

PERSONAL EXPENSE TRACKER APPLICATION



NALAIYA THIRAN PROJECT BASED LEARNING

On

**PROFESSIONAL READINESS FOR INNOVATION,
EMPLOYABILITY AND ENTREPRENEURSHIP**

A PROJECT REPORT

PNT2022TMID10256

HARI S	19110031
GUGAN K	19110029
BHARATHI R	19110012
AJITHKUMAR S	19110004

BACHELOR OF TECHNOLOGY

**IN
INFORMATION TECHNOLOGY**

**HINDUSTHAN COLLEGE OF ENGINEERING AND
TECHNOLOGY**

Approved by AICTE, New Delhi, Accredited with 'A' Grade by NAAC
(An Autonomous Institution, Affiliated to Anna University, Chennai)

COIMBATORE – 641 032

November 2022

TABLE OF CONTENTS

CHA	TITLE	PAGE NO.
1.	INTRODUCTION	1
	1.1 Project Overview	2
	1.2 Purpose	3
2.	LITERATURE SURVEY	
	2.1 Existing problem	4
	2.2 References	5
	2.3 Problem Statement Definition	6
3.	IDEATION & PROPOSED SOLUTION	
	3.1 Empathy Map Canvas	7
	3.2 Ideation & Brainstorming	8
	3.3 Proposed Solution	9
	3.4 Problem Solution fit	10
4.	REQUIREMENT ANALYSIS	
	4.1 Functional requirement	11
	4.2 Non-Functional requirements	12
5.	PROJECT DESIGN	
	5.1 Data Flow Diagrams	13
	5.2 Solution & Technical Architecture	14
	5.3 User Stories	15
6.	PROJECT PLANNING & SCHEDULING	
	6.1 Sprint Planning & Estimation	16
	6.2 Sprint Delivery Schedule	18
	6.3 Reports from JIRA	19

7.	CODING & SOLUTIONING	
	7.1 Feature	20
	7.2 Code for that feature	23
8.	TESTING	
	8.1 Test Cases	35
	8.2 User Acceptance Testing	36
9.	RESULTS	
	9.1 Result	37
10.	ADVANTAGES & DISADVANTAGES	42
11.	CONCLUSION	44
12.	FUTURE SCOPE	45
13.	APPENDIX	46
	Source code	
	Github and demo link	

1. INTRODUCTION

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.

Personal expense management is an important part of people's lives. However, everyone does not have the knowledge or time to manage their finances in a proper manner. And, even if a person has time and knowledge, they do not bother with tracking their expenses as they find it tedious and time-consuming.

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.

This Expense Tracker is a web application that facilitates the users to keep track and manage their personal expenses. This application helps the users to keep a digital diary. It will keep track of a user's income and expenses on a daily basis.

The user will be able to add his/her expenditures instantly and can review them anywhere and anytime with the help of the internet. It is time to stop using paper and excel sheets to keep track of your digital as well as cash payments.

Using paper is not easy to manage. It is common to delete files accidentally or misplace files. This expense tracker provides a complete digital solution to this problem. Excel sheets do very little to help in tracking expenses.

Furthermore, they don't have the advanced functionality of preparing graphical visuals automatically. Not only it will save the time of the people but also it will assure error-free calculations. The user just has to enter the income and expenditures and everything else will be performed by the system.

Now, you don't have to worry about managing your expenses, as you can get access to an expense tracker that will help in the active management of your finances.

1.1. PROJECT OVERVIEW

This a cloud based application for keeping track of a individual's expenses and an alert will be sent if they cross their predetermined limit to their expenses which is set by the users themselves.

This app helps people to spend their money in accordance to their income and not cross their limit by keeping tracks of their expenditures in almost all area that they spend money.

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.

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.

The skills required for this project is as follows:

- IBM Cloud,
- HTML ,
- CSS,
- JavaScript,
- IBM Cloud object Storage,
- Python-Flask,
- Kubernetes,
- Docker,
- IBM DB2,
- IBM Container Registry.

1.2. PURPOSE

The purpose of this application is as follows:

- 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.
- This Expense Tracker is a web application that facilitates the users to keep track and manage their personal expenses. This application helps the users to keep a digital diary. It will keep track of a user's income and expenses on a daily basis.
- The user will be able to add his/her expenditures instantly and can review them anywhere and anytime with the help of the internet. It is time to stop using paper and excel sheets to keep track of your digital as well as cash payments.
- Using paper is not easy to manage. It is common to delete files accidentally or misplace files. This expense tracker provides a complete digital solution to this problem. Excel sheets do very little to help in tracking expenses.
- Furthermore, they don't have the advanced functionality of preparing graphical visuals automatically. Not only it will save the time of the people but also it will assure error-free calculations. The user just has to enter the income and expenditures and everything else will be performed by the system.
- Now, you don't have to worry about managing your expenses, as you can get access to an expense tracker that will help in the active management of your finances.

2. LITRATURE SUPRVEY

2.1.EXISTING PROBLEM

The Expense tracker existing system does not provide the user portable device management level, existing system only used on desktop software so unable to update anywhere expenses done and unable to update the location of the expense details disruptive that the proposed system provides . In existing, we need to maintain the Excel sheets, CSVfiles for the user daily, weekly and monthly expenses. In existing, there is no as such complete solution to keep a track of its daily expenses easily. To do so a person as to keep a log in a diary or in a computer system, also all the calculations need to be done by the user which maysometimes results in mistakes leading to losses. The existing system is not user friendly because data is not maintained perfectly. But this project will not have any reminder to remain a person in a specific date,so that is the only drawback in which the remainder is not present. This project will be an unpopulated information because it has some disadvantages by not remind a person for each and every month. But it can used to perform calculation on income and expenses to overcome this problem we propose the new project.

PROPOSED SOLUTION

This new Online Income and Expense Tracker will eliminate all the demerits which are found under the existing system. To reduce manual calculations, we propose an application which is developed by python. Each user will be required to register on the system at registrationtime, the user will be provided id, which will be used to maintain the record of each unique user. Expense Tracker project which will keep a track of Income-Expense of a user on a day to day basis

This project takes Income from user and divides in daily expense allowed. If you exceed that expense limit it will cut it from your income and give new daily expense allowed amount. Expense tracker will generate report at the end of month to show Income-Expense via multiple graphs.

Expense tracking application system can generate report at the end of week or month to show Income-Expense via multiple graphs. It will let you add the savings amount which you had saved for some particular Festivals like Diwali, Birthdays.

If we exceed the target of our budget, it is automatically generating the notification that will be sent via E-mail.

An email will be sent to the user at the end of each month giving a brief summary of the monthly expenditure. The monthly, and year-wise comparison of expenditures will be done by the app which will let the user know the area where he is spending the most.

The user will be able to see the detailed analyses with the help of graphical visualizations. This project will provide a lot of benefits to the users with the help of which they will be surely able to keep track of each penny.

2.2. REFERENCES

1. IRJET-V6I31110 paper
2. IJIRT150860 paper
3. 8759-Article Text-15701-1-10-20210611
4. Spendee.com
5. <http://expense-manager.com/how-expense>

2.3.PROBLEM STATEMENT DEFINITION

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Hari	Set an alert and notification messages while exceeds her salary limit.	I am do not have awareness in spending money.	Then only she will not spend more money.	Excited.
PS-2	Gugan	To maintain a privacy in heraccount.	She needs privacy.	The account login details should not be hacked.	User friendly.
PS-3	Bharathi	The best investment Plans.	Who needs high returns from investment plan.	He need highreturns fromthe investment	More efficient.
PS-4	Ajith	An bug free app for easyaccess.	Needs bug free app for easy access.	She complete the transaction in faster way.	Curious.
PS-5	Kumar	To know thepremium schedule.	Alert for premium schedule.	To reduce the premium monthly expenses for paying premium	Not smart enough.

3. IDEATION & PROPOSED SOLUTION

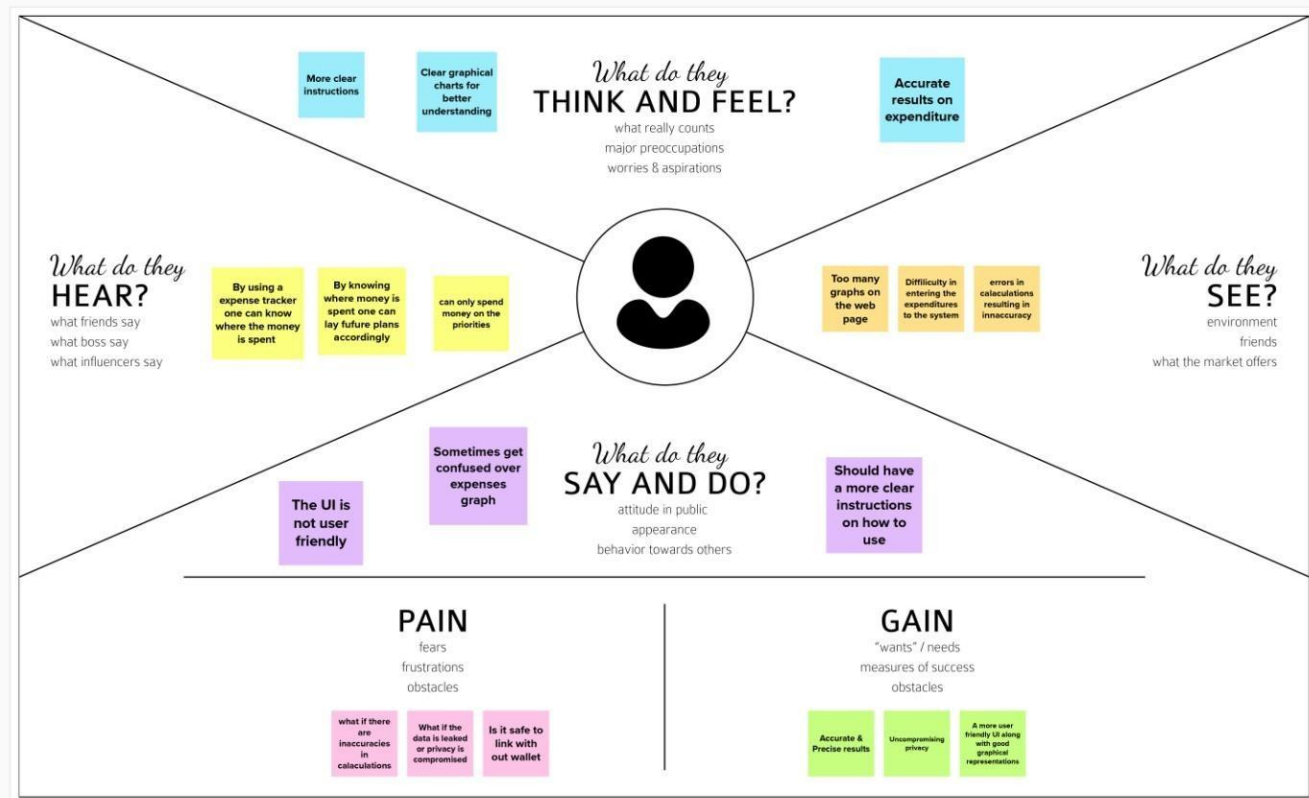
3.1. EMPATHY MAP CANVAS

An empathy map canvas helps brands provide a better experience for users by helping teams understand the perspectives and mindset of their customers. Using a template to create an empathy map canvas reduces the preparation time and standardizes the process so you create empathy map canvases of similar quality

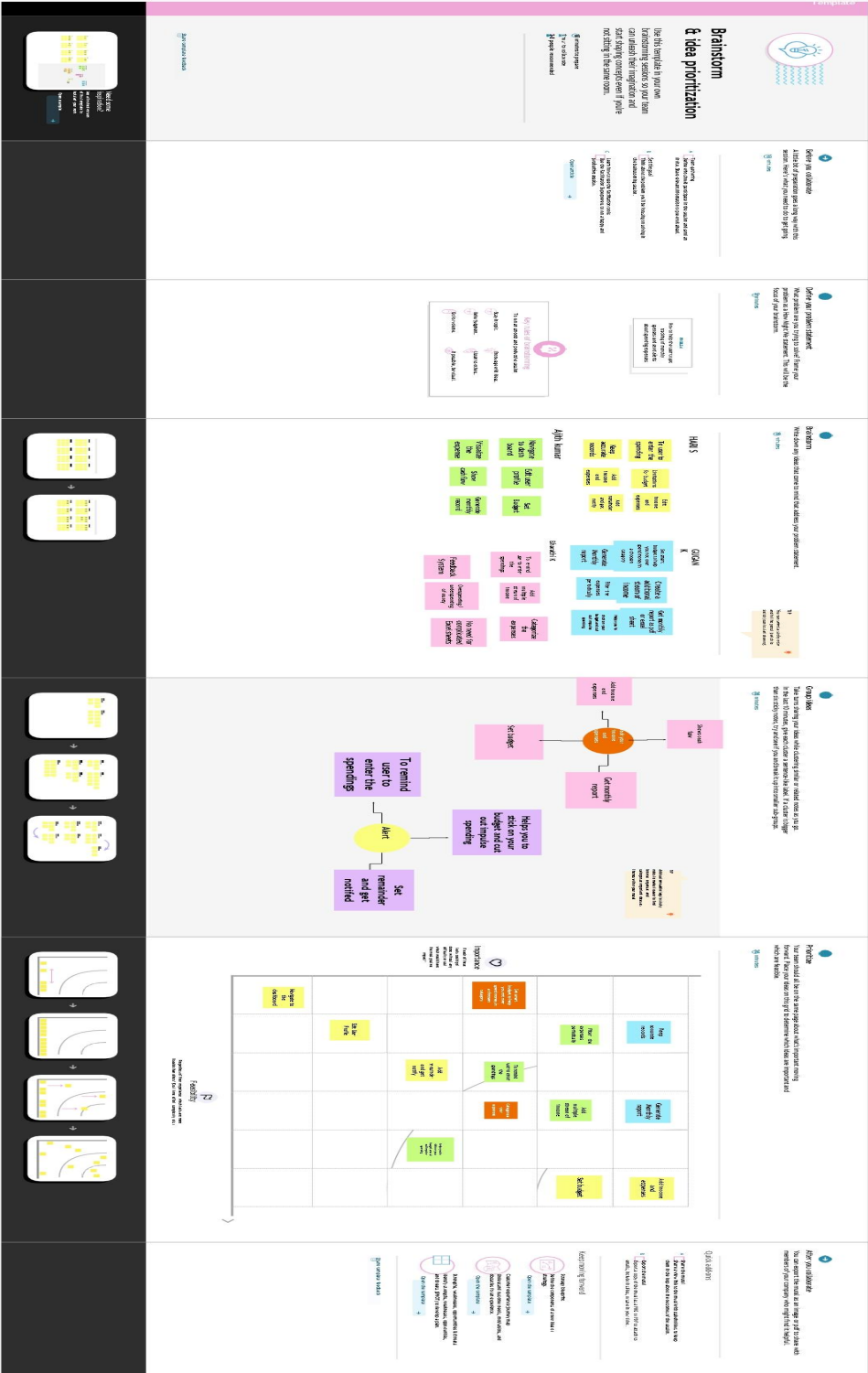
Empathy Map Canvas

Gain insight and understanding on solving customer problems.

Personal Expense Tracker Application



3.1. IDEATION & BRAINSTORMING



3.2. PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	There are a handful of budgeting tools online, but not all of them are effective in assisting users in actually creating and adhering to a budget. The ongoing maintenance, the consolidation of all user financial accounts and activity into a single dashboard are some of the negatives. However, a lot of this current software includes convoluted features that are difficult to use. The major problem is to take count paper receipts and calculate the expense statistics.
2.	Idea / Solution description	People tend to neglect their budgets due to their busy and chaotic lifestyle, which results in spending more than they planned. Future costs cannot be foreseen by the user. Their carelessness with money management will be a concern even though they can record their expense.
3.	Novelty/Uniqueness	Including all the expense including money spend using cash, Bank-cheques, etc. This application keeps track of all of your spending. To enter your expense, simply click and enter the data. To decrease human error, prevent data loss.
4.	Social Impact/ Customer Satisfaction	One can keep track of their costs and create a monthly or annual budget with this tool. The application will display the statistics and sent alert message, if your spending exceeds the specified limit that was specified.
5.	Business Model (Revenue Model)	The subscription/premium to access extra features of this application can be used by business people, and also adding advertisement to generate revenue for free access will help in Increasing customer base.
6.	Scalability of the Solution	Using IBM-cloud statistics can be shown to the user with high scalability, and with high accuracy.

3.3. PROBLEM SOLUTION FIT

Project Title: Personal Expense Tracker

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMID10256

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) Who is your customer? <i>Well-off professionals</i>	6. CUSTOMER CONSTRAINTS What constraints prevent your customers from taking action or limit their choices of solutions? <ul style="list-style-type: none"> • spending power • budget • no cash • network connection • available devices 	5. AVAILABLE SOLUTIONS Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? Solution available to the customers to get the job done is by CHAI BOT. If any concerns regarding application customer needs the admin or make reviews on social media as solution in the past. Pros: user friendly UI Cons: manual categorization of input data	Explore AS, differentiate

Focus on J&P, fit into BE, understand RC	2. JOBS-TO-BE-DONE / PROBLEMS Which jobs-to-be-done (or problems) do you address for your customers? <ul style="list-style-type: none"> • Tracking expenses • Visualizing • Budget planning 	9. PROBLEM ROOT CAUSE What is the real reason that this problem exists? What is the back story behind the need to do this job? Customers are: <ul style="list-style-type: none"> • Too busy to budget • Hard to keep up source of income and expenditures 	7. BEHAVIOUR What does your customers do to address the problem, adopt the job done? Directly related: find the right expense tracker, calculate usage and benefits. Indirectly associated: customers automatically gain knowledge on finances

3. TRIGGERS What triggers customers to act? <ul style="list-style-type: none"> • Seeing friends and family using the app • Inability to plan their expenses • Ease of tracking expenses 	10. YOUR SOLUTION To develop the expense tracker application to help customer for budgeting, accounting and creating awareness about money management and saving.	8. CHANNELS of BEHAVIOUR 8.1 ONLINE What kind of actions do customers take online? Search for user friendly expense tracking application. 8.2 OFFLINE What kind of actions do customers take offline? Gain finance knowledge by using the application and use them for their development.
4. EMOTIONS: BEFORE/ AFTER How do customers feel when they face a problem of a job and afterwards? Regret to save > desired to achieve FIRE goal Wasting money > saving money		

4. REQUIREMENT ANALYSIS

4.1. FUNCTIONAL REQUIREMENT

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement	Sub Requirement
1	User Registration	Registration through Form, Registration through Gmail.
2	User Confirmation	Confirmation viaEmail.
3	User Data	Data gathered in the application server is saved in the highsecurity cloud server.
4	Alert Notification	Alert messages through the Email.
5	User Monthly Budget Plan	Setting Monthly budget to manage their expenses.
6	Cloud Data Storage	To save the user valuable data high security cloud storage lineIBM Cloud are used.

4.2. NON-FUNCTIONAL REQUIREMENTS

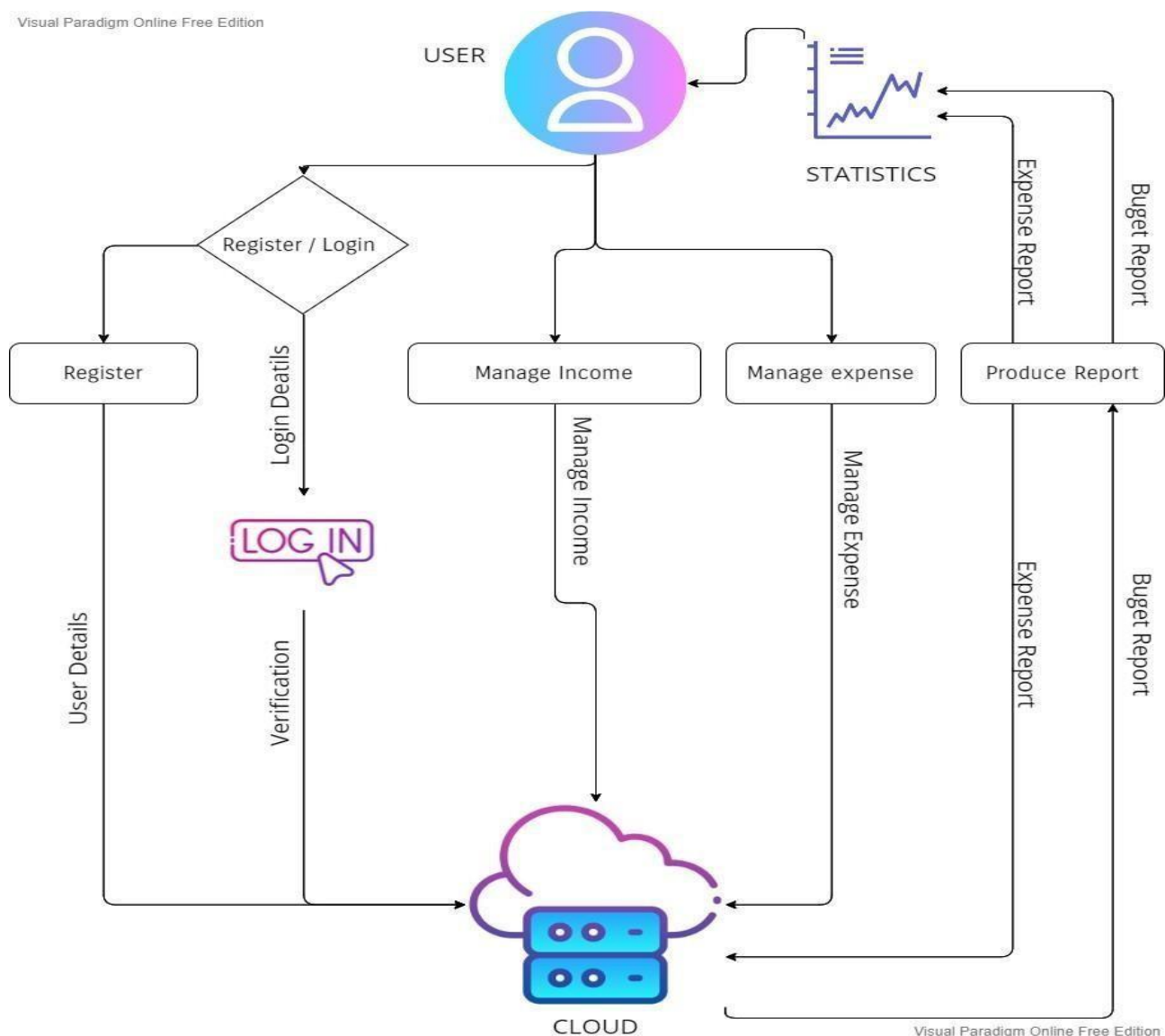
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
1	Usability	Effectiveness, efficiency, and overall experience of the userinteracting with our application should be maximized.
2	Security	Authentication, authorization, encryption of the data mustbe done in the application.
3	Reliability	Probability of error in the operations in a specified environment for a specified time should be minimized.
4	Performance	How the application is functioning accurately and effectivelythe application is to the end-users.
5	Availability	Using Cloud Storage and database, application reliability andthe user satisfaction will affect the solution
6	Scalability	Capacity of the application to handle growth, especially in handling more users.

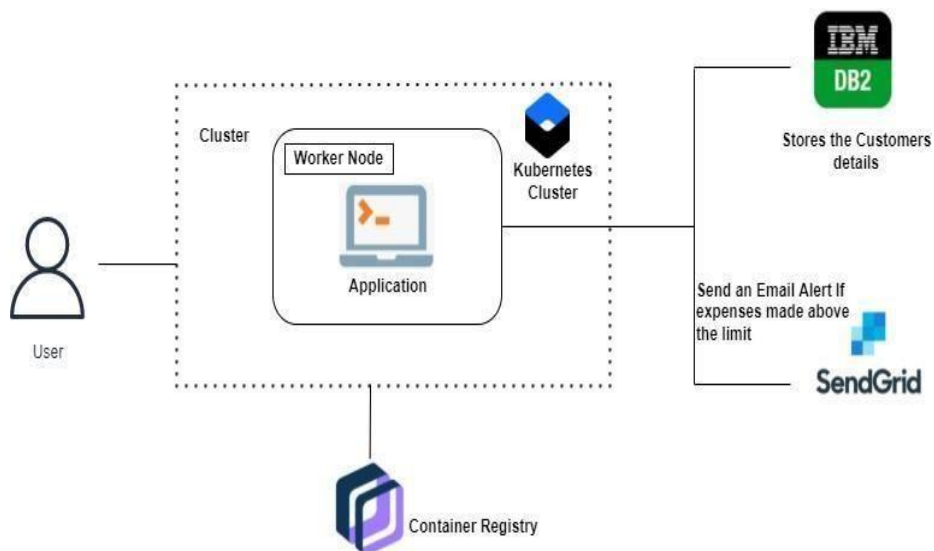
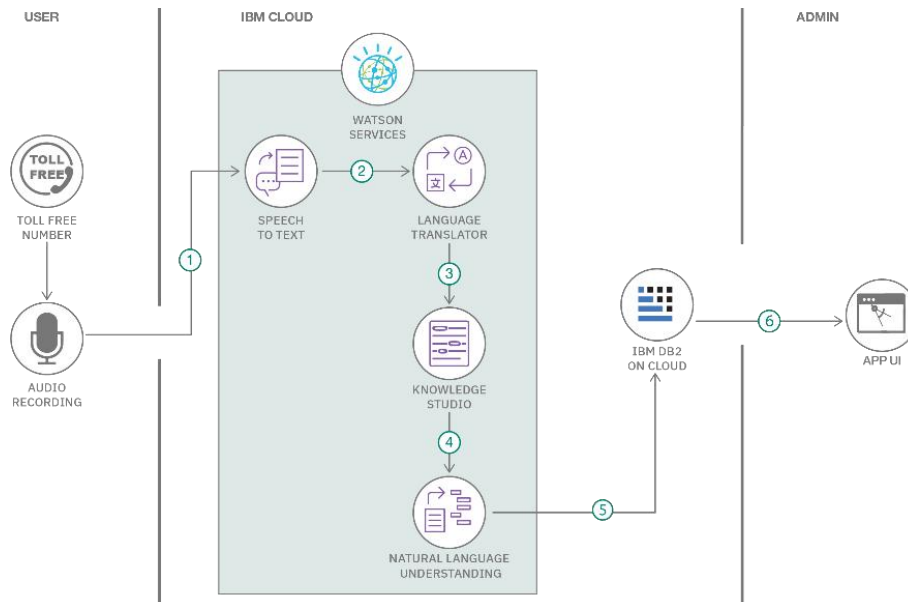
5. PROJECT DESIGN

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 enter and leave the system, what changes the information, and where data is stored.



5.2. SOLUTION AND TECHNOLOGY ARCHITECTURE



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	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-4	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard		As a user, I can access my detail, manage the expense, add budget, expense report from the app etc..		High	Sprint-1
Customer (Webuser)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	Login	USN-2	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-3	As a user, I can access my detail, manage the expense, add budget, expense report from the app etc..		High	Sprint-1
Customer Care Executive	Email or Customer Care number		As a user, I can contact the service administration for the support.	I can solve the Issue.	High	Sprint-3

6. PROJECT PLANNING AND SCHEDULING

6.1. SPRINT PLANNING AND ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	JSN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Hari S
Sprint-1		JSN-2	As a user, I will receive confirmation once I have registered for the application	1	High	Ajith
Sprint-1	Login	JSN-3	As a user, I can log into the application by entering email & password	1	High	Bharathi
Sprint-1	Dashboard	JSN-4	Logging in takes to the dashboard for the logged user	2	High	Ajith
Sprint-2	Workspace	JSN-1	Workspace for personal expense management	2	High	Hari S
Sprint-2	Charts	JSN-2	Creating various graphs and statistics of customer's	1	Medium	Bharathi
Sprint-2	Connecting to IBM DB2	JSN-3	Linking database with dashboard	2	High	Gugan K
Sprint-2		JSN-4	Making dashboard interactive with JS	2	High	Ajith

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Hari S
Sprint-3	Watson Assistant	USN-2	Creating chatbot for expense tracking and for clarifying user's query	1	Medium	Gugan K
Sprint-3	SendGrid	UNS-3	Using SendGrid to send mail to the user about their expenses	1	Low	Bharathi
Sprint-3		UNS-4	Integrating both frontend and backend	2	High	Ajith
Sprint-4	Docker	UNS-1	Creating image of website using docker	2	High	Hari S
Sprint-4	Cloud Registry	UNS-2	Upload docker image to IBM Cloud registry	2	High	Gugan K
Sprint-4	Kubernetes	UNS-3	Create container using the docker image and hosting the site	2	High	Ajith
Sprint-4	Exposing	UNS-4	Exposing IP/ports for the site	2	High	Bharathi

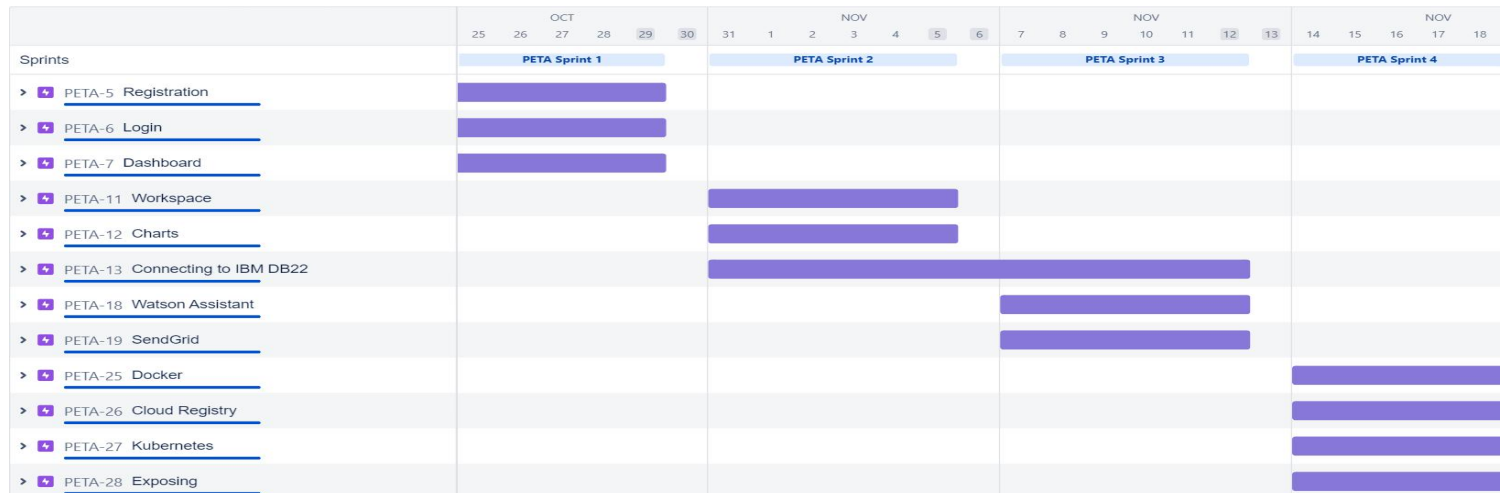
6.2 SPIRNT DELIVERY SCHEDULE

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

6.3.REPORTS FROM JIRA

BURNDOWN CHART:

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



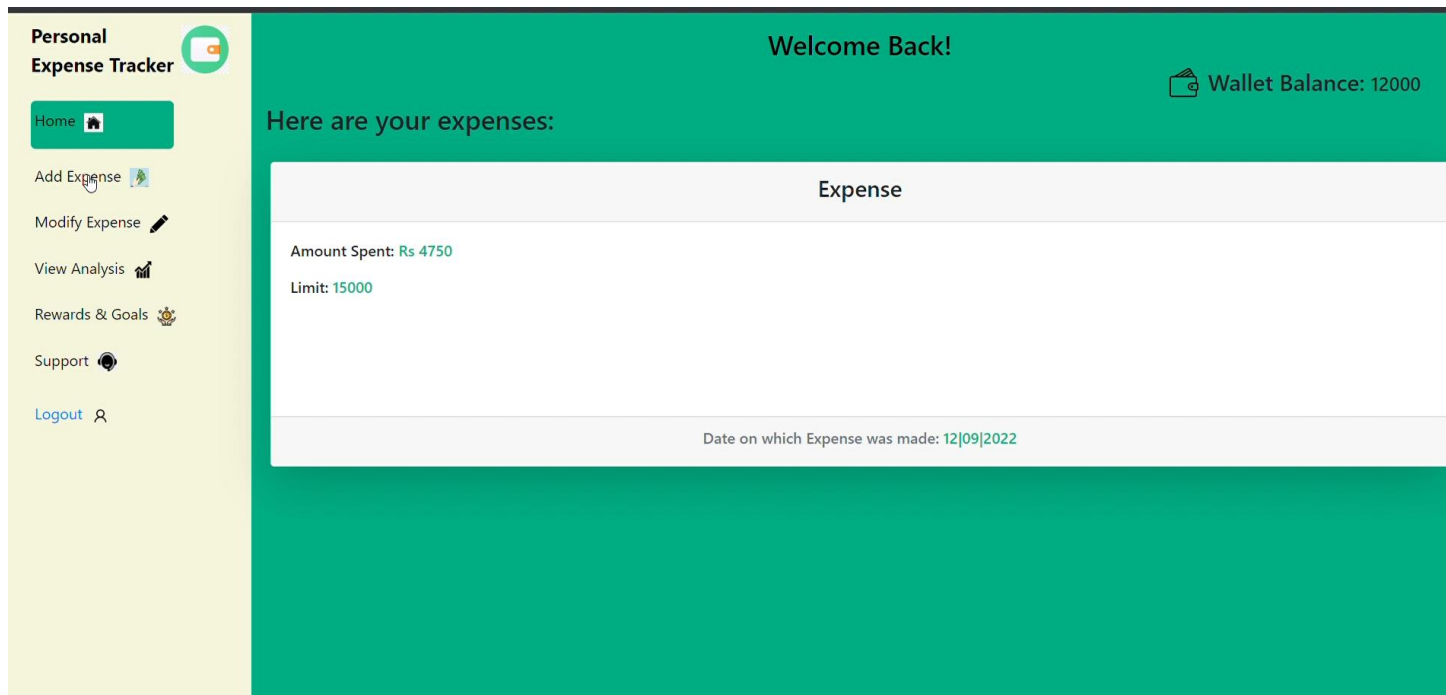
7. CODING AND SOLUTIONING

7.1. FEATURE 1

An alert email will be sent once the user crosses their expense limit that was set beforehand.

Pic 1:

Wallet and expense details



Pic 2:

Adding expense

The screenshot shows the 'Add expense' form in the Personal Expense Tracker app. The form is titled 'Expense Made' and is set against a teal background. It contains the following fields:

- Amount Spent: (Rs)**: A text input field containing '12000'.
- Expense Category:**: A dropdown menu with 'Shopping' selected.
- Date of Expense:**: A date picker showing '11/18/2022'.
- Description of Expense:**: A text input field containing 'Bought clothes'.

A green 'Submit Expense' button is located at the bottom right of the form. The left sidebar of the app is visible, showing navigation options: Home, Add Expense, Modify Expense, View Analysis, Rewards & Goals, Support, and Logout.

Pic 3:

Wallet and Expense updated.

The screenshot shows the 'Welcome Back!' screen in the Personal Expense Tracker app. The screen displays the following information:

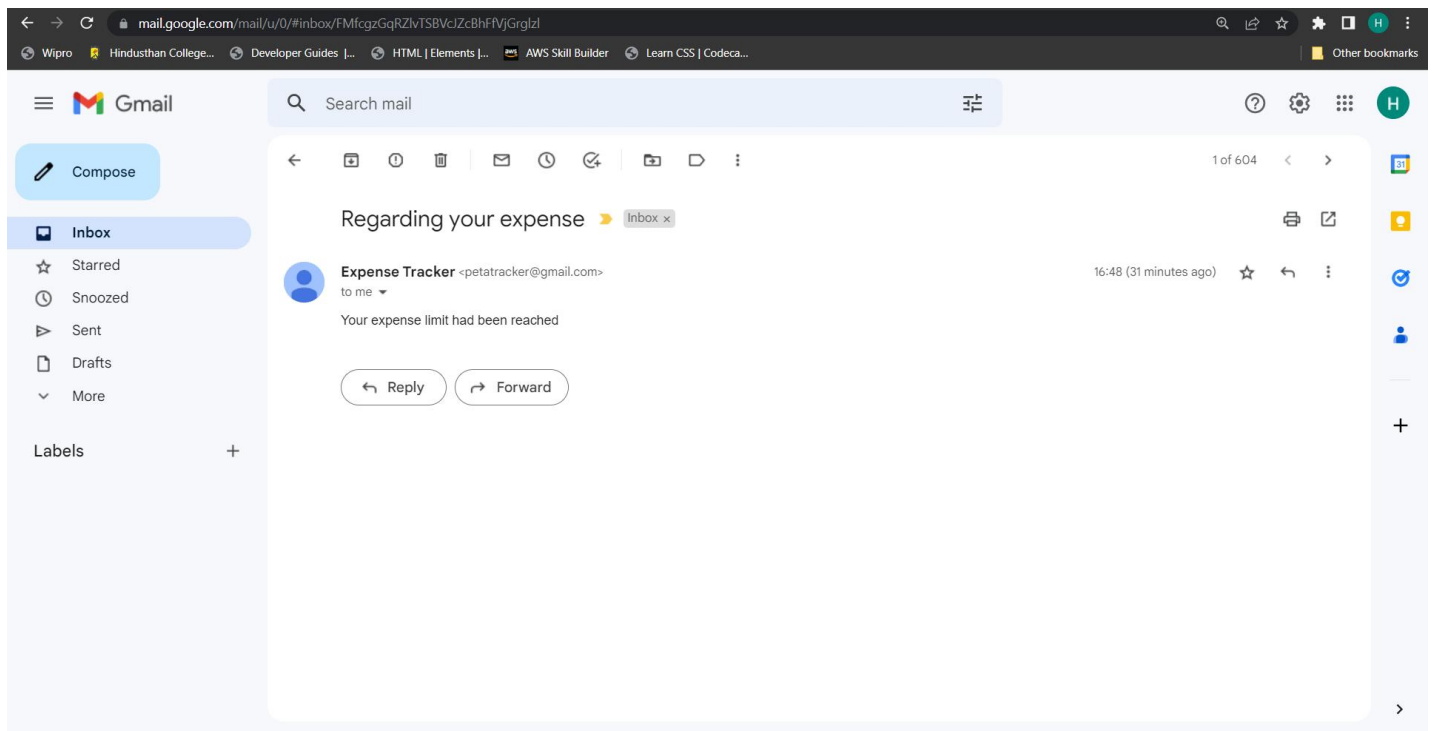
- Welcome Back!**: A greeting message at the top.
- Wallet Balance: 0**: A status indicator in the top right corner.
- Here are your expenses:**: A heading for the expense list.
- Expense**: A table with the following data:

Expense
Amount Spent: Rs 16750
Limit: 15000
Date on which Expense was made: 12/09/2022

The left sidebar of the app is visible, showing navigation options: Home, Add Expense, Modify Expense, View Analysis, Rewards & Goals, Support, and Logout.

Pic 4:

Alert email sent once the expense limit was crossed.



7.2. CODE FOR THAT FEATURE

HTML code for front end:

#dashboard

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

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfEpd3yD65Vohhpuc0mLASjC" crossorigin="anonymous">

    <!-- bootstrap for the cards -->
    <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">
    <title>Dashboard</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>

  <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: beige">
        <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; height: 50px; width: 150px; border-radius:
5px;">
              <a href="Dashboard.html" 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">
              <a href="add.html" 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">
              <a href="updatebalance.html" class="nav-link px-0 align-middle" style="color:black;">
                <span class="ms-1 d-none d-sm-inline">Modify Expense</span>
                
              </a>
            </li>
            <li class="nav-item mt-2">
              <a href="display.html" class="nav-link px-0 align-middle" style="color:black;">
                <span class="ms-1 d-none d-sm-inline">View Analysis</span>
                
              </a>
            </li>
          </ul>
        </div>
      </div>
    </div>
  </div>
```


#addexpense

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

    <!-- Bootstrap CSS -->
    <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>AddExpense</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>

  <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: beige">
        <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">
              <a href="dashboard.html" 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; height: 50px; width: 150px; border-
radius: 5px;">
              <a href="add.html" 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">
              <a href="updatebalance.html" class="nav-link px-0 align-middle" style="color:black;">
                <span class="ms-1 d-none d-sm-inline">Modify Expense</span>
              </a>
            </li>
            <li class="nav-item mt-2">
              <a href="display.html" 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">
              <a href="rewards.html" 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">
              <a href="support.html" class="nav-link px-0 align-middle" style="color:black;">
```

```

        <span class="ms-1 d-none d-sm-inline">Support</span>
        
    </a>
</li>
</li>
<li class="nav-item mt-2">
    <a href="logout.html" class="nav-link px-0 align-middle" style="color:black;">
        <a href="login.html"><span class="ms-1 d-none d-sm-inline">Logout</span></a>
        
    </a>
</li>
</ul>
</div>
</div>
<div class="col py-3" style="background-color: #00AD83">
    <h3 style="color:black; text-align: center;">Add expense</h3>
    <div class="container mt-3" style="width: 600px;">
        <div class="card shadow-lg bg-white rounded">
            <div class="card-header" style="text-align: center;">
                <span style="display:inline-flex"><h4>Expense Made</h4></span>
            </div>
            <div class="card-body">
                <form>
                    <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">
                    </div>
                    <div class="mb-3">
                        <label for="expensecategory" class="form-label">Expense Category: </label>
                        <input type="text" class="form-control" name="expensecategory" id="expensecategory"></input>
                    </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"></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>
                </form>
            </div>
            <div class="card-footer text-muted" style="text-align:center">
                <button type="submit" style="background-color:#00AD83; border-color:#00AD83; border-
radius:5px;">Submit Expense</button>
            </div>
        </div>
    </div>
</div>
</div>
</div>
</html>

```

#modify expense

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

    <!-- Bootstrap CSS -->
    <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>Update Balance</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>

  <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: beige">
        <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">
              <a href="Dashboard.html" 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" >
              <a href="add.html" 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; height: 50px; width: 150px; border-radius:
5px;">
              <a href="updatebalance.html" class="nav-link px-0 align-middle" style="color:black;">
                <span class="ms-1 d-none d-sm-inline">Modify Expense</span>
              </a>
            </li>
            <li class="nav-item mt-2">
              <a href="display.htmls" 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">
              <a href="rewards.html" 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">
              <a href="support.html" class="nav-link px-0 align-middle" style="color:black;">
```



```

s.starttls()
s.login("petatracker@gmail.com", "lxixbnpnxbkiemh")
message = 'Subject: {} \n\n {}'.format(SUBJECT, TEXT)
s.sendmail("il.petatracker@gmail.com", email, message)
s.quit()
def sendgridmail(user,TEXT):

    # from_email = Email("petatracker@gmail.com")
    from_email = Email("petatracker@gmail.com")
    to_email = To(user)
    subject = "Your expense limit had been reached"
    content = Content("text/plain",TEXT)
    mail = Mail(from_email, to_email, subject, content)

    # Get a JSON-ready representation of the Mail object
    mail_json = mail.get()
    # Send an HTTP POST request to /mail/send
    response = sg.client.mail.send.post(request_body=mail_json)
    print(response.status_code)
    print(response.headers)

```

#login.css

```

*{
    padding: 0;
    margin: 0;
    box-sizing: border-box;
}

body{
    font-family: 'Poppins', sans-serif;
    overflow: hidden;
}

.wave{
    position: fixed;
    bottom: 0;
    left: 0;
    height: 100%;
    z-index: -1;
}

.container{
    width: 100vw;
    height: 100vh;
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    grid-gap :7rem;
    padding: 0 2rem;
}

.img{
    display: flex;
    justify-content: flex-end;
    align-items: center;
}

.login-content{
    display: flex;
    justify-content: flex-start;

```



```

    align-items: center;
    text-align: center;
}

.img img{
    width: 500px;
}

form{
    width: 360px;
}

.login-content img{
    height: 100px;
}

.login-content h2{
    margin: 15px 0;
    color: #333;
    text-transform: uppercase;
    font-size: 2.9rem;
}

.login-content .input-div{
    position: relative;
    display: grid;
    grid-template-columns: 7% 93%;
    margin: 25px 0;
    padding: 5px 0;
    border-bottom: 2px solid #d9d9d9;
}

.login-content .input-div.one{
    margin-top: 0;
}

.i{
    color: #d9d9d9;
    display: flex;
    justify-content: center;
    align-items: center;
}

.i i{
    transition: .3s;
}

.input-div > div{
    position: relative;
    height: 45px;
}

.input-div > div > h5{
    position: absolute;
    left: 10px;
    top: 50%;
    transform: translateY(-50%);
    color: #999;
    font-size: 18px;
    transition: .3s;
}

.input-div:before, .input-div:after{
    content: '';
}

```

```

    position: absolute;
    bottom: -2px;
    width: 0%;
    height: 2px;
    background-color: #38d39f;
    transition: .4s;
}

.input-div:before{
    right: 50%;
}

.input-div:after{
    left: 50%;
}

.input-div.focus:before, .input-div.focus:after{
    width: 50%;
}

.input-div.focus > div > h5{
    top: -5px;
    font-size: 15px;
}

.input-div.focus > .i > i{
    color: #38d39f;
}

.input-div > div > input{
    position: absolute;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    border: none;
    outline: none;
    background: none;
    padding: 0.5rem 0.7rem;
    font-size: 1.2rem;
    color: #555;
    font-family: 'poppins', sans-serif;
}

.input-div.pass{
    margin-bottom: 4px;
}

a{
    display: block;
    text-align: right;
    text-decoration: none;
    color: #999;
    font-size: 0.9rem;
    transition: .3s;
}

a:hover{
    color: #38d39f;
}

.btn{
    display: block;

```

```

width: 100%;
height: 50px;
border-radius: 25px;
outline: none;
border: none;
background-image: linear-gradient(to right, #32be8f, #38d39f, #32be8f);
background-size: 200%;
font-size: 1.2rem;
color: #fff;
font-family: 'Poppins', sans-serif;
text-transform: uppercase;
margin: 1rem 0;
cursor: pointer;
transition: .5s;
}
.btn:hover{
background-position: right;
}

@media screen and (max-width: 1050px){
.container{
grid-gap: 5rem;
}
}

@media screen and (max-width: 1000px){
form{
width: 290px;
}

.login-content h2{
font-size: 2.4rem;
margin: 8px 0;
}

.img img{
width: 400px;
}
}

@media screen and (max-width: 900px){
.container{
grid-template-columns: 1fr;
}

.img{
display: none;
}

.wave{
display: none;
}

.login-content{
justify-content: center;
}
}

.container{
overflow: scroll
}

.container::-webkit-scrollbar {
display: none;
}

```

```

ul {
  position: relative;
  top: -20px;
  left: 0%;
  right: 10%;
  transform: translate(-50%, -50%);
  margin: 75px;
  padding: 0;
  display: flex;
  flex: auto;
}

ul li {
  list-style: none;
}

ul li a {
  position: relative;
  width: 60px;
  height: 60px;
  display: block;
  text-align: center;
  margin: 0 10px;
  border-radius: 50%;
  padding: 6px;
  box-sizing: border-box;
  text-decoration: none;
  box-shadow: 0 10px 15px rgba(0,0,0,0.3);
  background: linear-gradient(0deg, #ddd, #fff);
  transition: .5s;
}

ul li a: hover {
  box-shadow: 0 2px 5px rgba(0,0,0,0.3);
  text-decoration: none;
}

ul li a .fab {
  width: 100%;
  height: 100%;
  display: block;
  background: linear-gradient(0deg, #fff, #ddd);
  border-radius: 50%;
  line-height: calc(60px - 12px);
  font-size: 24px;
  color: #262626;
  transition: .5s;
}

ul li: nth-child(1) a: hover .fab {
  color: #3b5998;
}

ul li: nth-child(2) a: hover .fab {
  color: #00aced;
}

ul li: nth-child(3) a: hover .fab {
  color: #dd4b39;
}

ul li: nth-child(4) a: hover .fab {
  color: #007bb6;
}

```

```
}

ul li:nth-child(5) a:hover .fab {
  color: #e4405f;
}
.app{
  position: relative;
  top: -70px;
  height: 5%;
}
#app1{
  font-style: oblique;
  color:blue ;
}
#png{
  position: relative;
  top: -300px;
  right: 50px;
}

}
```

8. TESTING

8.1 TEST CASES

Testing is the process of evaluation a software item to detect differences between given input and expected output. Testing is a process should be done during the development phase.

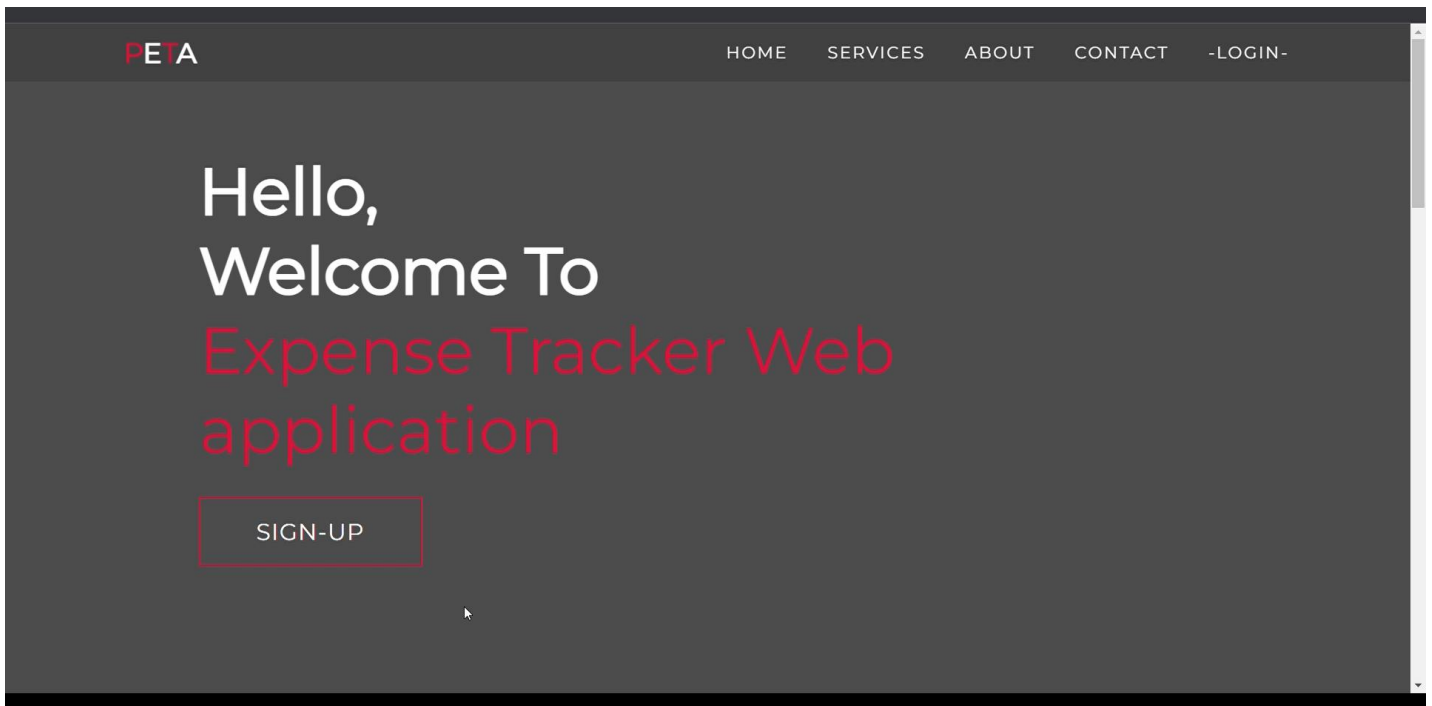
Test Case Id	Test Description	Input Test Data	Expected Result	Actual Result	Remarks
TC-1	Install PET app in android phone	Transfer PET app	Open Application with it home page	Application executed with home page	Pass
TC-2	Enter valid data in username and password field	Ajithkumar@gmail	Show home page for user	Displayed home page for user Ajith	Pass
TC-3	Enter a valid data in username and leave password field empty	Harisekar180@gmail.com	Show error	Didn't show any error	Pass

8.2.USER ACCEPTANCE TESTING

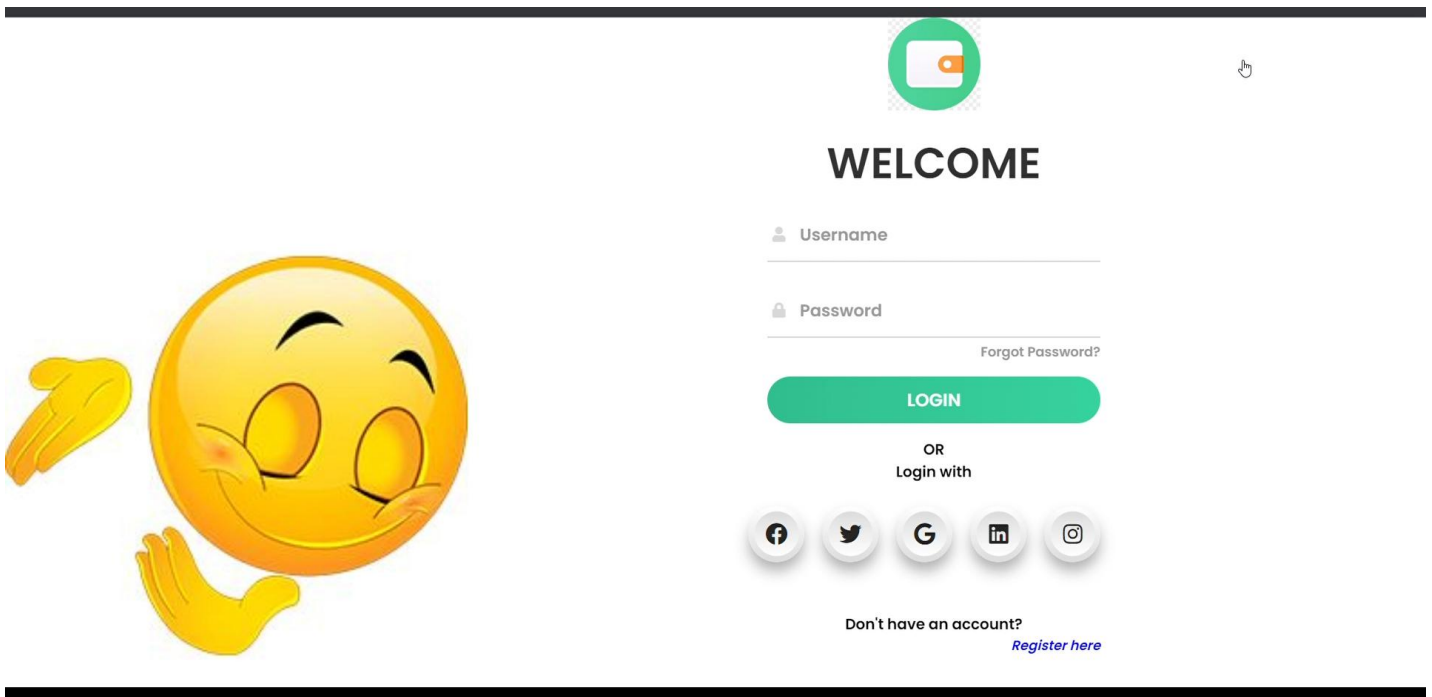
ID	TEST CASES	PASS/ FAIL	TESTED BY	TESTED ON
8.1	LOGIN	PASS	Ajith	11/11/2022
8.2	ADD EXPENSE	PASS	Bharathi	12/11/2022
8.3	MODIFY EXPENSE	PASS	Hari	12/11/2022
8.4	ANALYSIS	FAIL	Gugan	14/11/2022
8.5	LOGOUT & REDIRECTIONS	PASS	Hari	16/11/2022

9. RESULTS

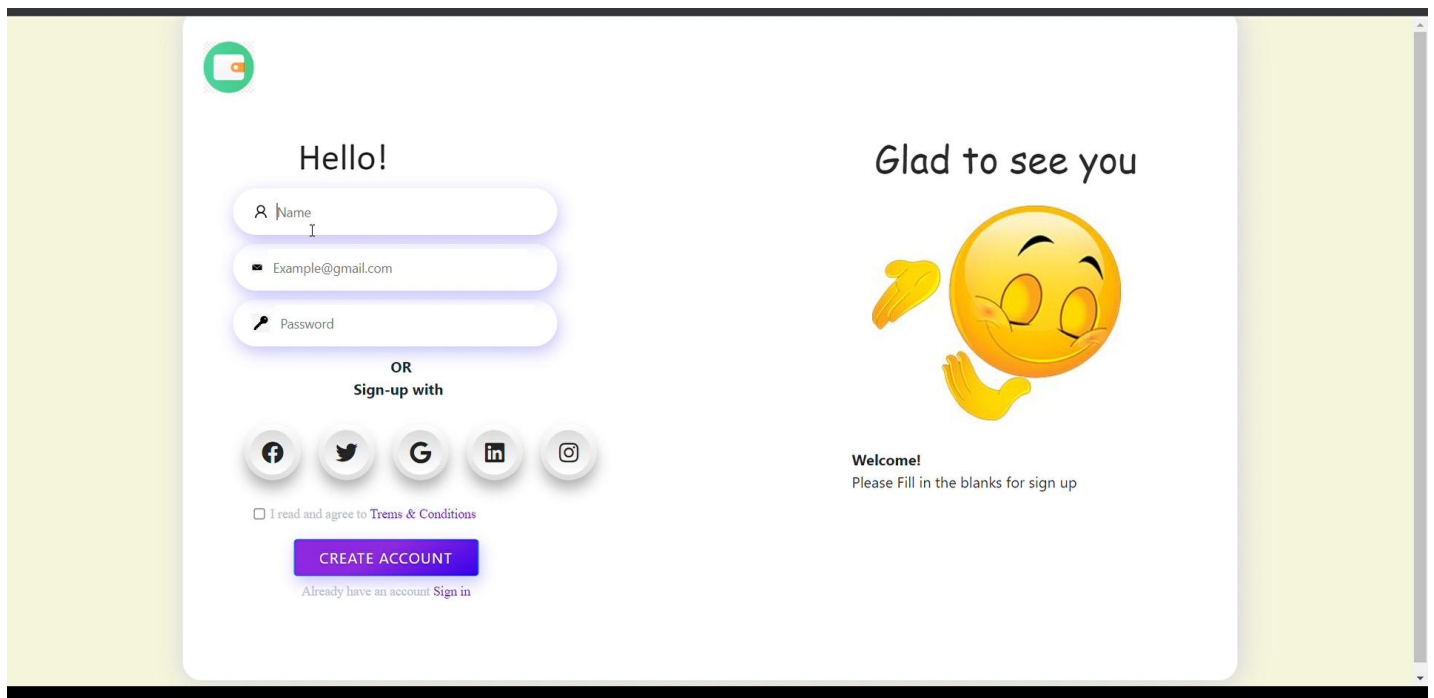
HOME-PAGE



LOGIN PAGE



SIGN UP PAGE



A sign-up page with a light green background. On the left, there's a white card with a green profile icon at the top left. The card contains the text "Hello!" followed by three input fields: "Name", "Email" (with placeholder "Example@gmail.com"), and "Password". Below these is a section "OR Sign-up with" with five social media icons: Facebook, Twitter, Google, LinkedIn, and Instagram. At the bottom of the card is a checkbox "I read and agree to Terms & Conditions", a purple "CREATE ACCOUNT" button, and a link "Already have an account Sign in". On the right, the text "Glad to see you" is above a large yellow smiling emoji with its hands up. Below the emoji, it says "Welcome!" and "Please Fill in the blanks for sign up".

Hello!

Name

Example@gmail.com

Password

OR
Sign-up with

Facebook Twitter Google LinkedIn Instagram

☐ I read and agree to Terms & Conditions

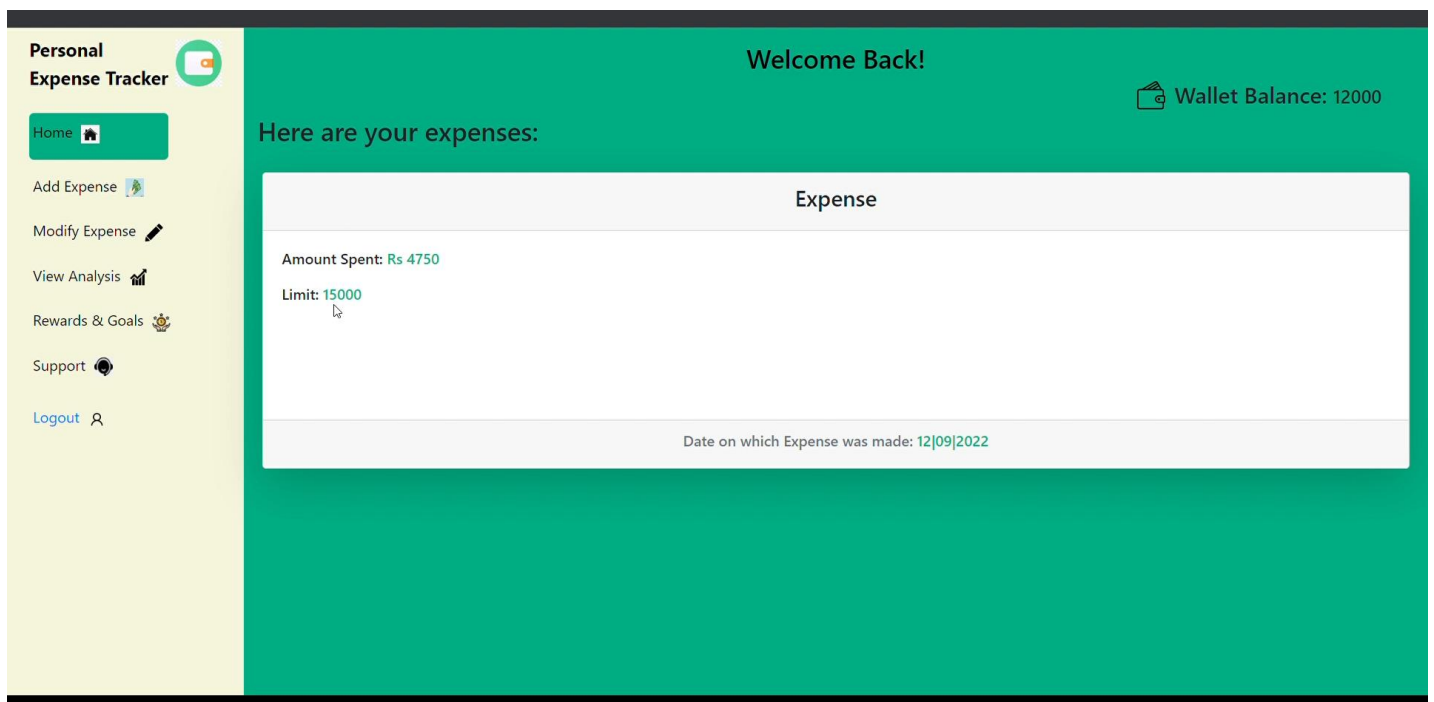
CREATE ACCOUNT

Already have an account Sign in

Glad to see you

Welcome!
Please Fill in the blanks for sign up

HOME DASHBOARD



A home dashboard for a "Personal Expense Tracker". The left sidebar is light green and contains a menu with "Home" (active), "Add Expense", "Modify Expense", "View Analysis", "Rewards & Goals", "Support", and "Logout". The main area has a teal header with "Welcome Back!" and "Wallet Balance: 12000". Below the header, it says "Here are your expenses:" followed by a white box. Inside the box, it shows "Amount Spent: Rs 4750" and "Limit: 15000". At the bottom of the box, it says "Date on which Expense was made: 12|09|2022".

Personal Expense Tracker

Home

Add Expense

Modify Expense

View Analysis

Rewards & Goals

Support

Logout

Welcome Back!

Wallet Balance: 12000

Here are your expenses:

Expense

Amount Spent: Rs 4750

Limit: 15000

Date on which Expense was made: 12|09|2022

ADD EXPENSE PAGE

Personal Expense Tracker

Home

Add Expense

Modify Expense

View Analysis

Rewards & Goals

Support

Logout

Add expense

Expense Made

Amount Spent: (Rs)

100.00

Expense Category:

Date of Expense:

mm/dd/yyyy

Description of Expense:

Submit Expense

MODIFY EXPENSE PAGE

Personal Expense Tracker

Home

Add Expense

Modify Expense

View Analysis

Rewards & Goals

Support

Logout

Update Balance

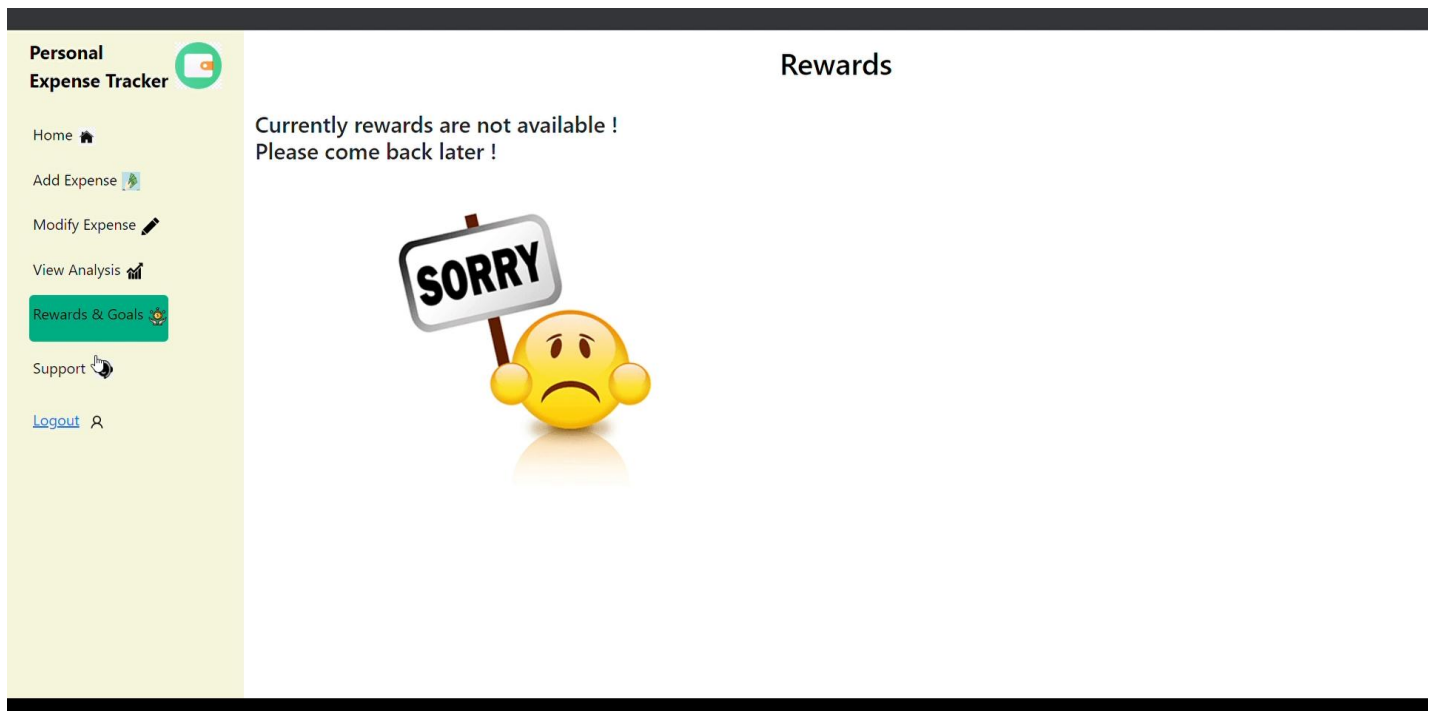
Wallet Balance

Current Balance: 12000

New Balance:

Update Balance

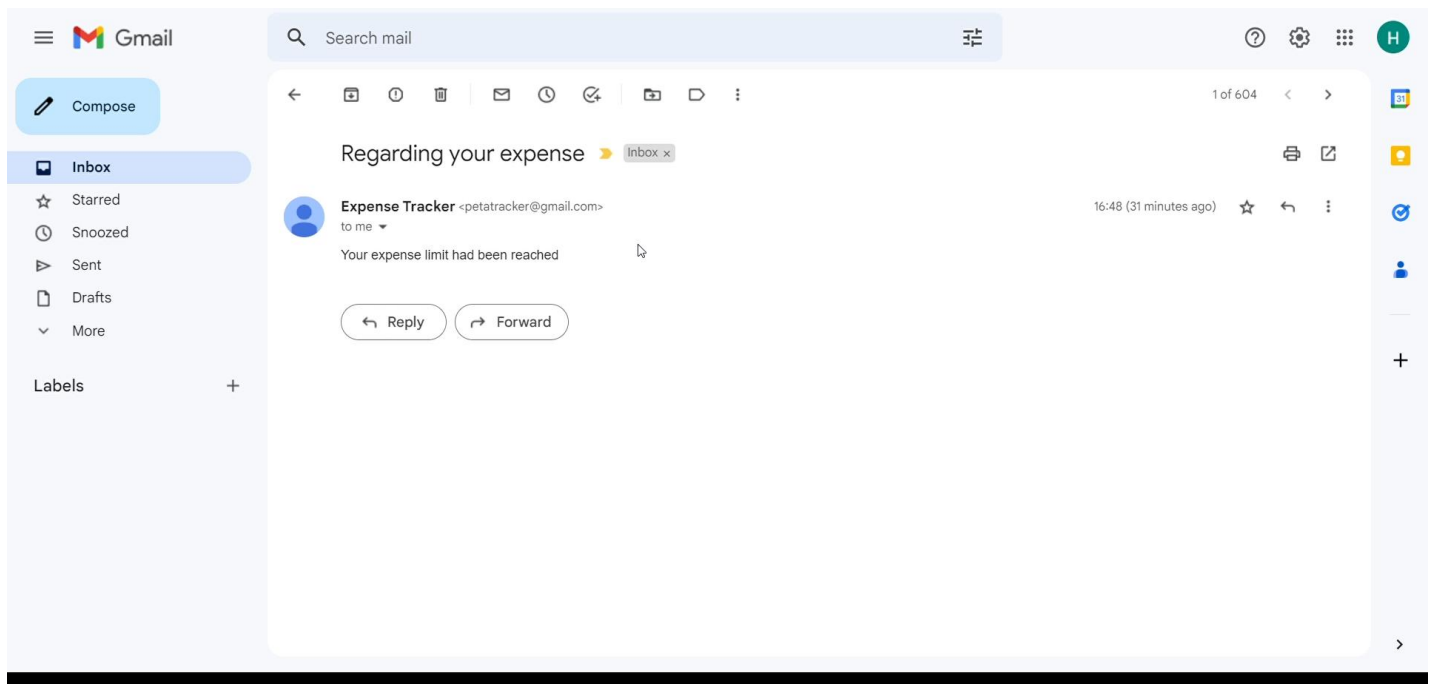
REWARD PAGE



SUPPORT PAGE



E-MAIL ALERT



10. ADVANTAGES AND DISADVANTAGES

ADVANTAGES

1. You have no control over your money

If you don't check your spending and create a budget, you will have no control whatsoever on your money. Instead, money will control you, and you will either have perpetual lack of funds or you will end up steeped in debt. A money manager app helps you decide between short-term and long-term spending .

2. You have no financial goals

If you are spending money frivolously, you will not have money to set financial goals. However, when you have a daily expense manager, you will be able to work with limited resources and use your money in a wise manner so that you can create financial goals and ensure you meet them.

3. You are unaware what is happening with your money

If you are clueless about how much is your inflow and how much you are spending, you will not know at the end of the month what happened to your money. An expense tracker helps you figure out what is happening to your money, and whether you can afford something you want.

4. You spend and save in a haphazard manner

If you don't have great financial management skills, you will not know how to categorize your expenses. However, tracking your expenses and budgeting them will help you become aware of how much you have to allocate to each expense category, and if you are short, you will be able to make adjustments with ease.

5. You have no clue about making your money work for you

In this day and age, when expenses are going through the roof, it has become crucial that you learn to make your money work for you so that you can create a nest egg for the future.

6. You don't have funds for emergencies

Remember, emergencies come when you least expect. Hence, if you don't have money stashed away for a rainy day, you will end up borrowing from family and friends. This way you could get into debt that will be difficult to pay back due to your poor money management skills

DISADVANTAGES

INTERNET CONNECTION IS REQUIRED

Since the application runs using cloud in order for using it you need to have consistent and stable internet connection for availing any and all the features available.

DATA SECURITY

Even though we have used IBM cloud for storage which has good security we can't claim that this application has impeccable security which cannot be hacked or cracked.

PRIVACY

Regardless of the user and their data , People won't feel comfortable sharing their each and every expenses to an third party online application which would affect their privacy.

ACCURACY

Even though the application has good level of accuracy in calculations but it's completely based on user's input which cannot be said to be completely accurate.

11. CONCLUSION

Monitoring your everyday expenses can set aside you cash, yet it can likewise help you set your monetary objectives for what's to come. On the off chance that you know precisely where your sum is going much of a stretch see where a few reductions and bargains can be made. Expense Tracker project is for keeping our day-to-day expenditures will helps us to keep record of our money daily. The project what we have created is work more proficient than the other income and expense tracker. The project effectively keeps away from the manual figuring for trying not to ascertain the pay and cost each month. It's a user-friendly application

After making this application we assure that this application will help its users to manage the cost of their daily expenditure. It will guide them and aware them about there daily expenses. It will prove to be helpful for the people who are frustrated with their daily budget management, irritated because of amount of expenses and wishes to manage money and to preserve the record of their daily cost which may be useful to change their way of spending money. In short, this application will help its users to overcome the wastage of money

12. FUTURE SCOPE

- 1) It will have various options to keep record (for example Food, Travelling Fuel, Salary etc.
- 2) Automatically it will keep on sending notifications for our daily expenditure.
- 3) In today's busy and expensive life, we are in a great rush to make moneys, but at the end of the month we broke off. As we are unknowingly spending money on title and unwanted things. So, we have come over with the plan to follow our profit.
- 4) Here user can define their own categories for expense type like food, clothing, rent and bills where they have to enter the money that has been spend and likewise can add some data in extra data to indicate the expense.

13. APPEDIX

SOURCE CODE

#app.py

```
from flask import Flask, render_template, request, redirect, session
from flask_mysql import MySQL
import MySQLdb.cursors
import re

from flask_db2 import DB2
import ibm_db
import ibm_db_dbi
from sendmail import sendgridmail, sendmail
from gevent.pywsgi import WSGIServer
import os

app = Flask(__name__)
db = DB2(app)

app.secret_key = 'a'

# app.config['MYSQL_HOST'] = 'remotemysql.com'
# app.config['MYSQL_USER'] = 'D2DxDUPBii'
# app.config['MYSQL_PASSWORD'] = 'r8XB04GsMz'
# app.config['MYSQL_DB'] = 'D2DxDUPBii'
"""
dsn_hostname = "b1bc1829-6f45-4cd4-bef4-10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
dsn_uid = "jjn48222"
dsn_pwd = "lZgvJpxduESZgt0A"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "32304"
dsn_protocol = "tcPIP"

dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
    "PORT={3};"
    "PROTOCOL={4};"
    "UID={5};"
    "PWD={6};"
).format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol, dsn_uid, dsn_pwd)
"""
# app.config['DB2_DRIVER'] = '{IBM DB2 ODBC DRIVER}'
app.config['database'] = 'bludb'
app.config['hostname'] = 'b1bc1829-6f45-4cd4-bef4-10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud'
app.config['port'] = '32304'
app.config['protocol'] = 'tcPIP'
app.config['uid'] = 'jjn48222'
app.config['pwd'] = 'lZgvJpxduESZgt0A'
app.config['security'] = 'SSL'
try:
```

```

mysql = DB2(app)
conn_str='database=bludb;hostname=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.clogj3sd0tgtu0lqde00.databases.appdomain.cloud;port=32304;protocol=tcPIP;\
uid=jjn48222;pwd=lzgvJpxduESZgt0A;security=SSL'
ibm_db_conn = ibm_db.connect(conn_str, '', '')

print("Database connected without any error !!")
except:
    print("IBM DB Connection error : " + DB2.conn_errormsg())

# app.config['']
# mysql = MySQL(app)

#HOME--PAGE
@app.route("/home")
def home():
    return render_template("home.html")

@app.route("/")
def add():
    return render_template("home.html")

#SIGN--UP--OR--REGISTER

@app.route("/signup")
def signup():
    return render_template("signup.html")

@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    print("Break point1")
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

        print("Break point2" + "name: " + username + "-----" + email + "-----" + password)

        try:
            print("Break point3")
            connectionID = ibm_db_dbi.connect(conn_str, '', '')
            cursor = connectionID.cursor()
            print("Break point4")
        except:
            print("No connection Established")

        # cursor = mysql.connection.cursor()
        # with app.app_context():
        #     print("Break point3")
        #     cursor = ibm_db_conn.cursor()
        #     print("Break point4")

    print("Break point5")
    sql = "SELECT * FROM register WHERE username = ?"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, username)

```

```

    ibm_db.execute(stmt)
    result = ibm_db.execute(stmt)
    print(result)
    account = ibm_db.fetch_row(stmt)
    print(account)

    param = "SELECT * FROM register WHERE username = " + "\"" + username + "\""
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    print("---- ")
    dictionary = ibm_db.fetch_assoc(res)
    while dictionary != False:
        print("The ID is : ", dictionary["USERNAME"])
        dictionary = ibm_db.fetch_assoc(res)

    # dictionary = ibm_db.fetch_assoc(result)
    # cursor.execute(stmt)
    # account = cursor.fetchone()
    # print(account)
    # while ibm_db.fetch_row(result) != False:
    #     # account = ibm_db.result(stmt)
    #     print(ibm_db.result(result, "username"))
    # print(dictionary["username"])

    print("break point 6")
    if account:
        msg = 'Username already exists !'
    elif not re.match(r'^@+@[^@]+\.[^@]+', email):
        msg = 'Invalid email address !'
    elif not re.match(r'[A-Za-z0-9]+', username):
        msg = 'name must contain only characters and numbers !'
    else:
        sql2 = "INSERT INTO register (username, email,password) VALUES (?, ?, ?)"
        stmt2 = ibm_db.prepare(ibm_db_conn, sql2)
        ibm_db.bind_param(stmt2, 1, username)
        ibm_db.bind_param(stmt2, 2, email)
        ibm_db.bind_param(stmt2, 3, password)
        ibm_db.execute(stmt2)
        # cursor.execute('INSERT INTO register VALUES (NULL, % s, % s, % s)', (username, email,password))
        # mysql.connection.commit()
        msg = 'You have successfully registered !'
    return render_template('signup.html', msg = msg)

#LOGIN--PAGE

@app.route("/signin")
def signin():
    return render_template("login.html")

@app.route('/login',methods =['GET', 'POST'])
def login():
    global userid
    msg = ''

    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        # cursor = mysql.connection.cursor()

```

```

# cursor.execute('SELECT * FROM register WHERE username = % s AND password = % s', (username, password ),)
# account = cursor.fetchone()
# print (account)

sql = "SELECT * FROM register WHERE username = ? and password = ?"
stmt = ibm_db.prepare(ibm_db_conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.bind_param(stmt, 2, password)
result = ibm_db.execute(stmt)
print(result)
account = ibm_db.fetch_row(stmt)
print(account)

param = "SELECT * FROM register WHERE username = " + "\"" + username + "\"" + " and password = " + "\"" + password
+ "\""

res = ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res)

# sendmail("hello sakthi","sivasakthisairam@gmail.com")

if account:
    session['loggedin'] = True
    session['id'] = dictionary["ID"]
    userid = dictionary["ID"]
    session['username'] = dictionary["USERNAME"]
    session['email'] = dictionary["EMAIL"]

    return redirect('/home')
else:
    msg = 'Incorrect username / password !'

return render_template('login.html', msg = msg)

#ADDING----DATA

@app.route("/add")
def adding():
    return render_template('add.html')

@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():

    date = request.form['date']
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category = request.form['category']

    print(date)
    p1 = date[0:10]
    p2 = date[11:13]
    p3 = date[14:]
    p4 = p1 + "-" + p2 + "." + p3 + ".00"
    print(p4)

    # cursor = mysql.connection.cursor()
    # cursor.execute('INSERT INTO expenses VALUES (NULL, % s, % s, % s, % s, % s)', (session['id'] ,date,
    expensename, amount, paymode, category))
    # mysql.connection.commit()

```

```

# print(date + " " + expensename + " " + amount + " " + paymode + " " + category)

sql = "INSERT INTO expenses (userid, date, expensename, amount, paymode, category) VALUES (?, ?, ?, ?, ?, ?)"
stmt = ibm_db.prepare(ibm_db_conn, sql)
ibm_db.bind_param(stmt, 1, session['id'])
ibm_db.bind_param(stmt, 2, p4)
ibm_db.bind_param(stmt, 3, expensename)
ibm_db.bind_param(stmt, 4, amount)
ibm_db.bind_param(stmt, 5, paymode)
ibm_db.bind_param(stmt, 6, category)
ibm_db.execute(stmt)

print("Expenses added")

# email part

param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current timestamp)
AND YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
res = ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res)
expense = []
while dictionary != False:
    temp = []
    temp.append(dictionary["ID"])
    temp.append(dictionary["USERID"])
    temp.append(dictionary["DATE"])
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    expense.append(temp)
    print(temp)
    dictionary = ibm_db.fetch_assoc(res)

total=0
for x in expense:
    total += x[4]

param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER BY id DESC LIMIT 1"
res = ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res)
row = []
s = 0
while dictionary != False:
    temp = []
    temp.append(dictionary["LIMITSS"])
    row.append(temp)
    dictionary = ibm_db.fetch_assoc(res)
    s = temp[0]

if total > int(s):
    msg = "Hello " + session['username'] + " , " + "you have crossed the monthly limit of Rs. " + s + "/- !!!" + "\n"
    + "Thank you, " + "\n" + "Team Personal Expense Tracker."
    sendmail(msg,session['email'])

return redirect("/display")

```

```

#DISPLAY---graph

@app.route("/display")
def display():
    print(session["username"],session['id'])

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

    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    return render_template('display.html' ,expense = expense)

#delete---the--data

@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
def delete(id):
    # cursor = mysql.connection.cursor()
    # cursor.execute('DELETE FROM expenses WHERE id = {0}'.format(id))
    # mysql.connection.commit()

    param = "DELETE FROM expenses WHERE id = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)

    print('deleted successfully')
    return redirect("/display")

#UPDATE---DATA

@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT * FROM expenses WHERE id = %s', (id,))
    # row = cursor.fetchall()

    param = "SELECT * FROM expenses WHERE id = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    while dictionary != False:
        temp = []

```

```

        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        row.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    print(row[0])
    return render_template('edit.html', expenses = row[0])

@app.route('/update/<id>', methods = ['POST'])
def update(id):
    if request.method == 'POST' :

        date = request.form['date']
        expensename = request.form['expensename']
        amount = request.form['amount']
        paymode = request.form['paymode']
        category = request.form['category']

        # cursor = mysql.connection.cursor()
        # cursor.execute("UPDATE `expenses` SET `date` = % s , `expensename` = % s , `amount` = % s , `paymode` = % s ,
        category` = % s WHERE `expenses`.`id` = % s ",(date, expensename, amount, str(paymode), str(category),id))
        # mysql.connection.commit()

        p1 = date[0:10]
        p2 = date[11:13]
        p3 = date[14:]
        p4 = p1 + "-" + p2 + "." + p3 + ".00"

        sql = "UPDATE expenses SET date = ? , expensename = ? , amount = ? , paymode = ? , category = ? WHERE id = ?"
        stmt = ibm_db.prepare(ibm_db_conn, sql)
        ibm_db.bind_param(stmt, 1, p4)
        ibm_db.bind_param(stmt, 2, expensename)
        ibm_db.bind_param(stmt, 3, amount)
        ibm_db.bind_param(stmt, 4, paymode)
        ibm_db.bind_param(stmt, 5, category)
        ibm_db.bind_param(stmt, 6, id)
        ibm_db.execute(stmt)

        print('successfully updated')
        return redirect("/display")

#limit
@app.route("/limit" )
def limit():
    return redirect('/limitn')

@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
    if request.method == "POST":
        number= request.form['number']

        # cursor = mysql.connection.cursor()
        # cursor.execute('INSERT INTO limits VALUES (NULL, % s, % s) ',(session['id'], number))

```

```

# mysql.connection.commit()

sql = "INSERT INTO limits (userid, limitss) VALUES (?, ?)"
stmt = ibm_db.prepare(ibm_db_conn, sql)
ibm_db.bind_param(stmt, 1, session['id'])
ibm_db.bind_param(stmt, 2, number)
ibm_db.execute(stmt)

return redirect('/limitn')

@app.route("/limitn")
def limitn():
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT limitss FROM `limits` ORDER BY `limits`.`id` DESC LIMIT 1')
    # x= cursor.fetchone()
    # s = x[0]

    param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER BY id DESC LIMIT 1"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = " /-"
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMITSS"])
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[0]

    return render_template("limit.html" , y= s)

#REPORT

@app.route("/today")
def today():
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT TIME(date) , amount FROM expenses WHERE userid = %s AND DATE(date) = DATE(NOW())', (str(session['id'])))
    # texpanse = cursor.fetchall()
    # print(texpanse)

    param1 = "SELECT TIME(date) as tn, amount FROM expenses WHERE userid = " + str(session['id']) + " AND DATE(date) = DATE(current timestamp) ORDER BY date DESC"
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpanse = []

    while dictionary1 != False:
        temp = []
        temp.append(dictionary1["TN"])
        temp.append(dictionary1["AMOUNT"])
        texpanse.append(temp)
        print(temp)
        dictionary1 = ibm_db.fetch_assoc(res1)

    # cursor = mysql.connection.cursor()

```



```

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

param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND DATE(date) = DATE(current timestamp)
ORDER BY date DESC"
res = ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res)
expense = []
while dictionary != False:
    temp = []
    temp.append(dictionary["ID"])
    temp.append(dictionary["USERID"])
    temp.append(dictionary["DATE"])
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    expense.append(temp)
    print(temp)
    dictionary = ibm_db.fetch_assoc(res)

total=0
t_food=0
t_entertainment=0
t_business=0
t_rent=0
t_EMI=0
t_other=0

for x in expense:
    total += x[4]
    if x[6] == "food":
        t_food += x[4]

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

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

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

    elif x[6] == "other":
        t_other += x[4]

print(total)

print(t_food)
print(t_entertainment)
print(t_business)
print(t_rent)
print(t_EMI)
print(t_other)

```

```

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

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

    param1 = "SELECT DATE(date) as dt, SUM(amount) as tot FROM expenses WHERE userid = " + str(session['id']) + " AND
    MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp) GROUP BY DATE(date) ORDER BY DATE(date)"
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpanse = []

    while dictionary1 != False:
        temp = []
        temp.append(dictionary1["DT"])
        temp.append(dictionary1["TOT"])
        texpanse.append(temp)
        print(temp)
        dictionary1 = ibm_db.fetch_assoc(res1)

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

    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current timestamp)
    AND YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0

```

```

t_other=0

for x in expense:
    total += x[4]
    if x[6] == "food":
        t_food += x[4]

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

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

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

    elif x[6] == "other":
        t_other += x[4]

print(total)

print(t_food)
print(t_entertainment)
print(t_business)
print(t_rent)
print(t_EMI)
print(t_other)

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

@app.route("/year")
def year():

    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT MONTH(date), SUM(amount) FROM expenses WHERE userid= %s AND YEAR(DATE(date))= YEAR(now())
GROUP BY MONTH(date) ORDER BY MONTH(date) ',(str(session['id'])))
    # texpanse = cursor.fetchall()
    # print(texpanse)

    param1 = "SELECT MONTH(date) as mn, SUM(amount) as tot FROM expenses WHERE userid = " + str(session['id']) + " AND
YEAR(date) = YEAR(current timestamp) GROUP BY MONTH(date) ORDER BY MONTH(date)"
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpanse = []

    while dictionary1 != False:
        temp = []
        temp.append(dictionary1["MN"])
        temp.append(dictionary1["TOT"])
        texpanse.append(temp)
        print(temp)
        dictionary1 = ibm_db.fetch_assoc(res1)

```

```

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

param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND YEAR(date) = YEAR(current timestamp)
ORDER BY date DESC"
res = ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res)
expense = []
while dictionary != False:
    temp = []
    temp.append(dictionary["ID"])
    temp.append(dictionary["USERID"])
    temp.append(dictionary["DATE"])
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    expense.append(temp)
    print(temp)
    dictionary = ibm_db.fetch_assoc(res)

total=0
t_food=0
t_entertainment=0
t_business=0
t_rent=0
t_EMI=0
t_other=0

for x in expense:
    total += x[4]
    if x[6] == "food":
        t_food += x[4]

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

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

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

    elif x[6] == "other":
        t_other += x[4]

print(total)

print(t_food)
print(t_entertainment)
print(t_business)
print(t_rent)

```

```

print(t_EMI)
print(t_other)

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

#log-out

@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    session.pop('email', None)
    return render_template('home.html')

port = os.getenv('VCAP_APP_PORT', '8080')
if __name__ == "__main__":
    app.secret_key = os.urandom(12)
    app.run(debug=True, host='0.0.0.0', port=port)

```

GIT-HUB LINK:

<https://github.com/IBM-EPBL/IBM-Project-12965-1659503618>

PROJECT DEMO LINK:

https://drive.google.com/file/d/1bHvOE93q_Rgtu50WyDI-Z4v-RHqqEMFC/view?usp=drivesdk