

**ALAGAPPA CHETTIAR GOVERNMENT COLLEGE OF  
ENGINEERING AND TECHNOLOGY**

**(An Autonomous Institution Affiliated to Anna University, Chennai)**

**KARAIKUDI – 630003**

**PROFESSIONAL READINESS FOR INNOVATION  
EMPLOYABILITY AND ENTREPRENEURSHIP**



**IBM PROJECT REPORT**

Submitted by

**Team ID: PNT2022TMID06088**

KARTHICK RAJA R	91761915011
PRAKASHRAJA T	91762015208
SINTHANAISELVAN P	91762015210
VIGNESH R S	91762015213

In partial fulfillment for the award of the degree  
Of  
**BACHELOR OF ENGINEERING**  
IN  
**COMPUTER SCIENCE AND ENGINEERING**  
**NOVEMBER 2022**

# **ALAGAPPA CHETTIAR GOVERNMENT COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(An Autonomous Institution Affiliated to Anna University, Chennai)**

**KARAIKUDI – 630003**

## **BONAFIDE CERTIFICATE**

Certified that this PROJECT REPORT “**PERSONAL EXPENSE TRACKER APPLICATION**” is the bonafide work of **KARTHICKRAJA R (91761915011)** **PRAKASHRAJA R (91762015208)** **SINTHANAISELVAN P (91762015210)** **VIGNESH RS (91762015213)** for **IBM NALAIYATHIRAN** in VII semester of B.E., degree course in Computer Science and Engineering branch during the academic year of 2022 - 2023.

**Staff-In charge**

**Dr.L.Rasikannan**

**Evaluator**

**Mrs.K.Chandrababha**

**Head of the Department**

**Mrs.K.Chandrababha**

## ACKNOWLEDGEMENT

We express our breathless thanks to our **Dr.P.K.PALANI, M.E, Ph.D**, the Dean, Alagappa Chettiar Government College Of Engineering And technology for giving constant motivation in succeeding in our goal.

We acknowledge our sincere thanks to Head of the Department **MRS.K.CHANDRAPRABHA, M.E** ,for giving us valuable suggestion and help towards us throughout thisProject.

We are highly grateful to thank our Project coordinator **DR.L.RASIKANNAN**, and our Project Evaluator **MRS.K.CHANDRAPRABHA, M.E**,Department of Computer Science and Engineering Alagappa Chettiar Government College Of Engineering And technology, for the coordinating us throughout this Project.

We are very much indebted to thank all the faculty members of Department of Computer science and Engineering in our Institute, for their excellent moral support and suggestions to complete our Project work successfully.

## **ABSTRACT**

For the project to succeed some resources are needed financial resources to start the project are the first one. This will involve the payment of the staff involved in the execution of the project. The second one is in buying the hardware equipment for installing the retail inventory programs.

The other resource needed is training the staff on how to use the retail inventory programs, as they are complicated and sensitive. The staff must be trained on security issues and policies governing the use of the retail inventory (King 2005).

It is also important for the company implementing these systems to have a policy governing the use of the systems to mitigate unethical use. It should spell out the disciplinary measures against anyone accessing the system illegally or leaving the systems insecure as this would provide an opportunity to distort the actual amount of the stock leading to loss and theft.

The major tasks in this project will be purchasing the Retail Inventory Management software from ERP vendors who will then use the software where the purchasing officer will do it. It will also involve the purchase of all the physical equipment needed for the success of this project.

This is because the software is meant for one-time use and once a client organization acquires the retail inventory management and integrates it into its day-to-day operations it becomes easy for the organization to work and improve on it such that it will not require any service from the vendor.

This ensures that the organization has well-integrated strategies to maintain and integrate the software in the operations. The software being a one-time sale means that diligence to ascertain the effectiveness and the capability of the software to meet the company's retail inventory needs is imperative.

The work of the purchasing team will be to assess the kind of Retail Inventory Management System needed and then bidding for a product that will meet those needs.

# TABLE OF CONTENTS

## 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

- 7.1 Feature 1
- 7.2 Feature 2

## 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

# **1.INTRODUCTION**

## **1.1. Project Overview**

Overview Personal Expense Tracker is a Windows store application to be designed for tracking daily personal and business expense and income related transaction. The main feature includes manage account book of different user, managing daily transaction, report generation, accounts management, supplier management, set limit for expense, etc. This application helps you to track all the expenses, incomes, accounts of different user and analyzing income/expense by generating reports. It is flexible and adaptive application suited to personal and business man.

In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management.

## **1.2. Purpose**

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.

## **2.LITERATURE SURVEY**

### **2.1.Existing problem**

Daily Expense is a simple application for Android devices. Writing in a user's pocket handbook is as convenient and easy as it is, because of this application, the user will be able to debit the account through his smart phone without any hassle under any circumstances. Users just need to enter revenue and expense as their revenue and speed, and the app calculates it for users. This application is very easy, fast and secure with money calculation and offline mode service.

#### **Related Works**

We looked through the Google play store and found some apps that are similar to our application but there are some features missing which this type of application need to have. So we combined all the features in our app and added some new features and make it useful with better UI.

Some related works are mentioned below with a short detail:

- Monefy Money Manager [1]: How to track your expenses successfully? We know that it's easy. You only need to add each expense you do... no more than that! And Monefy is going to help you. Just add new records when you are buying a coffee or taking a taxi. It's done in one click, because you don't need to fill anything except the amount. It has never been so quick and enjoyable!
- Expenses Tracker [2]: Expense Tracker - Money Manager & Budget can quickly and easily track your income and expenses, which will help you avoid making accidental expenses.
- Expense Manager [3]: Expense Manager is simple, intuitive, stable and feature-rich app that is just designed for you. Everything you need at your fingertips to manage the expenditures, chequebook and budgets.

#### **Comparative Studies**

We have studied some similar applications and find some problem these are not working in offline mode. There are some limitations to their application, which we do not have in our application. In some applications, there are no login and signup options, which is required for the

security of a user's information. However, in our application, we have a login and signup which is required for user data security and has a dashboard for monitoring the entire system.

This application is very simple and user-friendly application for the common people. The main goal of the project is to make the system offline and perform more tasks in short period of time.

## **2.2.References**

- [1] D. GRAZIANO, Gartner: Apple leads smartphone sales to new heights. BGR Media, <http://bgr.com/2012/02/15/gartner-apple-leads-smartphone-sales-to-new-heights/>, accessed October 2012, February 2012.
- [2] M. BROWNLOW, Smartphone statistics and market share. Email-Marketing-Reports, <http://www.email-marketing-reports.com/wireless-mobile/smartphone-statistics.htm>, accessed October 2012, October 2012.
- [3] THE WASHINGTON POST, iPhone 5 sales above 5 million, Apple says, surpassing iPhone 4S. The Washington Post, [http://www.washingtonpost.com/business/technology/iphone-5-sales-above-5-million-apple-says-surpassing-iphone-4s/2012/09/24/8b23322e-0648-11e2-afffd6c7f20a83bf\\_story.html](http://www.washingtonpost.com/business/technology/iphone-5-sales-above-5-million-apple-says-surpassing-iphone-4s/2012/09/24/8b23322e-0648-11e2-afffd6c7f20a83bf_story.html), accessed September 2012, September 2012.
- [4] C. HEATH AND J. B. SOLL, Mental budgeting and consumer decisions, J. Consumer Res., 23 (1996), pp. 40-52.
- [5] J. HASTINGS AND J. M. SHAPIRO, Mental accounting and consumer choice: Evidence from commodity price shocks. Unpublished report, 2012.
- [6] MINT, Homepage. Mint, <https://www.mint.com>, accessed October 2012, n.d.
- [7] APPLE, iOS technology overview. Apple, <http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html>, accessed October 2012, n.d.

## **2.3.Problem Statement Definition**

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love. A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

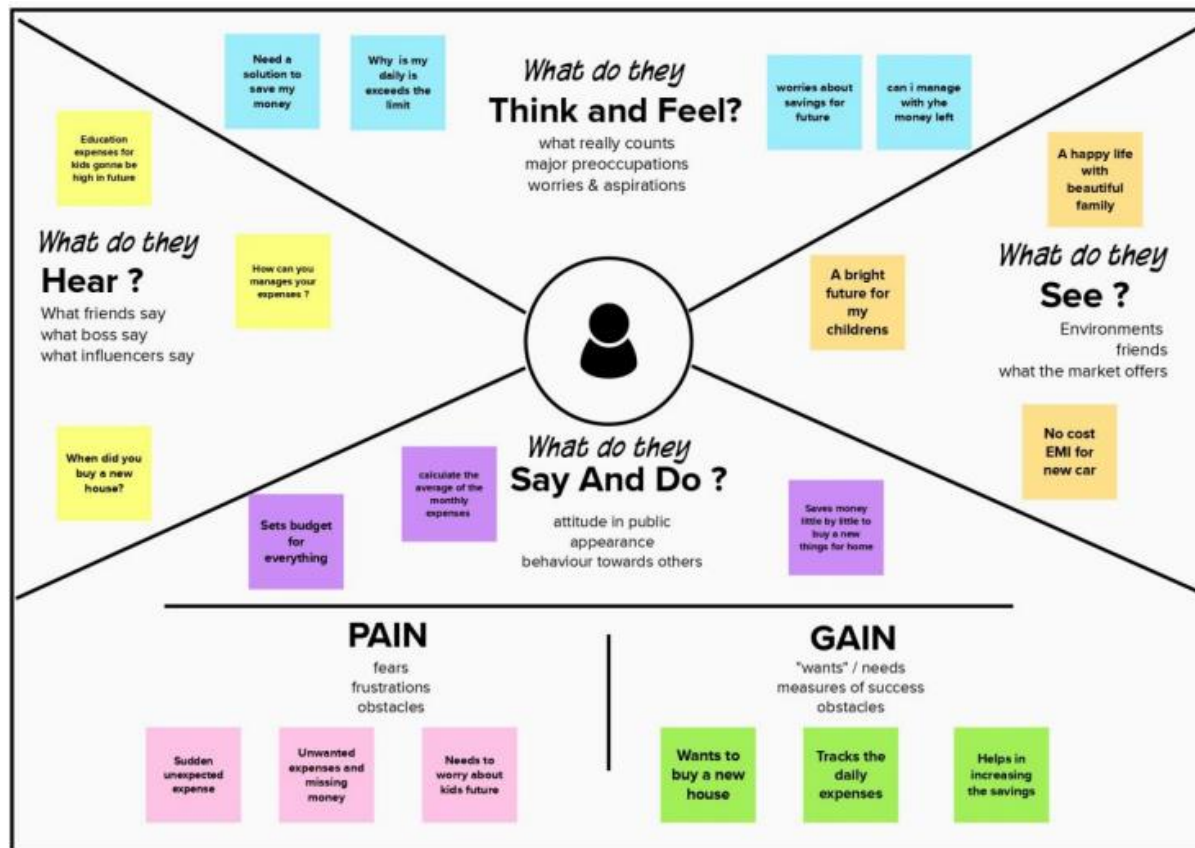




## 3.IDEATION & PROPOSED SOLUTION

### 3.1.Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



### **3.2.Ideation & Brainstorming**

Brainstorming provides a free and open environment that encourages everyone within a team

to participate in the creative thinking process that leads to problem solving. Prioritizing

volume over value, out-of-the-box ideas are welcome and built upon, and all participants are


encouraged to collaborate, helping each other develop a rich amount of creative solutions.

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.




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

Template




### 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 make the user to track expense and how might we define a remainder system for the user in simple way





#### Key rules of brainstorming

To run an smooth and productive session


 Stay in topic.

 Encourage wild ideas.

 Defer judgment.

 Listen to others.

 Go for volume.

 If possible, be visual.

## Step-2: Brainstorm, Idea Listing and Grouping

2

### Brainstorm

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

🕒 10 minutes

#### Karthick Raja



#### Sinthanaiselvan



#### Vignesh



#### Prakashraja

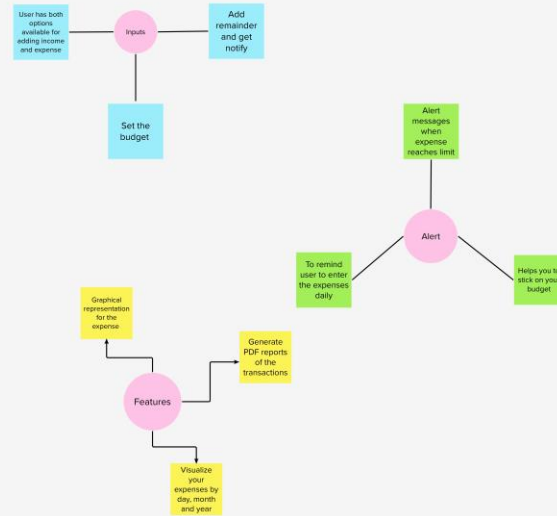


3

### 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 can break it up into smaller sub-groups.

🕒 20 minutes



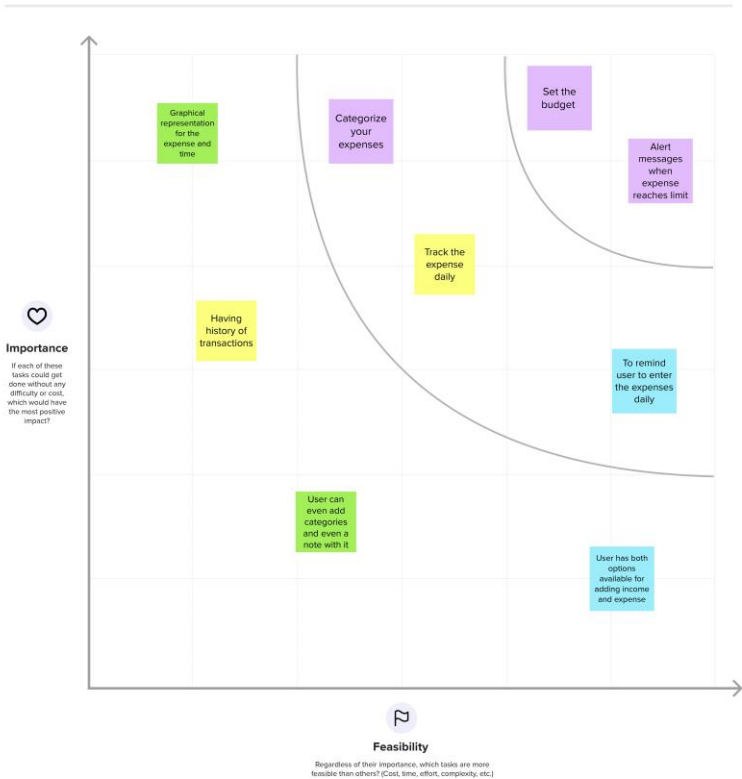
# Step-3: Idea Prioritization

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



→

## After you collaborate

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

### Quick add-ons

- Share the mural**  
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.
- Export the mural**  
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

### Keep moving forward

- Strategy blueprint**  
Define the components of a new idea or strategy.  
[Open the template →](#)
- Customer experience journey map**  
Understand customer needs, motivations, and obstacles for an experience.  
[Open the template →](#)
- Strengths, weaknesses, opportunities & threats**  
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.  
[Open the template →](#)

[Share template feedback](#)

### 3.3.Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	At the instant, there is no as such complete solution present easily or we should say free of cost which enables a person to keep a track of its daily expenditure easily. To do so a person has to keep a log in a diary or in a computer, also all the calculations needs to be done by the user which may sometimes results in errors leading to losses. Due to lack of a complete tracking system, there is a constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.
2.	Idea / Solution description	As the name itself suggests, this project is an attempt to manage our daily expenses in a more efficient and manageable way. The system attempts to free the user with as much as possible the burden of manual calculation and to keep the track of the expenditure. This system also eliminates sticky notes, bills.
3.	Novelty / Uniqueness	This personal expense tracker Application has features that enables the user to have an option to set a limit for the amount to be used. If the limit is exceeded the user will be notified with an Email and SMS alert.
4.	Social Impact / Customer Satisfaction	The user will be able to Stick to their Spending Limits. They can able to scan their bills anytime thus data loss is avoided. You might track expenses for a while just to get an idea of where your money's going, apps help you collect and classify your purchases so that you can identify areas that might be trimmed.
5.	Business Model (Revenue Model)	The user have to pay for the subscription of the premium version of the app. The premium version of the app gives the complete access to all the features and blocks the advertisement.
6.	Scalability of the Solution	Since this application is deployed on Cloud, it can handle multiple users at a time. With our application, the users can be able to manage their expenses more effectively.

### 3.4.Problem Solution fit

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why

<p><b>Define CS, fit into CC</b></p> <p><b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span> Who is your customer? i.e. working parents of 0-5 yrs. kids</p> <p>Peoples who wish to track their expenses like,</p> <ul style="list-style-type: none"> <li>Workers</li> <li>Student</li> </ul>	<p><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 services.</p> <ol style="list-style-type: none"> <li>Tracking future expenditure ideas.</li> <li>Data privacy and security.</li> <li>Adding Bank details.</li> </ol>	<p><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 used in the past? What pros &amp; cons do these solutions have? i.e. pen and paper is an alternative to digital note-taking</p> <p>Customer have to uses notes or paper to keep track their expenses or uses his mind.</p> <p><b>Explore AS, differentiate</b></p>
<p><b>Focus on J&amp;P, tap into BE, understand RC</b></p> <p><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</p> <ul style="list-style-type: none"> <li>People have to track their expenses regularly.</li> <li>They need to keep the receipts and bills to calculate the money spent.</li> <li>Also they need manual calculations</li> </ul>	<p><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.</p> <ol style="list-style-type: none"> <li>Due to lot of payments options, customers tends to forget when and where they spent.</li> <li>By tracking expenses they can save money and Time</li> </ol>	<p><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 time time on volunteering work (i.e. Greenpeace)</p> <ol style="list-style-type: none"> <li>Customer tries to keep track of their expenses on memory or note it in some papers and uses manual calculations. It may consumes time.</li> </ol> <p><b>Focus on J&amp;P, tap into BE, understand RC</b></p>
<p><b>Identify strong TR &amp; EM</b></p> <p><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.</p> <ol style="list-style-type: none"> <li>When customer spend too much and exceeds the budget he cannot save for future.</li> </ol> <p><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 -&gt; confident, in control - use it in your communication strategy &amp; design.</p> <p>BEFORE:    AFTER:</p> <ol style="list-style-type: none"> <li>Insecure    1. Peace</li> <li>Curious    2. Confident</li> <li>Doubtful    3. Happy</li> </ol>	<p><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.</p> <ol style="list-style-type: none"> <li>A Expenses Tracker App which keeps track of the expenses. This system attempts to free the user and avoids the manual calculations.</li> <li>No one alerts if their spending exceeds the budget.</li> </ol>	<p><b>8. CHANNELS of BEHAVIOUR</b> <span>CH</span></p> <p><b>8.1 ONLINE</b> What kind of actions do customers take online? Extract online channels from #?</p> <p><b>8.2 OFFLINE</b> What kind of actions do customers take offline? Extract offline channels from #? and use them for customer development</p> <ol style="list-style-type: none"> <li>Manual entry in Google sheets and keep tracks the history of the payments app.</li> <li>Also collects all the bills and receipts.</li> </ol> <p><b>Identify strong TR &amp; EM</b></p>



## 4.REQUIREMENT ANALYSIS

### 4.1.Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	Login	You need to enter your username and password here.
FR-3	Calendar	Table to add the information to their spending in a personal expense tracking application
FR-4	Expenct tracker	This application should graphically represent the expense in the form of report.
FR-5	Report generation	Graphical representation of report.
FR-6	Category	Users of this application will be able to add expense categories.

## 4.2.Non-functional Requirements

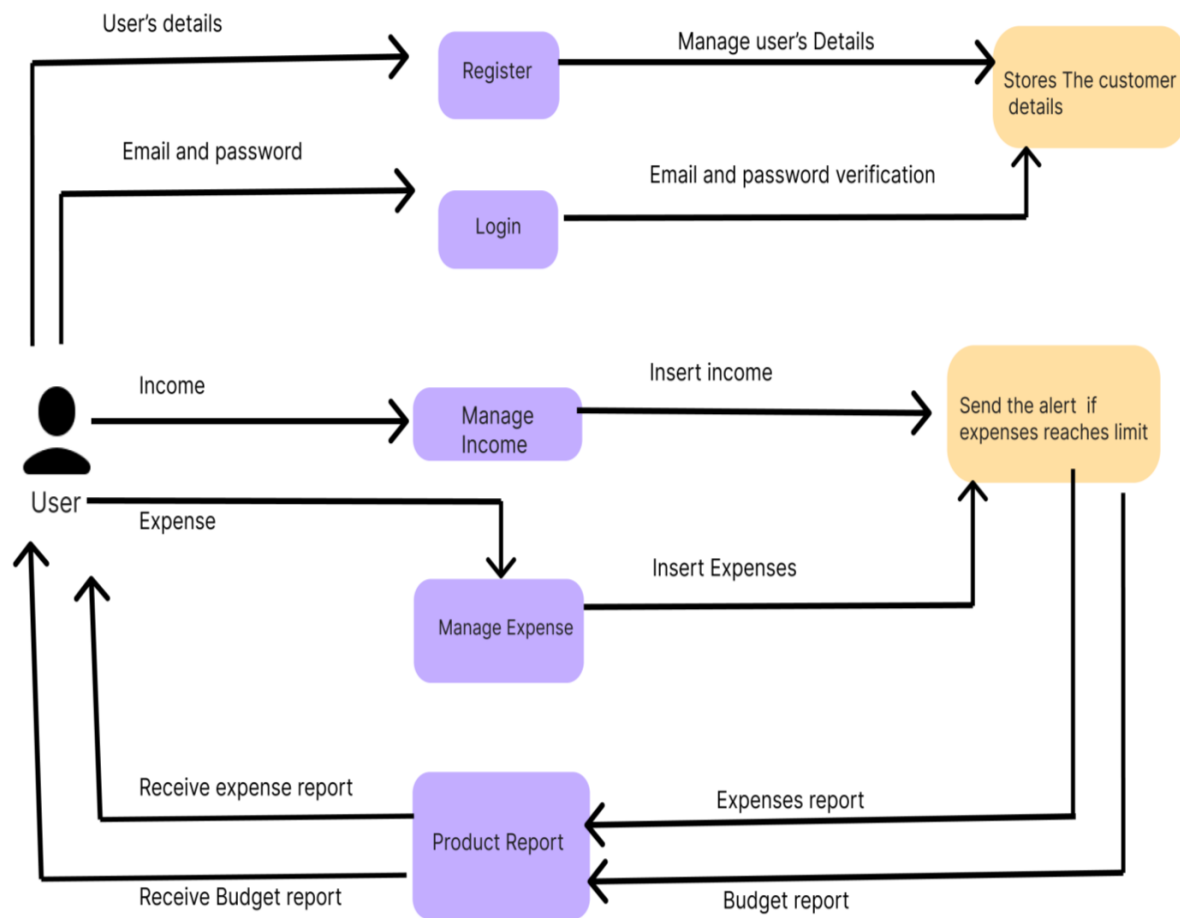
Following are the non-functional requirements of the proposed solution.

<b>FR No.</b>	<b>Non-Functional Requirement</b>	<b>Description</b>
NFR-1	<b>Usability</b>	Helps to keep an accurate record of your earnings and expenses.
NFR-2	<b>Security</b>	A detailed accounting of your income and expenses
NFR-3	<b>Reliability</b>	Each data record is stored on a well built efficient database schema. There is no risk of data of loss.
NFR-4	<b>Performance</b>	There are categories of expenses as well as an option. Because of lightweight database support, the system's throughput is increased.
NFR-5	<b>Availability</b>	The application must be completely operational at all times.
NFR-6	<b>Scalability</b>	The application must always function in its entirety

## 5.PROJECT DESIGN

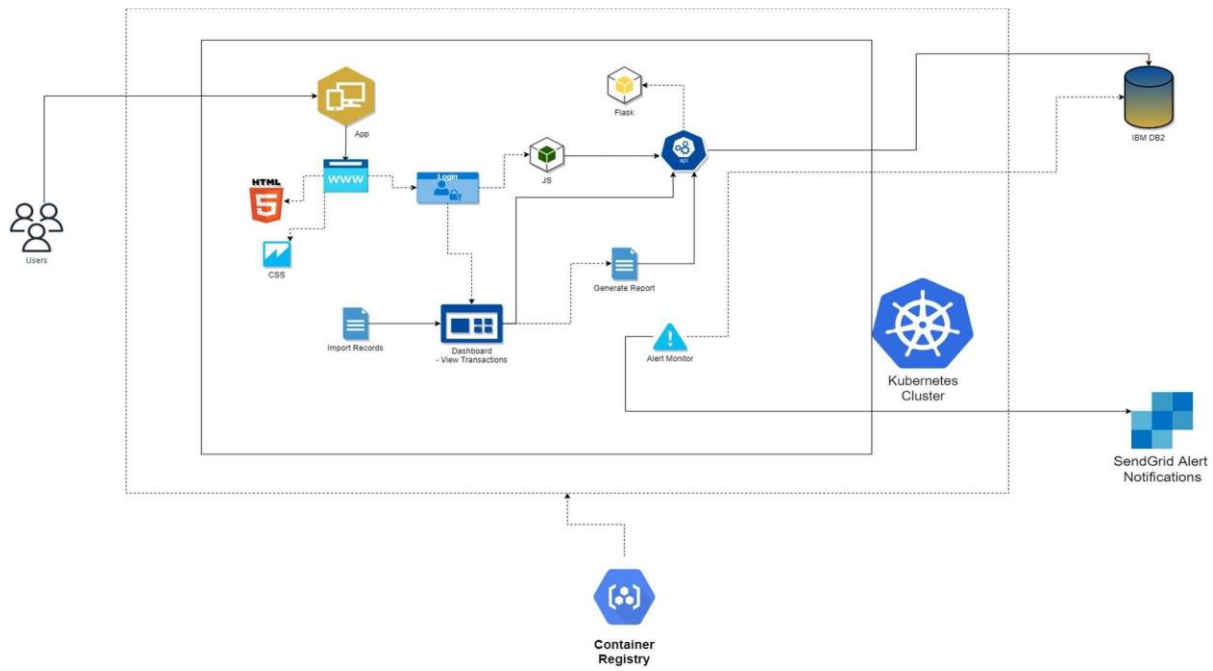
### 5.1.Data Flow Diagrams

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



## 5.2.Solution & Technical Architecture

### 5.2.1.Technical Architecture



**Table-1 : Components & Technologies:**

S. N o	Component	Description	Technology
1.	User Interface	How user interacts with application e.g.  Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript,  Bootstrap
2.	Application Logic-1	Logic for a process in the application	Python-Flask
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL
6.	Cloud Database	Database Service on Cloud	IBM DB2
7.	File Storage	File storage requirements	IBM Block Storage

8.	External API-1	Purpose of External API used in  the application	SendGrid API
9.	Infrastructure (Server /  Cloud)	Application Deployment on  Local  System / Cloud  Local Server Configuration:  Cloud Server Configuration :	Local Registry:  DockerHub  Cloud Registry:  Container  Registry  Cloud Server Configuration:  Kubernetes

**Table-2: Application Characteristics:**

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Flask Framework in Python is used to implement this Application.	Python-Flask
2.	Security Implementations	The user's financial information is extremely secure. IBM cloud's Container Registry can be used to accomplish this.	Container Registry, Kubernetes Cluster
3.	Scalable Architecture	This application 'Expense Tracker' has lifetime access. When a user's income is high, this product will be in more demand.	Container Registry, Kubernetes Cluster
4.	Availability	The user will have access to this application at any time.	Container Registry, Kubernetes Cluster

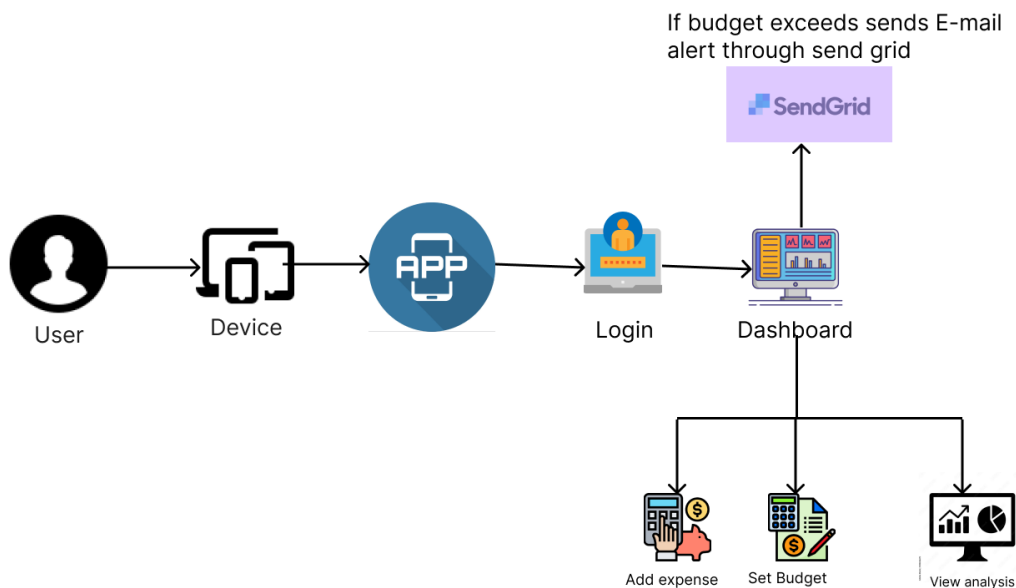
5.	Performance	<p>The performance will be high</p> <p>because there will be no</p> <p>network traffics in the</p> <p>application.</p>	<p>Container Registry,</p> <p>Kubernetes Cluster</p>
----	-------------	--	--



### 5.2.2.Solution Architecture

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

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



### 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 user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	
	Login	USN-2	As a user, I can log into the application by entering email & password	I can access the application	High	
	Dashboard	USN-3	As a user I can enter my income and expenditure details.	I can view my daily expenses	High	
Customer Care Executive		USN-4	As a customer care executive, I can solve the log in issues and other issues of the application.	I can provide support or solution at any time 24*7	Medium	
Administrator	Application	USN-5	As an administrator I can upgrade or update the application.	I can fix the bug which arises for the customers and users of the application	Medium	

## 6.PROJECT PLANNING & SCHEDULING

### 6.1.Sprint Planning & Estimation

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on theselected project & gathering information by referring the, technical papers, research publications etc.	09 October 2022
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the userPains & Gains, Prepare listof problem statements	09 October 2022
Ideation	List the by organizing thebrainstorming session andprioritize the top 3 ideas based on the feasibility &importance.	09 October 2022
Proposed Solution	Prepare the proposed solutiondocument, which includes thenovelty, feasibility of idea, business model, social impact,scalability of solution, etc.	15 October 2022
Problem Solution Fit	Prepare problem - solution fitdocument.	22 October 2022
Solution Architecture	Prepare solution architecturedocument.	22 October 2022

<b>Customer Journey</b>	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	10 November 2022
<b>Functional Requirement</b>	Prepare the functional requirement document.	25 October 2022
<b>Technology Architecture</b>	Prepare the technology architecture diagram.	10 November 2022
<b>Data Flow Diagrams</b>	Draw the data flow diagrams and submit for review.	10 November 2022
<b>Prepare Milestone &amp; Activity List</b>	Prepare the milestones & activity list of the project.	14 November 2022
<b>Project Development - Delivery of Sprint-1, 2, 3 &amp; 4</b>	Develop & submit the developed code by testing it.	14-19 November 2022

## 6.2.Sprint Delivery Schedule

Sprint planning is an event in scrum that kicks off the sprint. The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved. Sprint planning is done in collaboration with the whole scrum team.

### 6.2.1.Product Backlog, Sprint Schedule, and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Homepage	USN-1	AS a user I can view the index page to see the about of the Expense tracker	10	High	R. Karthick Raja
Sprint-1	Add expense	USN-2	As a User I will add my expense throughout the month I spend on	8	High	P. Sinthanaiselvan
Sprint-2	Login	USN-3	As a user, I need to login with user id and password to get in to the website	10	Medium	R.S. Vignesh
Sprint-2	Registration	USN-4	As a user, I can register for the application through Gmail	10	Medium	T. Prakash Raja
Sprint-3	Dashboard	USN-5	As a User, I will follow Co-Admin's instruction to reach the filling bin in short roots and save time	10	High	P. Sinthanaiselvan
Sprint-3	Total expense graph	USN-6	As a User I can view my expense in a graph of overview of the expense I spend.	5	Low	R.S. Vignesh
Sprint-4	Deployment in cloud	USN-7	As a User I can access the cloud to store my data of expense	15	High	R. Karthick Raja

### 6.2.2. 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	5 Days	10 Nov 2022	14 Nov 2022	20	15 Nov 2022
Sprint-2	20	5 Days	12 Nov 2022	15 Nov 2022	20	18 Nov 2022
Sprint-3	20	5 Days	15 Nov 2022	17 Nov 2022	20	18 Nov 2022
Sprint-4	20	5 Days	18 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)

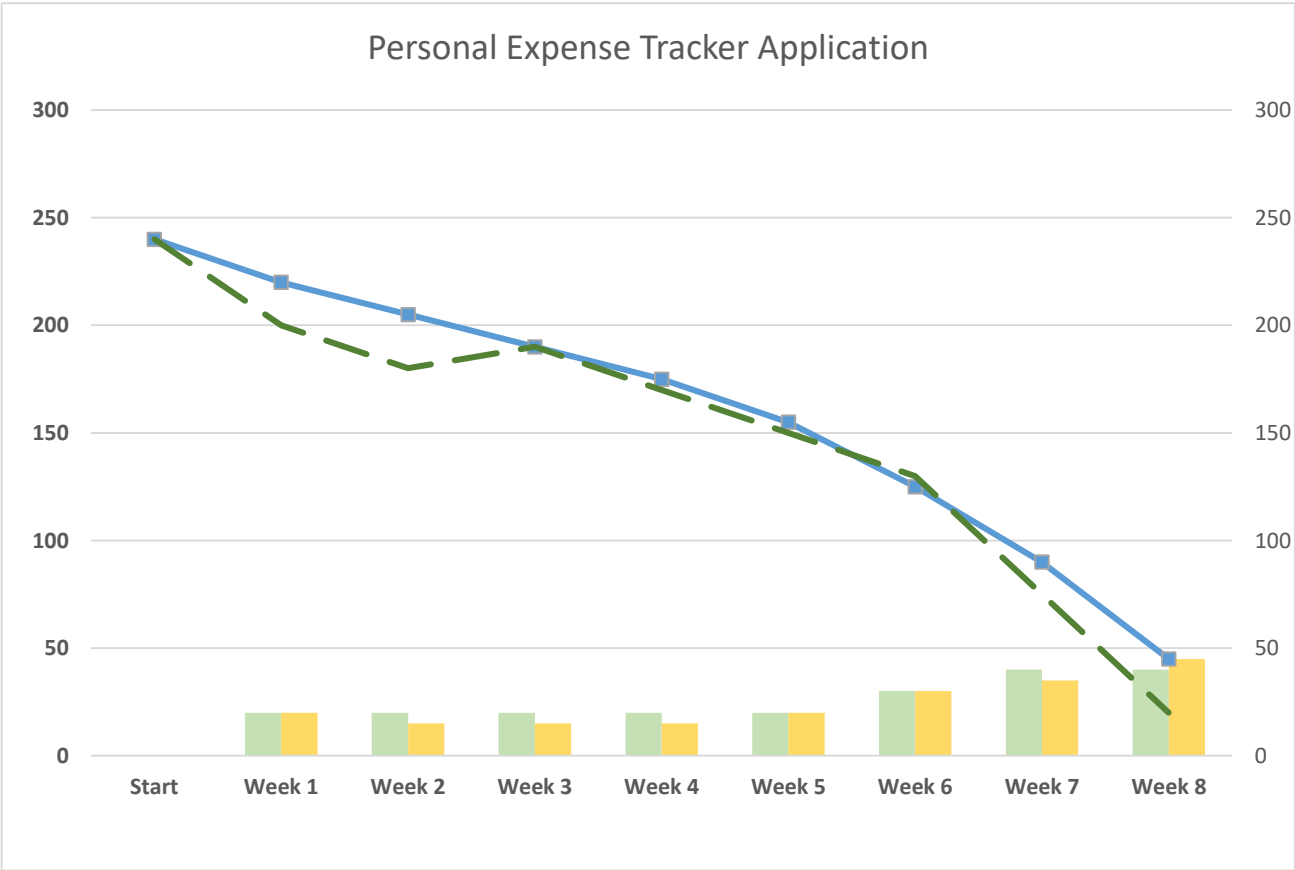
$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

Sprint	Story points	Duration	Average velocity
Sprint-1	20	5	4
Sprint-2	20	5	4
Sprint-3	20	5	4
Sprint-4	20	5	4
Total	80	20	16

Setting	Start	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8
Planned Hours		20	20	20	20	20	30	40	40
Actual Hours		20	15	15	15	20	30	35	45
Remaining Effort	240	220	205	190	175	155	125	90	45
Ideal Burndown	240	200	180	190	170	150	130	75	20

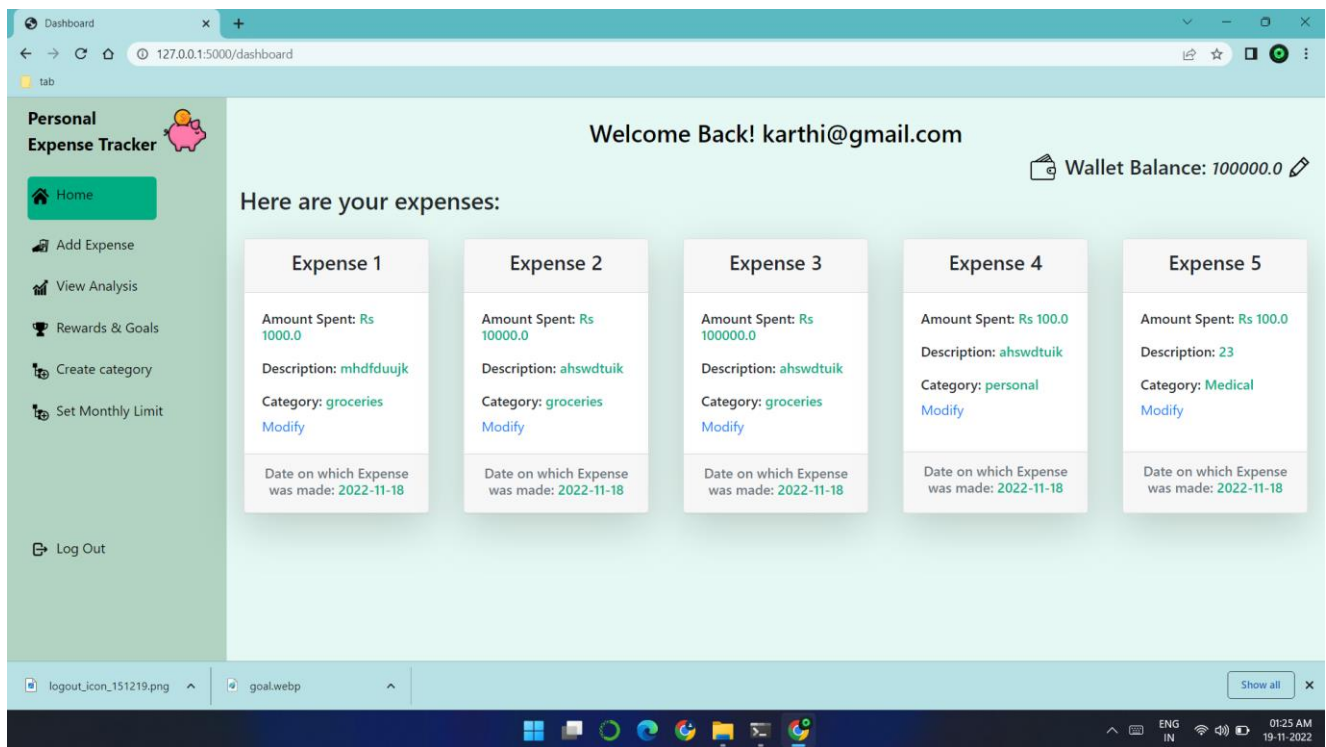
Feature	Initial Estimate	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Hours Left
Categories	60	20	8	5	1	5	5	5	1	10
Synchronization	60	10	5	2	2	5	5	15	10	6
Accounts	60	5	8	2	10	5	5	10	10	5
Reminders	60	10	12	2	3	5	5	5	10	8

**BURNDOWN CHART:**

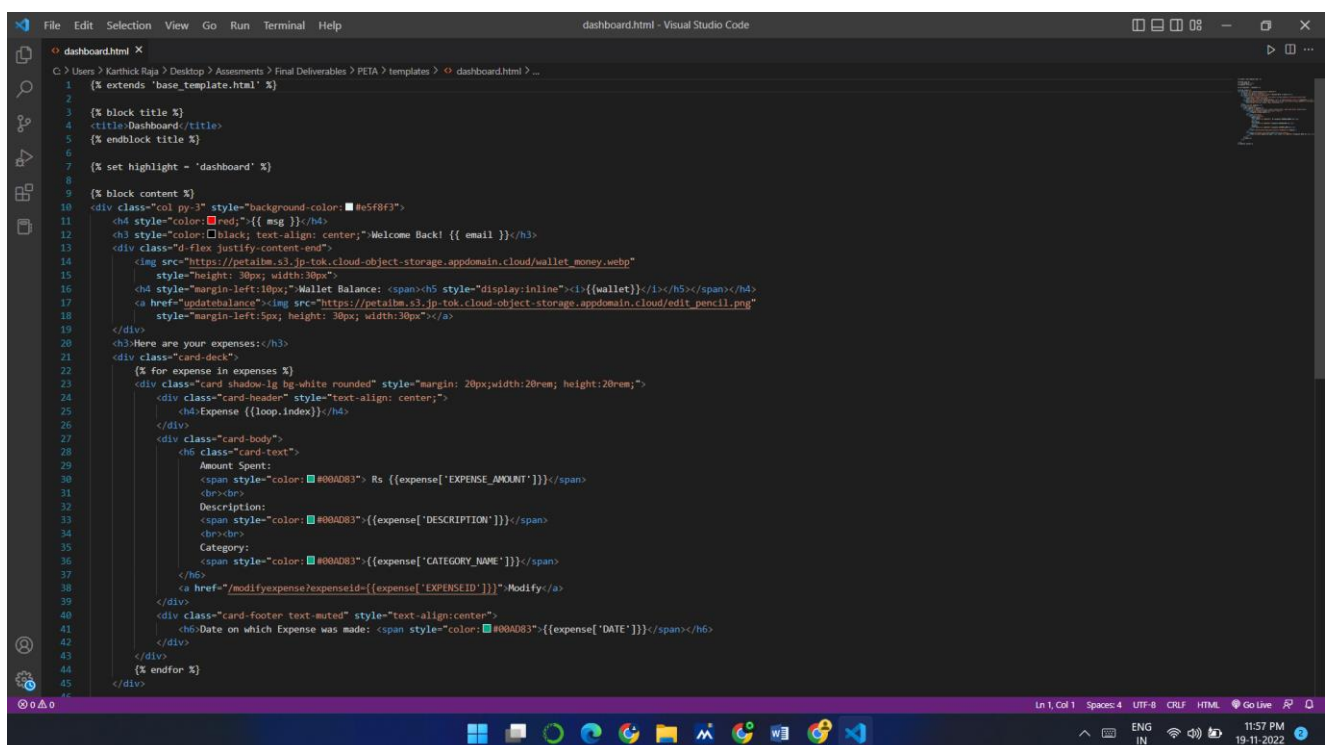


## 7.CODING & SOLUTIONING

### 7.1. FEATURE 1- DASHBOARD



### SOURCE CODE:





## App.py

```
from flask import Flask, render_template, request, redirect, url_for
from flask_mail import Mail, Message
from datetime import datetime
from flask_cors import CORS, cross_origin
import ibm_db
import json
import plotly
import plotly.graph_objs as go
import pandas as pd
from flask import send_file
from io import BytesIO
import matplotlib.pyplot as plt
import numpy as np
import base64
from PIL import Image
import time
import atexit
from datetime import datetime
from apscheduler.schedulers.background import BackgroundScheduler

app = Flask(__name__, template_folder='templates')
app.config['SECRET_KEY'] = 'top-secret!'
app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'apikey'
app.config['MAIL_PASSWORD'] = 'SG.rRPqo3ZyRhWUD6Rh1jE1CA.894zN6QMM9Uj0PgP1O-4KT-
_mjT9-KwXZ9ArygkEnis'
app.config['MAIL_DEFAULT_SENDER'] = 'karthi01rkr@gmail.com'
mail = Mail(app)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'

EMAIL = ''
USERID = ''
print()
try:
    conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=815fa4db-dc03-4c70-869a-
a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30367;SECURITY=SSL;S
SLServerCertificate=DigiCertGlobalRootCA.crt;UID=hbm00237;PWD=WdEDRgFEQ06uK41l", '', ''
)

except Exception as e:
    print(e)
print('hello')

def fetch_walletamount():
    sql = 'SELECT WALLET FROM PETA_USER WHERE EMAIL=?'
```

```
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, EMAIL)
ibm_db.execute(stmt)
user = ibm_db.fetch_assoc(stmt)
return user['WALLET']
```

```
def fetch_categories():
```

```
    sql = 'SELECT * FROM PETA_CATEGORY WHERE USERID = ?'
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, USERID)
    ibm_db.execute(stmt)

    categories = []
    while ibm_db.fetch_row(stmt) != False:
        categories.append([ibm_db.result(stmt, "CATEGORYID"),
                           ibm_db.result(stmt, "CATEGORY_NAME")])

    sql = 'SELECT * FROM PETA_CATEGORY WHERE USERID IS NULL'
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)

    while ibm_db.fetch_row(stmt) != False:
        categories.append([ibm_db.result(stmt, "CATEGORYID"),
                           ibm_db.result(stmt, "CATEGORY_NAME")])

    return categories
```

```
def fetch_userID():
```

```
    sql = 'SELECT USERID FROM PETA_USER WHERE EMAIL=?'
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, EMAIL)
    ibm_db.execute(stmt)
    user = ibm_db.fetch_assoc(stmt)
    return user['USERID']
```

```
def fetch_groups():
```

```
    sql = 'SELECT * FROM PETA_GROUPS'
    stmt = ibm_db.exec_immediate(conn, sql)
    groups = []
    while ibm_db.fetch_row(stmt) != False:
        groups.append([ibm_db.result(stmt, "GROUPID"),
                       ibm_db.result(stmt, "GROUPNAME")])
    return groups
```

```
def fetch_expenses():
```

```
    sql = 'SELECT * FROM PETA_EXPENSE where USERID = ' + str(USERID)
    stmt = ibm_db.exec_immediate(conn, sql)
    expenses = []
```

```

while ibm_db.fetch_row(stmt):
    category_id = ibm_db.result(stmt, "CATEGORYID")
    category_id = str(category_id)
    sql2 = "SELECT * FROM PETA_CATEGORY WHERE CATEGORYID = " + category_id
    stmt2 = ibm_db.exec_immediate(conn, sql2)
    category_name = ""
    while ibm_db.fetch_row(stmt2) != False:
        category_name = ibm_db.result(stmt2, "CATEGORY_NAME")
    expenses.append([ibm_db.result(stmt, "EXPENSE_AMOUNT"), ibm_db.result(
        stmt, "DATE"), ibm_db.result(stmt, "DESCRIPTION"), category_name])
return expenses

```

```

def fetch_limits():
    now = datetime.now()
    year = now.year

    limits = [0 for i in range(12)]

    sql = 'SELECT LIMITAMOUNT, LIMITMONTH FROM PETA_LIMIT WHERE USERID = ? AND
LIMITYEAR = ?'
    statement = execute_sql(sql, USERID, year)

    while ibm_db.fetch_row(statement):
        limit_amount = int(ibm_db.result(statement, 'LIMITAMOUNT'))
        limit_month = int(ibm_db.result(statement, 'LIMITMONTH'))
        limits[limit_month] = limit_amount

    return limits

```

```

def fetch_latest_expenses(expenses):
    latest_month = datetime.today().month
    latest_expenses = []
    for exp in expenses:
        if exp[1].month == latest_month:
            latest_expenses.append(exp)

    return latest_expenses

```

```

def fetch_monthly_expenses(expenses):
    latest_year = datetime.today().year
    monthly_expenses = {}

    for month in range(1, 13):
        monthly_expenses[month] = 0

    for exp in expenses:
        if exp[1].year == latest_year:
            monthly_expenses[exp[1].month] += exp[0]

```

```

    return monthly_expenses.values()

def draw_graph1(expenses):
    # TOTAL EXPENSE / DAY OF MONTH

    latest_expenses = fetch_latest_expenses(expenses)
    mp = {}
    for day in range(1, 31):
        mp[day] = 0

    for exp in latest_expenses:
        mp[exp[1].day] += exp[0]

    x = mp.keys()
    y = mp.values()

    plt.figure()
    plt.title('Expense recorded over the past month')
    plt.plot(x, y)
    plt.xlabel('Day of the month')
    plt.ylabel('Recorded expense')
    plt.xlim(1, 32)

    buffer = BytesIO()
    plt.savefig(buffer, format='png')

    encoded_img_data = base64.b64encode(buffer.getvalue())

    return encoded_img_data

def draw_graph2(expenses, limits):

    monthly_expenses = fetch_monthly_expenses(expenses)
    x = range(1, 13)
    y1 = limits
    y2 = monthly_expenses

    plt.figure()
    plt.title('Month-wise comparison of limit and expense')
    plt.plot(x, y1, label="Limit/month")
    plt.plot(x, y2, label="Expenses/month")
    plt.xlabel('Month')
    plt.legend()

    buffer = BytesIO()
    plt.savefig(buffer, format='png')

    encoded_img_data = base64.b64encode(buffer.getvalue())

```

```

    return encoded_img_data

scheduler = BackgroundScheduler()
scheduler.add_job(func=auto_renew, trigger="interval", seconds=3600 * 24)
print('hello')
scheduler.start()
print('hello')
atexit.register(lambda: scheduler.shutdown())

@app.route('/', methods=['GET', 'POST'])
@cross_origin()
def registration():
    global EMAIL
    print("hello")
    if request.method == 'GET':
        return render_template('registration.html')
    if request.method == 'POST':
        email = request.form['email']
        EMAIL = email
        password = request.form['password']
        wallet = request.form['wallet']
        sql = "INSERT INTO PETA_USER(EMAIL,PASSWORD,WALLET) VALUES(?,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.bind_param(stmt, 3, wallet)
        print(stmt)
        ibm_db.execute(stmt)
    return redirect(url_for('dashboard'))

@app.route('/login', methods=['GET', 'POST'])
def login():
    global EMAIL
    print("login")
    if request.method == 'POST':
        email = request.form['email']
        EMAIL = email
        print(EMAIL)
        password = request.form['password']
        sql = "SELECT * FROM PETA_USER WHERE email=? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        if account:
            return redirect(url_for('dashboard'))
        else:
            return redirect(url_for('login'))

```

```

        elif request.method == 'GET':
            return render_template('login.html')

@app.route('/logout', methods=['GET'])
def logout():
    if request.method == 'GET':
        global USERID
        global EMAIL
        USERID = ""
        EMAIL = ""
        return redirect(url_for('login'))

@app.route('/dashboard', methods=['GET'])
def dashboard():
    global USERID
    global EMAIL
    print("dashboard")
    if USERID == '' and EMAIL == '':
        print("null email")
        return render_template('login.html')
    elif USERID == '':
        USERID = fetch_userID()
        print(USERID)

    sql = "SELECT EXPENSEID, EXPENSE_AMOUNT, DESCRIPTION, CATEGORY_NAME, DATE FROM
PETA_EXPENSE, PETA_CATEGORY WHERE PETA_EXPENSE.USERID = ? AND PETA_EXPENSE.CATEGORYID
= PETA_CATEGORY.CATEGORYID"
    statement = execute_sql(sql, USERID)

    expenses = []
    while True:
        expense = ibm_db.fetch_assoc(statement)
        if expense:
            expenses.append(expense)
        else:
            break

    wallet = fetch_walletamount()
    return render_template('dashboard.html', expenses=expenses, wallet=wallet,
email=EMAIL)

@app.route('/updatebalance', methods=['GET', 'POST'])
def update_balance():
    if request.method == 'GET':
        wallet = fetch_walletamount()
        return render_template('updatebalance.html', wallet=wallet)
    elif request.method == 'POST':
        global EMAIL
        global USERID

```

```

        if EMAIL == '':
            return render_template('login.html', msg='Login before proceeding')
        if (USERID == ''):
            USERID = fetch_userID()

        new_balance = request.form['balanceupdated']
        sql = 'UPDATE PETA_USER SET WALLET = ? WHERE USERID = ?'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, new_balance)
        ibm_db.bind_param(stmt, 2, USERID)
        ibm_db.execute(stmt)

        return redirect(url_for('dashboard'))

@app.route('/addcategory', methods=['GET', 'POST'])
def add_category():
    if request.method == 'GET':
        return render_template('addcategory.html')

    elif request.method == 'POST':
        categoryname = request.form['category']
        sql = 'INSERT INTO PETA_CATEGORY(CATEGORY_NAME, USERID) VALUES(?,?)'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, categoryname)
        ibm_db.bind_param(stmt, 2, USERID)
        ibm_db.execute(stmt)

        return redirect(url_for('dashboard'))

@app.route('/addgroup', methods=['POST'])
def add_group():
    if request.method == 'POST':
        if USERID == '':
            return render_template('login.html', msg='Login before proceeding')
        sql = "INSERT INTO PETA_GROUPS(GROUPNAME, USERID) VALUES(?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, request.form['groupname'])
        ibm_db.bind_param(stmt, 2, USERID)
        ibm_db.execute(stmt)

        group_info = {}

        sql = "SELECT * FROM PETA_GROUPS WHERE GROUPNAME=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, request.form['groupname'])
        ibm_db.execute(stmt)
        group_info = ibm_db.fetch_assoc(stmt)
        return {"groupID": group_info['GROUPID'], 'groupname':
group_info['GROUPNAME']}

```

```

@app.route('/addexpense', methods=['GET', 'POST'])
def add_expense():
    if request.method == 'GET':
        groups = fetch_groups()
        categories = fetch_categories()
        if len(categories) == 0:
            return redirect(url_for('add_category'))
        return render_template('addexpense.html', categories=categories,
groups=groups)

    elif request.method == 'POST':
        global EMAIL
        global USERID
        if EMAIL == '':
            return render_template('login.html', msg='Login before proceeding')
        if (USERID == ''):
            USERID = fetch_userID()

        amount_spent = request.form['amountspent']
        category_id = request.form.get('category')
        description = request.form.get('description')
        date = request.form['date']

        groupid = request.form.get('group')
        groupid = None if groupid == '' else groupid

        print(amount_spent, category_id, description, date, groupid, USERID)

        sql = "INSERT INTO PETA_EXPENSE(USERID, EXPENSE_AMOUNT, CATEGORYID, GROUPID,
DESCRIPTION, DATE) VALUES(?,?,?,?,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, USERID)
        ibm_db.bind_param(stmt, 2, amount_spent)
        ibm_db.bind_param(stmt, 3, category_id)
        ibm_db.bind_param(stmt, 4, groupid)
        ibm_db.bind_param(stmt, 5, description)
        ibm_db.bind_param(stmt, 6, date)
        ibm_db.execute(stmt)
        print(date, amount_spent, category_id)
        sql = "UPDATE PETA_USER SET WALLET = WALLET - ? WHERE USERID = ?"
        statement = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(statement, 1, amount_spent)
        ibm_db.bind_param(statement, 2, USERID)
        ibm_db.execute(statement)

        return redirect(url_for('dashboard'))

@app.route('/analysis', methods=['GET', 'POST'])
def analyse():
    if request.method == 'GET':

```



```

        expenses = fetch_expenses()
        limits = fetch_limits()

        graph1 = draw_graph1(expenses=expenses)
        graph2 = draw_graph2(expenses=expenses, limits=limits)

        return render_template("analysis.html", img_data1=graph1.decode('utf-8'),
img_data2=graph2.decode('utf-8'))

    elif request.method == 'POST':
        return render_template('analysis.html')

def execute_sql(sql, *args):
    stmt = ibm_db.prepare(conn, sql)
    for i, arg in enumerate(args):
        ibm_db.bind_param(stmt, i + 1, arg)
    ibm_db.execute(stmt)
    return stmt

def check_monthly_limit(month, year):
    sql = 'SELECT SUM(EXPENSE_AMOUNT) FROM PETA_EXPENSE WHERE USERID = ? AND
MONTH(DATE) = ? AND YEAR(DATE) = ?'
    statement = execute_sql(sql, USERID, month, year)
    amt_spent = ibm_db.fetch_tuple(statement)

    sql = 'SELECT LIMITAMOUNT FROM PETA_LIMIT WHERE USERID = ? AND LIMITMONTH = ? AND
LIMITYEAR = ?'
    statement = execute_sql(sql, USERID, month, year)
    monthly_limit = ibm_db.fetch_tuple(statement)

    if amt_spent and monthly_limit and int(amt_spent[0]) > int(monthly_limit[0]):
        diff = int(amt_spent[0]) - int(monthly_limit[0])
        msg = Message('Monthly limit exceeded', recipients=[EMAIL])
        msg.body = (
            f'Monthly limit exceeded by {diff} for the month of {month}, {year}')
        mail.send(msg)

def update_monthly_limit(monthly_limit, month, year):
    sql = 'SELECT LIMITAMOUNT FROM PETA_LIMIT WHERE USERID = ? AND LIMITMONTH = ? AND
LIMITYEAR = ?'
    statement = execute_sql(sql, USERID, month, year)

    if ibm_db.fetch_row(statement):
        sql = 'UPDATE PETA_LIMIT SET LIMITAMOUNT = ? WHERE USERID = ? AND LIMITMONTH
= ? AND LIMITYEAR = ?'
        execute_sql(sql, monthly_limit, USERID, month, year)
    else:
        sql = 'INSERT INTO PETA_LIMIT VALUES(?, ?, ?, ?)'
        execute_sql(sql, USERID, monthly_limit, month, year)

```

```
check_monthly_limit(month, year)
```

```
@app.route('/setmonthlylimit', methods=['GET', 'POST'])
```

```
def set_monthly_limit():
```

```
    if request.method == 'GET':
```

```
        return render_template('setmonthlylimit.html')
```

```
    elif request.method == 'POST':
```

```
        new_monthly_limit = request.form['monthlylimit']
```

```
        now = datetime.now()
```

```
        update_monthly_limit(new_monthly_limit, now.month, now.year)
```

```
        return redirect(url_for('dashboard'))
```

```
@app.route('/modifyexpense', methods=['GET', 'POST'])
```

```
def modify_expense():
```

```
    if request.method == 'GET':
```

```
        expenseid = request.args.get('expenseid')
```

```
        sql = "SELECT * FROM PETA_EXPENSE WHERE EXPENSEID = ?"
```

```
        statement = execute_sql(sql, expenseid)
```

```
        expense = ibm_db.fetch_assoc(statement)
```

```
        categories = fetch_categories()
```

```
        groups = fetch_groups()
```

```
        return render_template('modifyexpense.html', expense=expense,
```

```
categories=categories, groups=groups)
```

```
    elif request.method == 'POST':
```

```
        amount_spent = request.form['amountspent']
```

```
        category_id = request.form.get('category')
```

```
        description = request.form['description']
```

```
        date = request.form['date']
```

```
        groupid = request.form.get('group')
```

```
        expenseid = request.form['expenseid']
```

```
        old_amount_spent = request.form['oldamountspent']
```

```
        sql = "UPDATE PETA_EXPENSE SET EXPENSE_AMOUNT = ?, CATEGORYID = ?, GROUPID =  
?, DESCRIPTION = ?, DATE = ? WHERE EXPENSEID = ?"
```

```
        execute_sql(sql, amount_spent, category_id,
```

```
groupid, description, date, expenseid)
```

```
        sql = "UPDATE PETA_USER SET WALLET = WALLET + ?"
```

```
        execute_sql(sql, float(old_amount_spent) - float(amount_spent))
```

```
        return redirect(url_for('dashboard'))
```

```
def fetch_goals():
```

```
    sql = 'SELECT * FROM PETA_GOALS WHERE USERID = ?'
```

```
    statement = execute_sql(sql, USERID)
```

```
    goals = []
```

```

while True:
    goal = ibm_db.fetch_tuple(statement)
    if goal:
        goals.append(goal[2:])
    else:
        break

print(goals)
return goals

@app.route('/rewards', methods=['GET'])
def rewards_and_goals():
    goals = fetch_goals()
    return render_template('rewards.html', goals=goals)

@app.route('/addgoal', methods=['GET', 'POST'])
def add_goal():
    if request.method == 'GET':
        return render_template('addgoal.html')
    elif request.method == 'POST':
        goal_amount = request.form['goal_amount']
        date = request.form['date']
        reward = request.form['reward']
        sql = "INSERT INTO PETA_GOALS(USERID, GOAL_AMOUNT, DATE, REWARD) VALUES(?, ?,
?, ?)"
        execute_sql(sql, USERID, goal_amount, date, reward)
        return redirect(url_for('dashboard'))

def check_goals():
    sql = "SELECT A.GOALID, A.USERID, A.GOAL_AMOUNT, A.DATE, A.REWARD, B.WALLET FROM
PETA_GOALS AS A, PETA_USER AS B WHERE A.USERID = B.USERID"
    statement = execute_sql(sql)

    now = datetime.now()
    while True:
        row = ibm_db.fetch_assoc(statement)
        if not row:
            break
        if row['DATE'] == now:
            if row['GOAL_AMOUNT'] <= row['WALLET']:
                msg = Message('Goal achieved!', recipients=[EMAIL])
                msg.body = (
                    f'You are eligible for your reward:\n{row["REWARD"]}')
                mail.send(msg)
            else:
                msg = Message('Goal limit exceeded', recipients=[EMAIL])
                msg.body = (
                    f'You are not eligible for the reward:\n{row["REWARD"]}\nBetter
luck next time!')

```

```

        mail.send(msg)
        sql = "DELETE FROM PETA_GOALS WHERE GOALID = ?"
        execute_sql(sql, row['GOALID'])

scheduler.add_job(func=check_goals, trigger="interval", seconds=3600 * 24)

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)

```

## 7.2. FEATURE 2-WATSON

### SOURCE CODE:

<script>

```
window.watsonAssistantChatOptions = {
```

```

integrationID: "9dbf7c82-7565-4808-abc9-a42a2b831075", // The ID of this
integration.region: "au-syd", // The region your integration is hosted
in.serviceInstanceID: "61b88af9-0969-4812-a514-0d770256682b", // The ID of your
service instance.onLoad: function(instance) { instance.render(); }

```

```
};
```

```
setTimeout(function(){
```

```
const t=document.createElement('script');
```

```

t.src=https://web-chat.global.assistant.watson.appdomain.cloud/versions/
+(window.watsonAssistantChatOptions.clientVersion || 'latest')
+"/WatsonAssistantChatEntry.js";

```

```
document.head.appendChild(t);
```

```
});
```

</script>

## 8. TESTING

### 8.1 Test Cases

#### 1. Application level:

This level deals with all the activities of sale like sale of an item, available discount, coupons, payment, receipt printing, more precisely functions of application level is sales and payment.

Types of testing used at this level are as:

- **Functionality Testing:**

It is a type of testing which test that all functions of the application are working as per the requirements of the system. In the POS system, testers validate the working of all functions using functionality testing, the functions which the POS system performs are printing receipt of the purchased item, returning an item, selling, scanning an item for its price and available discount, and payment.

- **Compatibility Testing:**

This testing is non-functional testing that is used to test whether the system is capable to work with other OS, hardware, software, mobile devices, etc. As the POS system is connected to several hardware components so testers test compatibility with all hardware and a new version of OS.

- **Payment Gateway Testing:**

POS system supports online payment modes of the transaction so it needs to follow PCI complaints. Testers validate that the system is working successfully with various payment modes.

#### 2. Enterprise Level:

The enterprise level is a broad term that deals with functions like payrolls, total transaction throughout the day, offers which attracted a large number of customers, database management, inventory list management, and accounts management.

### **Type of testing used at this level are as:**

- **Performance Testing:**

It is a type of testing which test the performance of the system in terms of response time, working with connected devices, speed, and scalability when workload increases. POS system goes through this testing to validate the responsiveness of the system and when the system crash if the load increases.

- **Interoperability Testing:**

POS system operates on various software and hardware, and testers validate that the system is interacting with other related software and hardware as per requirements.

- **Compliance Testing:**

It is also called conformance testing, it is used to determine that the application meets all the set standards. The POS system offers credit and debit card payment modes so. it needs to fulfill all the parameters set by PCI standards for payments, testers validate that the system is following all those set standards.

- **Data Migration Testing:**

This form of testing is used when data is migrated from the original database to the new database storage system, it validates that all data is replaced without any loss of data, and no duplicate data is made.

- **Upgradation Testing:**

This form of testing determines that the new versions of the software are compatible to upgrade the old version of the system. A POS system needs to be upgraded with a new OS, software, and hardware version so testers validate that the system can be timely upgrade and support new features to remain in the race of changing technology with time.

## 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 the Personal Expense Tracker project at the 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 were resolved

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

### 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
Login	3	0	0	3
Registration	4	0	0	4
Dashboard	1	0	0	1
Expense Addition	3	0	1	2
Set monthly limit	2	0	0	2
View Analysis	3	0	1	2
Recurring Expense Addition	2	0	0	2
View Recurring Expense	2	0	0	2
Rewards and Goals	4	0	0	4
Category Creation	2	0	0	2
Modifying an Expense	2	0	0	2

### Who Performs UAT?

Users or client – This could be either someone who is buying a product (in the case of commercial software) or someone who has had a software custom-built through a software service provider or the end-user if the software is made available to them ahead of the time and when their feedback is sought out.

The team can be comprised of beta testers or the customer should select UAT members internally from every group of the organization so that each and every user role can be tested accordingly.



## 9. RESULTS

### 9.1 Performance Metrics

The project Personal Expense Tracker Application performance Metrics are validated based on the

#### - Latency

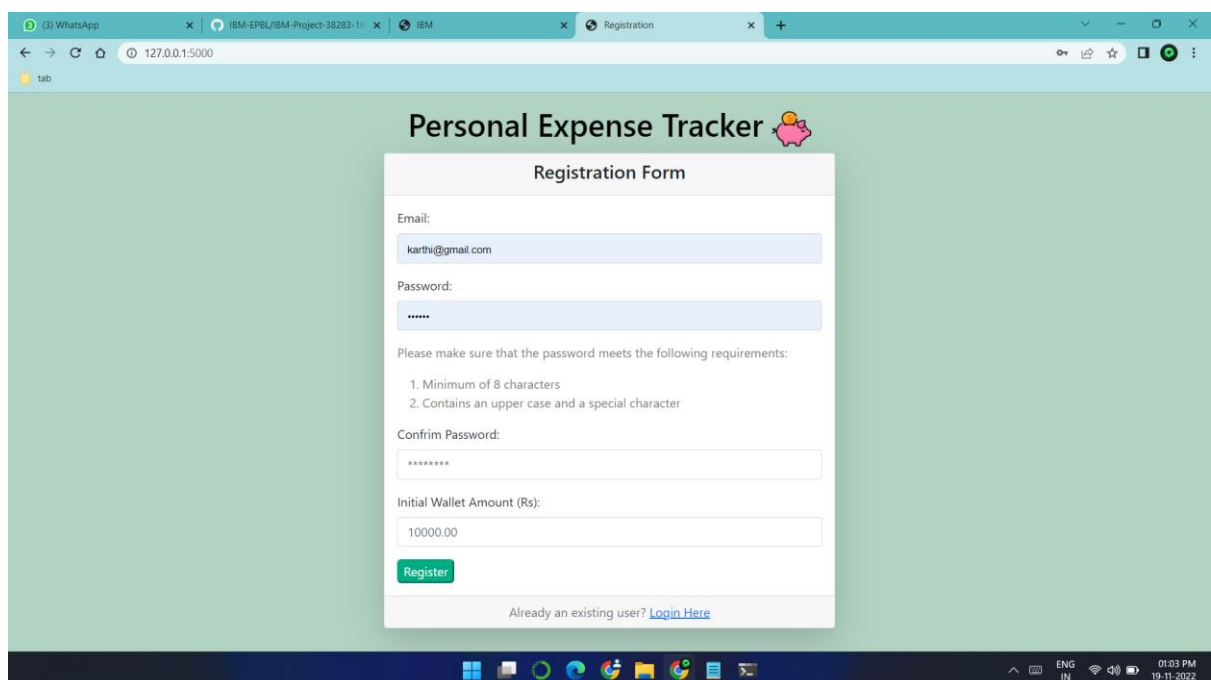
Personal Expense Tracker application deployed on the IBM Kubernetes Cluster offers up to 150 - 200ms in the average traffic and 200 -250ms in the stressed loaded environment with Ngnix container acting as the service in the cluster.

#### - Rate

Personal Expense Tracker Application can handle up to 300K to 400K requests at a time and the over the handle can wear down but satisfies the request upon delayed manner.

#### - No. of Failures

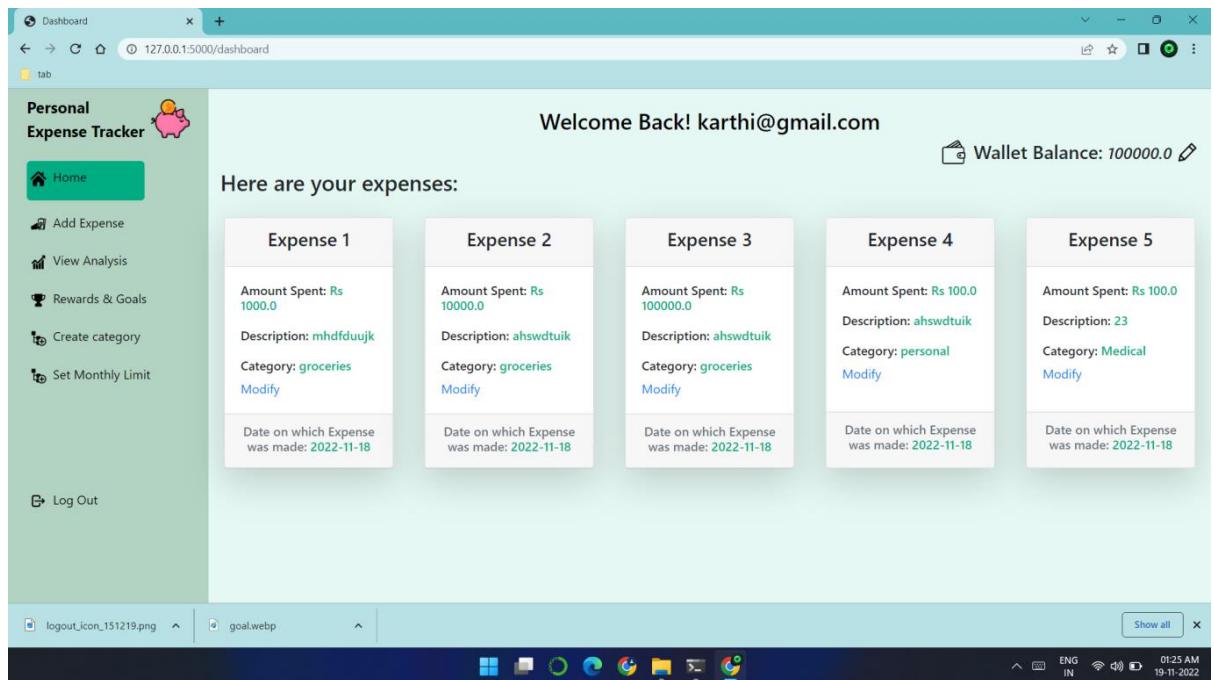
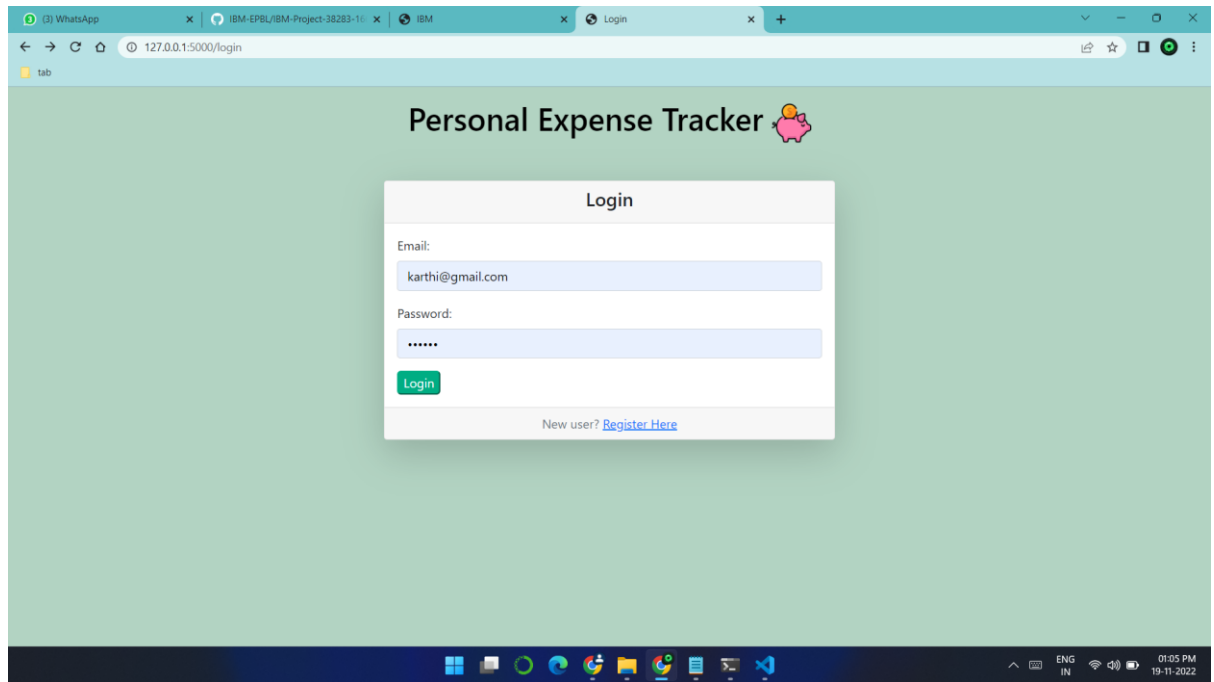
Personal Expenses Tracker Application is designed in such a way that it can handle different input and respond accordingly. The application is aggressively tested with different test cases so that it reduces the rate of failure up to 0.75%

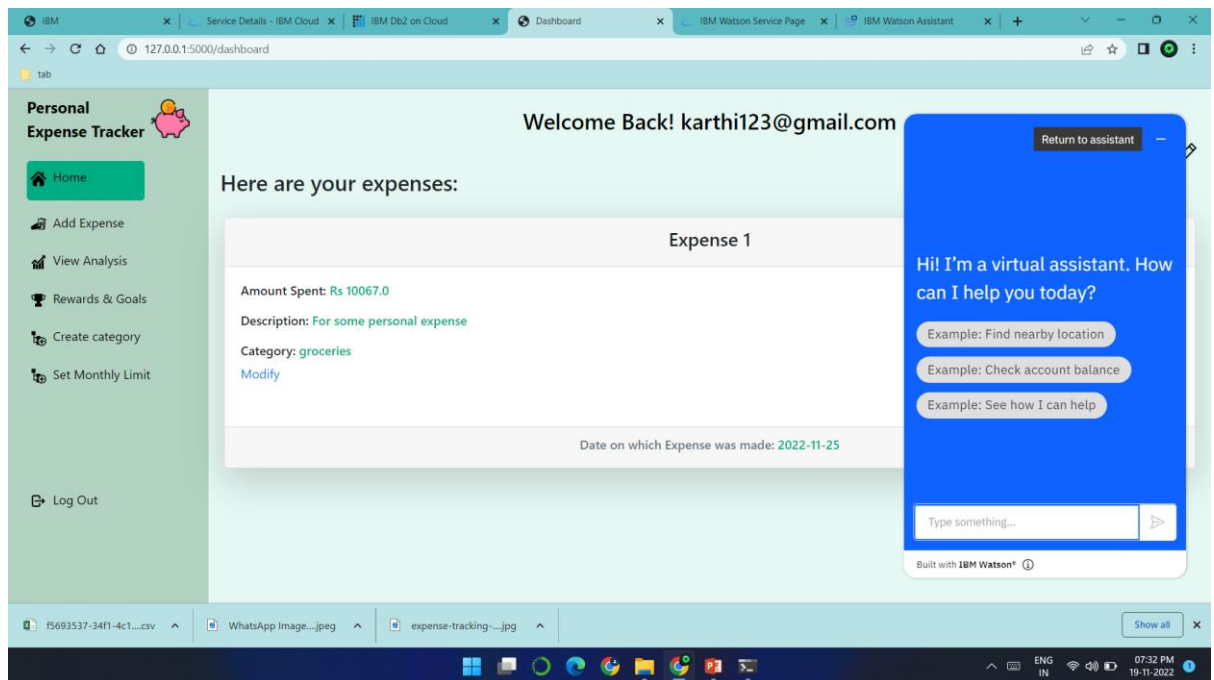
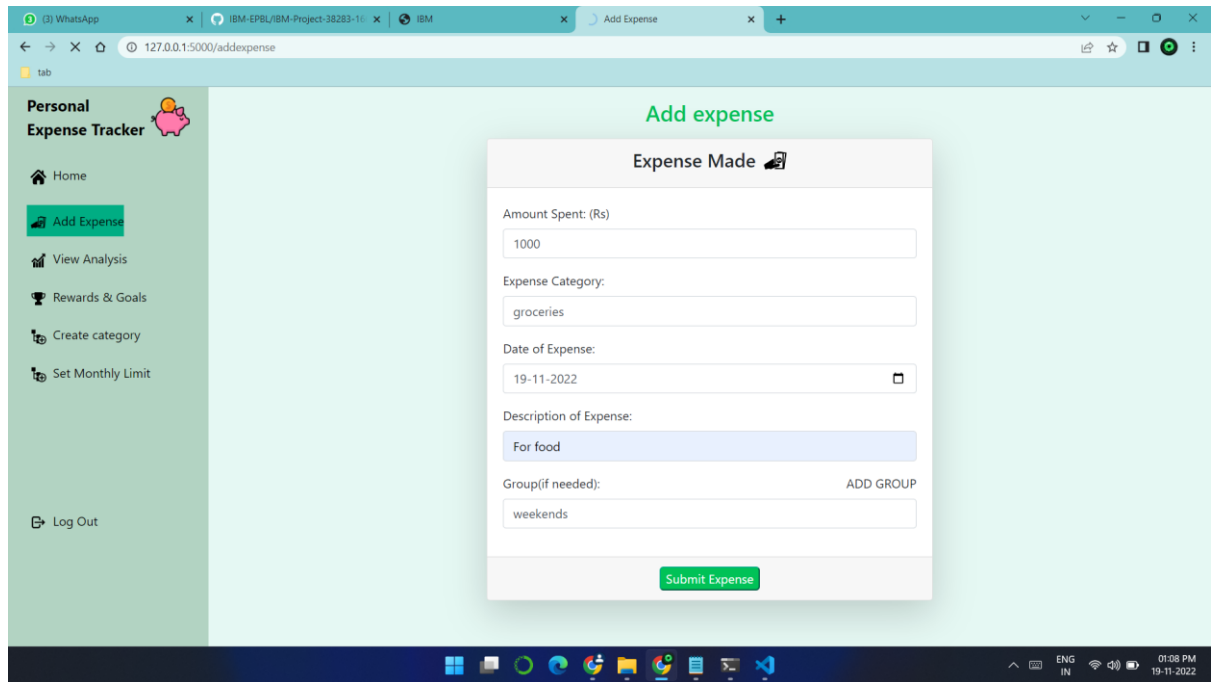


The screenshot shows a web browser window with the URL 127.0.0.1:5000. The page title is "Personal Expense Tracker" with a piggy bank icon. The main content is a "Registration Form" with the following fields and text:

- Email:
- Password:
- Please make sure that the password meets the following requirements:
  - 1. Minimum of 8 characters
  - 2. Contains an upper case and a special character
- Confirm Password:
- Initial Wallet Amount (Rs):
- 
- Already an existing user? [Login Here](#)

The browser's taskbar at the bottom shows the Windows logo, several application icons, and system status icons including network, volume, and battery, along with the time 01:03 PM on 19-11-2022.





## **10.ADVANTAGES & DISADVANTAGES**

### **ADVANTAGES & DISADVANTAGES**

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

## **11.CONCLUSION**

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

## **12.FUTURE SCOPE**

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

1. The application is unable to maintain the backup of data once it is uninstalled.
2. This application does not provide higher decision capability.

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

1. Multiple language interfaces.
2. Provide backup and recovery of data.
3. Mobile app advantage.

## 13.APPENDIX

### 13.1. Source Code

#### Login.html:

```
<!doctype html>
<html lang="en">
  <head>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">

    <title>Login</title>
  </head>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min
.js" integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>

  <body style="background-color:#B2D3C2">
    <div class="container mt-3">
      <h1 style="color: black; text-align: center;">
        Personal Expense Tracker 
      </h1>
      <div class="container mt-5" style="width: 600px;">
        <h4>{{ msg }}</h4>
        <div class="card shadow-lg bg-white rounded">
          <div class="card-header" style="text-align: center;">
            <h4>Login</h4>
          </div>
          <div class="card-body">
            <form action="/login" method="POST">
              <div class="mb-3">
                <label for="email" class="form-label">Email:
</label>
```

```

        <input type="email" class="form-control"
name="email" id="email" placeholder="abc@gmail.com">
    </div>
    <div class="mb-3">
        <label for="passowrd" class="form-
label">Password: </label>
        <input type="password" class="form-control"
name="password" id="password"></input>
    </div>
    <button type="submit" style="background-
color:#00AD83; border-color:#00AD83;color: white; border-
radius:5px;">Login</button>
    </form>
</div>
<div class="card-footer text-muted" style="text-
align:center">
    New user? <span><a href="/">Register Here</a></span>
</div>
</div>
</div>
</body>
</html>

```

## Register.html:

```

<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfSpd3yD65VohhpuuC0mLASjC"
crossorigin="anonymous">

        <title>Registration</title>
    </head>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min
.js" integrity="sha384-
Mrcw6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>

```



```

<body style="background-color:#B2D3C2">
  <div class="container mt-3">
    <h1 style="color: black; text-align: center;">
      Personal Expense Tracker 
    </h1>
    <div class="container mt-2" style="width: 600px;">
      <div class="card shadow-lg bg-white rounded">
        <div class="card-header" style="text-align: center;">
          <h4>Registration Form</h4>
        </div>
        <div class="card-body">
          <form action="/" method="POST">
            <div class="mb-3">
              <label for="email" class="form-label">Email:
</label>
              <input type="email" class="form-control"
name="email" id="email" placeholder="abc@gmail.com">
            </div>
            <div class="mb-3">
              <label for="passowrd" class="form-
label">Password: </label>
              <input type="password" class="form-control"
name="password" id="password"></input>
              <p style="color: gray;" class="mt-3">Please
make sure that the password meets the following requirements:</p>
              <ol style="color: gray;"><li>Minimum of 8
characters</li><li>Contains an upper case and a special character</li></ol>
            </div>
            <div class="mb-3">
              <label for="confirmpassword" class="form-
label">Confrim Password: </label>
              <input type="password" class="form-control"
name="confirmpassword" id="confirmpassword" placeholder="*****">
            </div>
            <div class="mb-3">
              <label for="wallet" class="form-label">Initial
Wallet Amount (Rs): </label>
              <input type="number" class="form-control"
name="wallet" id="wallet" placeholder="10000.00">
            </div>
            <button type="submit" style="background-
color:#00AD83; border-color:#00AD83;color: white; border-
radius:5px;">Register</button>
          </form>
        </div>
      </div>
    </div>
  </div>

```

```

                <div class="card-footer text-muted" style="text-align:center">
                    Already an existing user? <span><a href="login">Login Here</a></span>
                </div>
            </div>
        </div>
    </div>
</body>
</html>

```

## Base.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWFspD3yD65VohhpUuCOmLASjC" crossorigin="anonymous">

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

    {% block title %}
        <title>Base Template</title>
    {% endblock title %}
</head>
<body>
    <div class="container-fluid">
        <div class="row flex-nowrap">

```

```

        <div class="col-auto col-md-3 col-xl-2 px-sm-2 px-0"
style="background-color: #B2D3C2">
            <div class="d-flex flex-column align-items-center align-items-
sm-start px-3 pt-2 min-vh-100" style="color:black">
                <p class="d-flex align-items-center pb-3 mb-md-0 me-md-
auto text-white text-decoration-none">
                    <span class="fs-5 d-none d-sm-inline"
style="color:black; font-weight: bold;">Personal Expense Tracker</span>
                    
                </p>
                <ul class="nav nav-pills flex-column mb-sm-auto mb-0
align-items-center align-items-sm-start" id="menu">
                    <li class="nav-item mt-2" style="background-color:
{{'#00AD83' if highlight == 'dashboard'}}; height: 50px; width: 150px; border-
radius: 5px;">
                        <a href="dashboard" class="nav-link align-middle
px-0" style="color:black;">
                            
                            <span class="ms-1 d-none d-sm-
inline">Home</span>
                        </a>
                    </li>
                    <li class="nav-item mt-2" style="background-color:
{{'#00AD83' if highlight == 'addexpense'}};">
                        <a href="addexpense" class="nav-link px-0 align-
middle" style="color:black;">
                            
                            <span class="ms-1 d-none d-sm-inline">Add
Expense</span>
                        </a>
                    </li>
                    <li class="nav-item mt-2" style="background-color:
{{'#00AD83' if highlight == 'analysis'}};">
                        <a href="analysis" class="nav-link px-0 align-
middle" style="color:black;">
                            
                            <span class="ms-1 d-none d-sm-inline">View
Analysis</span>
                        </a>

```

```

        </li>

        <li class="nav-item mt-2" style="background-color:
{{'#00AD83' if highlight == 'rewards'}};">
            <a href="rewards" class="nav-link px-0 align-
middle" style="color:black;">
                
                <span class="ms-1 d-none d-sm-inline">Rewards
& Goals</span>
            </a>
        </li>

        <li class="nav-item mt-2" style="background-color:
{{'#00AD83' if highlight == 'addcategory'}};">
            <a href="addcategory" class="nav-link px-0 align-
middle" style="color:black;">
                
                <span class="ms-1 d-none d-sm-inline">Create
category</span>
            </a>
        </li>

        <li class="nav-item mt-2" style="background-color:
{{'#00AD83' if highlight == 'setmonthlylimit'}};">
            <a href="setmonthlylimit" class="nav-link px-0
align-middle" style="color:black;">
                
                <span class="ms-1 d-none d-sm-inline">Set
Monthly Limit</span>
            </a>
        </li>
    </ul>

    <ul class="nav nav-pills flex-column mb-sm-auto mb-0
align-items-center align-items-sm-end" id="menu">
        <li class="nav-item mt-2">
            <a href="logout" class="nav-link px-0 align-
middle" style="color:black;">
                

```

```

Out</span>
    <span class="ms-1 d-none d-sm-inline">Log
    </a>
  </li>
</ul>

</div>
</div>
{% block content %}
  <h1>This needs to be overridden</h1>
{% endblock content %}
</div>
</div>

{% block script %}
<script></script>
<script>
  window.watsonAssistantChatOptions = {
    integrationID: "9dbf7c82-7565-4808-abc9-a42a2b831075", // The ID of this
integration.
    region: "au-syd", // The region your integration is hosted in.
    serviceInstanceID: "61b88af9-0969-4812-a514-0d770256682b", // The ID of
your service instance.
    onLoad: function(instance) { instance.render(); }
  };
  setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/"
+ (window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
  });
</script>
{% endblock script %}
</body>
</html>

```

## Addcategory.html:

```

{% extends 'base_template.html' %}

{% block title %}

```

```

<title>Add Expense</title>
{% endblock title %}

{% set highlight = 'addexpense' %}

{% block content %}
<div class="col py-3" style="background-color:#e5f8f3">
    <h3 style="color:#00c257; text-align: center;">Add expense</h3>
    <div class="container mt-3" style="width: 600px;">
        <div class="card shadow-lg bg-white rounded">
            <form action="/addexpense" method="POST">
                <div class="card-header" style="text-align: center;">
                    <span style="display:inline-flex"><h4>Expense
Made</h4></span>
                </div>
                <div class="card-body">
                    <div class="mb-3">
                        <label for="amountspent" class="form-label">Amount
Spent: (Rs) </label>
                        <input type="number" class="form-control"
name="amountspent" id="amountspent" placeholder="100.00" required>
                    </div>
                    <div class="mb-3">
                        <label for="expensecategory" class="form-
label">Expense Category: </label>
                        <select name="category" id="category" class="form-
control" placeholder="Select a category" required>
                            <option value="">Select a category</option>
                            {% for cat in categories %}
                                <option value="{{ cat[0] }}">{{ cat[1]
}}</option>
                            {% endfor %}
                        </select>
                    </div>
                    <div class="mb-3">
                        <label for="date" class="form-label">Date of Expense:
</label>
                        <input type="date" class="form-control" name="date"
id="date" required></input>
                    </div>
                    <div class="mb-3">
                        <label for="description" class="form-
label">Description of Expense: </label>
                        <input type="text" class="form-control"
name="description" id="description"></input>
                    </div>

```

```

        <div class="mb-3">
            <label for="group" class="form-label">Group(if
needed): </label>
            <div title="New group" style="float:right"
value="Create group" onclick="addGroup()">ADD GROUP</div>
            <br/>

            <select name="group" id="group" class="form-control">
                <option value="">Select existing group</option>
                {% for group in groups %}
                    <option value="{{ group[0] }}">{{ group[1]
}}</option>

                {% endfor %}
            </select>
        </div>
    </div>
    <div class="card-footer text-muted" style="text-align:center">
        <button type="submit" value="submit" style="background-
color:#00c257; border-color:#00AD83;color: white; border-radius:5px;">Submit
Expense</button>
    </div>
</form>
</div>
</div>
</div>
{% endblock content %}

{% block script %}
<script>
    function addGroup(e) {
        group = window.prompt('Enter group name: ')
        console.log('PROMPT WINDOW SHOWN'+group);

        const formData = new FormData();
        formData.append("groupname", group);

        const xhttp = new XMLHttpRequest();
        xhttp.onload = function() {
            if (this.readyState == 4 && this.status == 200) {
                var groupid= JSON.parse(this.responseText);
                console.log(groupid);
                const newOption = document.createElement('option');
                const optionText =
document.createTextNode(groupid['groupname']);
                newOption.appendChild(optionText);
                newOption.setAttribute('value',groupid['groupId']);
                const selectDropdown = document.getElementById('group');
                selectDropdown.appendChild(newOption);
            }
        }
        xhttp.open('POST', '/add_group', true);
        xhttp.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
        xhttp.send(formData.toString());
    }
</script>
</div>
</div>
</div>

```

```

        console.log('GROUPID :'+ groupid['groupID']);
    }
}
xhttp.open("POST", "http://localhost:5000/addgroup");
xhttp.send(formData);
}
document.querySelector('#date').valueAsDate = new Date();
</script>
{% endblock script %}

```

## Analysis.html:

```

{% extends 'base_template.html' %}

{% block title %}
<title>Analysis</title>
{% endblock title %}

{% set highlight = 'analysis' %}

{% block content %}
<div class="col-auto px-0 col-lg-10 col-md-6 col-sm-4">
    <div class="card min-vh-100" style="background-color: #ffffff">
        <h4 class="card-header" style="color:#ffffff;background:
#00c257;">Analysis of my expenses</h4>
        <div class="card-body">
            <div class="row flex-nowrap">
                <div class="col col-lg-5 col-md-3 px-4" >
                    
                </div>
                <div class="col col-lg-5 col-md-3 px-4" >
                    
                </div>
            </div>
        </div>
    </div>
</div>

```



```
</div>
{% endblock content %}

{% block script %}
<script type="text/javascript">
  function generate_graph1() {}
</script>
{% endblock script %}
```

## 13.2.GitHub & Project Demo Link

### GitHub link

<https://github.com/IBM-EPBL/IBM-Project-38283-1660377159>

### Demo Link

[https://drive.google.com/file/d/1CXLrdNcUltulgo3\\_eonx5JkqQM2ESShv/view?usp=sharing](https://drive.google.com/file/d/1CXLrdNcUltulgo3_eonx5JkqQM2ESShv/view?usp=sharing)