

IBM-Project-43884-1663922812

**PERSONAL EXPERIENCE TRACKING
APPLICATIONS**

TEAM DETAILS:

Team ID : PNT2022TMID45221

College Name : M.A.M College of Engineering and Technology

Department : INFORMATION TECHNOLOGY

TEAM MEMBERS :

Team Lead - Nitish S

Team Member 1 - Akash E

Team Member 2 - Gokulnath G

Team Member 3 - Rahavan R

1. **INTRODUCTION**
 - 1.1 Project Overview
 - 1.2 Purpose
2. **LITERATURE SURVEY**
 - 2.1 Existing problem
 - 2.2 References
 - 2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
 - 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 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 Link

INTRODUCTION

The daily expense tracker is a web Application which is used to maintain data of daily, weekly, monthly and yearly expenses in an eye-catching way. This project is aimed at developing a web application which will be helpful to users who run out of resources due to mismanagement and also find it difficult to maintain records of their expenses. So daily expense tracker will help them manage their needs and spending in a better way by accessing the web application directly from web browsers. It is designed and developed in a way that it is compatible with each and every device. We have also considered disinterest or the lack of time of users to maintain such records and thus made the web application voice powered as well making sure that there is something for each and every type of users. The application doesn't need any extended hardware or software support to run and thus a user with minimal resources can also make use of the application to make a difference into their life.

1.1 Project Overview

Tracking regular expense is a key factor to maintain a budget. People often track expense using pen and paper method or take notes in a mobile phone or a computer. These processes of storing expense require further computations and processing for these data to be used as a trackable record. In this work, we are proposing an automated system named as expense to store and calculate these data. Expnese is an application that runs on Android smartphones. By using this application, users can save their expense by simply scanning the bills or receipt copies. This application extracts the textual information from the receipts and saves the amount and description for further processing. It also monitors user's income by tracking the received SMS's from the user's saving accounts. By calculating income and expense it produces the user's balance in monthly and yearly basis. Overall, this is a smart automated solution for tracking expense.

1.2 Purpose

Tracking daily expense isn't therefore innovative. several ancient and technological approach is found to trace our expenses and budget with their own practicality. From decades ago and these days we've been writing our expenditure in a very register to calculate the profit or saving. Not solely this several desktop and mobile applications has been developed for this purpose. Quicken and Microsoft cash were the primary desktop applications was developed decades ago however wasn't therefore acquainted with the users. My budget book application is employed to check the expenses in chart or graphs with the calendar system. QuickBooks were the appliance for the little business holder to finish off their whole business. YNAB and Penny were the most recent application that were

embedded with AI and applicable for commerce expenses mechanically. However, Mint was the one that was wide used and trusty

LITERATURE SURVEY

2.1 Existing Problem

The problem of current generation population is that they can't remember where all of the money they earned have gone and ultimately have to live while sustaining the little money they have left for their essential needs. In this time there is no such perfect solution which helps a person to track their daily expenditure easily and efficiently and notify them about the money shortage they have. For doing so have to maintain long ledgers or computer logs to maintain such data and the calculation is done manually by the user, which may generate error leading to losses.

Not having a complete tracking.

Reference

- <https://nevonprojects.com/daily-expense-tracker-system/>
- <https://data-flair.training/blogs/expense-tracker-python/>
- <https://phpgurukul.com/daily-expense-tracker-using-php-and-mysql/>
- <https://ijarsct.co.in/Paper391.pdf>
- https://kandi.openweaver.com/?landingpage=python_all_projects&utm_source=google&utm_medium=cpc&utm_campaign=promo_kandi_ie&utm_content=kandi_ie_search&utm_term=python_devs&gclid=Cj0KCQiAgribBhDkARIsAASA5bukrZgbI9UZxzpoyf0PofB1mZNxzCokUP-3TchpYMclHTYFYiqP8aAmmwEALw_wcB

2.3 Problem Statement Definition

In the busy schedule people are not ready to waste their time on tracking the unwanted usage. so we gonna create the tracker application that will show the important and relevant track expense to the user with user friendly interface and they are getting quick pick of the day.

IDEATION & PROPOSED SOLUTION

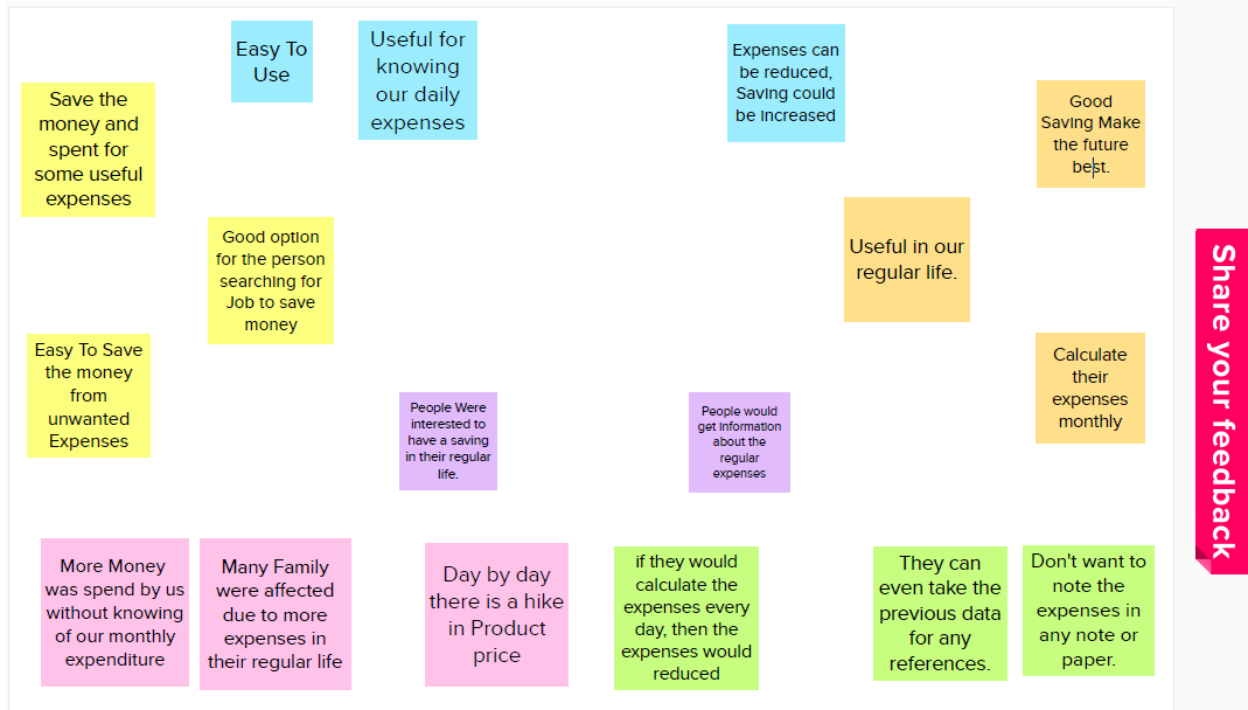
3.1 Empathy Map Canvas

Empathy Map Canvas

Gain insight and understanding on solving customer problems.

1


Build empathy and keep your focus on the user by putting yourself in their shoes.



3.2 Ideation & Brainstorming

Ideation and the practise of brainstorming, a particular method for coming up with fresh ideas, are frequently closely related. The main distinction between ideation and brainstorming is that whereas brainstorming is nearly often done in groups, ideation is typically seen as being more of a solitary endeavour. A group of people are frequently gathered for a brainstorming session to generate either fresh, general ideas or solutions to specific problems or circumstances. On instance, a large firm that has discovered it is the target of a significant lawsuit might wish to consult with its top executives to come up

with ideas for how to publicly respond to the case being filed. In a brainstorming session, participants are encouraged to freely share any ideas that may come to mind. According to the theory, by coming up with a lot of ideas, the brainstorming group is more likely to find a workable solution to the problem they are trying to solve.




Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- 🕒 10 minutes to prepare
- 🕒 1 hour to collaborate
- 👤 2-8 people recommended

[Share template feedback](#)



Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →

1


Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

🕒 5 minutes


PROBLEM


How might we [your problem statement]?





Key rules of brainstorming


To run a smooth and productive session


 Stay in topic.

 Encourage wild ideas.

 Defer judgment.

 Listen to others.

 Go for volume.

 If possible, be visual.

Brainstorm

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

10 minutes

TIP You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing!

3

Group Ideas

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

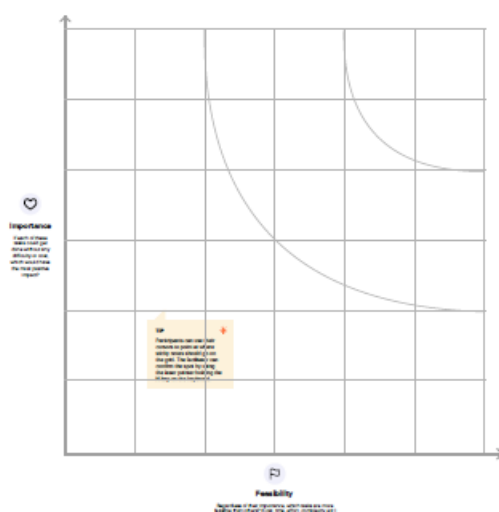
20 minutes

TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Priority

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.

30 minutes



After you collaborate

You can export the model as an image or pdf to share with members of your company who might find it helpful.

[Quick add-ons](#)

[View the manual](#)

Share the mural
Share a video link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

[Report this record](#)

Report the value
Report a copy of the record as a PDF or PPT in attach or email. Include in slides, or save in your drive.

Keep moving forward

Strategy Navigator

 Define the components of a narrative or scenario

[Open this template in](#)

 Understand customer needs, motivations, and responses for an experience.

Given the importance of

- Identify strengths, weaknesses, opportunities, and threats (SWOT) in developing a plan.

[Open this template](#)

⇒ Share complete feedback



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	In the busy schedule people are not ready to waste their time on tracking the unwanted usage.so we gonna create the tracker application that will shows the important and relevant track expense to the user with user friendly interface and they are getting quick pick of the day.
2.	Idea / Solution description	We will provide them application effective financial management requires the proper tracking of income and expenses
3.	Novelty / Uniqueness	A personal expense tracker app is where you can link your financial information to keep hold of expenses you are occurring. It helps in managing your everyday expenses or you can say it is an intermediary to keep track of your expenses without any need for third-party intervention.
4.	Social Impact / Customer Satisfaction	Customer satisfied With a personal expense tracker app, get a clear image of your spending and be your financial planner. As there are many expense tracker app free of cost, it is a simple decision to start using them.
5.	Business Model (Revenue Model)	Expense tracker apps put more of an emphasis on your spending. These apps usually categorize your expenses and help you get a good idea of your purchasing behavior
6.	Scalability of the Solution	Best graphical user interface, no buffering of information, low latency, Updating of daily tracking very quickly, friendly user interface.

3.4 Problem Solution fit

Problem-Solution Fit - this occurs when you have evidence that customers care about certain jobs, pains, and gains. At this stage you've proved the existence of a problem and have designed a value proposition that addresses your customers' jobs, pains and gains. Unfortunately you still do not have clear evidence that your customer really care enough about your value proposition enough to buy it.

1. CUSTOMER SEGMENT(S) CS The person who is busy and couldn't manage their expenses regularly and we will keep track of the expenses regularly and will notify them.	6. CUSTOMER LIMITATIONS CL <small>EG. BUDGET, DEVICES</small> Constant network connection	5. AVAILABLE SOLUTIONS AS <small>PROS & CONS</small> The application can be extended to include scanning of barcode on the price tag which decreases the effort of entering the data in the input fields. A notification system can be enabled in case when the expenses crosses over the income generated by the user to warn him or her about the situation.
2. PROBLEMS / PAINS PR <small>+ ITS FREQUENCY</small> Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert.	9. PROBLEM ROOT / CAUSE RC People think that their bank details might have sold to an unauthorised person. People think that their personal details might have sold	7. BEHAVIOR BE <small>+ ITS INTENSITY</small> The customer believes more in manual tracking of their expenditure rather than virtual tracking applications. The customer will exhibit this behaviour until an authenticated application serves it's purpose rightly.
3. TRIGGERS TO ACT TR The customer is triggered by their surrounding talking about the approach of tracking the expenses.	10. YOUR SOLUTION SL The proposed system makes a novel attempt to track the user expenses daily and if their expenses exceeds the fixed budget we will notify them through mail and user will get an analysed report. If the user spends large amount of money in a particular area continuously, we will notify them to reduce the spending in that particular area.	8. CHANNELS of BEHAVIOR CH ONLINE The customer will exhibit this behaviour until an authenticated application serves it's purpose rightly.
4. EMOTIONS EM <small>BEFORE / AFTER</small> BEFORE: Fear of spending lot of money and couldn't manage their expenses. AFTER: They can manage their expense regularly.		OFFLINE Maintain a separate diary, note the expenses at the moment and calculate the daily expenses at the end of the day.

REQUIREMENT ANALYSIS

What is Requirement Analysis: It is the process of determining user expectations for a system under consideration.

These should be quantifiable and detailed.

Requirement Analysis:

- Serves as a foundation for test plans and project plan
- Serves as an agreement between developer and customer
- Process to make stated and unstated requirements clear
- Process to validate requirement for completeness, ambiguity and feasibility.

4.1 Functional requirement

Functional requirements specify what a system should be able to do through computations, technical details, data manipulation and processing, and other specialised functions. Use cases, which are used to represent behavioural requirements, explain all the instances in which the system makes use of the functional requirements. Non-functional requirements, commonly referred to as "quality requirements," which place restrictions on the design or execution, support functional requirements (such as performance requirements, security, or reliability). Non-functional requirements often take the form "system shall be," while functional needs are typically articulated in the form "system must do." While non-functional needs are defined in the system architecture, the plan for accomplishing functional requirements is detailed in the system design. Functional requirements, as used in requirements engineering, outline specified outcomes of a system.

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. So, it's important to make them clear both for the development team and the stakeholders.

Generally, functional requirements describe system behaviour under specific conditions. For example:

- The system sends an approval request after the user enters personal information.
- A search feature allows a user to hunt among various invoices if they want to credit an issued invoice.
- The system sends a confirmation email when a new user account is created.

4.2 Non- Functional requirement

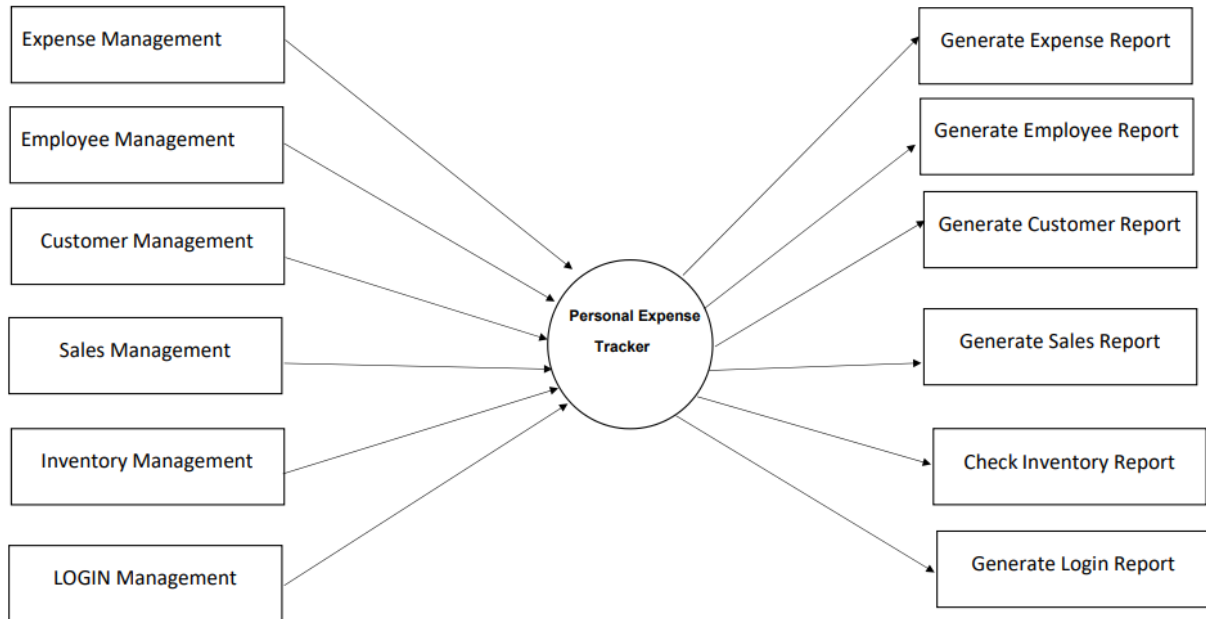
In general, non-functional requirements outline what a system is supposed to be rather than what it should be able to perform. Functional requirements are typically expressed as "system shall do," an individual action or component of the system, maybe explicitly in terms of a mathematical function, or as a black box description of an input, output, process, and control functional model, also known as an IPO Model. Non-functional requirements, on the other hand, have the form of "system shall be," which refers to a general characteristic of the system as a whole or of a particular aspect rather than a specific function. The overall characteristics of the system frequently determine whether a development project is a success or a failure.

Non-functional requirements are frequently referred to as a product's "quality traits" in error.

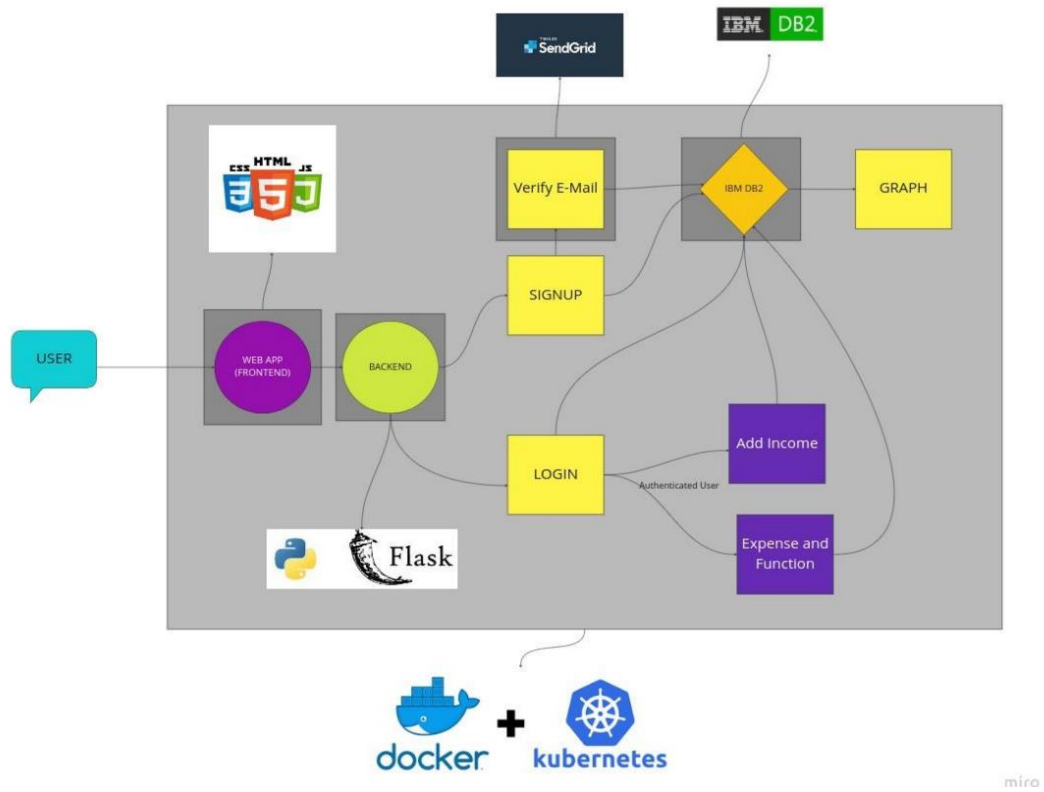
- Non-functional requirements, not related to the system functionality, rather define how the system should perform. Some examples are:
- The website pages should load in 3 seconds with the total number of simultaneous users <5 thousand.
- The system should be able to handle 20 million users without performance deterioration.
- Here's a brief comparison and then we'll proceed to a more in-depth explanation of each group.

PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



Technology Architecture

S.No	Components	Description	Technology
1	User Interface	User interacts with WebUI	HTML,CSS,JavaScript
2	Application Logic-1	The application contains the sign-in/sign-up logic and login to the dashboard	Java/Python
3	Application Logic-2	To send the email	SendGrid
4	Database	To store the data	MySQL
5	Cloud Database	Database Service on Cloud	IBM DB2
6	File Storage	To keep track of expenses	IBM Block Storage

Application Characteristics

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Flask Framework in Python is used	Python-Flask
2.	Security Implementations	This Application Provides high security to the user Financial data. It can be done by using the Container Registry in IBM cloud	ContainerRegistry, Kubernetes Cluster
3.	Scalable Architecture	It provide no isolate the internal code dependencies	Python
4.	Availability	Runs everywhere and user friendly	Docker
5.	Performance	Orchestrate containerized application to run the cluster of hosts	Kubernets

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a customer, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	login	USN-2	As a customer, I can login to the application by entering correct email and password.	I can access my account/dashboard.	High	Sprint-1
	Dashboard	USN-3	As a customer, I can see all the orders raised by me.	I get all the info needed in my dashboard.	Low	Sprint-2
	Order creation	USN-4	As a customer, I can place my order with the detailed description of my query	I can ask my query	Medium	Sprint-2
	Address Column	USN-5	As a customer, I can have conversations with the assigned agent and get my queries clarified	My queries are clarified.	High	Sprint-3
	Forgot password	USN-6	As a customer, I can reset my password by this option incase I forgot my old password.	I get access to my account again	Medium	Sprint-4
Agent (web user)	Order details	USN-7	As a Customer ,I can see the current stats of order.	I get abetter understanding	Medium	Sprint-4
	Login	USN-1	As an agent I can login to the application by entering Correct email and password.	I can access my account / dashboard.	High	Sprint-3
	Dashboard	USN-2	As an agent, I can see the order details assigned to me by admin.	I can see the tickets to which I could answer.	High	Sprint-3
	Address column	USN-3	As an agent, I get to have conversations with the customer and clear his/er dobuts	I can clarify the issues.	High	Sprint-3
	Forgot password	USN-4	As an agent I can reset my password by this option in case I forgot my old password.	I get access to my account again.	Medium	Sprint-4

Admin (Mobile user)	Login	USN-1	As a admin, I can login to the appliaction by entering Correct email and password	I can access my account/dashboard	High	Sprint-1
	Dashboard	USN-2	As an admin I can see all the orders raised in the entire system and lot more	I can assign agents by seeing those order.	High	Sprint-1
	Agent creation	USN-3	As an admin I can create an agent for clarifying the customers queries	I can create agents.	High	Sprint-2
	Assignment agent	USN-4	As an admin I can assign an agent for each order created by the customer.	Enable agent to clarify the queries.	High	Sprint-1
	Forgot password	USN-5	As an admin I can reset my password by this option in case I forgot my old password.	I get access to my account.	High	Sprint-1

PROJECT PLANNING & SCHEDULING

‘Project Planning and Scheduling’, though separate, are two sides of the same coin in project management. Fundamentally, ‘Project planning’ is all about choosing and designing effective policies and methodologies to attain project objectives. While ‘Project scheduling’ is a procedure of assigning tasks to get them completed by allocating appropriate resources within an estimated budget and time-frame. The basis of project planning is the entire project. Unlikely, project scheduling focuses only on the project-related tasks, the project start/end dates and project dependencies. Thus, a ‘project plan’ is a comprehensive document that contains the project aims, scope, costing, risks, and schedule. And a project schedule includes the estimated dates and sequential project tasks to be executed.

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	5	High	Nitish
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	2	Low	Akash
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	4	High	Gokul Nath
Sprint-2	Dashboard	USN-5	As a user, I can view my expenses in the web UI shows weekly expense	5	High	Akash
Sprint-2		USN-6	As a User, I can to see my expenses as a Graph based on the predefined categories	10	High	Nitish
Sprint-3	Add Expense or Income	USN-7	As a user, I can add Expenses and income.	5	High	Rahavan
Sprint-3		USN-8	Modify expenses and Income	5	Medium	Akash
Sprint-3	Maximum limit	USN-9	As a user, I want to set an upper limit in the Expenses.	3	High	Nitish
Sprint-3		USN-10	As a user, I want to receive a email when my expenses exceed the limit	3	High	Gokul Nath
Sprint-4	User Profile	USN-4	As a user, I can edit my details and change my password	4	High	Rahavan
Sprint-4	Chatbot	USN-11	As a User, it will be helpful to have a interactive Chatbot	4	Low	Gokul Nath

6.2 Sprint Delivery Schedule

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	11	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	15	
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	16	
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	8	

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)

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

CODING & SOLUTIONING

7.1 Feature 1

Feature 1: Add Expense

Feature 2: Update Expense

Feature 3: Delete Expense Feature 4: Set Limit

Feature 5: Send Alert Emails to users

7.2 Other Features

Track your expenses anywhere, anytime. Seamlessly manage your money and budget without any financial paperwork. Just click and submit your invoices and expenditures. Access, submit, and approve invoices irrespective of time and location. Avoid data loss by scanning your tickets and bills and saving in the app. Approval of bills and expenditures in real-time and get notified instantly. Quick settlement of claims and reduced human errors with an automated and streamlined billing process.

Code

```
from flask import (
    Flask,
    render_template,
    send_file,
    request,
    redirect,
    url_for,
    session,
    flash,
)

import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
import ibm_db
import re
from matplotlib import pyplot as plt
```

```

from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
from matplotlib.figure import Figure
from io import BytesIO

app = Flask(__name__)
app.secret_key = "Zenik"

conn = ibm_db.connect(
    "DATABASE=bludb;HOSTNAME=2f3279a5-73d1-4859-88f0-
a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=30756;SECURITY=S
SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;
UID=dwd46636;PWD=vPVuK2TjmxDVRVeA",'',''
)

@app.route("/", methods=["POST", "GET"])
@app.route("/home")
def home():
    return render_template("home.html")

@app.route("/login", methods=["GET", "POST"])
def login():
    msg = ""
    if request.method == "POST":
        username = request.form["username"]
        password = request.form["password"]
        print(username,password)
        sql = "SELECT * FROM REGISTER WHERE email=? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session['loggedin'] = True
            session['id'] = account['EMAIL']
            session["email"] = account["EMAIL"]
            session['username'] = account['USERNAME']
            msg = 'Logged in successfully !'

            return redirect("/dashboard")
        else:

```

```

        msg = "Incorrect login credentials"
        flash(msg)
        return render_template("login.html", title="Login")

@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"]
        password1 = request.form["password1"]
        sql = "SELECT * FROM REGISTER WHERE username =? or email=? "
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)

        if account:
            msg = "Account already exists"
        elif password1 != password:
            msg = "re-entered password doesnt match"
        elif not re.match(r"[A-Za-z0-9]+", username):
            msg = "Username should be only alphabets and numbers"
        else:
            sql = "INSERT INTO REGISTER(USERNAME,EMAIL,PASSWORD) VALUES(?,?,?)"
            stmt = ibm_db.prepare(conn, sql)
            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)

            message = Mail(
                from_email=os.environ.get('MAIL_DEFAULT_SENDER'),
                to_emails=email,
                subject='Welcome',
                html_content='<p>Hello, Your Registration was successfull.  
<br><br> Thank you for choosing us.</p>')
            try:
                sg = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
                response = sg.send(message)

```

```

        print(response.status_code)

    except Exception as e:
        print(e.message)

    return redirect("/login")

flash(msg)
return render_template("register.html", title="Register")

@app.route("/logout")
def logout():
    session.clear()
    return redirect("/")

def isLoggedIn():
    return session["Loggedin"]

@app.route("/dashboard")
def dashboard():
    if isLoggedIn:
        return render_template("dashboard.html", title="Dashboard")
    else:
        flash("Login to go to dashboard")
        return redirect("/login")

@app.route("/changePassword/", methods=["POST", "GET"])
def changePassword():
    msg = "Enter the new password"
    if request.method == "POST":
        pass1 = request.form["pass1"]
        pass2 = request.form["pass2"]
        if pass1 == pass2:
            sql = "UPDATE REGISTER SET password=?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, pass1)
            # ibm_db.bind_param(stmt, 2, session["id"])
            if ibm_db.execute(stmt):
                msg = "Successfully Changed Password!!!!"

        else:

```

```

        msg = "Passwords not equal"
    flash(msg)
    return redirect(url_for("dashboard"))

@app.route("/changeBudget/", methods=["POST", "GET"])
def changeBudget():
    msg = "Enter the new budget"
    if request.method == "POST":
        budgetAmount = request.form["budgetAmount"]
        sql = "UPDATE BUDGETS SET maxBudget=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, budgetAmount)
        # ibm_db.bind_param(stmt, 2, session["id"])
        if ibm_db.execute(stmt):
            session["budget"] = budgetAmount
            msg = "Successfully Changed Budget!!!"
        else:
            msg = "Budget not changed"
    flash(msg)
    return redirect(url_for("dashboard"))

@app.route("/addBudget/", methods=["POST", "GET"])
def addBudget():
    msg = "Enter the budget"
    if request.method == "POST":
        budgetAmount = request.form["budgetAmountToAdd"]
        sql = "INSERT INTO BUDGETS(id,maxbudget) VALUES(?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, session["id"])
        ibm_db.bind_param(stmt, 2, budgetAmount)
        if ibm_db.execute(stmt):
            session["budget"] = budgetAmount
            msg = "Successfully Set The Budget!!!"
        else:
            msg = "Budget not set yet"
    flash(msg)
    return redirect(url_for("dashboard"))

def fetchall(stmt):
    ibm_db.bind_param(stmt, 1, session["id"])
    ibm_db.execute(stmt)
    results = []

```

```

        result_dict = ibm_db.fetch_assoc(stmt)
        results.append(result_dict)
        while result_dict is not False:
            result_dict = ibm_db.fetch_assoc(stmt)
            results.append(result_dict)
        results.pop()
        return results

def getTotal(table):
    sql = "SELECT SUM(AMOUNT) FROM " + table + " where USER_ID=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, session["id"])
    ibm_db.execute(stmt)
    result = ibm_db.fetch_assoc(stmt)
    print(result)
    return result["1"]

@app.route("/logtoday")
def logtoday():
    # if isLogged():
        sql = "SELECT AMOUNT,CATEGORY,NEED FROM TRANSACTIONS WHERE USER_ID=? AND DATEADDED=CURRENT_DATE"
        stmt = ibm_db.prepare(conn, sql)
        expenseData = fetchall(stmt)
        print(expenseData)
        expenseTotal = getTotal("TRANSACTIONS")
        sql = "SELECT AMOUNT FROM income WHERE ID=? AND DATEADDED=CURRENT_DATE"
        stmt = ibm_db.prepare(conn, sql)
        incomeData = fetchall(stmt)
        print(incomeData)
        return render_template(
            "logtoday.html",
            title="Today's Log",
            expenseData=expenseData,
            incomeData=incomeData,
            expenseTotal=expenseTotal,
        )
    # else:
    #     flash("Login First")
    #     return redirect("/login")

@app.route("/addExpense/", methods=["POST", "GET"])

```

```

def addExpense():
    msg = ""
    if request.method == "POST":
        amount = request.form["Amount"]
        need = request.form["Need/Want"]
        category = request.form["category"]
        sql = "INSERT INTO TRANSACTIONS(USER_ID,AMOUNT,NEED,CATEGORY,DATEADDED)
VALUES(?,?,?,?,CURRENT_DATE)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, session["id"])
        ibm_db.bind_param(stmt, 2, amount)
        ibm_db.bind_param(stmt, 3, need)
        ibm_db.bind_param(stmt, 4, category)
        if ibm_db.execute(stmt):
            msg = "Successfully Added Expense!!!"
        else:
            msg = "Expense not added"

    flash(msg)
    return redirect(url_for("logtoday"))
# return redirect(url_for('logtoday'))

@app.route("/addIncome/", methods=["POST", "GET"])
def addIncome():
    msg = ""
    if request.method == "POST":
        amount = request.form["AmountIncome"]
        sql = "INSERT INTO INCOME(ID,AMOUNT,DATEADDED) VALUES(?,?,CURRENT_DATE)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, session["id"])
        ibm_db.bind_param(stmt, 2, amount)
        if ibm_db.execute(stmt):
            msg = "Successfully Added Income!!!"
        else:
            msg = "Income not added"

    flash(msg)
    return redirect(url_for("logtoday"))

# @app.route("/Edit")
###Visualization functions

```

```

@app.route("/reports")
def reports():
    return render_template("reports.html", title="Reports")

@app.route("/needVwant/")
def needVwant():
    sql = "SELECT Sum(amount) AS amount, need FROM transactions WHERE
DAYS(CURRENT_DATE)-DAYS(DATEADDED)<29 AND user_id = ? GROUP BY NEED ORDER BY
need"
    stmt = ibm_db.prepare(conn, sql)
    transactions = fetchall(stmt)
    values = []
    labels = []
    print(transactions)
    for transaction in transactions:
        values.append(transaction["AMOUNT"])
        labels.append(transaction["NEED"])
    fig = plt.figure(figsize=(10, 7))
    plt.pie(values)
    plt.title("Need v Want")
    plt.legend(["WANT", "NEED"])
    canvas = FigureCanvas(fig)
    img = BytesIO()
    fig.savefig(img)
    img.seek(0)
    return send_file(img, mimetype="image/png")

@app.route("/categoriesChart/")
def categoriesChart():
    sql = "SELECT Sum(amount) AS amount, category FROM transactions WHERE
DAYS(CURRENT_DATE)-DAYS(DATEADDED)<29 AND user_id = ? GROUP BY category ORDER BY
category"
    stmt = ibm_db.prepare(conn, sql)
    transactions = fetchall(stmt)
    values = []
    labels = []
    print(transactions)
    for transaction in transactions:
        values.append(transaction["AMOUNT"])
        labels.append(transaction["CATEGORY"])
    fig = plt.figure(figsize=(10, 7))
    plt.pie(values, labels=labels)
    plt.title("Categories")

```



```

plt.legend()
canvas = FigureCanvas(fig)
img = BytesIO()
fig.savefig(img)
img.seek(0)
return send_file(img, mimetype="image/png")

##edit the legend... all visualizations workkkkkkk!!!!!!
@app.route("/dailyLineChart/")
def dailyLineChart():
    sql = "SELECT Sum(amount) AS amount, DAY(dateadded) as dateadded FROM
transactions WHERE DAYS(CURRENT_DATE)-DAYS(DATEADDED)<29 AND user_id = ? GROUP
BY dateadded ORDER BY dateadded"
    stmt = ibm_db.prepare(conn, sql)
    transactions = fetchall(stmt)
    x = []
    y = []
    print(transactions)
    for transaction in transactions:
        y.append(transaction["AMOUNT"])
        x.append(transaction["DATEADDED"])
    ##get budget
    sql = "SELECT MAXBUDGET FROM budgets WHERE id = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, session["id"])
    ibm_db.execute(stmt)
    budget = ibm_db.fetch_assoc(stmt)
    print(budget)
    fig = plt.figure(figsize=(10, 7))
    plt.scatter(x, y)
    plt.plot(x, y, "-")
    if budget:
        plt.axhline(y=budget["MAXBUDGET"], color="r", linestyle="-")
    plt.xlabel("Day")
    plt.ylabel("Transaction")
    plt.title("Daily")
    plt.legend()
    canvas = FigureCanvas(fig)
    img = BytesIO()
    fig.savefig(img)
    img.seek(0)
    return send_file(img, mimetype="image/png")

```

```
# if __name__ == "__main__":
#     app.debug = True
#     app.run()

if __name__=="__main__":
    port = int(os.environ.get('PORT',5000))
    app.run(port=port,host="0.0.0.0")
```

TESTING:

8.1 Test Cases

- Login Page (Functional)
- Login Page (UI)
- Expense Page (Functional)

8.2 User Acceptance Testing:

1.Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of [product name] project time of the release to user acceptance testing (UAT)

2.Defect Analysis

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

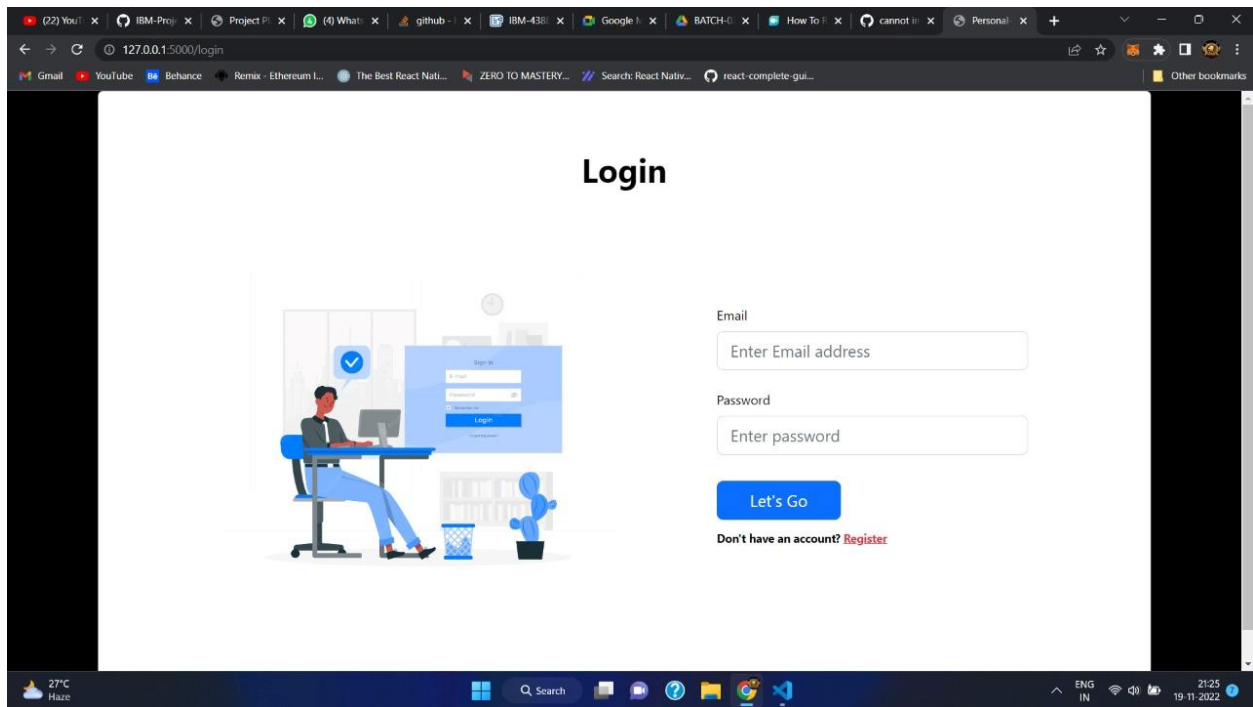
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	8	15
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	9	2	4	11	20
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	0	1	8
Totals	22	14	11	22	51

3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Interface	7	0	0	7
Login	43	0	0	43
Logout	2	0	0	2
Limit	3	0	0	3

Login page:



Signing page:

Register

Your Username

Your Email

Password

Repeat your password

[Get Started](#)

Have an account? [Log In](#)

Dashboard page:

Budget Tracker Dashboard Log's Today Reports Log Out

Dashboard

Welcome Gokul Nath

Name : Gokul Nath
Email : gokul.it19@mamcet.com
Budget :

[Change Password](#) [Add Budget](#)

Docker page:

The screenshot shows a Docker Playground interface with a container named `cds69bu0_cds69g60qau000ahdqpg`. The container's IP is `192.168.0.13` and port `5000` is open. The container's memory usage is `31.81% (1.243GiB / 3.906GiB)` and CPU usage is `0.17%`. The SSH command is `ssh ip172-18-0-18-cds69bu0qau000ahdp0@direct.labs.pla`. The terminal output shows the following:

```
* Tip: There are .env or .flaskenv files present. Do "pip install python-dotenv" to use them.
* Serving Flask app 'app' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://172.17.0.2:5000/ (Press CTRL+C to quit)
172.18.0.1 - - [19/Nov/2022 05:10:45] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [19/Nov/2022 05:10:45] "GET /index.css HTTP/1.1" 404 -
172.18.0.1 - - [19/Nov/2022 05:10:45] "GET /static/styles.css HTTP/1.1" 200 -
172.18.0.1 - - [19/Nov/2022 05:10:47] "GET /favicon.ico HTTP/1.1" 404 -
172.18.0.1 - - [19/Nov/2022 06:26:39] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [19/Nov/2022 06:26:39] "GET /static/styles.css HTTP/1.1" 304 -
172.18.0.1 - - [19/Nov/2022 06:26:39] "GET /index.css HTTP/1.1" 404 -
```

LogToday page:

The screenshot shows the LogToday web application interface. The navigation bar includes `Budget Tracker`, `Dashboard`, `Log's Today`, `Reports`, and `Log Out`. The main heading is `Today's Log`. The `Expenses` section displays a table with the following data:

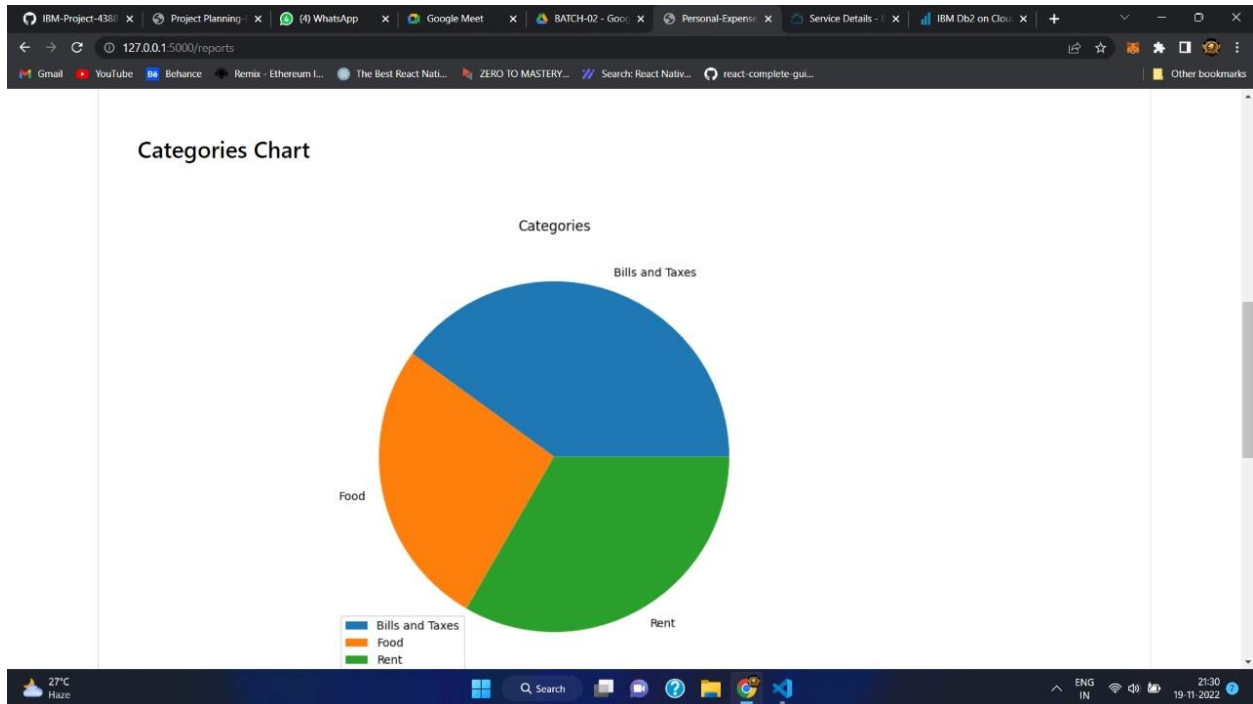
AMOUNT	CATEGORY	NEED
2000	Food	TRUE
2500	Rent	FALSE
3000	Bills and Taxes	FALSE

The total expense is `Total : 7500`. The `Income` section displays a table with the following data:

AMOUNT
500000

At the bottom, there are buttons for `Add Expense` and `Add Income`.

Reports page:



Database:

The screenshot shows the IBM Db2 on Cloud console. The left sidebar displays a list of tables, with 'SIGNUP' selected. The main area shows the 'Table definition' for the 'SIGNUP' table, which is located in the 'DWD46636' schema. The table has four columns: 'USERNAME' (CHAR, 15, nullable), 'Roll Number' (INTEGER, nullable), 'EMAIL' (VARCHAR, 32, nullable), and 'PASSWORD' (VARCHAR, 32, nullable). A 'View data' button is visible at the bottom right of the table definition panel.

Name	Data type	Nullable	Length	Scale
USERNAME	CHAR	Y	15	0
Roll Number	INTEGER	Y		0
EMAIL	VARCHAR	Y	32	0
PASSWORD	VARCHAR	Y	32	0

ADVANTAGES:

One of the major pros of tracking spending is always being aware of the state of one's personal finances. Tracking what you spend can help you stick to your budget, not just in a general way, but in each category such as housing, food, transportation and gifts. While a con is that manually tracking all cash that is spent can be irritating as well as time consuming, a pro is that doing this automatically can be quick and simple. Another pro is that many automatic spending tracking software programs are available for free.

DISADVANTAGES:

A con with any system used to track spending is that one may start doing it then taper off until it's forgotten about all together. Yet, this is a risk for any new goal such as trying to lose weight or quit smoking. If a person first makes a budget plan, then places money in savings before spending any each new pay period or month, the tracking goal can help. In this way, tracking spending and making sure all receipts are accounted for only needs to be done once or twice a month. Even with constant tracking of one's spending habits, there is no guarantee that financial goals will be met. Although this can be considered to be a con of tracking spending, it could be changed into a pro if one makes up his or her mind to keep trying to properly manage all finances.

CONCLUSION

A comprehensive money management strategy requires clarity and conviction for decision-making. You will need a defined goal and a clear vision for grasping the business and personal finances. That's when an expense tracking app comes into the picture. An expense tracking app is an exclusive suite of services for people who seek to handle their earnings and plan their expenses and savings efficiently. It helps you track all transactions like bills, refunds, payrolls, receipts, taxes, etc., on a daily, weekly, and monthly basis.

FUTURE SCOPE

- Achieve your business goals with a tailored mobile app that perfectly fits your business.
- Scale-up at the pace your business is growing.
- Deliver an outstanding customer experience through additional control over the app.
- Control the security of your business and customer data.
- Open direct marketing channels with no extra costs with methods such as push notifications.
- Boost the productivity of all the processes within the organization.
- Increase efficiency and customer satisfaction with an app aligned to their needs.
- Seamlessly integrate with existing infrastructure.
- Ability to provide valuable insights.
- Optimize sales processes to generate more revenue through enhanced data collection.
- Chats: Equip your expense tracking app with a bot that can understand and answer all user queries and address their needs such as account balance, credit score, etc.
- Prediction: With the help of AI, your mobile app can predict your next purchase, according to your spending behavior. Moreover, it can recommend products and provide unique insights on saving money. It brings out the factors causing fluctuations in your expenses.