

# **WEB PHISHING DETECTION USING MACHINE LEARNING**

## **CLOUD APPLICATION DEVELOPMENT DOMAIN**

**TEAM ID: PNT2022TMID07421**

**A PROJECT REPORT**

*Submitted by*

**ABINANDHAN.M**

**DHARANITHARAN.A.P**

**DURUVANTHRAJ.B.G**

**GANESHKUMAR.G**

**COMPUTER SCIENCE AND ENGINEERING**  
**P. A. COLLEGE OF ENGINEERING AND TECHNOLOGY**  
**(Autonomous)**  
**Pollachi, Coimbatore Dt. - 642 002**



**NOVEMBER 2022**



# **P. A. COLLEGE OF ENGINEERING AND TECHNOLOGY**

## **BONAFIDE CERTIFICATE**

Certified that this project report “**Personal Expense Tracker Application**” is the work of “**ABINANDHAN.M (721719104001), DHARANITHARAN.A.P(721719104016), DURUVANTHRAJ.B.G (721719104025), GANESHKUMAR.G (721719104027)**” who carried out the project work under our supervision.

### **SIGNATURE**

**Dr. D. CHITRA**

Professor

**HEAD OF THE DEPARTMENT**

Computer Science and Engineering

P. A. College of Engineering and  
Technology

### **SIGNATURE**

**FACULTY MENTOR**

Dr.A.KALIAPPAN

Associate Professor

Computer Science and Engineering

P. A. College of Engineering and  
Technology

### **SIGNATURE**

**FACULTY EVALUATOR**

Mr.S.SURESH KUMAR

Assistant Professor

Computer Science and Engineering

P. A. College of Engineering and  
Technology

Submitted to the Viva- Voce Examination held on -----

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

# **TABLE OF CONTENTS**

1

## **1. INTRODUCTION**

- 1.1 Project Overview
- 1.2 Purpose

## **2. LITERATURE SURVEY**

- 2.1 Existing problem
- 2.2 Reference s
- 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 t
- 4.2 Non-Functional requirements

## **5. PROJECT DESIGN**

- 5.1 Data Flow Diagrams
- 5.2 Solution & Technical Architecture
- 5.3 User Stories

## **6. PROJECT PLANNING & SCHEDULING**

- 6.1 Sprint Planning & Estimation
- 6.2 Sprint Delivery Schedule
- 6.3 Reports from JIRA

## **7. CODING & SOLUTIONING (Explain the features added in the project along with code)**

- 7.1 Feature 1
- 7.2 Feature 2
- 7.3 Database Schema (if Applicable)

## **8. TESTING**

- 8.1 Test Cases
- 8.2 User Acceptance Testing

## **9. RESULTS**

- 9.1 Performance Metrics

## **10. ADVANTAGES & DISADVANTAGES**

## **11. CONCLUSION**

## **12. FUTURE SCOPE**

## **13. APPENDIX**

Source Code

GitHub & Project Demo Lin

## **INTRODUCTION**

### **1.1 Project Overview**

As everyone are busy these days we are spending alot and not giving importance to tracking the expenses .So,here the Personal expense tracker help in tracking all the expenses.The expenses like daily,weekly,monthly and gives a report on the financial actions of our life.So everyone should have their track on spending because they can be stable financially.This application helps everyone to track and they can use for day-to-day purpose.

### **1.2 Purpose**

Many people spend unwantedly and they don't know where their money is going and they are weak in calculating the expenses.This make to spend more and does not make them to save money.So our application make them to track their expense and make the track of their each and every expense they making.Without recognising it, people frequently overspend, which can be harmful. You may keep track of how much money you spend every day and on what by using a daily spending tracker. You'll know exactly where your money is going atthe end of the month. One of the best ways to control your spending and create some semblance of order in your finances is to do this.

**2.1 Existing Solutions:****LITERATURE SURVEY – 1**

**TOPIC :** A Smart Approach to Track Daily Expenses

**AUTHOR :** AK Gupta, UP Singh, Dr. B. Balamurugan

**OVERVIEW :**

- A Java GUI-based application was suggested in this research to guarantee that it will aid users in managing the cost of their everyday expenses. They would be led by it and made aware of their everyday spending. The fundamental modules for adding and displaying costs as well as controlling expense categories were included in the suggested design. CRUD operations on expenditure data are supported.

**ADVANTAGES :**

- Category-wise management of expenses.
- Daily, monthly, annual basis tracking.
- Simple and user-friendly.

**DISADVANTAGES :**

- Lack of visual analytics for expense data.
- Lack of support for splitting group expenses.
- Supports manual data monitoring only.

## **LITERATURE SURVEY – 2**

**TOPIC** : Expense Tracker

**AUTHOR** : Lekshmi P, Dr. Mahalekshmi T, Prof Miriam Thomas

### **OVERVIEW :**

- The Daily Expense Tracker System is intended to keep track of an organization's income and expenses on a daily basis. This system splits income depending on daily costs. If the daily expenditure exceeds the daily allowance, the system will compute income and issue a new daily spending allowance. At the end of the month, the daily spending monitoring system will provide a report that displays an income-expense graph. Employees also send reports to the boss for verification. Manager sends final reports to administrator. The system forecasts the next month's expenses based on the final reports. It will assist in keeping track of all expenses and revenue.

### **ADVANTAGES :**

- Maintenance of expense data in the form of Excel sheets, CSV files, thereby avoiding entering individual expenses manually.
- Better visual analytics of data for various timelines.
- Supports handling for reimbursements.
- Least squares regression, a statistical procedure, is used to predict the expense limits.

### **DISADVANTAGES :**

- Suitable for organization scale, too complex for personal use.
- Expense prediction is not really necessary for small transactions made on personal use.
- Involves the participation of 3 roles Admin, Manager, Employee.

## **LITERATURE SURVEY – 3**

**TOPIC** : Expense Tracker Application

**AUTHOR** : Mrs.P.Usha ,Velmurugan.R

### **OVERVIEW :**

- This is an android-based application that allows users to keep a computerised diary to monitor spending on a daily basis in order to remain on budget and know expenses that are shown via a graphical representation with unique capabilities for categorising expenses suited for the user. Java, XML, and MySQL are utilised. View analytics, filtering transaction views, and a PDF report are all available.

### **ADVANTAGES :**

- Has various components of updating and viewing users expenditure.
- User can track his expenses by choosing a day and using various filtering options to study expenses.
- Visualization using pie chart with percentage view shows graphical representation.

### **DISADVANTAGES :**

- Doesn't support upcoming android versions.
- If a particular data is deleted, it cannot be viewed again.
- Statistics about income and expense detail of user can be prepared.

### **LITERATURE SURVEY – 4**

**TOPIC :** Online Income and Expense Tracker

**AUTHOR :** S. Chandini, T. Poojitha, D. Ranjith, V.J. Mohammed Akram, M.S. Vani, V. Rajyalakshmi

### **OVERVIEW :**

- It is a web application which is helpful to manage out income and expense as a daily or periodically or else whenever we want to remind and acts as an indicator or reminder 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.

### **ADVANTAGES :**

- Generates report at the end of week or month to show Income.
- Expense via multiple graphs.
- There is also an option to view owe and lend expenses which adds or gets deducted from the overall budget according without bothering the user.
- User friendly and data is maintained efficiently.

### **DISADVANTAGES :**

- Does not provide any option to handle shared expense of a group.
- Effort has to be made to include each and every transaction into the input field.

### **LITERATURE SURVEY – 5**

**TOPIC :** A Review on Budget Estimator Android Application

**AUTHOR :** Priyanka Joshi, Aditya Kamble ,Namita Jagtap,

**OVERVIEW :**

- The Budget Estimator system is intended to manage the programme user's daily spending in a more effective and manageable manner. This project is about a mobile application Expenses system with geolocation tracking. Based on the user's location, it uses Google Places to verify the availability of stores in the vicinity and delivers a notice for offers. In terms of security design, this system may include a login authentication such as an OTP message to your mobile device; this feature may provide the user with more security trust. We propose an android-developed application to reduce manual calculations. This programme enables users to keep a digitally automated journal.

**ADVANTAGES :**

- In this paper, an algorithm was proposed to show offers in nearby places using geo- location tracking.
- This mobile application with 2-step verification method provides the security to the users.

**DISADVANTAGES :**

- Suitable for only Personal use.
- It does not provide any analytics.

**2.2 Proposed Solution**

Personal finance encompasses all financial decisions and activities that a Finance app facilitates by assisting you in managing your finances efficiently. A personal finance app will not only assist you with budgeting and accounting, but will also provide you with valuable information about money management.

Personal finance applications will ask users to enter their expenses, and their wallet balance will be updated based on their expenses, which will be visible to the user. Users can also get a graphical breakdown of their spending. They can set a limit for the amount to be used for that month, and if the limit is exceeded, the user will receive an email alert.



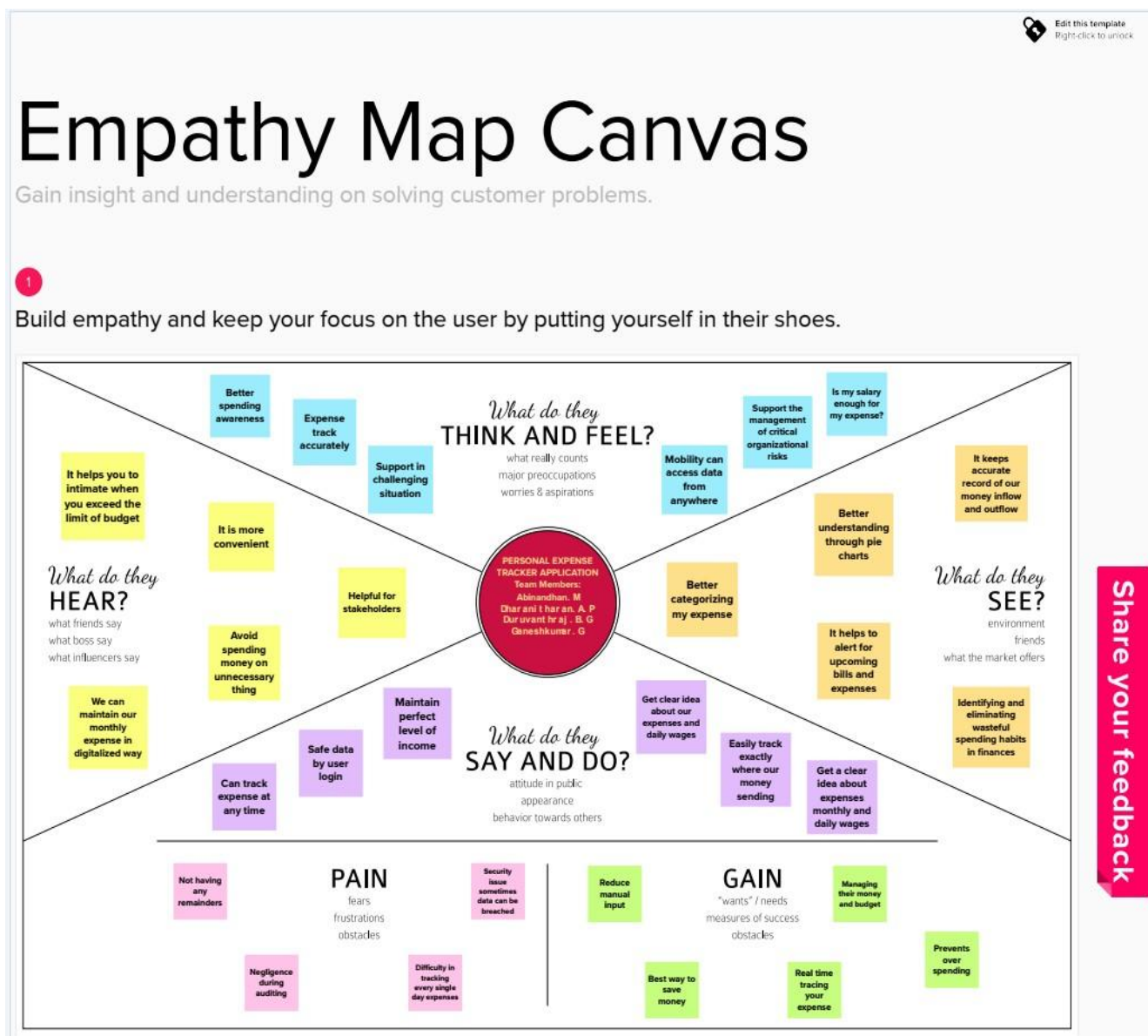
## 2.2 References

S.NO	JOURNAL TITLE	FIRST AUTHOR	CITATION
1	A Smart Approach to Track Daily Expenses	UP Singh	UP Singh, AK Gupta, Dr. B. Balamurugan (2021) - Spending Tracker : A Smart Approach to Track Daily Expenses - Turkish Journal of Computer and Mathematics Education Vol.12 No.6 , 5095-5103
2	Expense Tracker	Prof Miriam Thomas	Prof Miriam Thomas, Lekshmi P, and Dr. Mahalekshmi T (2022) -Expense Tracker - International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) Volume 9, Issue 4, September 2020 ISSN (Online) 2581-9429
3	Expense Tracker Application	Velmurugan.R	Velmurugan.R , Mrs.P.Usha (2021) - Expense Tracker Application - International Journal of Innovative Research in Technology (IJIRT) Volume 7, Issue 10, March 2021 ISSN: 2349-6002
4	Online Income and Expense Tracker	S. Chandini	S. Chandini, T. Poojitha, D. Ranjith, V.J. Mohammed Akram, M.S. Vani, V. Rajyalakshmi -Online Income and Expense Tracker, International Research Journal of Engineering and Technology (IRJET) Volume 6, Issue 3, March 2019 e-ISSN: 2395-0056, p-ISSN: 2395-0072
5	Budget Estimator Android Application	Namita Jagtap	Namita Jagtap, Priyanka Joshi, Aditya Kamble (2019)- A Review on Budget Estimator Android Application- International Journal of Innovative Research in Technology (IJIRT) Volume 6, Issue

			4, March 2021 ISSN: 2395-0056
--	--	--	-------------------------------

## IDEATION & PROPOSED SOLUTION

### 3.1 Empathy Map Canvas



### 3.1 Ideation & Brainstorming

2

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

10 minutes

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

**DURUVANTHRAJ B G**  
Filter the expenses graphically  
Edit income and expenses  
Keep accurate records  
Create a additional stream of income  
Shows cash flow  
Generate Monthly report

**DHARANITHARAN A P**  
Set smart budget to help you not over spend money in a chosen category  
No need for complicated Excel sheets  
Categorize your expenses  
Feedback System  
Get monthly report as pdf or excel sheet  
Over spending / under spending of money

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

3

**Group ideas**  
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.  

20 minutes

Track Your Income and Expenses

Add income and expenses

Shows Cash Flow

Set budget

Get monthly report

Alert

To remind user to enter the spendings

Set remainder and get notified

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

Access

Filter the expense periodically

Edit user profile

Visualize the expenses

1

2

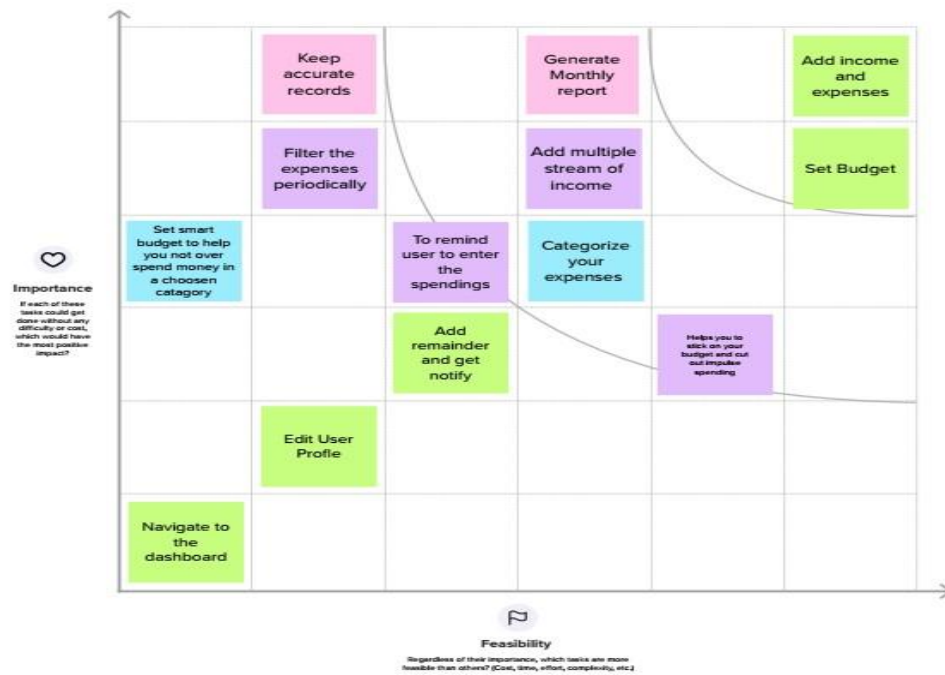
3

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



### 3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The problem is that people find it difficult to maintain track of their monthly expenses and to avoid obsessive spending.
2.	Idea / Solution description	The user can enter income and expenses into the personal cost tracker programme. As a result, the expense wallet is updated. A graphical breakdown of the expense could be obtained. The user is also notified if the budget's monthly maximum is surpassed.
3.	Novelty / Uniqueness	The personal expense tracker application helps the user not only in budgeting and accounting; it also provides the insights about money management through the analysis. The user also gets notified if the monthly limit is exceeded.
4.	Social Impact / Customer Satisfaction	The personal spending tracker programme not only assists the user in budgeting and accounting, but it also provides insights into money management through analysis. If the monthly limit is surpassed, the user is also notified.
5.	Business Model (Revenue Model)	The application may have a free and premium version, with the user having the option to upgrade to the premium version to gain access to additional features. Furthermore, the premium version may be ad-free.
6.	Scalability of the Solution	This application is not only for personal use, but it can also be extended to business organisations.

## 3.4 Problem Solution fit

### Problem-Solution fit canvas 2.0

PERSONAL EXPENSE TRACKER APPLICATION - TEAM ID: PNT2022TMID07421

Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span> Who is your customer? i.e. working parents of 0-5 y.o. kids <ul style="list-style-type: none"> <li>Customers are people who spend money either carelessly or with difficulty keeping track of it.</li> <li>Provides a whole lot of different categories of expenditure types to avoid mismatch of expenditure.</li> <li>The Need for Financial Management for Common People.</li> </ul>	<b>6. CUSTOMER CONSTRAINTS</b> <span>CC</span> What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash network connection, available devices. <ul style="list-style-type: none"> <li>The majority of online solutions include numerous ads that restrict their effectiveness.</li> <li>The approach proposed here features a function that allows you to view expenses visually.</li> <li>It also has a functionality that notifies you throughout email if a spending exceeds a predetermined limit.</li> <li>Devices That Are Available.</li> <li>Network Relationship</li> </ul>	<b>5. AVAILABLE SOLUTIONS</b> <span>AS</span> Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking <ul style="list-style-type: none"> <li>Applications that track expenses and are accessible for both iOS and Android.</li> <li>A personal expense tracking tool was created for this project.</li> <li>Calculating the total spendings of the user. Alerting the user nearing the budget. Notifying the user of spending above budget. Providing useful financial tips for better savings. Providing reports for assessments</li> </ul>	Explore AS, differentiate
	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <span>J&amp;P</span> Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides. <ul style="list-style-type: none"> <li>This application's goal is to make it possible for users to keep track of their spending.</li> <li>The categories for the expenses are made available to the clients.</li> <li>They also have the choice of viewing the costs as a graphical depiction for the duration of a year, six months, etc.</li> <li>Fixed by establishing a cap on the amount that can be spent in a given month; if the cap is surpassed, the user will be notified through email.</li> </ul>	<b>9. PROBLEM ROOT CAUSE</b> <span>RC</span> What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations. <ul style="list-style-type: none"> <li>Inappropriate expenses result in high taxes.</li> <li>Easy company forecasting; significant cost savings; difficulty in manually tracking expenses due to the abundance of payment options</li> <li>An opportunity lost</li> <li>A reduction in savings</li> <li>A poor investment</li> <li>No comprehensive and simple way to keep track of everyday spending</li> <li>excessive spending without effective management</li> <li>insufficient financial knowledge</li> <li>mistake prone and it takes time.</li> </ul>	<b>7. BEHAVIOUR</b> <span>BE</span> What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer; calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace) <ul style="list-style-type: none"> <li>Start utilising the cost tracker software.</li> <li>Classify expenses as they are incurred to save money.</li> <li>Set a monthly spending cap and maintain separate in-hand wallet and online accounts.</li> <li>Ask your neighborhoods or coworkers for information.</li> <li>Obtain recommendations from professionals who are knowledgeable in the finance sector.</li> </ul>	
<b>3. TRIGGERS</b> <span>TR</span> What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.  Knowing that these expenditure applications can help clients save a lot of money.	<b>10. YOUR SOLUTION</b> <span>SL</span> If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.  Create a flask-based personal cost tracker application, use the sendgrid framework to enable email-based expense notifications, and offer a graphical expense display option.	<b>8. CHANNELS OF BEHAVIOUR</b> <span>CH</span> <b>8.1 ONLINE</b> What kind of actions do customers take online? Extract online channels from #7  Virtual budget trackers have numerous advertising that, when clicked, capture information including account numbers if they are provided.  <b>8.2 OFFLINE</b> What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development. <ul style="list-style-type: none"> <li>Access to data that has already been downloaded.</li> <li>Make sure they are familiar with the tax laws by having them read the available books on taxes.</li> </ul>	Extract online & offline CH of BE	
<b>4. EMOTIONS: BEFORE / AFTER</b> <span>EM</span> How do customers feel when they face a problem or a job and afterwards? i.e. lost, insecure > confident, in control - use it in your communication strategy & design.  Before: Users are in a depressive state prior. After: Users feel ready to handle the cost.				

Identify strong TR & EM

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

Extract online & offline CH of BE

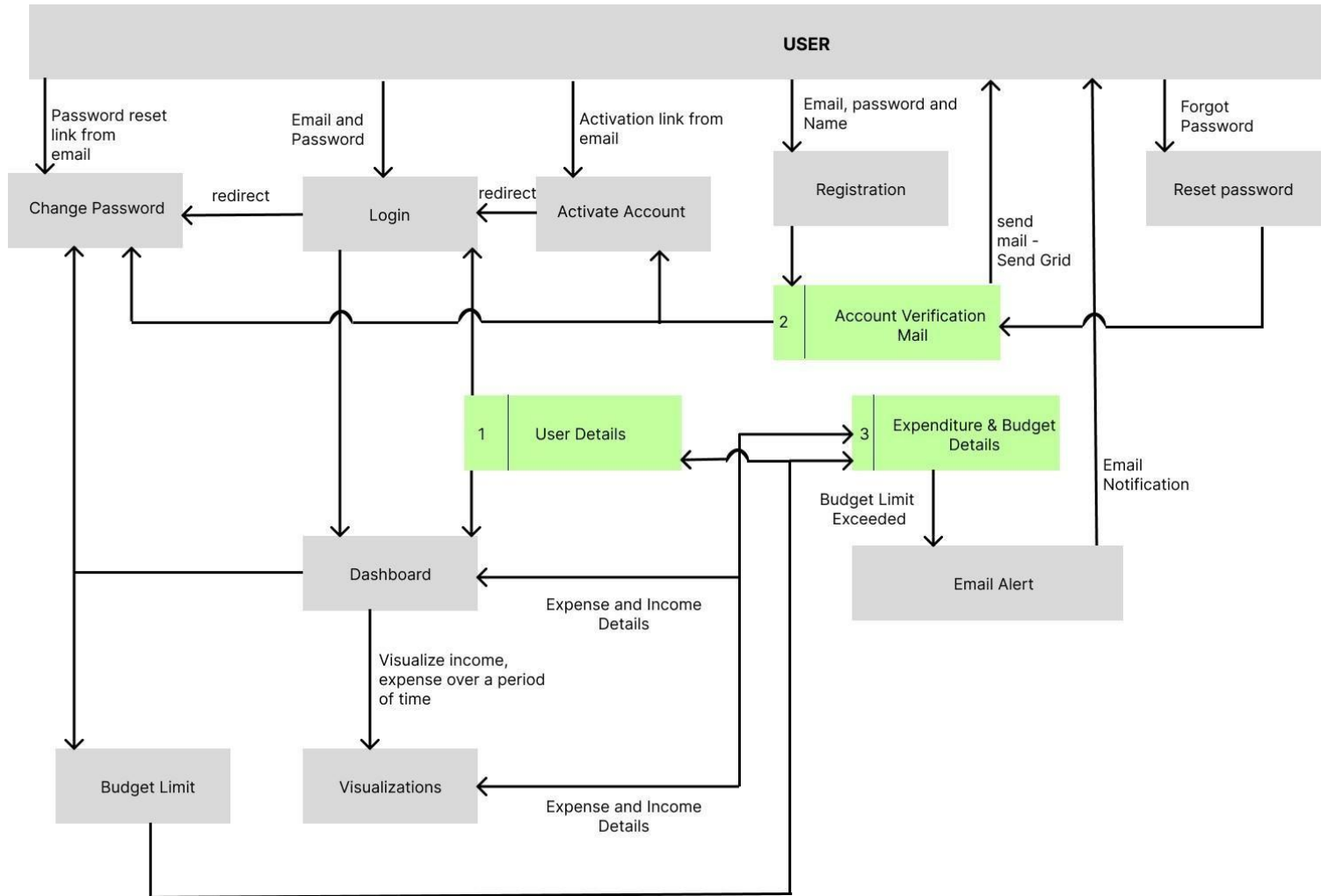
**4.1 Functional Requirements**

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration Form for collecting details.
FR-2	User Login	Enter username and password.
FR-3	Forget Password	Reseting the password by sending an OTP to user's mail.
FR-4	Calendar	Personal expense tracker application must allow user to add the data to their expenses.
FR-6	Dashboard	User can add the expense and can evaluate them using the provided options.
FR-5	Expense Tracker	This application must graphically represent the expense like report.
FR-6	Report generation	Report must be generated in a graphical form.
FR-7	Category	This application shall allow users to add categories of their expenses.
FR-8	Result Page	Show the user result.

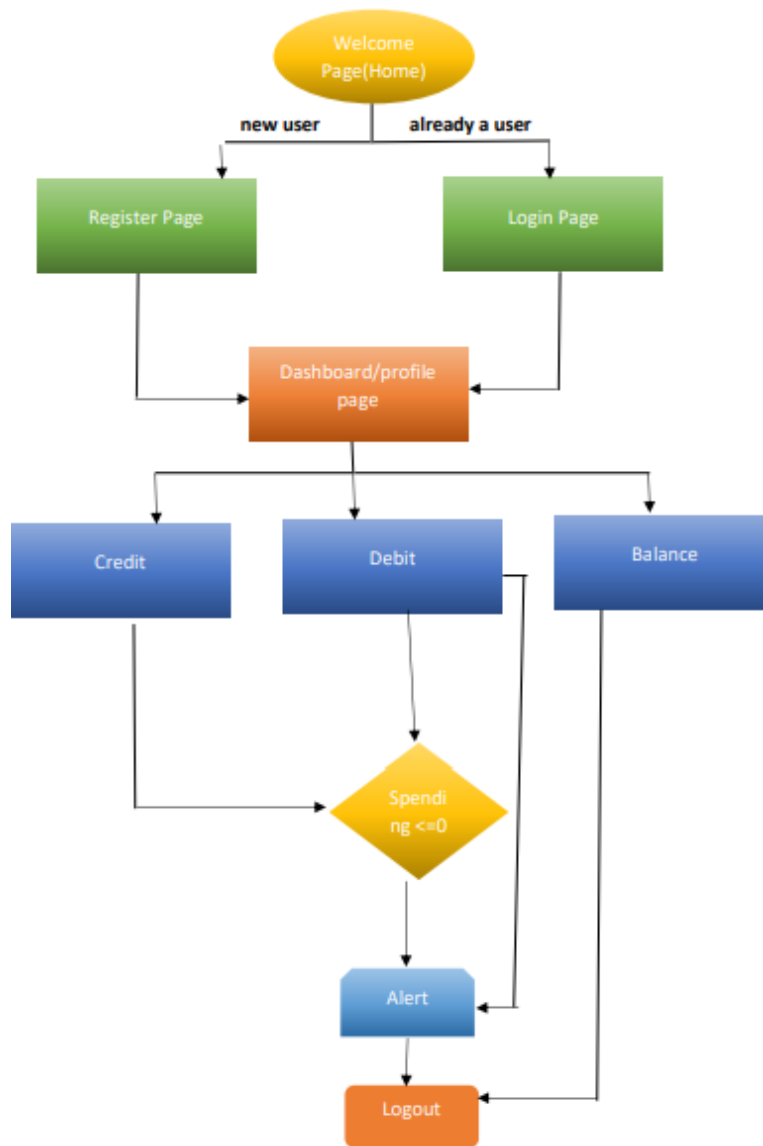
## 4.2 Non-Functional Requirements

FR No.	Non-Functional Requirement	Description
NFR-1	<b>Usability</b>	Most web browsers allow users to access the application. The application's attractive and detailed user interface makes it easier to use. It makes it easier for you to monitor your earnings and expenses.
NFR-2	<b>Security</b>	Customers are required to set up an account for themselves using their email, which is secured by a longer password of six characters. This application might prevent you from engaging in online crimes.
NFR-3	<b>Reliability</b>	Each data record is kept in an effective database schema that is well built. No chance of data loss exists.
NFR-4	<b>Performance</b>	Expense types include categories and an option. The system's throughput is increased thanks to the lightweight database support.

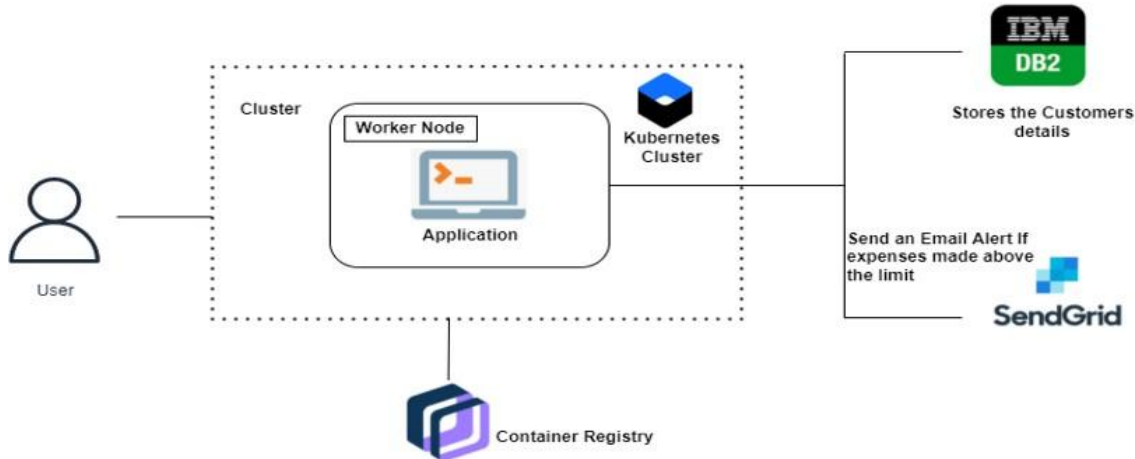


**5.1 Data Flow Diagrams**

## 5.2 Solution & Technical Architecture



### 5.3 User stories



User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	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
Customer	Account Activation	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
Customer	Login	USN-3	As a user, I can log into the application by entering email & password	I can login on correct credentials	High	Sprint-1
Customer	Dashboard	USN-4	As a user, I can add expenses and income	I can enter the amount and category to save	High	Sprint-2
Customer	Login	USN-5	As a user, I can change my password	I can change password if I'm logged in	Low	Sprint-4
Customer	Dashboard	USN-6	As a user, I can view my past expenses and income over a period of time, and visualize them	I can view past records provided if records available	High	Sprint-2

Customer	Notification	USN-7	As a user, When my monthly limit exceeds, i get a email notification	I can get alert notification, if i have set a limit	Low	Sprint-4
Customer	Dashboard	USN-8	As a user, I can set a monthly expense limit	I can set a valid limit	Low	Sprint-4
Customer	Forgot password	USN-9	As a user, I can get a reset password link if i forget it through mail	I need to have access to my email	High	Sprint-3
Customer	Forgot password	USN-10	As a user, I can change my password if I forget it	The link should be valid	High	Sprint-3

## **6 PROJECT PLANNING AND SCHEDULING**

### **6.1 Product Backlog, Sprint Schedule, and Estimation**

<b>Sprint</b>	<b>Functional Requirement (Epic)</b>	<b>User Story Number</b>	<b>User Story / Task</b>	<b>Story Points</b>	<b>Priority</b>	<b>Team Members</b>
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	Abinandhan M
Sprint-3	Security	USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Abinandhan M
Sprint-3	Registration	USN-3	As a user, I can register for the application through Facebook	2	Low	Dharanitharan A P
Sprint-1	Registration	USN-4	As a user, I can register for the application through Gmail	2	Medium	Dharanitharan A P
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	1	High	Duruvanthraj B G
Sprint- 2	Dashboard	USN-6	As a user, I can view, edit, delete my expenses and budget.	2	High	Duruvanthraj B G
Sprint- 3	Report	USN-7	As a user, I can group the expenses	1	Medium	Ganeshkumar G

Sprint- 4	Graphical Representation	USN-8	As a administrator, I can view if there are anyspam accounts and I should be able to ban them	2	Medium	Ganeshkumar G
-----------	--------------------------	-------	---	---	--------	---------------

## 6.1 Project Tracker,Velocity & Burndown chart

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

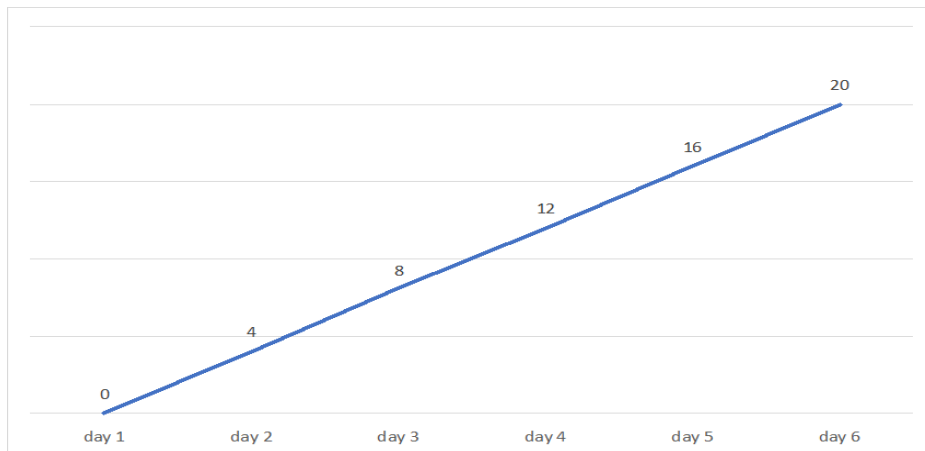
### Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

### **Burdown Chart:**

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



### **Sprint duration**

## 7.

## CODING AND SOLUTIONING

### 7.1 Server Side

```
from flask import Flask, render_template, request, redirect, session
import ibm_db
import re
from datetime import datetime

app = Flask(__name__)

app.secret_key = 'safste5eyhrsgh'
try:
    conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnk39u98g.databases.appdomain.cloud;PORT=31249;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=mwz43368;PWD=Xt9nRoZW5RemaaTu",",")
except Exception as e:
    print(e)

#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 = "
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

        query = 'SELECT * FROM register WHERE username =?;'
        stmt=ibm_db.prepare(conn,query)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'^[^\s@]+@[^\s@]+\.[^\s@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            query = "INSERT INTO register (username,email,password) VALUES (?,?,?)"
            stmt=ibm_db.prepare(conn,query)
            ibm_db.bind_param(stmt,1,username)
            ibm_db.bind_param(stmt,2,email)
            ibm_db.bind_param(stmt,3,password)
            ibm_db.execute(stmt)
            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']
    query = "SELECT * FROM register WHERE username = ? AND password = ?;"
    stmt = ibm_db.prepare(conn,query)
    ibm_db.bind_param(stmt,1,username)
    ibm_db.bind_param(stmt,2,password)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_tuple(stmt)
    print (account)

    if account:
        session['loggedin'] = True
        session['id'] = account[0]
        userid= account[0]
        session['username'] = account[1]

        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']
date = str(datetime.strptime(date.replace("T", " "), "%Y-%m-%d %H:%M"))
expensename = request.form['expensename']
amount = request.form['amount']
paymode = request.form['paymode']
category = request.form['category']
print(date + " " + expensename + " " + amount + " " + paymode + " " + category)
query = 'INSERT INTO expenses (userid,date,expensename,amount,paymode,category) VALUES (?, ?, ?, ?, ?, ?)'
stmt = ibm_db.prepare(conn,query)
ibm_db.bind_param(stmt,1,session['id'])
ibm_db.bind_param(stmt,2,date)
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)
# cursor.execute('INSERT INTO expenses VALUES (NULL, % s, % s, % s, % s, % s, % s)', (session['id'] ,date,
expensename, amount, paymode, category))

return redirect("/display")

```

#DISPLAY---graph

```

@app.route("/display")
def display():
    print(session["username"],session['id'])
    # query = 'SELECT * FROM expenses WHERE userid = ' + str(session['id']) + ' ORDER BY date DESC'
    # stmt = ibm_db.prepare(conn,query)
    # ibm_db.bind_param(stmt,1,(str(session['id'])))
    # ibm_db.execute(stmt)
    # tuple = ibm_db.fetch_tuple(stmt)
    # expense = []
    # while tuple != False:
    #     expense.append(list(tuple))
    #     tuple = ibm_db.fetch_tuple(stmt)
    # query = 'SELECT SUM(amount) FROM expenses WHERE userid= ?'
    # stmt = ibm_db.prepare(conn,query)
    # ibm_db.bind_param(stmt,1,(str(session['id'])))
    # ibm_db.execute(stmt)
    # texpense = []
    # tuple = ibm_db.fetch_tuple(stmt)
    # while tuple!=False:

```

```

# print(tuple)
# texpanse.append(tuple)
# tuple = ibm_db.fetch_tuple(stmt)
# print(texpanse)
# 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 += float(x[4])
#     if x[6] == "food":
#         t_food += float(x[4])
#
#     elif x[6] == "entertainment":
#         t_entertainment += float(x[4])
#
#     elif x[6] == "business":
#         t_business += float(x[4])
#     elif x[6] == "rent":
#         t_rent += float(x[4])
#
#     elif x[6] == "EMI":
#         t_EMI += float(x[4])
#
#     elif x[6] == "other":
#         t_other += float(x[4])

```

```

param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " ORDER BY date DESC"
res = ibm_db.exec_immediate(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"].strip())
expense.append(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]
if expense:
    return render_template('display.html', expense = expense, title="History", 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)
return redirect('/add')

```

```

# #delete---the--data

```

```

@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
def delete(id):
    query = 'DELETE FROM expenses WHERE id = ?'

```

```

stmt = ibm_db.prepare(conn,query)
ibm_db.bind_param(stmt,1,str(id))
ibm_db.execute(stmt)
# cursor.execute('DELETE FROM expenses WHERE id = {0}'.format(id))
# mysql.connection.commit()
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()
    query = 'SELECT * FROM expenses WHERE id =?'
    stmt = ibm_db.prepare(conn,query)
    ibm_db.bind_param(stmt,1,str(id))
    ibm_db.execute(stmt)
    row = []
    tuple = ibm_db.fetch_tuple(stmt)
    while tuple!=False:
        row.append(tuple)
        tuple = ibm_db.fetch_tuple(stmt)
    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']

```

```

        query = 'UPDATE expenses SET date = ? , expensename = ? , amount = ? , paymode = ? , category = ? WHERE
expenses.id = ? '

```

```

        # 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()
stmt = ibm_db.prepare(conn,query)
ibm_db.bind_param(stmt,1,date)
ibm_db.bind_param(stmt,2,expensename)
ibm_db.bind_param(stmt,3,amount)
ibm_db.bind_param(stmt,4,str(paymode))
ibm_db.bind_param(stmt,5,str(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']
        print(number)
        query = 'INSERT INTO limits (userid,limitss) VALUES (?, ?) '
        stmt = ibm_db.prepare(conn,query)
        # cursor.execute('INSERT INTO limits VALUES (NULL, % s, % s) ',(session['id'], number))
        # mysql.connection.commit()
        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():
    query = "SELECT limitss FROM limits ORDER BY limits.id DESC LIMIT 1 "
    stmt = ibm_db.prepare(conn,query)

```

```

# cursor.execute('SELECT limitss FROM `limits` ORDER BY `limits`.`id` DESC LIMIT 1')
ibm_db.execute(stmt)
x= ibm_db.fetch_tuple(stmt)
if x:
    s = x[0]
    return render_template("limit.html" ,title="Limit", y=s)
else:
    return render_template("limit.html",title="Limit" , y=0)

# #REPORT

@app.route("/today")
def today():
    query = 'SELECT TIME(date), amount FROM expenses WHERE userid = ? AND DATE(date) = DATE(NOW())'
    # cursor.execute('SELECT TIME(date), amount FROM expenses WHERE userid = %s AND DATE(date) =
DATE(NOW())',(str(session['id'])))
    stmt = ibm_db.prepare(conn,query)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    tuple = ibm_db.fetch_tuple(stmt)
    texpanse = []
    while tuple!=False:
        texpanse.append(tuple)
        tuple = ibm_db.fetch_tuple(stmt)
    print(texpanse)
    # query = 'SELECT * FROM expenses WHERE userid = ? AND DATE(date) = DATE(NOW()) ORDER BY
expenses.date DESC'
    # stmt = ibm_db.prepare(conn,query)
    # ibm_db.bind_param(stmt,1,session['id'])
    # ibm_db.execute(stmt)
    # expense = []
    # while tuple!=False:
    #     temp = []
    #     temp.append(tuple["ID"])
    #     temp.append(tuple["USERID"])
    #     temp.append(tuple["DATE"])
    #     temp.append(tuple["EXPENSENAME"])
    #     temp.append(tuple["AMOUNT"])
    #     temp.append(tuple["PAYMODE"])
    #     temp.append(tuple["CATEGORY"].strip())
    #     expense.append(temp)
    #     tuple = ibm_db.fetch_tuple(stmt)
    #     print(expense)
    # 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(conn, param)
dictionary = ibm_db.fetch_assoc(res)
print(dictionary)
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"].strip())
    expense.append(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 += float(x[4])

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

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

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

```



```

        elif x[6] == "other":
            t_other += float(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",title="Today",texpanse = texpanse, expense = expense, total = total ,
        t_food = t_food,t_entertainment = t_entertainment,
        t_business = t_business, t_rent = t_rent,
        t_EMI = t_EMI, t_other = t_other )

@app.route("/month")
def month():
    query = 'SELECT DATE(date), SUM(amount) FROM expenses WHERE userid= ? AND MONTH(DATE(date))=
MONTH(now()) GROUP BY DATE(date) ORDER BY DATE(date) '
    # 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'])))
    stmt = ibm_db.prepare(conn,query)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    tuple = ibm_db.fetch_tuple(stmt)
    texpanse = []
    while tuple!=False:
        texpanse.append(tuple)
        tuple = ibm_db.fetch_tuple(stmt)
    print(texpanse)

    # cursor = mysql.connection.cursor()
    # query = 'SELECT * FROM expenses WHERE userid = ? AND MONTH(DATE(date))= MONTH(now()) ORDER
BY expenses.date DESC'
    # # cursor.execute('SELECT * FROM expenses WHERE userid = % s AND MONTH(DATE(date))=
MONTH(now()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
    # stmt = ibm_db.prepare(conn,query)
    # ibm_db.bind_param(stmt,1,session['id'])
    # ibm_db.execute(stmt)

```

```

# expense = []
# tuple = ibm_db.fetch_tuple(stmt)
# while tuple!=False:
#     expense.append(tuple)
#     tuple = ibm_db.fetch_tuple(stmt)
# 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(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"].strip())
    expense.append(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 += float(x[4])
    if x[6] == "food":
        t_food += float(x[4])

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

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

```

```

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

elif x[6] == "other":
    t_other += float(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("month.html",title="Month", texpanse = texpanse, expense = expense, total = total ,
    t_food = t_food,t_entertainment = t_entertainment,
    t_business = t_business, t_rent = t_rent,
    t_EMI = t_EMI, t_other = t_other )

@app.route("/year")
def year():
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT MONTH(date), SUM(amount) FROM expenses WHERE userid= %s AND
YEAR(DATE(date))= YEAR(now()) GROUP BY MONTH(date) ORDER BY MONTH(date) ',(str(session['id'])))
    # texpanse = cursor.fetchall()
    # print(texpanse)
    query = 'SELECT DATE(date), SUM(amount) FROM expenses WHERE userid= ? AND YEAR(DATE(date))=
YEAR(now()) GROUP BY DATE(date) ORDER BY DATE(date)'
    stmt = ibm_db.prepare(conn,query)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    texpanse = []
    tuple = ibm_db.fetch_tuple(stmt)
    while tuple!=False:
        texpanse.append(tuple)
        tuple = ibm_db.fetch_tuple(stmt)

    # 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()
# query = 'SELECT * FROM expenses WHERE userid = ? AND YEAR(DATE(date))= YEAR(now()) ORDER BY
expenses.date DESC'
# stmt = ibm_db.prepare(conn,query)
# ibm_db.bind_param(stmt,1,session['id'])
# ibm_db.execute(stmt)
# expense = []
# tuple = ibm_db.fetch_tuple(stmt)
# while tuple!=False:
#     expense.append(tuple)
#     tuple = ibm_db.fetch_tuple(stmt)
    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND YEAR(date) = YEAR(current
timestamp) ORDER BY date DESC"
    res = ibm_db.exec_immediate(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"].strip())
        expense.append(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:
    print(x)
    total += float(x[4])
    if x[6] == "food":
        t_food += float(x[4])

    elif x[6] == "entertainment":

```

```

        t_entertainment += float(x[4])

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

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

    elif x[6] == "other":
        t_other += float(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",title="Year", texpanse = texpanse, expense = expense, total = total ,
        t_food = t_food,t_entertainment = t_entertainment,
        t_business = t_business, t_rent = t_rent,
        t_EMI = t_EMI, t_other = t_other )

#log-out

@app.route('/logout')

def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    return render_template("home.html")

if __name__ == "__main__":
    app.run()

```

8.1 Acceptance Testing

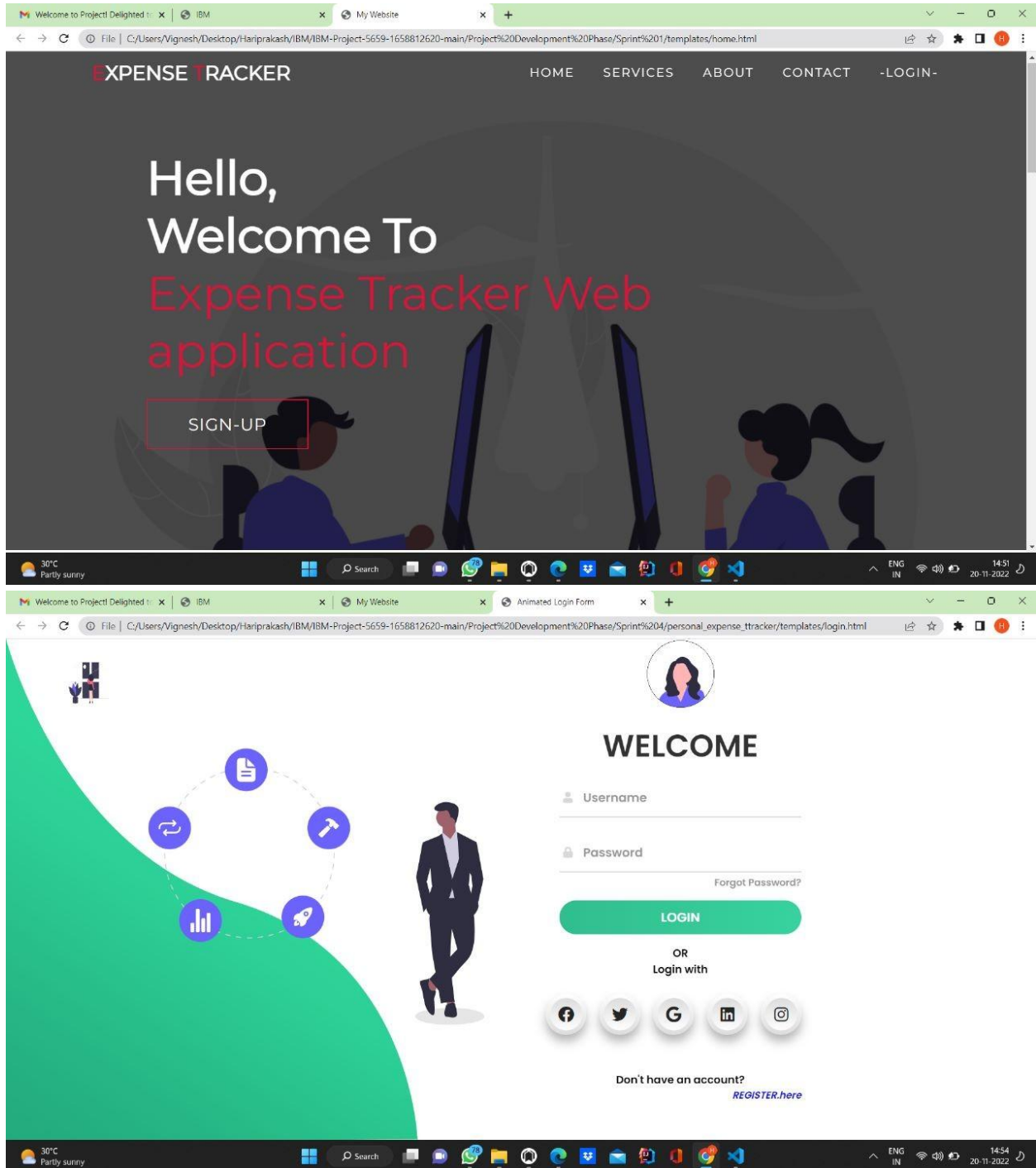
S.No	Test Cases	Yes/ No
1.	Keyword driven	Yes
2.	Responds in manually drafted rules	Yes
3.	Manages multiple users	Yes
4.	Conversational Paradigm	Yes
3.	Learns from real interactions	No
4.	Training via historical data	No
5.	Has decision-making skills	No

8.1 TestCase Report

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	1	0	0	0	1
Duplicate	1	0	0	0	1
External	3	1	0	0	4
Fixed	4	1	0	0	5
Not Reproduced	0	0	0	0	1
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	9	2	0	0	11

## 9. Results

### 9.1 Sample Screens:



## **10. ADVANTAGES & DISADVANTAGES**

### **10.1 Advantages:**

- Enhanced clientele services
- Cloud-based approach
- Order completion

### **10.2 Disadvantages:**

- System Conflict
- less frequent physical audits
- There is no available fix to shorten the service cycle or improve the bottleneck.

## **11.**

## **FUTURE SCOPE**

- 1) It will have different record-keeping options.
- 2) It will continue to send notifications about our daily spending automatically.
3. Despite being in a haste to make money in today's hectic and expensive world, we eventually gave up. As we naively waste money on unnecessary items and titles. We so came over with the intention of following our profit.

## **12. REFERENCES**

- 1) <https://github.com/IBM-EPBL/IBM-Project-53220-1661319171>