PERSONAL EXPENSE TRACKER APPLICATION

NALAIYA THIRAN PROJECT BASED LEARNING ON PROFESSIONAL READLINESS FOR INNOVATION, EMPLOYMENT AND ENTERPRENEURSHIP

A PROJECT REPORT

SAHUL HAMEED A (922119104036)

RAMPRASANTH A (922119104035)

SATHEESHKUMAR G(922119104041)

VIGNESHWARAN V(922119104048)

TEAM ID: PNT2022TMID33144

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

SSM INSTITUTE OF ENGINEERING AND TECHNOLOGY, DINDIGUL

TAMIL NADU 624003

CONTEXT

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

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

8. TESTING

- 8.1. Test Cases
- 8.2. User Acceptance Testing

9. RESULTS

- 9.1. Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

CHAPTER 1

INTRODUCTION

Now a day's people are concerned about regularity of their daily expenses. This is done mainly for keep a track of the users' daily expenses to have a control of users' monthly expenses. Personal Expense Tracker Application is used to manage the user's daily expenses in a more coherent and manageable way. This application will help us to reduces the manual calculations for their daily expenses and also keep the track of the expenses. With the help of this application, user can calculate his total expenses per day and these results will stored for unique user. As the traditional methods of budgeting, we need to maintain the Excel sheets, Word Documents, notes, and files for the user daily and monthly expenses. There is no as such full-fledged solution to keep a track of our daily expenses easily. Keeping a log in diary is a very monotonous process and also may sometimes lead into problems due to the manual calculations. Looking on all the above given conditions, we are trying to satisfy the user requirements by building a mobile application which will help them reduces their burdens. "Personal Expense Tracker Application" is an application where one can enter their daily expenses and end of the day, they know their expenses in charts.

1.1 Project Overview

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.

1.2 Purpose

In this paper, we develop a mobile application developed for the android platform that keeps record of user personal expenses, user contribution in group expenditures, top investment options, view of the current stock market, read authenticated financial news and grab the best ongoing offers in the market in popular categories. The proposed application would eliminate messy sticky notes, spreadsheets confusion and data handling inconsistency problems while offering the best overview of your expenses. With our application can manage their expenses and decide on their budget more effectively.

CHAPTER 2

LITERATURE SURVEY

A literature survey or a literature review in a project report is that section which shows the various analyses and research made in the field of your interest and the results already published, taking into account the various parameters of the project and the extent of the project. It helps you set a goal for your analysis - thus giving you your problem statement.

2.1 Existing problem

1. EXPENDITURE MANAGEMENT SYSTEM: (2022)

AUTHOR: DR.V.GEETHA, G.NIKITHA, H.SRI LASYA, DR.C.K.GOMATHY

Expense tracker is a web application used to track user expenses and generates periodical reports about the savings and expenditure. In this project, we propose an application known as "Expense Tracker," which is helpful to manage our income and expense daily or periodically or else whenever we want to remind. It also acts as an indicator or reminder example in the fastest world in which we cannot remember what the things we have to do for the end of the month are and the payments we have to pay for the particular month. Due to some conflict or other stress, we sometimes forget what the income is, the payments we have to pay.

2. EXPENSE TRACKER: (2021)

AUTHOR: AMAN GARG, MUKUL, SAGAR MITTAL, MR. SHEKAR SINGH

Expense Tracker is a web application that facilitates the users to keep track and manage their personal as well as business expenses. 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. User can easily import transactions from user mobile wallets without risking their information and efficiently protecting user’s privacy. User can see the accurate duration for how long a particular product is being used by them. The monthly, and yearly comparison of expenditures will be done by the app which will let the user know the area where user is spending the most.

3. AN ANDROID BASED MOBILE APPLICATION FOR TRACKING DAILY EXPENSES

AUTHOR: ADEPEGBA, O. A., FAYEMIWO, M.A., ODUW

YEAR: 2018

This study is aimed at developing an android based mobile application capable of monitoring and controlling personal expenses, as well as cautioning the user against reckless and unbudgeted spending. The developed system was designed using system flowchart, use case diagram, sequence diagram, class diagram and system architecture diagram. It was implemented using Java programming language on android studio and My SQL.

` The developed system was evaluated based on basic functionality tests performed on the individual modules,the integrated testing as well as the overall function testing. The results of testing the functionalities of the developed system showed that all the modules worked properly when tested individually. They rejected invalid inputs and responded promptly to user requests. Database operations such as insert, update, delete and add that were performed yielded expected results, and data consistency / integrity are maintained in the reports generated. Thus, the developed system provides an easy to use, portable and secured means of enhancing financial sustainability and promotes individual and societal economic growth via fiscal discipline.

As technological innovation develops, numerous Information Technology (IT) based applications are developed to aid individuals and organizations in performing tasks, especially those being carried out on daily basis. This android based mobile application for tracking daily expenses aims to automate the record keeping and monitoring of daily expenses. In those days, a costs day book was used to monitor day to day costs, periodic costs and ascertain the financial plan manually.

4. EXPENSE TRACKER: A SMART APPROACH TO TRACK EVERY EXPENSE:

AUTHORS: HRITHIK GUPTA, ANANT PRAKASH SINGH, NAVNEET KUMAR, J. ANGELIN BLESSY

YEARS: 2020

Expense Tracker is a day-to-day expense management system designed to easily and efficiently track the daily expenses of unpaid and unpaid staff through a computerized system that eliminates the need for manual paper tasks that systematically maintains records and easily accesses data stored by the user.

The language databases we use to develop this system are Java (Apache Netbeans 11.3) and MySQL Workbench 8.0CE. This application is a GUI (Graphics User Interface)based application. If you are a window user, you can download the application and work accordingly. This system is used by any person to control his income expenditure from daily to annual basics. And to keep an eye on their spending. This app is very easy to use and mutli-language. The main feature of this app is that you can track by day and category. You can use it according to your category.

It will guide them and aware them about their 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.

5. A REVIEW ON BUDGET ESTIMATOR ANDROID APPLICATION

AUTHOR: NAMITA JAGTAP , PRIYANKA JOSHI, ADITYA KAMBLE

YEAR: 2019

In existing, we need to maintain the excel sheets, csv etc. files for the user daily and monthly expenses. In existing, there is no as such complete solution to keep a track of its daily expenditure easily.

To do so a person as 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. This project is about mobile application Expenses system with geo-location tracking ,Based on the location of the user, it using Google Places, to check, the available store in the area, provides a notification for offers purpose.

In term of security design, this system may implement a login authentication such as OTP message to your mobile device, this function may bring more security confidence to user. , we propose an application which is developed by android. this application allows users to maintain a digital automated diary.

Each user will be required to register on the system at registration time, the user will be provided id,which will be used to maintain the record of each unique user.

Expense Tracker application which will keep a track of income-expense of a user on a day to day basis. this application takes income from user and divides in daily expense allowed.

If you exceed that days expense it will cut if from your income and give new daily expense allowed amount, and if that days expense is less it will add it in savings.

6. PERSONALIZED EXPENSE MANAGING ASSISTANT USING ANDROID

AUTHORS:N.ZAHIRAJAHAN, K.I.VINODHINI

YEARS: 2016

Mobile applications stood top among usability and user convenience. Many applications are available in the market to manage personal and group expenses. Not many applications provide a comprehensive view of both use cases. In this project, we develop a mobile application that keeps track of user personal expenses, his/her personal contribution towards group expenses; maintain monthly incomes, recurring and ad-hoc payments. It provides information of "who owes who and by how much".

The proposed application would eliminate sticky note, spreadsheet and ledger that cause confusions, data inconsistency problems while recording and splitting of expenses. With our application user can manage his expenses more effectively. This application will not only help users to manage their expenses but also help marketing executives to plan marketing according to the needs of users.

This is also a unique feature of expense manager. The estimations of expenses of user are suggested to them. This would help the user to adjust his spending accordingly within their budget. However, this application will not consider some expense types like buying electronic appliances, automobiles and other expensive products which do not occur frequently.

2.2 References

i) "Expenditure Management System" by Journal Of Engineering, Computing & Architecture.

ii) "Expense Tracker" by Aman Garg, published in Volume 9, Issue IV, April 2021 in International Journal for Research in Applied Science & Engineering Technology.

iii) "An Android Based Mobile Application For Tracking Daily Expenses" on global ecosystem for nurturing multidisciplinary research innovations at Ghana.

iv) "eExpense: A Smart Approach to Track Everyday Expense" on 2018 4th International Conference on Electrical Engineering and Information & Communication Technology.

v) "A Review on Budget Estimator Android Application" published on International Research Journal of Engineering and Technology (IRJET) by Dept. of computer Engineering, K.J.Somaiya Institute Of Engineering & IT , Maharashtra, Mumbai.

vi) "Personalized Expense Managing Assistant Using Android" published on International Journal of Computer Techniques -– Volume 3 Issue 2, Mar- Apr 2016 by Department of Computer Applications, Nandha Engineering College/Anna University, Erode.

2.3 Problem Statement Definition

The customer problem statement template gives you the guidelines to create a problem statement. As part of the Design Thinking methodology, the problem statement is essential to put yourself in your customer's shoes and gain empathy when building services or products, besides tackling the real issues behind your customers' needs.

I am	Describe customer with 3-4 key characteristics - who are they?	Describe the customer and their attributes here
I'm trying to	List their outcome or "job" the care about - what are they trying to achieve?	List the thing they are trying to achieve here
but	Describe what problems or barriers stand in the way – what bothers them most?	Describe the problems or barriers that get in the way here
because	Enter the "root cause" of why the problem or barrier exists – what needs to be solved?	Describe the reason the problems or barriers exist
which makes me feel	Describe the emotions from the customer's point of view – how does it impact them emotionally?	Describe the emotions the result from experiencing the problems or barriers

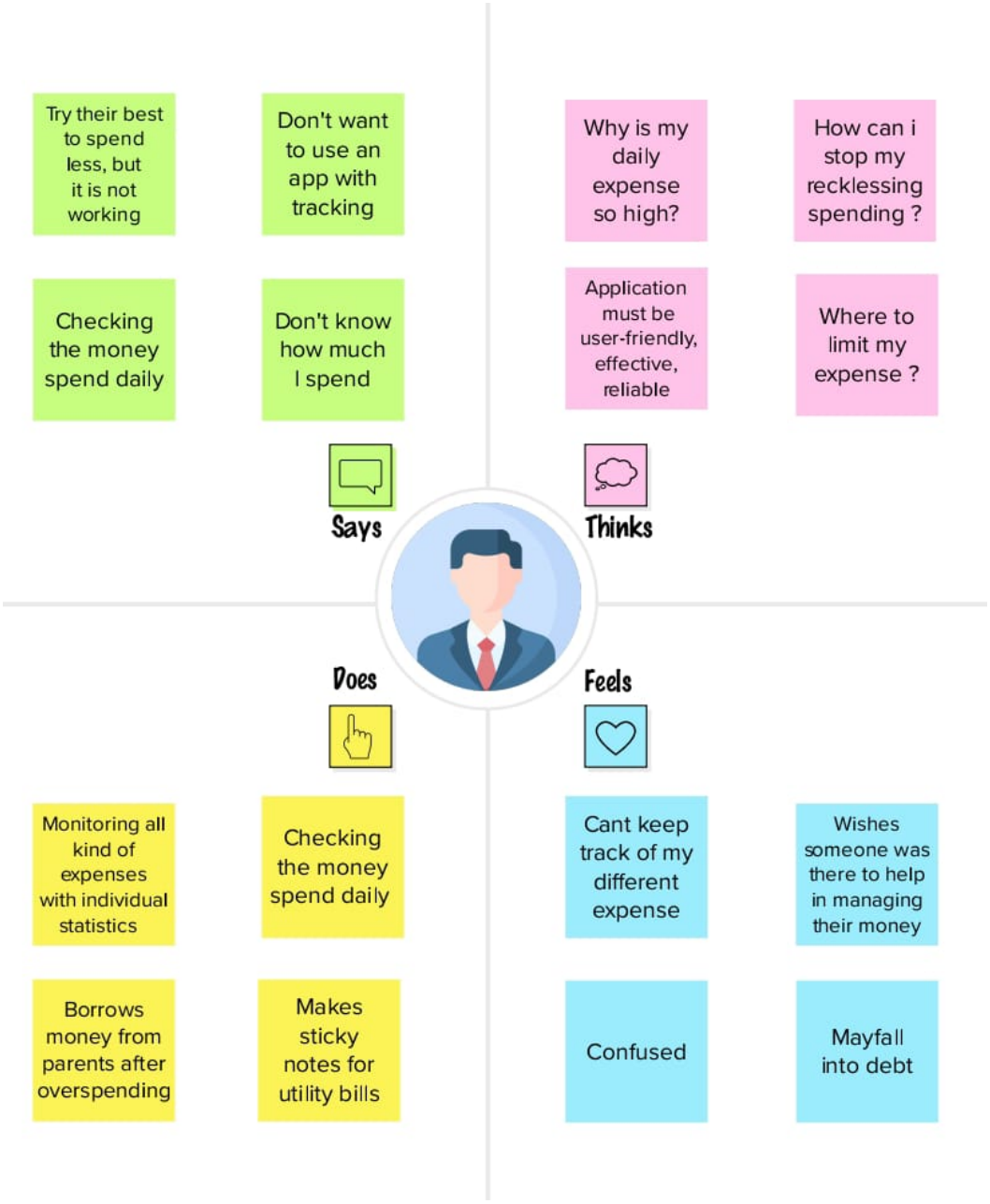
Problem Statement (PS)	Iam (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	An employee	Make a monthlybudget	There are no facilities to set a budget	I need to save money for my future plans	Frustrated
PS-2	Client- Geetha	The best investment Plans	Who needs high returns from investment plan	I need high returns from the investment	More efficient
PS-3	A Manager	To know the premium schedule.	Alert for premium schedule.	To reduce the premium monthly expenses for paying premium	Not smart enough
PS-4	Client - Loki	To record the expenses and income of individual	Can't able to access	Lack of internet connection	Disappointed
PS-5	Client- Maha	To record the expenses and income of individual	Request denied	Limited accessibility in tracking	Frustated
PS-6	Client- Nirm	Keep track of my expenses	Can't categorize the various types of expenses	There is no option to organize the expenses	Uncomfortable

CHAPTER 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.



3.2 Ideation & Brainstorming

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

Step-2: Brainstorm, Idea Listing and Grouping

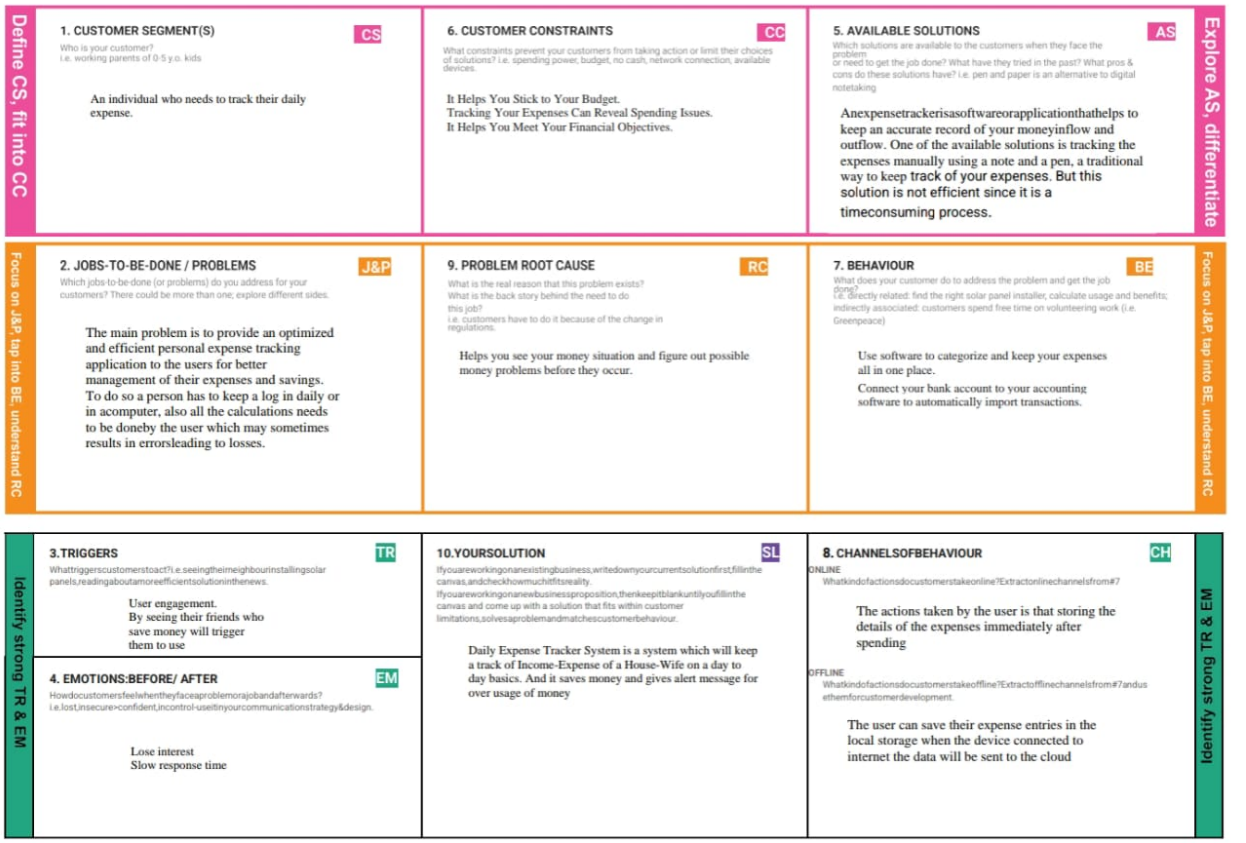
Step-3: Idea Prioritization

3.3 Proposed Solution

S.No	Parameter	Description
1.	Problem Statement	In Expense Tracker System in paper-based, it is difficult to track our monthly expenses manually. There are many budgeting tools online, but not all of them are effective in assisting users in actually creating and adhering to a budget. This Expense records may get lost in case of fire accidents, flood etc.
2.	Idea / Solution description	This application can handle large number of users and data with high performance and security. This application can adapt for both large-scale and small-scale purposes. Easily available in all kinds of devices.
3.	Novelty / Uniqueness	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. 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.
4.	Social Impact / Customer Satisfaction	The user gets notified when their expense exceeds the limit and also it reminds the user when they forgot to make entry. Tracking expenses through SMS. Data analytics on expenses. Future expense prediction.
5.	Business Model (Revenue Model)	The application should be able to generate reports of their spending and notify users if they have exceeded their budget. It is designed to be dynamic to produce the prediction. It also provides users' personal information, their income as well as their expenses. This application can create awareness among common people about finance and stuffs. This application also helps user to be financially responsible. It Reduces time.

6.	Scalability of the Solution	This Application is provided for free of cost. But It will have some advertisement. In premium version there is no advertisement and contains some additional features.

3.4 Problem Solution fit



CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Form for collecting details
FR-2	Login	Enter username and password
FR-3	Calendar	Personal expense tracker application must allow user to add the data to their expenses.
FR-4	Category	This application shall allow usersto add categories of their expenses.
FR-5	Expense Tracker	This application should graphically represent the expense in the form of report.
FR-6	Notification	Access System and Real time alerting
FR-7	Report	Graphical representation of report must be generated.

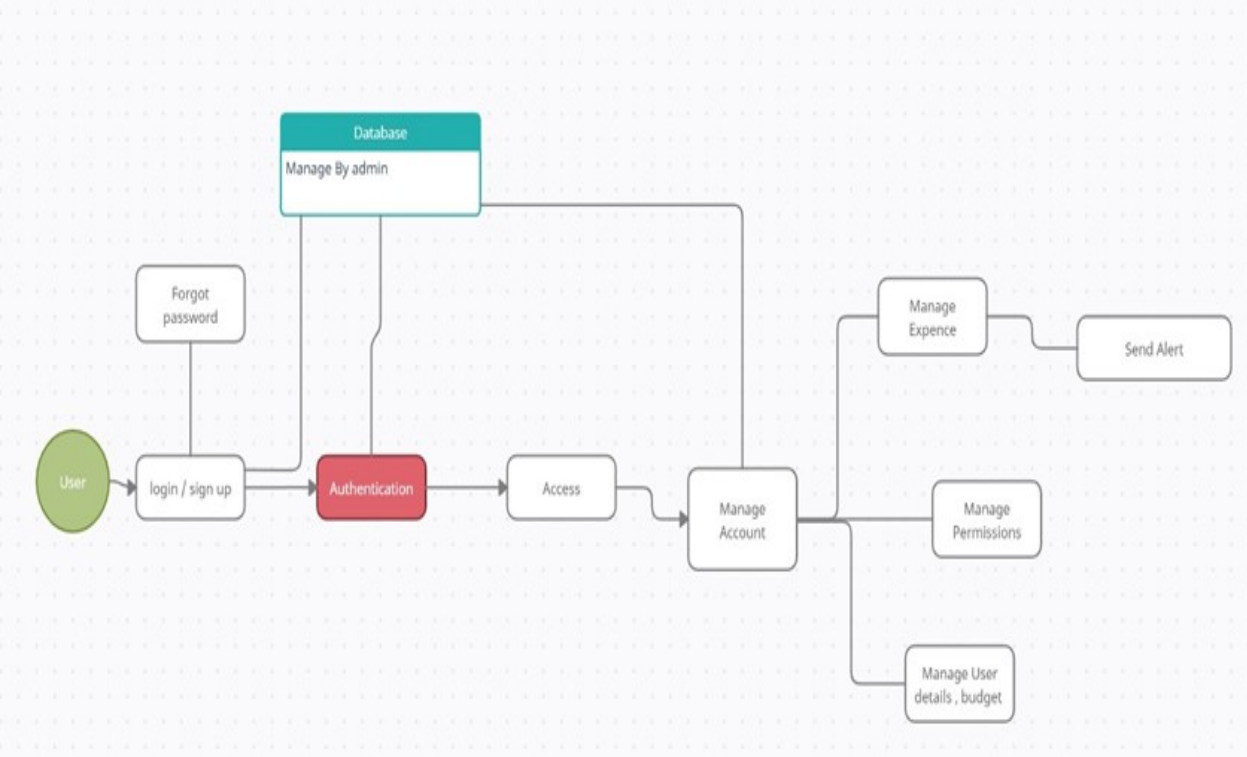
4.2 Non-Functional requirements

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Helps to keep an accurate recordof your income and expenses.
NFR-2	**Security**	Maintain user personal details in a encrypted manner by using data security algorithms.
NFR-3	**Reliability**	Each data record is stored on a well-built efficient database schema. Also maintains tacking of day-to-day expenses.
NFR-4	**Performance**	The types of expense are categories along with an option. Throughput of the system is increased due to light weight database support.
NFR-5	**Availability**	The application must have a 100% up-time. Using charts and graphs may help you monitor your budgeting and assets.
NFR-6	**Scalability**	The abilityto appropriately handleincreasing demands. Automate all the recurrent expenses and remind you on a timely basis.

CHAPTER 5

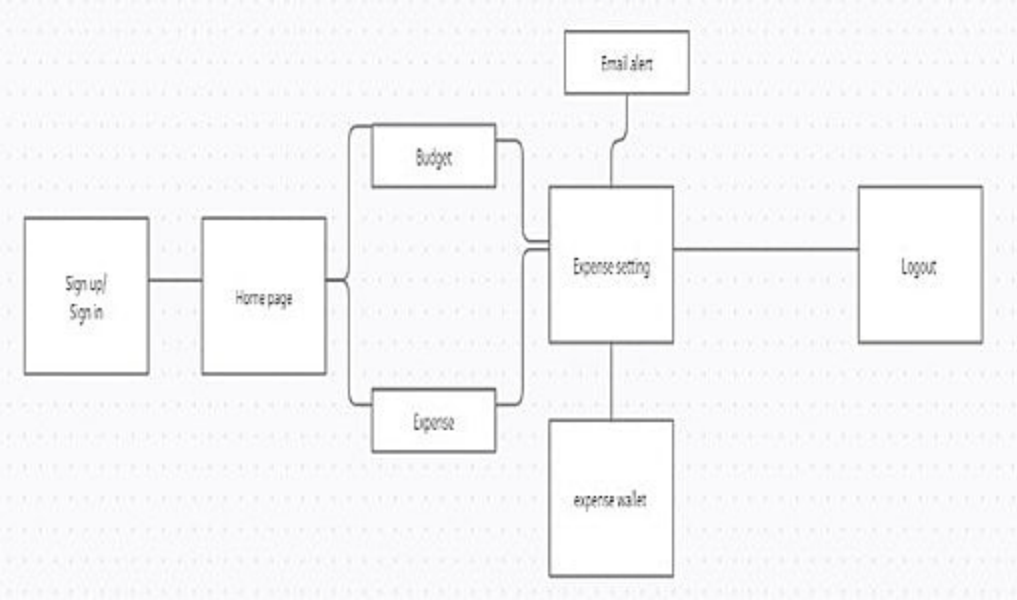
PROJECT DESIGN

5.1 Data Flow Diagrams

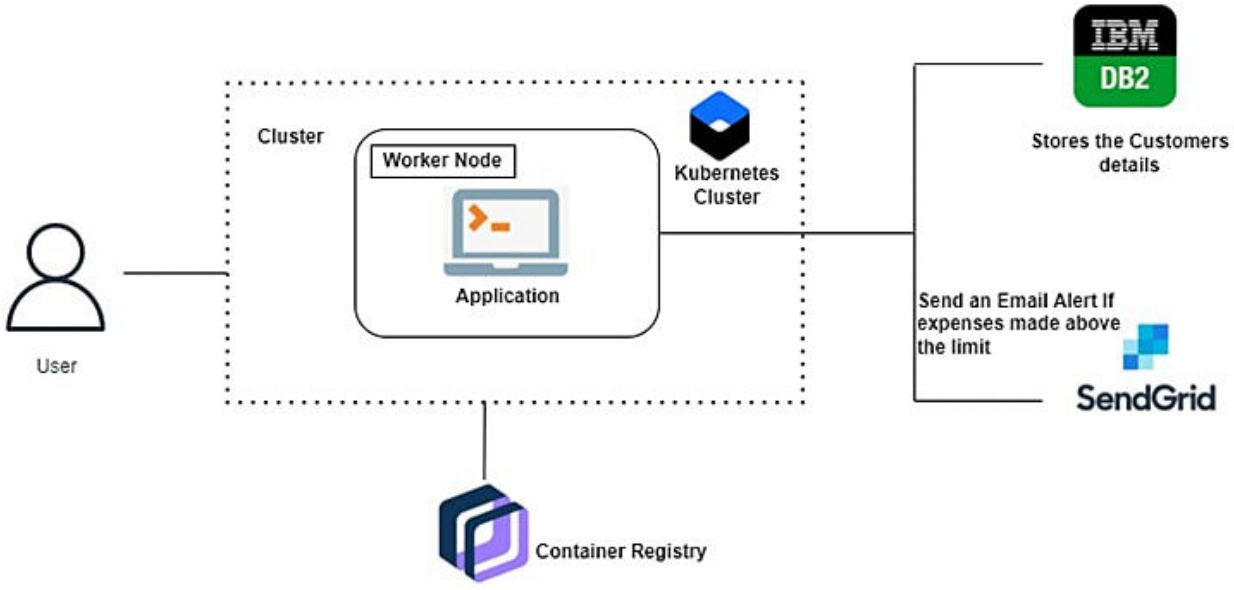


5.2 Solution and Tehnical Archirtecture

Solution Archirtecture



Tehnical Architecture



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user & Web user)	Registration	USN - 1	As a user, I can register for the application by entering my email, and confirming my password.	I can access my account / dashboard.	High	Sprint - 1
		USN - 2	As a user, I can track my expenses and manage my monthly budget.	I can track my expenses and manage my monthly budget.	High	Sprint–3
		USN - 3	As a user, I can see if there is an excessive expense and if there is such condition, I will be notified via e-mail.	I can receive e-mail, if there is an excessive expense.	Low	Sprint–3
	Login	USN - 4	As a user, I can login to user dashboard and see the info about my incomes and expenses.	I can login to user dashboard and see the information.	High	Sprint–1
	Dashboard	USN - 5	As a user, I can enter my income and expense details.	I can view my daily expenses.	High	Sprint–2
Customer Care Executive		USN – 6	As a customer care executive, I can solve the log in issues & issues of the application.	I can provide support or solution at any time 24*7	Medium	Sprint–4
Administrator	Application	USN - 7	As an administrator, I can update the application.	I can fix the bug in the application	Medium	Sprint - 4

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Sahul Hameed
		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Vigneshwaran
	Login	USN-3	As a user, I can register for the application through Gmail	2	Medium	Ramprasanth
	Dashboard	USN-4	As a user, I can log into the application by entering email & password	2	High	Satheeshkumar

Sprint-2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Satheeshkumar
	Charts	USN-2	Creating various graphs and statistics of customer's data	2	Low	Ramprasanth
	Connecting to IBM DB2	USN-3	Linking database with dashboard	1	Medium	Vigneshwaran
		USN-4	Making dashboard interactive with JS	2	High	Sahul Hameed
Sprint-3		USN-1	Wrapping up the server side works of frontend	1		Sahul Hameed
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user's query	2		Vigneshwaran
	SendGrid	USN-3	Using SendGrid to send mail to the user about their expenses	2		Satheeshkumar
		USN-4	Integrating both frontend and backend	1		Ramprasanth

Sprint-4	Docker	USN-1	Creating image of website using docker	2	High	Rampras anth
	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Vignesh waran
	Kubernet es	USN-3	Create container using the docker image and hosting the site	2	High	Sahul Hameed
	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Sathees hkumar

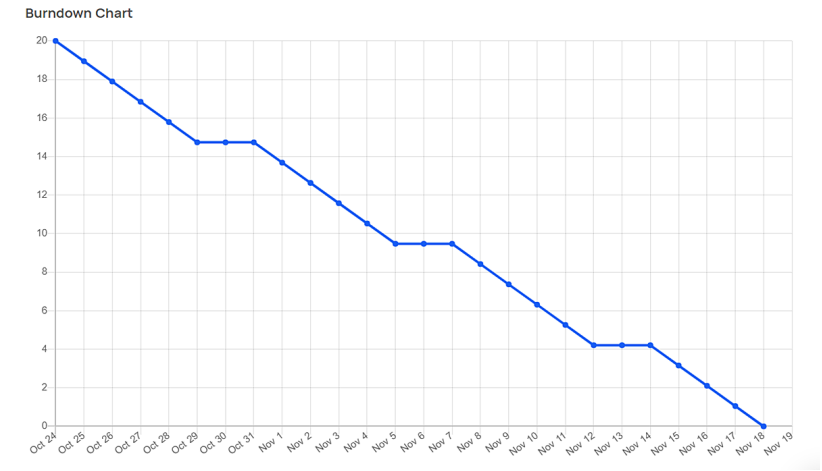
6.2 Sprint Delivery Schedule

Sprint Delivery Plan

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

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.



6.3 Reports from JIRA

CHAPTER 7

CODING & SOLUTIONING

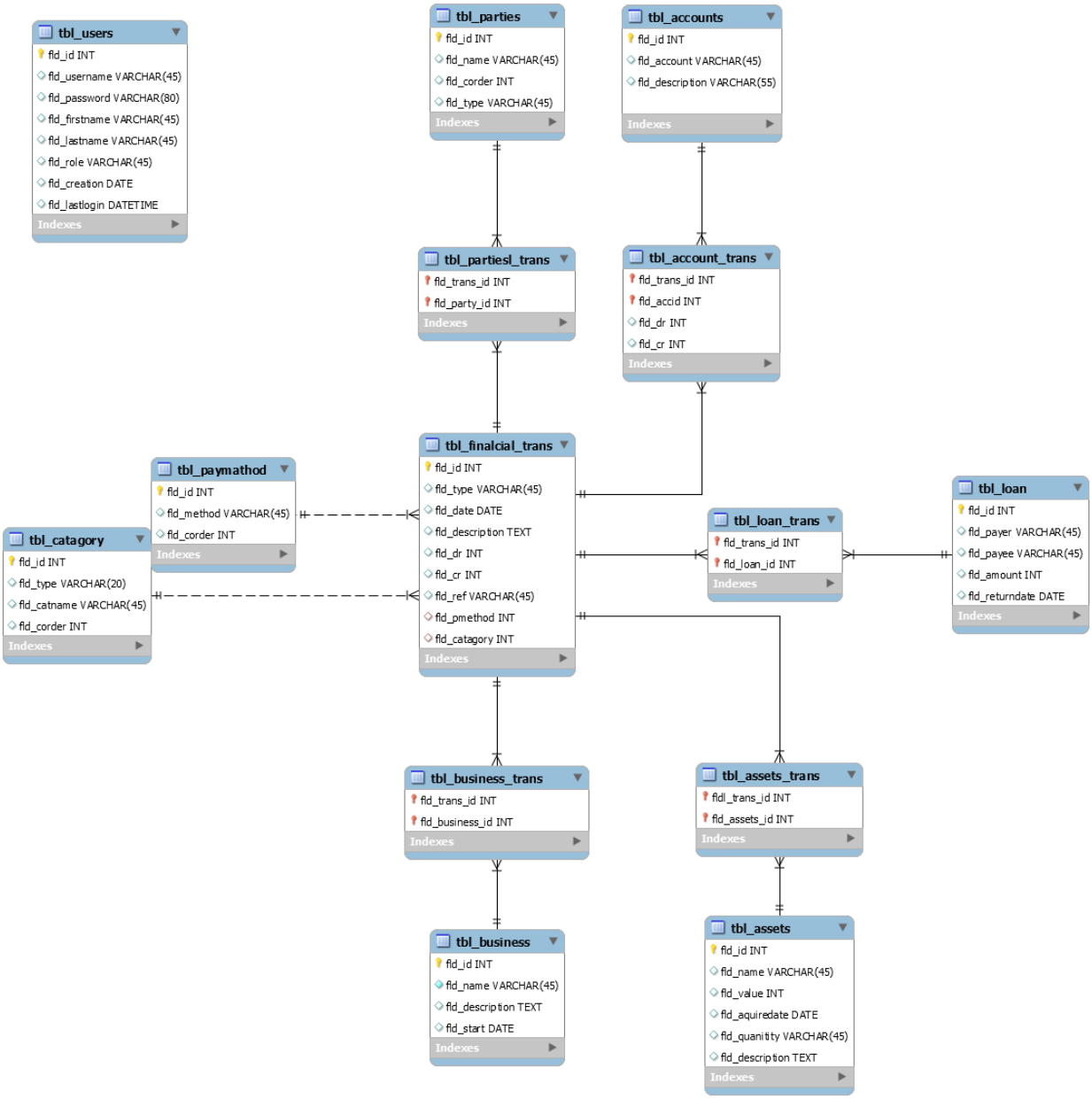
7.1. Feature 1

Handle Documents It’s time to stop using paper and excel spreadsheets for keeping records of your cash payments and online transactions! Papers are tough to handle and more dangerous for the environment.On the other hand, excel sheets may offer an online solution but don’t do much help in money handling. So, it’s better to develop money management software that collects insights from the data and helps make business decisions.Tracks Receipts You always can’t find the cash and digital payments made by you and this is a big issue with tracking expenses.So, if you want to keep a track of your monetary investments, you should go using a business expense tracker app. This helps store all receipts by only clicking their images in your expenses handling app.Prevents Data Losses and Frauds Manual handling of personal expenses and finances can’t check every transaction detail accurately.For this reason, fraud cases happen many times. Using an expense tracking and budgeting app, the workflow of money and finance handling becomes automated. This not just prevents fraudulence but also makes the procedure more accurate and transparent.

7.2. Feature 2 :

Mitigates Human Errors We cannot afford mistakes when it comes to handling budgets and finances. However, humans may make some errors because of misunderstanding, carelessness, or negligence.With the help of an expense handling app, you can lower as well as prevent every mistake caused because of carelessness.Offers Precise Analytics Excel spreadsheets might track and store data and create helpful charts or graphs from it but still have no advanced functionality. On the other hand, human engagement brings the possibility of mistakes.So, it’s best to build a business expense tracker app that carries out prediction analysis and helps you make efficient business decisions.

7.3.Database Schema:



CHAPTER 8

TESTING

8.1. Test Cases

With a concerted effort, I conducted research on general well-being to have a rudimentary grasp on health management, as well as the existing apps in order to get an understanding of what is already existing in the market, the characteristics, specialties, and usability. There are a considerable number of apps tracking apps existing in the market. They aim to track daily spends intake by logging info given to achieve users' preset goals. To log spend, users can input the expense in the app, or scan the barcode of a package. Most apps allow users to connect with associated activities apps to track spend progress. With a premium upgrade, users can get access to tailor-made saving according to health goals or specified way to spend.In order to build a realistic initial target group, I wanted to conduct some usability tests with 5 users that regularly engage in buying activity and spending tracking, including users.

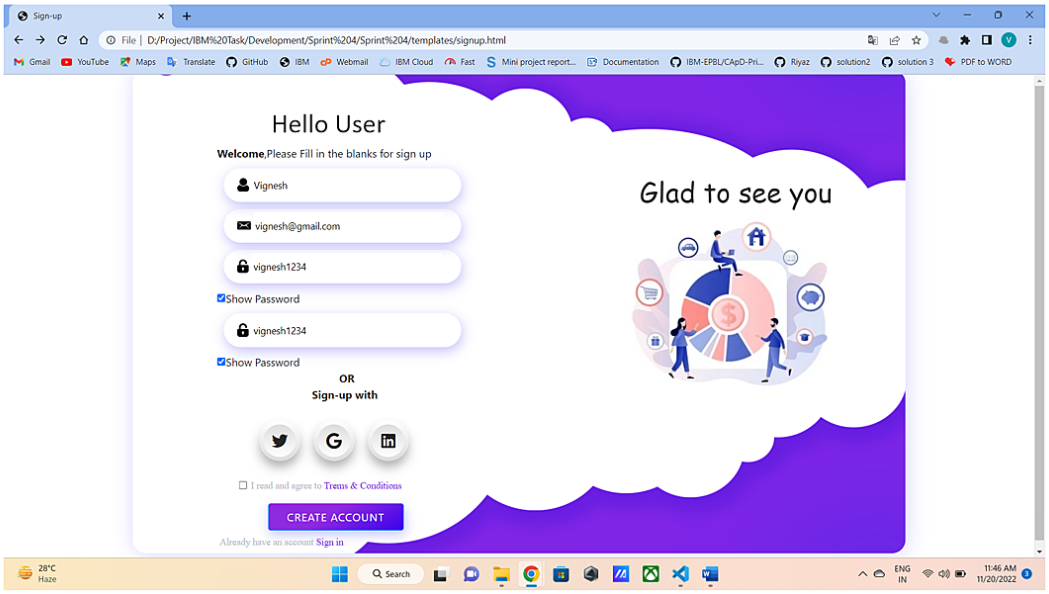
8.2. User Acceptance Testing

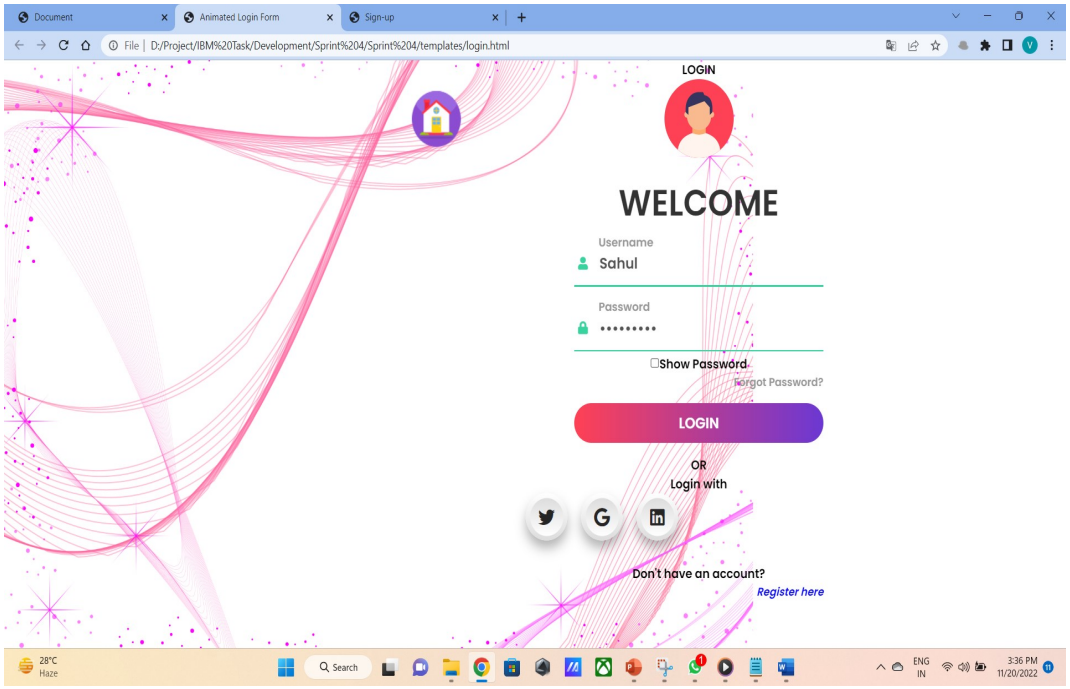
Must-have features of a app I wanted to address the user pain points by including (and improving) the core features of the application. Personal profiles After downloading the app, a user needs to register and create an account. At this stage, users should fill in personal information like name, gender, age, height, weight, spend preferences, spend logging and dashboard Allowing users to analyze their spending habits. They should be able to log expenses and money intake and see their progress on a dashboard that can track overall spends.Push notifications Push notifications are an effective tool for increasing user engagement and retention. To motivate users to keep moving toward their goals, it's pertinent to deliver information on their progress toward the current goal and remind them to log what they spend on.money counter Enabling the application to calculate spent amount of users have gone and done based on the data they've logged. Let users count money and see accurate spend information via a built-in barcode scanner.

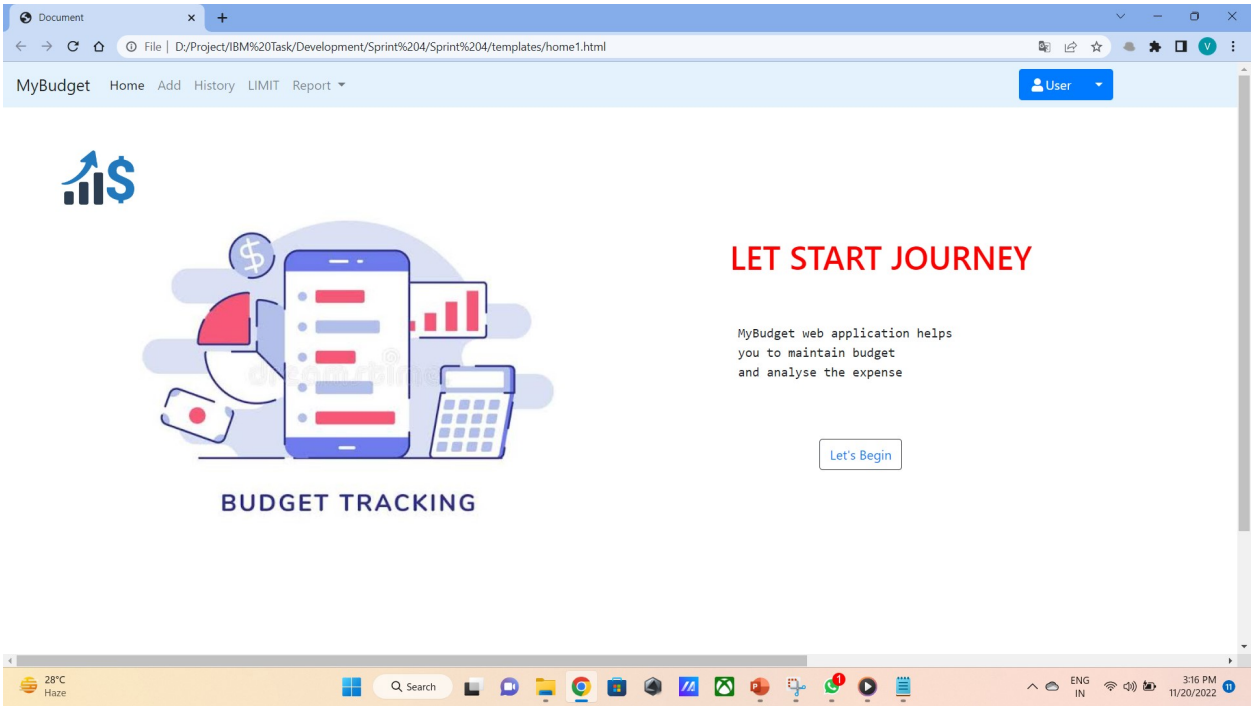
CHAPTER 9

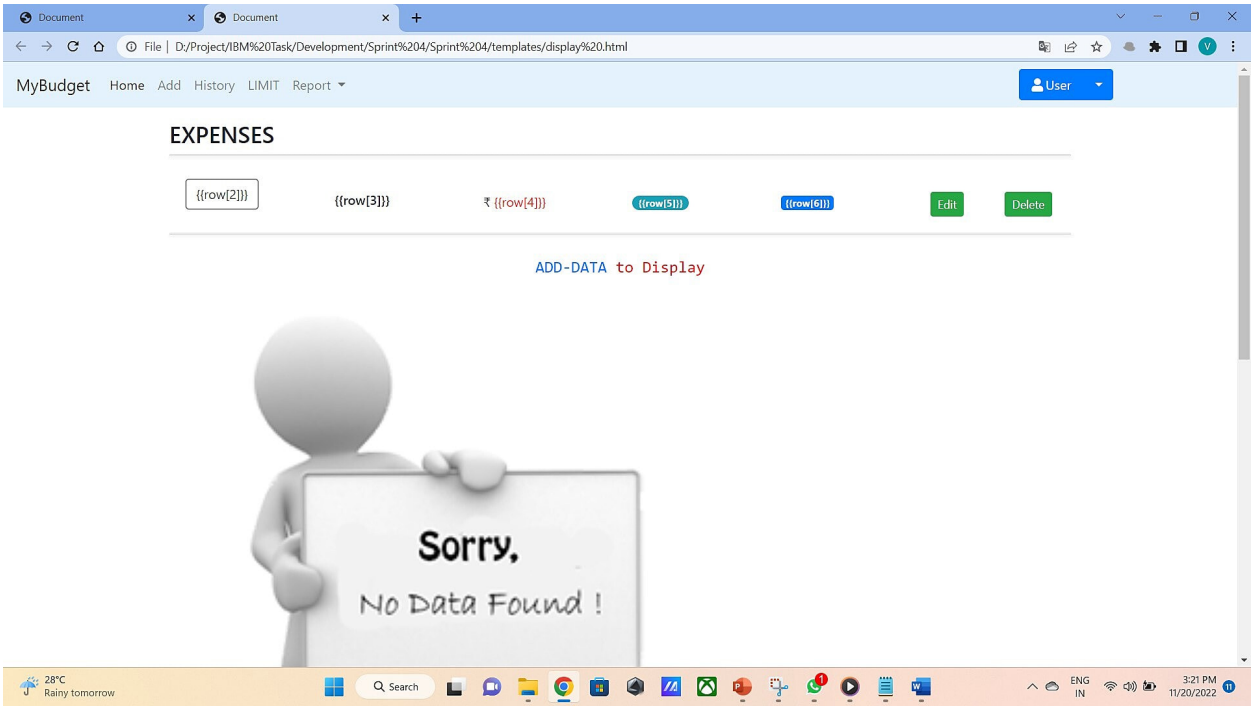
RESULTS

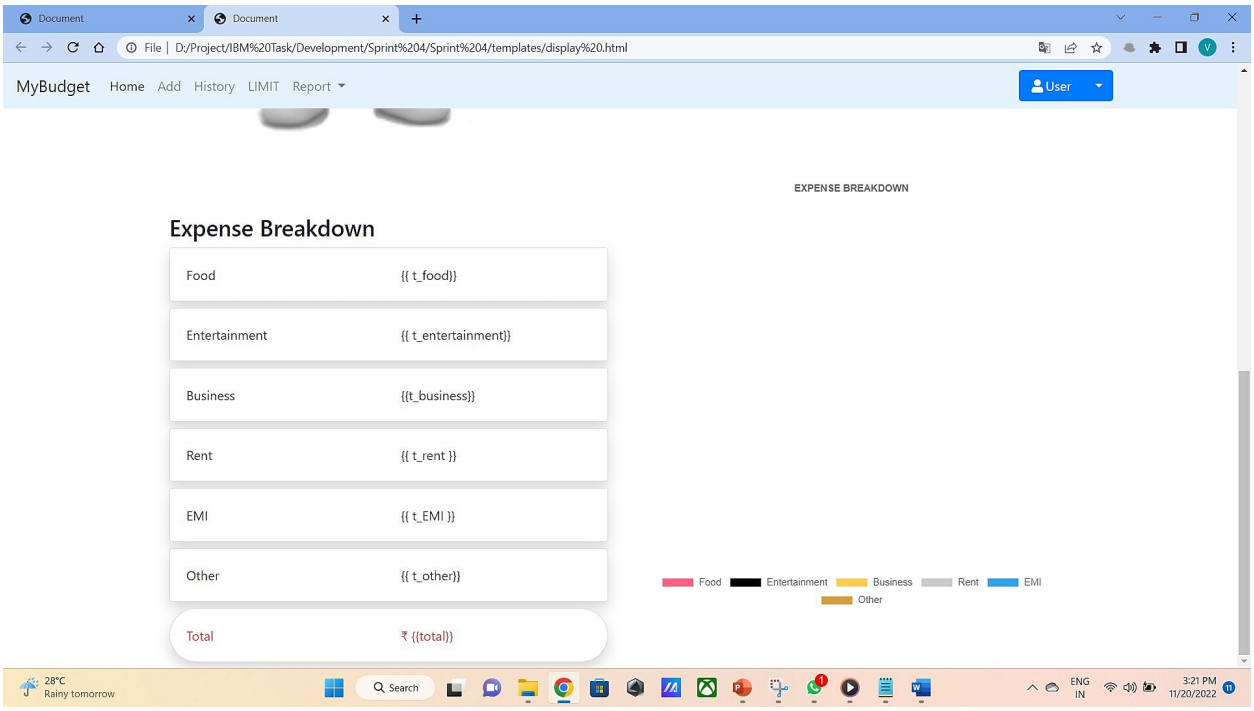
9.1 Performance Metrics

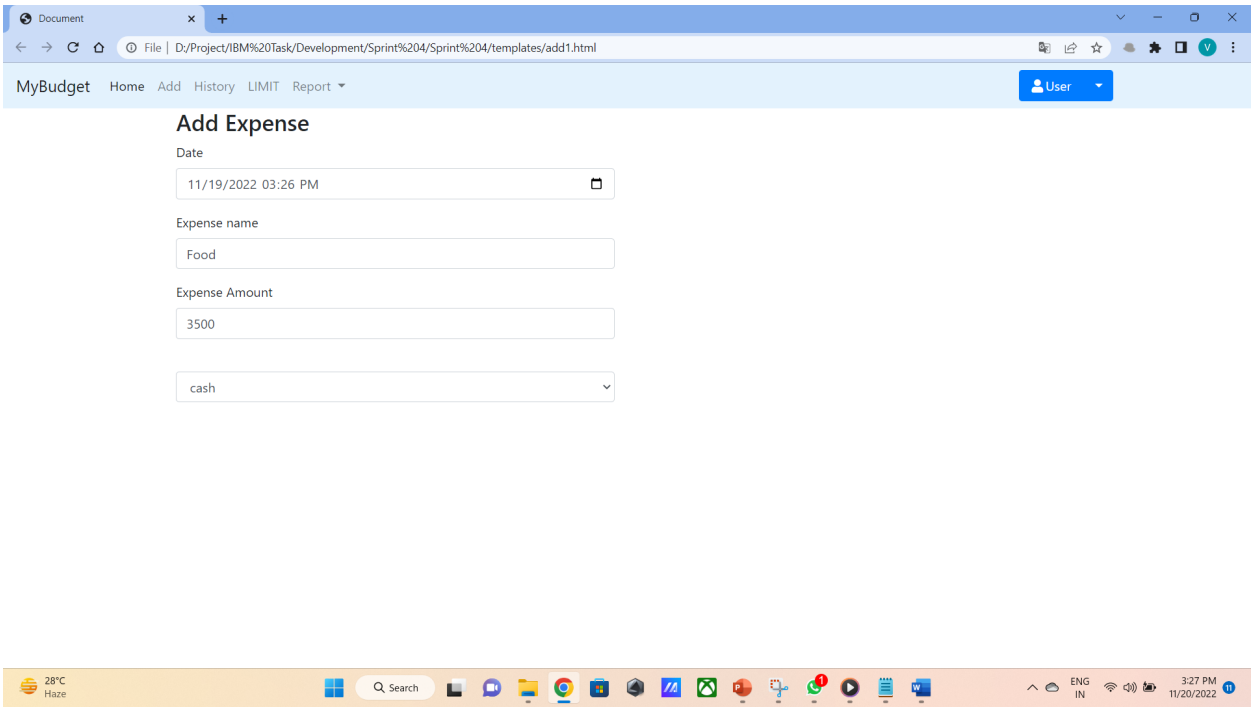





Document
File | D:/Project/IBM%20Task/Development/Sprint%204/Sprint%204/templates/home1.html
MyBudget
Home
Add
History
LIMIT
Report
User
BUDGET TRACKING
LET START JOURNEY
MyBudget web application helps
you to maintain budget
and analyse the expense
Let's Begin
28°C
Haze
Search
ENG
IN
3:16 PM
11/20/2022


Document
Document
File | D:/Project/IBM%20Task/Development/Sprint%204/Sprint%204/templates/display%20.html
MyBudget
Home
Add
History
LIMIT
Report
User
EXPENSES
{{row[2]}}
{{row[3]}}
₹ {{row[4]}}
{{row[5]}}
{{row[6]}}
Edit
Delete
ADD-DATA to Display
Sorry,
No Data Found !
28°C
Rainy tomorrow
Search
ENG
IN
3:21 PM
11/20/2022


Document
Document
File | D:/Project/IBM%20Task/Development/Sprint%204/Sprint%204/templates/display%20.html
MyBudget
Home
Add
History
LIMIT
Report
User
Expense Breakdown
Food
{{ t_food}}
Entertainment
{{ t_entertainment}}
Business
{{t_business}}
Rent
{{ t_rent }}
EMI
{{ t_EMI }}
Other
{{ t_other}}
Total
₹ {{total}}
EXPENSE BREAKDOWN
Food
Entertainment
Business
Rent
EMI
Other
28°C
Rainy tomorrow
Search
ENG
IN
3:21 PM
11/20/2022


Document
File | D:/Project/IBM%20Task/Development/Sprint%204/Sprint%204/templates/add1.html
MyBudget
Home
Add
History
LIMIT
Report
User
Add Expense
Date
11/19/2022 03:26 PM
Expense name
Food
Expense Amount
3500
cash
28°C
Haze
Search
ENG
IN
3:27 PM
11/20/2022

CHAPTER 10

ADVANTAGES & DISADVANTAGES

10.1 Advantages

I have a cheaper cell phone plan, using a smaller provider than the big monopoly providers.I don't watch tv, so no cable, though my husband has a Netflix account and I'll watch comedy specials or sci-fi series on occasionI don't subscribe to any music streaming or video games.Come to think of it, I've never been that much into entertainment – I don't buy tickets to concerts, sports games, or movies (I am in the minority for sure – a lot of people around me love watching movies but I forget movies so fast so I hardly go to the theater). What do I do for fun? I take walks with my family, work in the garden, read, surf online, nap, and I have a lot of occupations to keep me busy.I don't buy books, I borrow them at the library. I always have a lot of books on hold or on renewal for my family. I listen to a ton of audiobooks when I drive to and from work everyday, they are such a source of delight for me.I don't have healthcare premiums – I live in Canada.I don't eat out too much – we do batch cooking, I always bring my own lunch, snacks, and coffee in a thermos to work. Plus I am very picky about what I buy at the grocery store. We do not eat processed junk food or soda, I mostly buy high-quality meat, fruit, and dairy. Rice is very economical. I also grind my own coffee beans and bring my coffee to work, I NEVER buy Starbucks and I never eat at the cafeteria. Why should I pay more for inferior food and drink prepared by people who don't have health as a priority I have a SodaStream, which is one of life's joys. I love fizzy sparkling water, and now I never have to buy club soda again. I can use tap water and my SodaStream carbonates it for me! I even drink more water now because of it, and I bring it in a water bottle when I'm out and about.I don't shop for clothes – at age 41, I have enough clothes already and still fit and wear the same size clothes as when I was a teenager. Since I always use a drying rack instead of the dryer, my clothes never wear out either. Over the years I've donated the ones that I don't wear, and kept the ones that I do. Plus, the hospital provides sterile scrubs for work which is free! I am happy with my wardrobe and usually wear cheap Uniqlo leggings with a dress that is 20 years old.

10.2 Disadvantages

Your information is less secure, and probably being used and sold. If the service is free, then the product is you. Mint.com, like other financial apps, is a free service. They have to pay their bills somehow, so regardless of what their privacy policy may or may not say, just assume that your spending history and trends are going to be recorded and analyzed, by someone, somewhere. Now, you shouldn't have to worry about credit card fraud or identity theft, these companies are large enough and secure enough that you'll never have to worry about something like that. Just recognize that your information, most likely anonymous, will be used and potentially even sold. Personally, I have no problem with that, but if you do, then make sure you avoid these types of services. Automating everything to do with your finances can make you financially lazy. If your bills are paid automatically and your finances are track automatically, then what is there left for you to do? Not a lot, to be honest. So you might stop caring about what you're spending and where your money is going. Eventually you may look at your Mint data and realize that you've blown your budget over the last two months, but by then it is too late. So if you do choose to use this program, ensure that you are also being diligent in checking in on your finances. Set up a weekly or biweekly check for yourself to go through your finances and hit on all the important points.

CHAPTER 11

CONCLUSION

In this paper, we proposed a framework for job recommendation task. This framework facilitates the understanding of job recommendation process as well as it allows the use of a variety of text processing and recommendation methods according to the preferences of the job recommender system designer. Moreover, we also contribute mak-ing publicly available a new dataset containing job seekers pro les and job vacancies.Future directions of our work will focus on performing a more exhaustive evaluation considering a greater amount of methods and data as well as a comprehensive evaluation of the impact of each professional skill of a job seeker on the received job recommendation.

CHAPTER 12

FUTURE SCOPE

Automatically it will keep on sending notificationsnfor our daily expenditure. In today's busy and expensive life, we are in a great rush to make moneys, bu t 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. 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. Provision to add different currencies will be added so that also can be used worldwide and the currency converters will be designed and added in order to convert the different currency rates. A new tab named "Search" will be implemented so that if the user searches for any vendor, category or subcategory by name, he can see the expenses made on that particular search in a table view list with the total number of transactions made and the total expense amount for that search. This would provide a lot more flexibility for the users to track the particular expenses on particular items. Also, the graph reports show the expenses and income graphs separately in the current version. In the future, a comparison between the income made and expense will be shown graphically providing the user more options to see what they are making and what they are spending accordingly. A PDF feature would be implemented so that the user can see the total expenses/incomes in a much simpler PDF format in one file.

CHAPTER 13

APPENDIX

Source Code :

```
from flask import Flask, render_template, request, redirect, session
# from flask_mysqldb import MySQL
# import MySQLdb.cursors
import re
from flask_db2 import DB2
import ibm_db
import ibm_db_dbi
from sendemail import sendgridmail,sendmail

# from gevent.pywsgi import WSGIServer
import os
app = Flask(__name__)
app.secret_key = 'a'

"""
dsn_hostname = "ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
dsn_uid = "vmk08423"
dsn_pwd = "3KfJl6HGDtPdbIWy"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "31321"
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'] ='764264db-9824-4b7c-82df-
40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'

app.config['port'] = '32536'
app.config['protocol'] = 'tcpip'
app.config['uid'] = 'hpw99087'
app.config['pwd'] = 'NE5OkpBwb8rOk08k'
app.config['security'] = 'SSL'
try:
    mysql = DB2(app)
    conn_str='database=bludb;hostname=764264db-9824-4b7c-82df-
40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=32536;
protocol=tcpip;\
            uid=hpw99087;pwd=NE5OkpBwb8rOk08k;security=SSL'
    ibm_db_conn = ibm_db.connect(conn_str,'','')
    print("Database connected without any error !!")
except:
    print("IBM DB Connection error   :     " + DB2.conn_errormsg())
# app.config['']

# mysql = MySQL(app)
```

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

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

#SIGN--UP--OR--REGISTER
@app.route("/signup")
def signup():
   return render_template("signup.html")

@app.route('/register', methods =['GET', 'POST'])
def register():
   msg = ''
   print("Break point1")
   if request.method == 'POST' :
      username = request.form['username']
      email = request.form['email']
      password = request.form['password']
      print("Break point2" + "name: " + username + "------" + email + "------" +
password)
      try:
         print("Break point3")
         connectionID = ibm_db_dbi.connect(conn_str, '', '')
         cursor = connectionID.cursor()
         print("Break point4")
      except:
         print("No connection Established")

      # cursor = mysql.connection.cursor()
      # with app.app_context():
```

```
        #     print("Break point3")
        #     cursor = ibm_db_conn.cursor()
        #     print("Break point4")

        print("Break point5")
        sql = "SELECT * FROM register WHERE username = ?"
        stmt = ibm_db.prepare(ibm_db_conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        result = ibm_db.execute(stmt)
        print(result)
        account = ibm_db.fetch_row(stmt)
        print(account)
        param = "SELECT * FROM register WHERE username = '" + "\'" + username +
"\'"
        res = ibm_db.exec_immediate(ibm_db_conn, param)
        print("---- ")
        dictionary = ibm_db.fetch_assoc(res)
        while dictionary != False:
            print("The ID is : ", dictionary["USERNAME"])
            dictionary = ibm_db.fetch_assoc(res)

        print("break point 6")
        if account:
            msg = 'Username already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            sql2 = "INSERT INTO register (username, email,password) VALUES (?, ?, ?)"
            stmt2 = ibm_db.prepare(ibm_db_conn, sql2)
            ibm_db.bind_param(stmt2, 1, username)
            ibm_db.bind_param(stmt2, 2, email)
```

```
            ibm_db.bind_param(stmt2, 3, password)
            ibm_db.execute(stmt2)
            # cursor.execute('INSERT INTO register VALUES (NULL, % s, % s, % s)',
(username, email,password))
            # mysql.connection.commit()
            msg = 'You have successfully registered !'
        return render_template('signup.html', msg = msg)

 #LOGIN--PAGE
@app.route("/signin")
def signin():
    return render_template("login.html")

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

    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        # cursor = mysql.connection.cursor()
        # cursor.execute('SELECT * FROM register WHERE username = % s AND
password = % s', (username, password ),)
        # account = cursor.fetchone()
        # print (account)

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

```
        print(account)

        param = "SELECT * FROM register WHERE username = " + "\'" + username +
"\'" + " and password = " + "\'" + password + "\'"
        res = ibm_db.exec_immediate(ibm_db_conn, param)
        dictionary = ibm_db.fetch_assoc(res)

        # sendmail("hello sakthi","sivasakthisairam@gmail.com")
        if account:
            session['loggedin'] = True
            session['id'] = dictionary["ID"]
            userid = dictionary["ID"]
            session['username'] = dictionary["USERNAME"]
            session['email'] = dictionary["EMAIL"]
            return redirect('/home')
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)

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

@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():
    date = request.form['date']
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category = request.form['category']
    print(date)
    p1 = date[0:10]
    p2 = date[11:13]
```

```
    p3 = date[14:]
    p4 = p1 + "-" + p2 + "." + p3 + ".00"
    print(p4)
    # cursor = mysql.connection.cursor()
    # cursor.execute('INSERT INTO expenses VALUES (NULL,  % s, % s, % s, % s, % s, % s)', (session['id'] ,date, expensename, amount, paymode, category))
    # mysql.connection.commit()
    # print(date + " " + expensename + " " + amount + " " + paymode + " " + category)
    sql = "INSERT INTO expenses (userid, date, expensename, amount, paymode, category) VALUES (?, ?, ?, ?, ?, ?)"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, session['id'])
    ibm_db.bind_param(stmt, 2, p4)
    ibm_db.bind_param(stmt, 3, expensename)
    ibm_db.bind_param(stmt, 4, amount)
    ibm_db.bind_param(stmt, 5, paymode)
    ibm_db.bind_param(stmt, 6, category)
    ibm_db.execute(stmt)
    print("Expenses added")
    # email part
    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
```

```
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)
    total=0
    for x in expense:
        total += x[4]
    param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) +
" ORDER BY id DESC LIMIT 1"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = 0
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMITSS"])
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[0]
    if total > int(s):
        msg = "Hello " + session['username'] + " , " + "you have crossed the monthly
limit of Rs. " + s + "/- !!!" + "\n" + "Thank you, " + "\n" + "Team Personal Expense
Tracker."
        sendmail(msg,session['email'])
    return redirect("/display")

#DISPLAY---graph
@app.route("/display")
def display():
    print(session["username"],session['id'])
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT * FROM expenses WHERE userid = % s AND date
ORDER BY `expenses`.`date` DESC',(str(session['id'])))
```

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

#delete---the--data
@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
def delete(id):
    #  cursor = mysql.connection.cursor()
    #  cursor.execute('DELETE FROM expenses WHERE  id = {0}'.format(id))
    #  mysql.connection.commit()
    param = "DELETE FROM expenses WHERE  id = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    print('deleted successfully')
    return redirect("/display")

#UPDATE---DATA
@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
```

```
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT * FROM expenses WHERE  id = %s', (id,))
    # row = cursor.fetchall()

    param = "SELECT * FROM expenses WHERE  id = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        row.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)
    print(row[0])
    return render_template('edit.html', expenses = row[0])

@app.route('/update/<id>', methods = ['POST'])
def update(id):
  if request.method == 'POST' :
      date = request.form['date']
      expensename = request.form['expensename']
      amount = request.form['amount']
      paymode = request.form['paymode']
      category = request.form['category']
    #   cursor = mysql.connection.cursor()
    #   cursor.execute("UPDATE `expenses` SET `date` = % s , `expensename` = % s
, `amount` = % s, `paymode` = % s, `category` = % s WHERE `expenses`.`id` = % s
```

```
",(date, expensename, amount, str(paymode), str(category),id))



 #   mysql.connection.commit()
     p1 = date[0:10]
     p2 = date[11:13]
     p3 = date[14:]
     p4 = p1 + "-" + p2 + "." + p3 + ".00"
     sql = "UPDATE expenses SET date = ? , expensename = ? , amount = ?,
paymode = ?, category = ? WHERE id = ?"
     stmt = ibm_db.prepare(ibm_db_conn, sql)
     ibm_db.bind_param(stmt, 1, p4)
     ibm_db.bind_param(stmt, 2, expensename)
     ibm_db.bind_param(stmt, 3, amount)
     ibm_db.bind_param(stmt, 4, paymode)
     ibm_db.bind_param(stmt, 5, category)
     ibm_db.bind_param(stmt, 6, id)
     ibm_db.execute(stmt)
     print('successfully updated')
     return redirect("/display")

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

@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
    if request.method == "POST":
       number= request.form['number']
       #  cursor = mysql.connection.cursor()
       #  cursor.execute('INSERT INTO limits VALUES (NULL, % s, % s) ',(session['id'],
number))
```

```
        #  mysql.connection.commit()

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

@app.route("/limitn")
def limitn():
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT limitss FROM `limits` ORDER BY `limits`.`id` DESC
LIMIT 1')
    # x= cursor.fetchone()
    # s = x[0]
    param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) +
" ORDER BY id DESC LIMIT 1"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = " /-"
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMITSS"])
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[0]
    return render_template("limit.html" , y= s)

#REPORT
@app.route("/today")
def today():
    #   cursor = mysql.connection.cursor()
```

```
    #   cursor.execute('SELECT TIME(date)   , amount FROM expenses  WHERE
userid = %s AND DATE(date) = DATE(NOW()) ',(str(session['id'])))
    #   texpense = cursor.fetchall()
    #   print(texpense)

      param1 = "SELECT TIME(date) as tn, amount FROM expenses WHERE userid
= " + str(session['id']) + " AND DATE(date) = DATE(current timestamp) ORDER BY
date DESC"
      res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
      dictionary1 = ibm_db.fetch_assoc(res1)
      texpense = []
      while dictionary1 != False:
         temp = []
         temp.append(dictionary1["TN"])
         temp.append(dictionary1["AMOUNT"])
         texpense.append(temp)
         print(temp)
         dictionary1 = ibm_db.fetch_assoc(res1)
    #   cursor = mysql.connection.cursor()
    #   cursor.execute('SELECT * FROM expenses WHERE userid = % s AND
DATE(date) = DATE(NOW()) AND date ORDER BY `expenses`.`date`
DESC',(str(session['id'])))
    #   expense = cursor.fetchall()

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

```
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    expense.append(temp)
    print(temp)
    dictionary = ibm_db.fetch_assoc(res)


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

for x in expense:
    total += x[4]
    if x[6] == "food":
        t_food += x[4]
    elif x[6] == "entertainment":
        t_entertainment  += x[4]
    elif x[6] == "business":
        t_business  += x[4]
    elif x[6] == "rent":
        t_rent  += x[4]
    elif x[6] == "EMI":
        t_EMI  += x[4]
    elif x[6] == "other":
        t_other  += x[4]
print(total)
print(t_food)
print(t_entertainment)
```

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

@app.route("/month")
def month():
   #   cursor = mysql.connection.cursor()
   #   cursor.execute('SELECT DATE(date), SUM(amount) FROM expenses WHERE
userid= %s AND MONTH(DATE(date))= MONTH(now()) GROUP BY DATE(date)
ORDER BY DATE(date) ',(str(session['id'])))
   #   texpense = cursor.fetchall()
   #   print(texpense)
    param1 = "SELECT DATE(date) as dt, SUM(amount) as tot FROM expenses
WHERE userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current
timestamp) AND YEAR(date) = YEAR(current timestamp) GROUP BY DATE(date)
ORDER BY DATE(date)"
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpense = []
    while dictionary1 != False:
       temp = []
       temp.append(dictionary1["DT"])
       temp.append(dictionary1["TOT"])
       texpense.append(temp)
       print(temp)
       dictionary1 = ibm_db.fetch_assoc(res1)
   #   cursor = mysql.connection.cursor()
   #   cursor.execute('SELECT * FROM expenses WHERE userid = % s AND
```

```
MONTH(DATE(date))= MONTH(now()) AND date ORDER BY `expenses`.`date`
DESC',(str(session['id'])))
   #   expense = cursor.fetchall()

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



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


    for x in expense:
```

```
        total += x[4]
        if x[6] == "food":
            t_food += x[4]

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

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

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

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

    print(total)

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



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

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

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

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



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

     param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + "
AND YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
```

```
res = ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res)
expense = []
while dictionary != False:
    temp = []
    temp.append(dictionary["ID"])
    temp.append(dictionary["USERID"])
    temp.append(dictionary["DATE"])
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    expense.append(temp)
    print(temp)
    dictionary = ibm_db.fetch_assoc(res)


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

for x in expense:
    total += x[4]
    if x[6] == "food":
        t_food += x[4]
    elif x[6] == "entertainment":
        t_entertainment  += x[4]
    elif x[6] == "business":
        t_business  += x[4]
    elif x[6] == "rent":
```

```
            t_rent  += x[4]
        elif x[6] == "EMI":
            t_EMI  += x[4]
        elif x[6] == "other":
            t_other  += x[4]
    print(total)
    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)
    return render_template("today.html", texpense = texpense, expense = expense,
total = total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )

#log-out

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


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

GitHub :

https://github.com/IBM-EPBL/IBM-Project-1034-1658335396.git

Demo Link:

https://drive.google.com/file/d/1eaw8E9Ml8Es_aa76dmPc7AZHRlawfivx/view