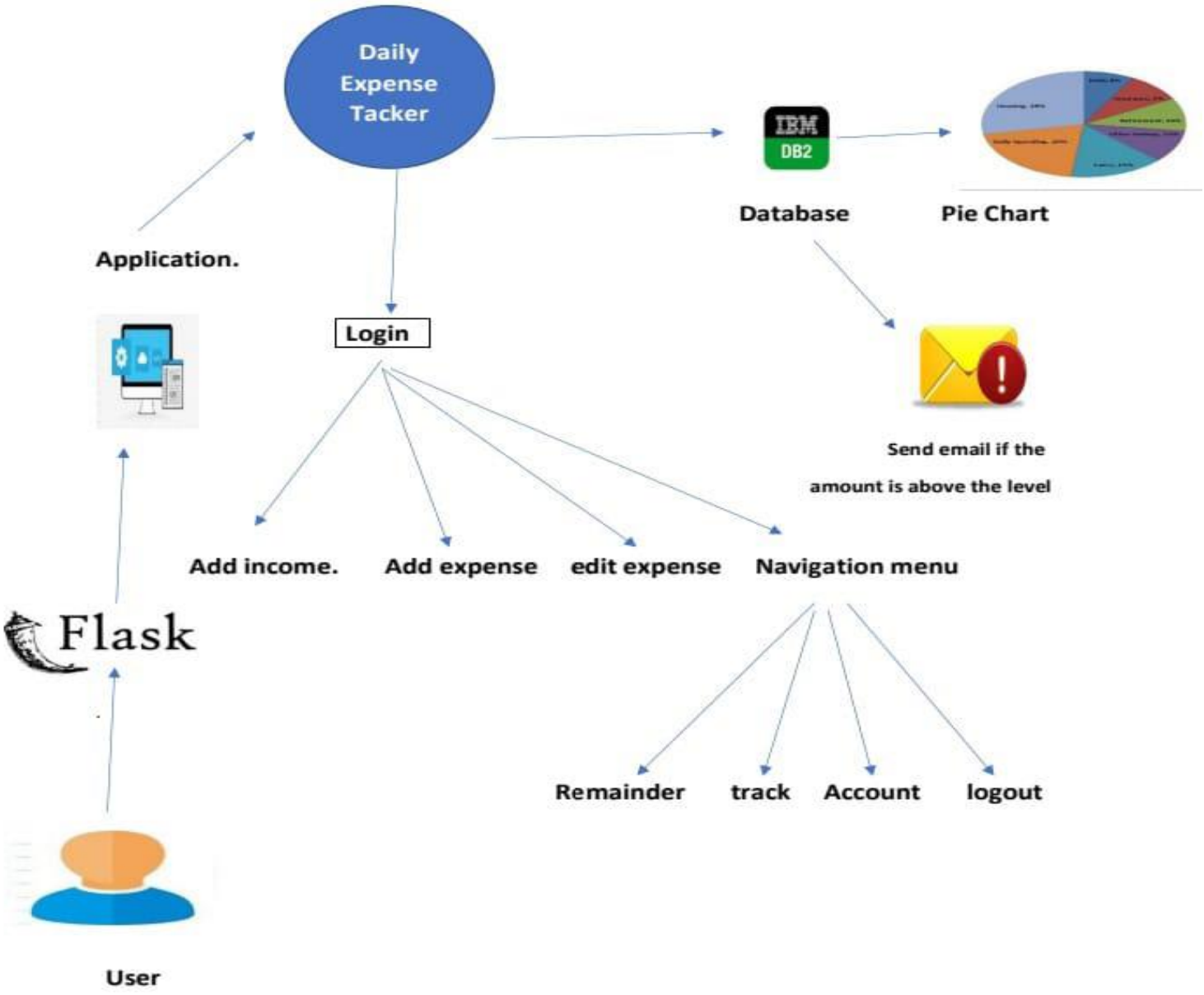
5.1 Data Flow Diagram

Data Flow diagram:

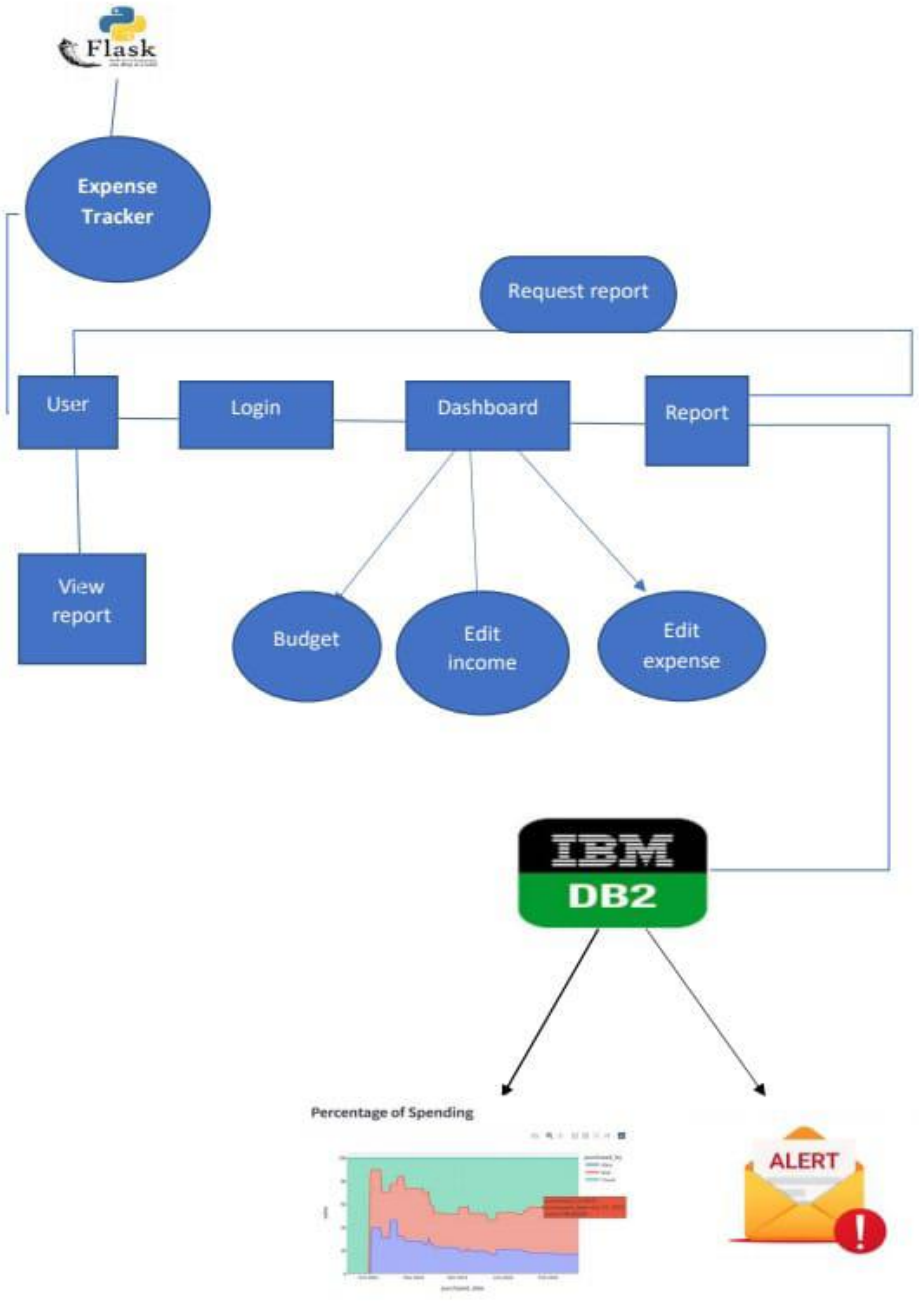


5.2 Solution & Technical Architecture

Solution Architecture


Daily Expense Tacker
IBM DB2
Database
Pie Chart
Application.
Login
Send email if the amount is above the level
Add income.
Add expense
edit expense
Navigation menu
Flask
Remainder
track
Account
logout
User

Technical architecture



5.3 User Stories

User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user and web 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	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through form	I can register by entering the details	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access my dashboard	High	Sprint-1
	Dashboard	USN-6	As a user ,I can log into the dashboard and manage income	I can add, delete and modify the income	High	Sprint-1
		USN-7	As a user, I can log into the dashboard and manage expense	I can add, delete and modify the expenses	High	Sprint-1
		USN-8	As a user, I can get a report is based on the details	I can manage my money by viewing this report	Medium	Sprint-1
Administrator	Alert message	USN-9	As a user, I can get an email if the money level is above the limit	I can receive alert email	High	Sprint-1
	Database	USN-10	As a user , I can't able to see the database but the details are automatically stored on the database	Based on the details on the database, I can get the details of money monthly through email	High	Sprint-1

6.Project Planning &Scheduling

6.1 Planning and estimation

Sprint	Functional Requirement(Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	App Development	USN-1	Create expense tracker app	10	High	Iswarya, Reyma Jerusha
Sprint-1		USN-2	Design and app for tracking daily expenses	9	Medium	Iswarya, Reyma Jerusha
Sprint-2	Containerized the app	USN-3	The purpose is to provide secure reliable,highweight runtime environment	8	High	Ashika Jebi,Anusha
Sprint-3	Create UI	USN-4	Create an UI to interact with the application	9	Medium	Ashika Jebi,Iswarya
Sprint-4	Database connection	USN-5	Connect the project using IBM DB2 database	8	High	Reyma Jerusha,Anusha

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	10	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	10	6 Days	31 Oct 2022	05Nov 2022	20	05 Nov 2022
Sprint-3	10	6 Days	07 Nov 2022	12 Nov 2022	20	12Nov 2022
Sprint-4	10	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:

$$AV = \frac{sprint\ duration}{velocity}$$

AV=6/10=0.6

6.2 Sprint Delivery Schedule

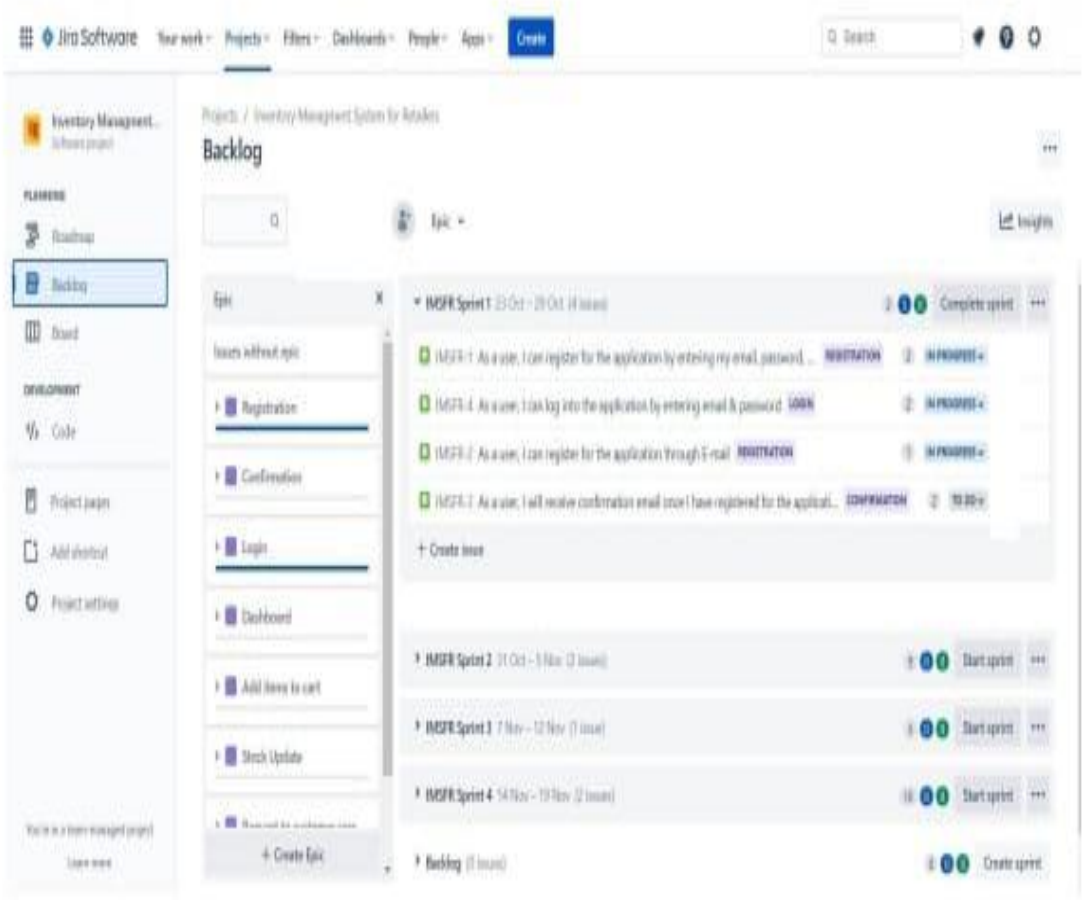
Milestone Activity Plan.

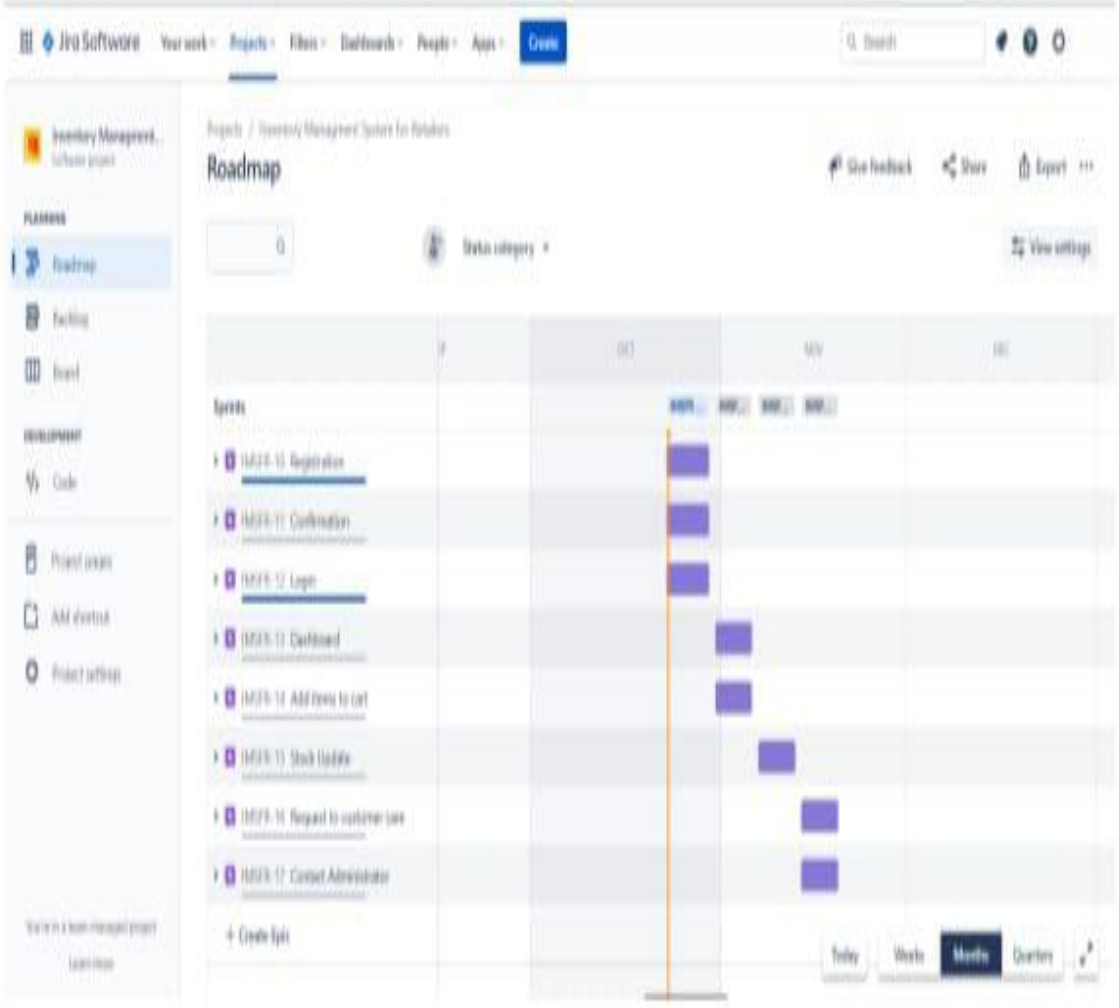
Milestone	Function (Epic)	Milestone Story Number	Story/ Task
Milestone -1	App development	**Ml**	Which keep track of income-Expense of a person on a day to day basis and help to manage monthly budget
Milestone-2	Containarize the app	M2	Containerization is the packaging of software code with operating system. A container can run on any infrastructure. It is an operating system level virtualization methods.
Milestone-3	Create UI	M3	Create an UI to interact with the application to help the customers.
Milestone-4	Database connection	M4	Allow client software to talk to database server software, whether on same machine or not. Here we use IBMDB, database for connection.

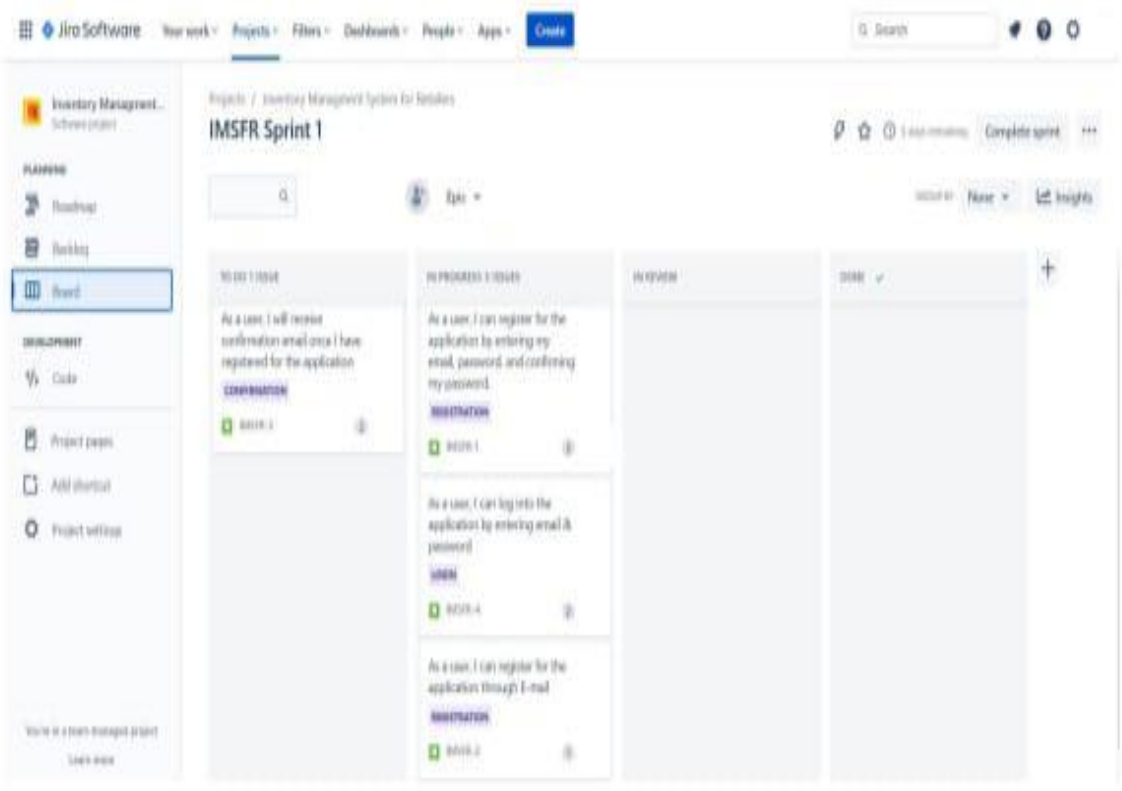
Milestone-5	Application layer	M5	Build the flask application and the HTML pages.
Milestone-6	Train CNN model	M6	Register for IBM Cloud and train Image Classification Model.
Milestone-6	Final result	M7	To ensure all the activities and resulting the final output.

6.3 Reports from JIRA

Project Tool: JIRA




Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Roadmap
Give feedback
Share
Export
Status category
View settings
Sprints
Today
Weeks
Months
Quarters


Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Inventory Managment...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / Inventory Management System for Retailers
IMSFR Sprint 1
Complete sprint
Epic
None
Insights
TO DO 1 ISSUE
As a user, I will receive confirmation email once I have registered for the application
CONFIRMATION
IMSFR-3
IN PROGRESS 3 ISSUES
As a user, I can register for the application by entering my email, password, and confirming my password.
REGISTRATION
IMSFR-1
As a user, I can log into the application by entering email & password
LOGIN
IMSFR-4
As a user, I can register for the application through E-mail
REGISTRATION
IMSFR-2
IN REVIEW
DONE

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'
"""
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)
```

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:

8.1 Test Cases

8.2 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
Print Engine	7	0	0	7
Client Application	20	0	0	20
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	8	0	0	8
Final Report Output	4	0	0	4

9. Results

9.1 Performance Metrices

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

ii. Transaction Receipts: Capture and organize your payment receipts to keep track of your expenditure.

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

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

10.Advantages & Disadvantages

1.One of the beat reason to track your spending is because that helps you stay involved in your finances.

2.Software or application that helps to track an accurate record of your money inflow and outflow.

3.Consistently tracking your expenses will help you maintain control of finances and promote better financial habits like saving and investing.

4.Monitoring your expenses throughout the month holds you accountable for your finances in a few key ways.

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

6 . Control the security of your business and customer data

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

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 Scope

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-40544-1660631023