PERSONAL EXPENSE TRACKER APPLICATION

IBM PROJECT REPORT

submitted by

ARUNKUMAR C

ARUNKUMAR M

BALASUBRAMANIYAN R

GOKULNATH G

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1. PROJECT OVERVIEW

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.

1.2. PURPOSE

Personal finance 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. 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.

Also known as expense manager and money manager, an expense tracker is a software or application that helps to keep an accurate record of your money inflow and outflow. Many people in India live on a fixed income, and they find that towards the end of the month they don't have sufficient money to meet their needs. While this problem can arise due to low salary, invariably it is due to poor money management skills.

People tend to overspend without realizing, and this can prove to be disastrous. Using a daily expense manager can help you keep track of how much you spend every day and on what. At the end of the month, you will have a clear picture where your money is going. This is one of the best ways to get your expenses under control and bring some semblance of order to your finances.

CHAPTER 2

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1. EXISTING PROBLEM

Some research and journals have been reviewed throughout this project to make out a distinct image of it. These journals in short, works as a guide for this project to implement Least Square Method. Based on article [4], it discusses about regression models. Basically, it holds a concept where we forecast the time series of interest at y-axis assuming that it has a linear relationship with other time series at x-axis. The author [4] also stated that, the least squares principle provides a way of determining the coefficients effectively by minimizing the sum of the squared errors. A study carried out by author [9], it introduces tools and methods for both finance and accounting that help with asset pricing, corporate finance, options and futures, and conducting financial accounting research. How least square method works and implied in financial forecasting is discussed. The author [2] applied several of statistical time series models to observe forecast errors in the demand of juice production are within the expected limit and to choose a forecasting technique which has a less relative error. The author [2] proved that Least Square Method is more accurate than the others. Article [3] also did the study in order to forecast milk production in India using statistical time series modeling[1]Double Exponential Smoothing and Auto-regressive Integrated Moving Average and concluded that Auto Regressive Integrated Moving Average performed better. 8 In a paper studied by [7] explains that Batch-mode Least Squares SVM (LSSVM) is often associated with unbounded number of support vectors (SVs). This, makes it unsuitable for applications if it involves large-scale streaming data. In this paper [7], it explains how to train the limited-scale LSSVM dynamically. By applying a budget online LSSVM (BOLSSVM) algorithm, methodologically, by setting a fixed budget for SVs, LSSVM model is updated according to the current SVs set dynamically without re-training from scratch. This way, the proposed BOLSSVM algorithm is especially useful for online prediction tasks. Thus, batch-mode learning methods were compared, the computational complexity of the proposed BOLSSVM method is significantly reduced. The validity and effectiveness of the proposed BOLSSVM algorithm is shown by the experimental results of classification and regression on benchmark data-sets and real-world applications. The paper [10] aims to describe a computerized system that can predict the budget for the new year based on past budgets by using time series analysis. It will then show results with most minimum errors and controls the budget during the year. Through the ability to control exchange, compared to the scheme with the investigator and calculating the deviation, measurement of performance ratio and the growth of some indicators relating to budgets, it is possible to achieve the objective. For example, this article [10] uses the rate of condensation of capital, the growth rate and profitability ratio and gives a clear indication whether these ratios are good or not.

One of the most common existing software that is related to this project is MINT. Mint was formally introduced in September 2007. it is a server-based web, but this software also can be used using PC or smart-phone. Based on a research from author [1], MINT is aware of users' daily expense and if they have a future goal of buying something, user can reduce your current spending according to it. Most importantly, it keeps a track on users' credit bills, home bills and savings. This budgeting software also will notify users whenever user are due to pay a bill or payment. This will lower the chance for users to forget to make payment. Despite having some great advantages, MINT also comes with a plenty of drawbacks such as there no guarantee of the security in this online software. The chances of getting their account hacked is worrisome as this software stores users' financial account. The rivalry from other potential software also becomes one of the big factor. Website has too many advertisements while browsing through finances.

2.2. REFERENCES

[1] Morgan L. (2017). Why Do People Think Mint is Bad For Budgeting. Investing Education.[online] Retrieved from: http://www.investinged.com/why-do-people-think-mint[1]is-bad-for-budgeting/ [Accessed 1 April 2017].

[2] Kumar, R and Mahto, D 2013, 'A case study : Application of Proper Forecasting Technique in Juice Production', Global Journal of Researches in Engineering, vol. 13, no. 4, pp. 1-6

[3] Pal, S, Ramasubramanian, V and Mehta, SC 2007, 'Statistical Models for Forecasting Milk Production in India', Journal of Indian Society of Agricultural Statistics, vol.61, no.2, pp. 80- 83.

[4] Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice.

[5] Thanapal, P., Patel, M., Lokesh Raj, T., & Satheesh Kumar, J. (2015). Income and Expense Tracker. Indian Journal Of Science And Technology, 8(S2), 118-122.

[6] Chauhan, B. D., Rana, A., & Sharma, N. K. (2017, September). Impact of development

methodology on cost & risk for development projects. In 2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO) (pp. 267-272). IEEE

[7] Jian, L., Shen, S., Li, J., Liang, X., & Li, L. (2016). Budget online learning algorithm for least squares SVM. IEEE transactions on neural networks and learning systems, 28(9), 2076-2087.

[8] Gaither, G. H., Dukes, F. O., & Swanson, J. R. (1981). ENROLLMENT FORECASTING: USE OF A MULTIPLE-METHOD MODEL FOR PLANNING AND BUDGETING. Decision Sciences, 12(2), 217-230.

[9] Lee, C. F., Chen, H. Y., & Lee, J. (2019). Econometric Approach to Financial Analysis, Planning, and Forecasting. In Financial Econometrics, Mathematics and Statistics (pp. 125- 157). Springer, New York, NY

[10] Elbasheer, F. A., & Samani, A. T. (2014). Forecasting Budget Estimated Using Time[1]Series. Intelligent Information Management, 2014, 6, 142-148, Published Online May 2014 in SciRes. http://www. scirp. org/journal/iim.

2.3. PROBLEM STATEMENT DEFINITION

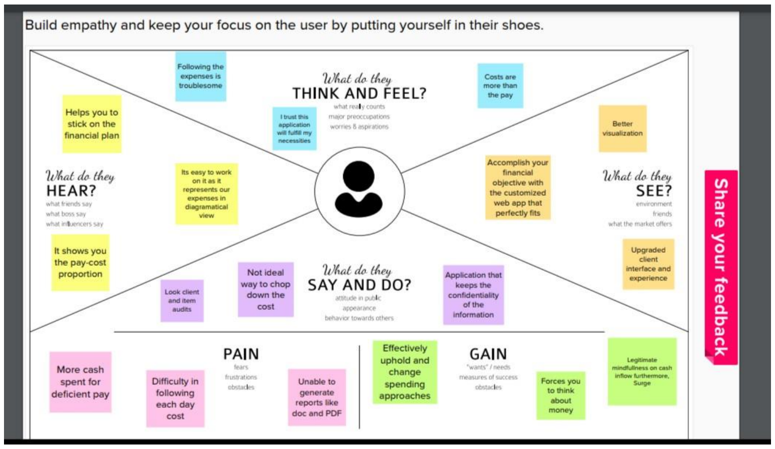
For a very long time, managing finances and accounts has been a real problem. People are less inclined to use spreadsheets or chequebooks to track their spending. Despite this, modern technology and the internet's increasing accessibility have given it a new viewpoint during the past few decades. We can track our spending with the use of an expense tracker. Additionally, it can assist us in identifying spending patterns and keeping track of forthcoming bill payments. It is a web[1]based programme that can monitor their expenditure and ascertain whether they are staying within their allotted budget. The necessary information, including the expense amount, merchant, category, and date the expense was made, must be entered by potential users. This mobile system is a comprehensive expense tracking tool that will not only assist users in keeping tabs on their spending but also reduce unnecessary spending, hence promoting a responsible lifestyle.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.IDEATION AND PROPOSED SOLUTION

3.1. EMPATHY MAP CANVAS



Build empathy and keep your focus on the user by putting yourself in their shoes.
Following the expenses is troublesome
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Costs are more than the pay
I trust this application will fulfill my necessities
Helps you to stick on the financial plan
Better visualization
What do they HEAR?
what friends say
what boss say
what influencers say
Its easy to work on it as it represents our expenses in diagramatical view
Accomplish your financial objective with the customized web app that perfectly fits
What do they SEE?
environment
friends
what the market offers
It shows you the pay-cost proportion
Upgraded client interface and experience
Not ideal way to chop down the cost
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
Look client and item audits
Application that keeps the confidentiality of the information
Share your feedback
PAIN
fears
frustrations
obstacles
More cash spent for deficient pay
Difficulty in following each day cost
Unable to generate reports like doc and PDF
Effectively uphold and change spending approaches
GAIN
"wants" / needs
measures of success
obstacles
Forces you to think about money
Legitimate mindfulness on cash inflow furthermore, Surge

Fig-3.1 Empathy map

3.2. IDEATION & BRAINSTORMING

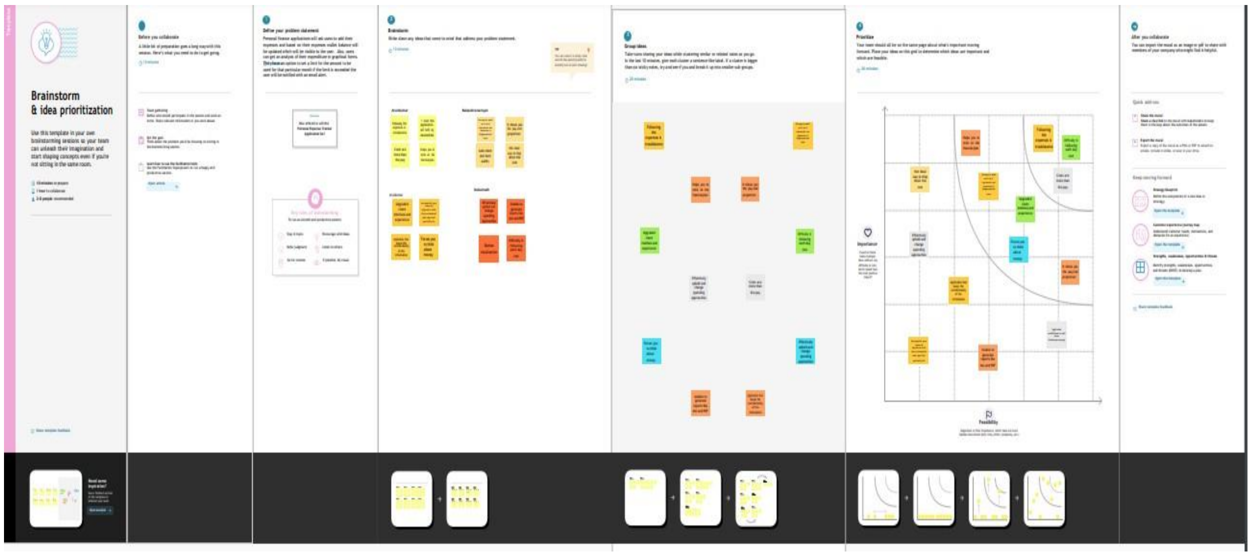


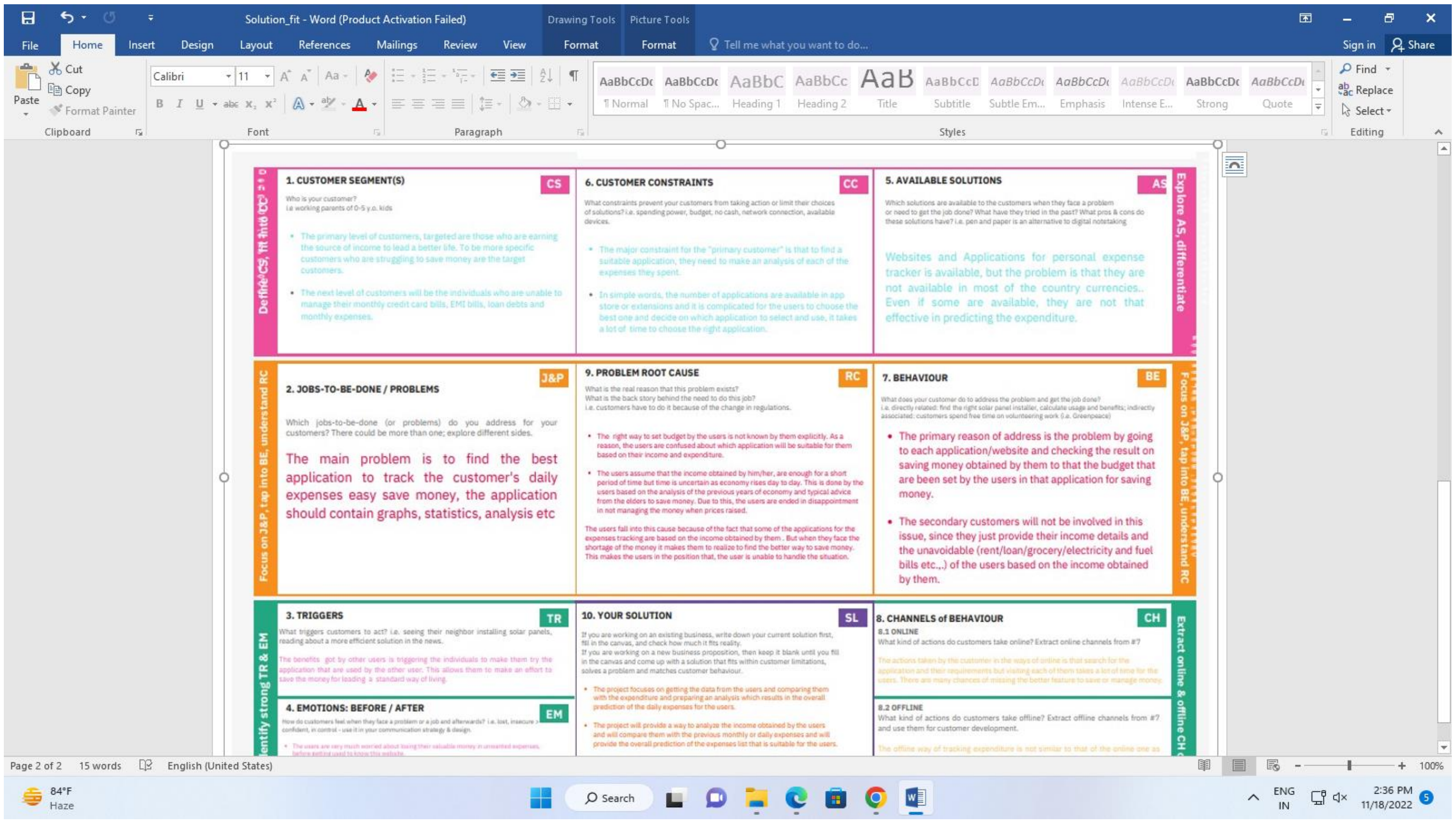
Fig-3.2 Ideation & Brainstorming

3.3. PROPOSED SOLUTION

S.no	Parameter	Description
1	**Problem Statement**	● By tracking expenses and following a plan, a budget makes it easier to pay bills on time, build an emergency fund, and save for major expenses such as a car or home. ● A Daily Expense Tracker is a one kind of digital diary that helps to keep an eye on all of our money related transitions and also provides all financial activities report daily, weekly, monthly and yearly. ● At the instant, there is no as such complete solution present easily or we should say free of cost which enables a person to keep a track of its daily expenditure easily.
2	**Idea/Solution Description**	● To develop a systematic system that will help to improve users' financial management and forecast future budget planning. ● To test and evaluate the reliability of the system to
		generate monthly report and forecast budget for the users. ● Precisely keeping track of user's expenses as well as their budgeting.

3	**Novelty /Uniqueness**	1. This application is a very simple and user-friendly application for the common people. 2. user data security and has a dashboard for monitoring the entire system. 3. Backup and Restore all information.
4	**Social Impact**	1. This application helps the user to avoid unwanted expenses and bad financial situations. 2. It will guide them and make them aware about their daily expenses. 3. This application will help its users to overcome the wastage of money.
5	**Business Model**	1. This system can only be used by individuals as it includes only personal expenses. And only admin is allowed to manage the maintenance of the system. 2. Expenses Tracker is a way that can help us to keep up with our spending. Not only that, it can help us pinpoint areas that we have been spending and track upcoming bill payments
6	**Scalability of the Solution**	1. Cost effectiveness - Cloud providers only charge for what an organization uses, so there is no need to pay for obsolete or redundant equipment. 2. Reliability - Organizations can rest assured they will see high performance, as scalable architecture can meet sudden increases or decreases in demand.

3.4. PROBLEM SOLUTION FIT



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e working parents of 0-5 y.o. kids

- The primary level of customers, targeted are those who are earning the source of income to lead a better life. To be more specific customers who are struggling to save money are the target customers.
- The next level of customers will be the individuals who are unable to manage their monthly credit card bills, EMI bills, loan debts and monthly expenses.

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

- The major constraint for the "primary customer" is that to find a suitable application, they need to make an analysis of each of the expenses they spent.
- In simple words, the number of applications are available in app store or extensions and it is complicated for the users to choose the best one and decide on which application to select and use, it takes a lot of time to choose the right application.

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face a problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

Websites and Applications for personal expense tracker is available, but the problem is that they are not available in most of the country currencies.. Even if some are available, they are not that effective in predicting the expenditure.

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

The main problem is to find the best application to track the customer's daily expenses easy save money, the application should contain graphs, statistics, analysis etc

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

- The right way to set budget by the users is not known by them explicitly. As a reason, the users are confused about which application will be suitable for them based on their income and expenditure.
- The users assume that the income obtained by him/her, are enough for a short period of time but time is uncertain as economy rises day to day. This is done by the users based on the analysis of the previous years of economy and typical advice from the elders to save money. Due to this, the users are ended in disappointment in not managing the money when prices raised.

The users fall into this cause because of the fact that some of the applications for the expenses tracking are based on the income obtained by them . But when they face the shortage of the money it makes them to realize to find the better way to save money. This makes the users in the position that, the user is unable to handle the situation.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

- The primary reason of address is the problem by going to each application/website and checking the result on saving money obtained by them to that the budget that are been set by the users in that application for saving money.
- The secondary customers will not be involved in this issue, since they just provide their income details and the unavoidable (rent/loan/grocery/electricity and fuel bills etc.,.) of the users based on the income obtained by them.

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

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbor installing solar panels, reading about a more efficient solution in the news.

The benefits got by other users is triggering the individuals to make them try the application that are used by the other user. This allows them to make an effort to save the money for leading a standard way of living.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards? i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

- The users are very much worried about losing their valuable money in unwanted expenses, before getting used to know this website.

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

- The project focuses on getting the data from the users and comparing them with the expenditure and preparing an analysis which results in the overall prediction of the daily expenses for the users.
- The project will provide a way to analyze the income obtained by the users and will compare them with the previous monthly or daily expenses and will provide the overall prediction of the expenses list that is suitable for the users.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

The actions taken by the customer in the ways of online is that search for the application and their requirements but visiting each of them takes a lot of time for the users. There are many chances of missing the better feature to save or manage money.

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

The offline way of tracking expenditure is not similar to that of the online one as

Extract online & offline CH

Fig-3.3 problem solution fit

CHAPTER 4

REQUIREMENT ANALYSIS

4. REQUIREMENT ANALYSIS

4.1. FUNCTIONAL REQUOREMENTS

FR No.	Functional Requirement (Epic)	Sub Requirement (Story/ Sub-Task)
FR-1	**User Registration**	Form for collecting details
FR-2	**Login**	Enter usernameand password
FR-3	**Calendar**	Personal expense tracker application must allow user to addthe data to their expenses.
FR-4	**Expense Tracker**	This application should graphically represent the expense inthe formof report.
FR-5	**Report generation**	Graphical representation of report mustbe generated.
FR-6	**Category**	This application shall allow users to add categories oftheir expenses.

4.2. Non-functional Requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Helps to keep an accurate recordof your incomeand expenses.
NFR-2	**Security**	Budget tracking apps are considered very safe from thosewho commit cybercrimes.
NFR-3	**Reliability**	Each data record is stored on a wellbuilt efficientdatabase schema. There is no risk of data loss.

NFR-4	**Performance**	The types of expense arecategories along with an option. Throughput of the systemis increased due tolight weight database support.
NFR-5	**Availability**	The application musthave a 100% up-time.
NFR-6	**Scalability**	The abilityto appropriately handleincreasing demands.

CHAPTER 5

PROJECT DESIGN

5. PROJECT DESIGN

5.1. DATA FLOW DIAGRAMS

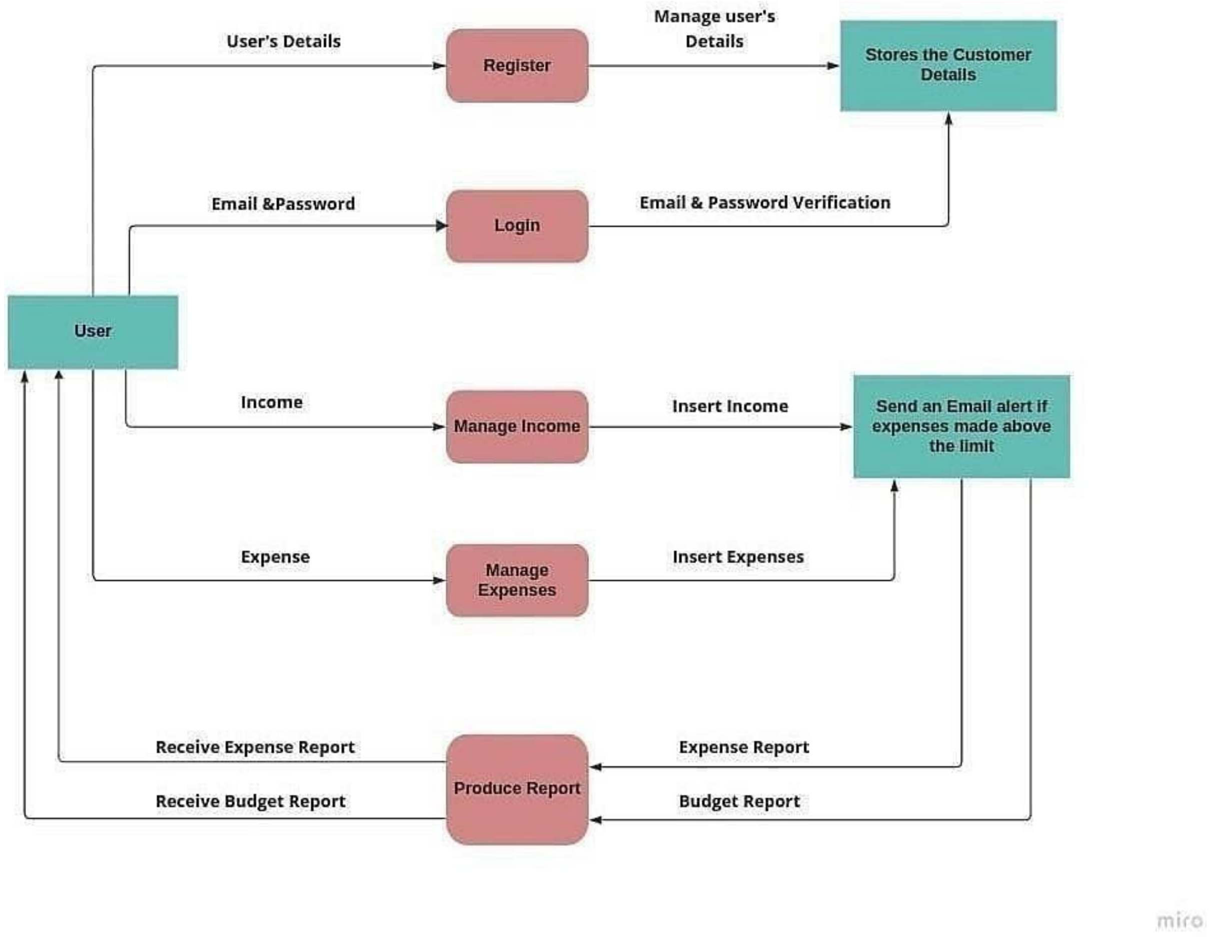


Fig 5.1 dataflow diagram

5.2 SOLUTION & TECHNICAL ARCHITECTURE

The Deliverable shall include the architectural diagramas below and the information as per the table1 &table 2

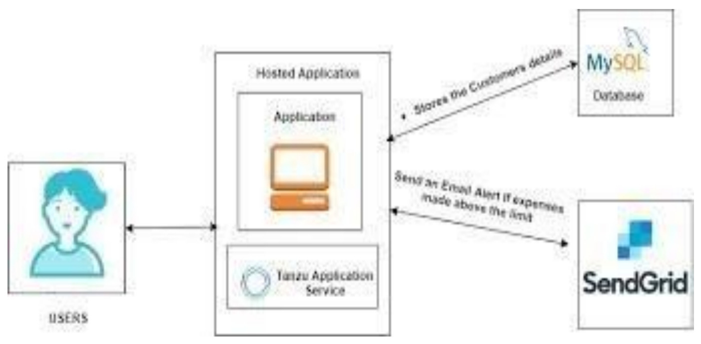

Hosted Application
Application
Tanzu Application Service
USERS
Stores the Customers details
MySQL
Database
Send an Email Alert if expenses made above the limit
SendGrid

Fig 5.2 technical architecture

Guidelines:

1. Include all the processes (As an application logic / Technology Block)
2. Provide infrastructural demarcation (Local / Cloud)
3. Indicateexternal interfaces (thirdparty API's etc.)
4. Indicate Data Storage components / services
5. Indicateinterface to machinelearning models (if applicable)

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 byenteringmy email, password, and confirming my password.	I can access my account/dashboard	High	
	Login	USN-2	As a user,I can log into the applicationby entering email& password	I can access the applicati on	High	
	Dashboard	USN-3	As a user I can enter my income and expenditure details.	I can view my daily expenses	High	
Customer Care Executive		USN-4	As a customer care executive, I can solvethe log inissues and other issuesof the application.	I can provide support or solution at any time24*7	Medium	
Administrator	Application	USN-5	As an administratorI can upgradeor update the application.	I can fix the bug whicharises for the customers and users of the application	Medium	

CHAPTER 6
PROJECT PLANNING
&
SCHEDULING

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	Arunkumar C, Arunkumar M, Balasubramaniyan,Gokulnath
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Arunkumar C, Arunkumar M, Balasubramaniyan,Gokulnath
Sprint-2		USN-3	As a user, I can register for the application through Facebook	2	Low	Arunkumar C, Arunkumar M, Balasubramaniyan, Gokulnath
Sprint-1		USN-4	As a user, I can register for the application throughGmail	2	Medium	Arunkumar C, Arunkumar M, Balasubramaniyan, Gokulnath
Sprint-1	Login	USN-5	As a user, I can log into the application by enteringemail& password	1	High	Arunkumar C, Arunkumar M, Balasubramaniyan, Gokulnath

Sprint-3	Login	USN-5	As a, Customer Care Executive I can log into the application by entering serveremail & password	2	High	Arunkumar C, Arunkumar M, Balasubramaniya n,Gokulnath
Sprint-4	Login	USN-5	As a Administrator, I can log into the application by entering sever email & password	2	High	Arunkumar C, Arunkumar M, Balasubramaniya n,Gokulnath

6.2 SPRINT DELIVERY SCHEDULE

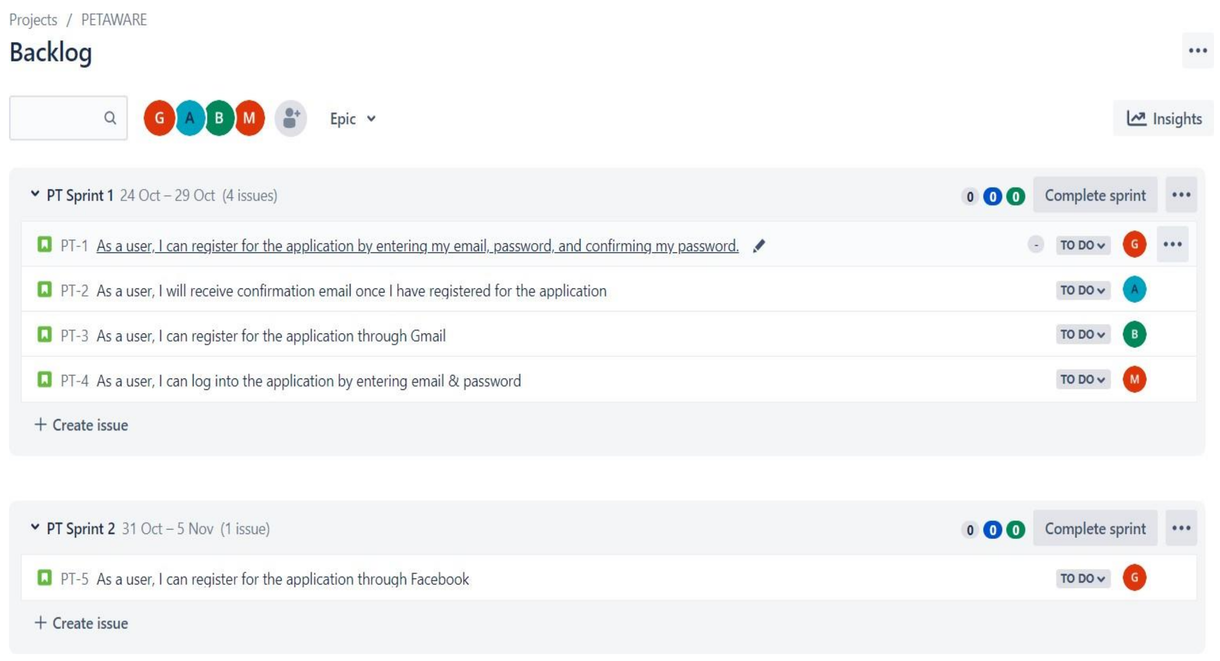
Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date (Planne d)**	**Story Points Completed (as onPlannedEnd Date)**	**Sprint Release Date (Actual)**
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	30 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	06 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	14 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.2.1. VELOCITY:

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

6.3 REPORTS FROM JIRA

SPRINTS:



Projects / PETAWARE

Backlog

Epic

Insights

PT Sprint 1 24 Oct – 29 Oct (4 issues) 0 0 0 Complete sprint

PT-1 As a user, I can register for the application by entering my email, password, and confirming my password. TO DO G

PT-2 As a user, I will receive confirmation email once I have registered for the application TO DO A

PT-3 As a user, I can register for the application through Gmail TO DO B

PT-4 As a user, I can log into the application by entering email & password TO DO M

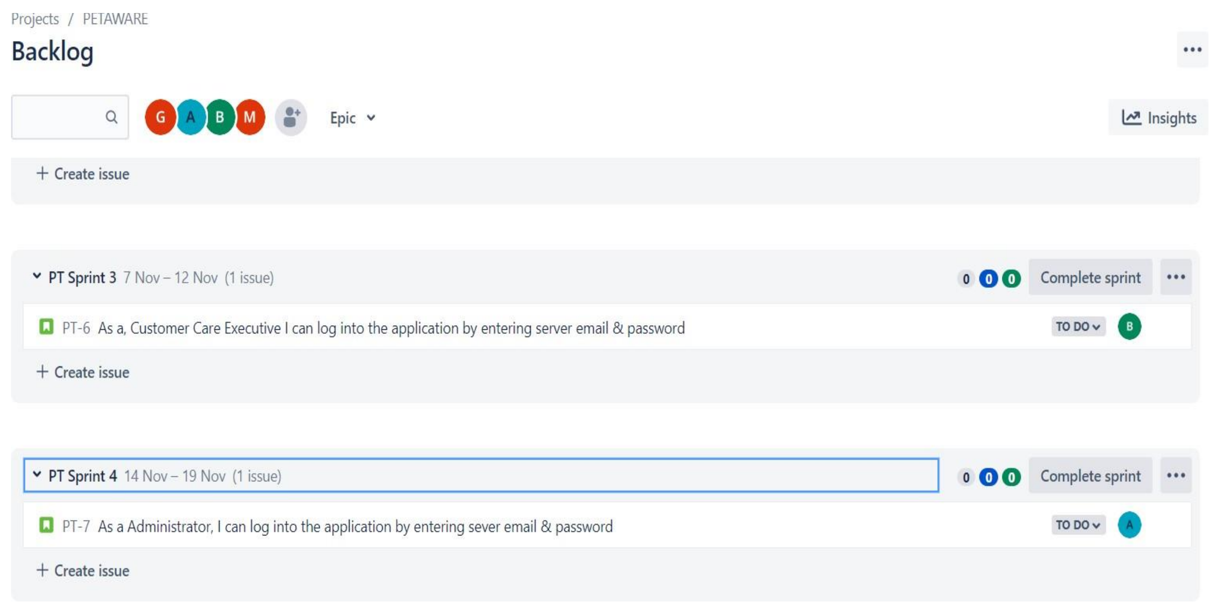
+ Create issue

PT Sprint 2 31 Oct – 5 Nov (1 issue) 0 0 0 Complete sprint

PT-5 As a user, I can register for the application through Facebook TO DO G

+ Create issue

fig 6.1 sprints-1


Projects / PETAWARE

Backlog

PT Sprint 3 7 Nov – 12 Nov (1 issue)

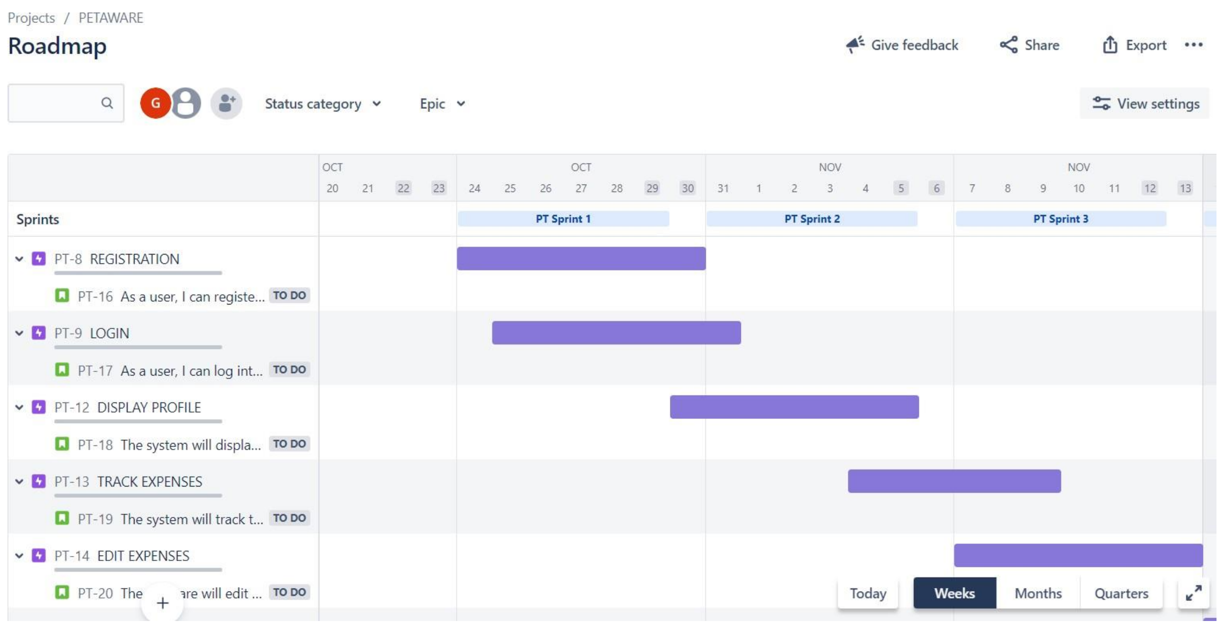
PT-6 As a, Customer Care Executive I can log into the application by entering server email & password — TO DO — B

PT Sprint 4 14 Nov – 19 Nov (1 issue)

PT-7 As a Administrator, I can log into the application by entering sever email & password — TO DO — A

fig 6.2 sprints-2

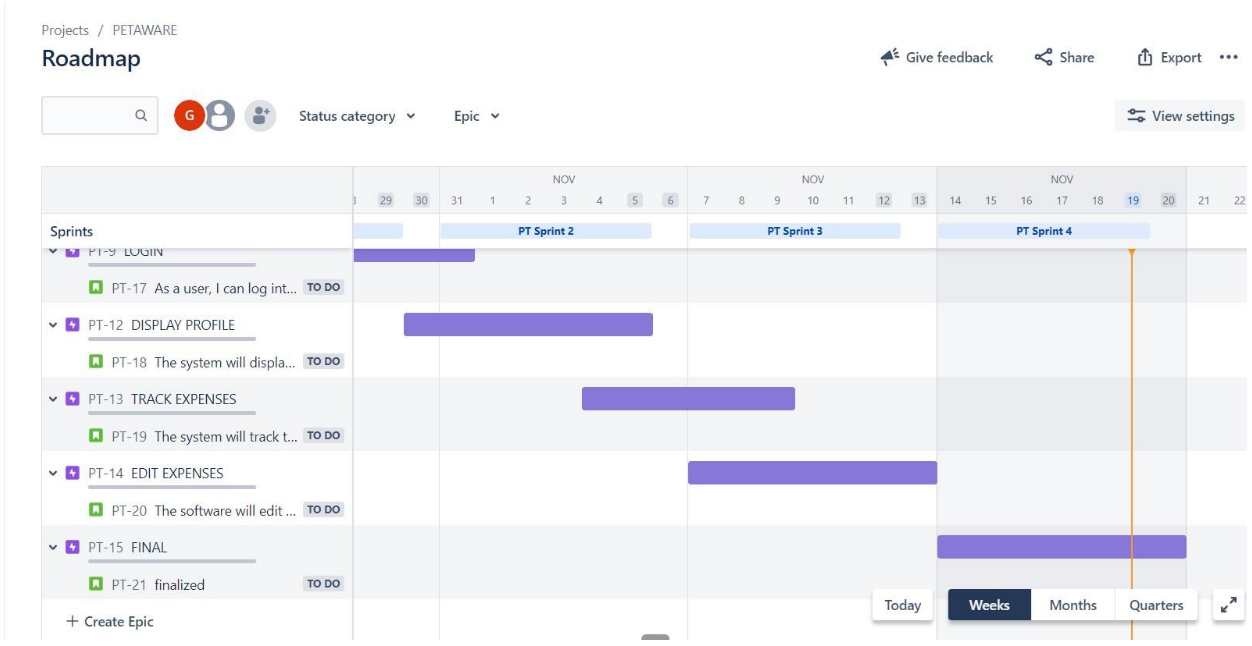
ROADMAP


Projects / PETAWARE

Roadmap

Sprints	PT Sprint 1	PT Sprint 2	PT Sprint 3
PT-8 REGISTRATION			
PT-16 As a user, I can registe... TO DO			
PT-9 LOGIN			
PT-17 As a user, I can log int... TO DO			
PT-12 DISPLAY PROFILE			
PT-18 The system will displa... TO DO			
PT-13 TRACK EXPENSES			
PT-19 The system will track t... TO DO			
PT-14 EDIT EXPENSES			
PT-20 The ...are will edit ... TO DO			

fig 6.3 roadmap-1


Projects / PETAWARE

Roadmap

PT-9 LOGIN
PT-17 As a user, I can log int... TO DO
PT-12 DISPLAY PROFILE
PT-18 The system will displa... TO DO
PT-13 TRACK EXPENSES
PT-19 The system will track t... TO DO
PT-14 EDIT EXPENSES
PT-20 The software will edit ... TO DO
PT-15 FINAL
PT-21 finalized TO DO

fig 6.4 roadmap-2

CHAPTER 7
CODING & SOLUTIONING

7. CODING AND SOLUTIONING

7.1. FEATURE 1

Front-End Development:

A front-end development architects and develops websites and applications using web technologies (i.e., HTML, CSS, DOM, and JavaScript), which run on the Open Web Platform or act as compilation input for non-web platform environments.

The main features that we added in our software are

i. registration.html

ii. dashboard.html

FEATURE 1

registraton.html

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Sign Up Form</title>

  <link href='https://fonts.googleapis.com/css?family=Nunito:400,300'
rel='stylesheet' type='text/css'>
  <link rel="stylesheet" href="{{ url_for('static',
filename='css/register.css') }}">
</head>
<body>

<form action="/register" method="POST">

  <h1>Sign Up</h1>

  <fieldset>
    <legend><span class="number">1</span>Your basic info</legend>
    <label for="name">Name:</label>
    <input type="text" name="name" id="name" required>
    <label for="name">DOB:</label>
    <input type="date" id="name" name="dob" required>
    <label for="name">Occupation:</label>
    <input type="text" id="name" name="occupation" required>
    <label>Sex:</label>
    <input type="radio" id="under_13" value="male" name="gender"><label
```

```
for="under_13" class="light">Male</label> &nbsp
    <input type="radio" id="over_13" value="female" name="gender"><label
for="over_13" class="light">Female</label><br><br>

    <label for="mail">Email:</label>
    <input type="text" id="mail" name="email" required>

    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>
    <label for="name">Initial amount:</label>
    <input type="number" id="name" min="10000"
           max="10000000" name="intial_amount" required >
    <label for="name">Phone Number:</label>
    <input type="text" id="name" name="number" title="Enter 10 digit
number" pattern="[1-9]{1}[0-9]{9}" required>
    <label for="bio">Address:</label>
    <textarea id="bio" name="address" rows="4" columns="30"
required></textarea>
    </fieldset>
  <button type="submit" class="bl">Sign Up</button>
  <button type="reset" class="bl">Reset</button>
  <a href="/logging" class="al">Already registered?</a>
</form>

</body>
</html>
```

dashboard.html

```
<!DOCTYPE html>

<html>

<head>
      <title>Simple web Development Template</title>

      <style>
            * {
                  margin: 0;
                  padding: 0;
            }

            .navbar {
                  display: flex;
                  position: sticky;
                  top: 0;
                  cursor: pointer;
            }

            .background {
                  background: black;
                  background-blend-mode: darken;
                  background-size: cover;
            }

            .nav-list {
                  width: 70%;
                  display: flex;
                  margin-left:0px;
            }

            .logo {
                  padding-left:0px;
            }



            .nav-list li {
                  list-style: none;
                  padding: 26px 30px;
            }

            .nav-list li a {
                  text-decoration: none;
                  color: white;
            }

            .nav-list li a:hover {
                  color: grey;
            }

            .rightnav {
                  width: 30%;
                  text-align: right;
            }

            #search {
                  padding: 5px;
                  font-size: 17px;
                  border: 2px solid grey;
                  border-radius: 9px;
            }
```

```
.firstsection {
      height: 300px;
}

.secondsection {
      height: 400px;
}

.box-main {
      display: flex;
      justify-content: center;
      align-items: center;
      color:#EF32D9;
      max-width: 80%;
      margin: auto;
      height: 80%;
}

.firsthalf {
      width: 100%;
      display: flex;
      flex-direction: column;
      justify-content: center;
}

.secondhalf {
      width: 30%;
}

.secondhalf img {
      width: 70%;
      border: 4px solid white;
      border-radius: 150px;
      display: block;
      margin: auto;
}

.text-big {
      font-family:cursive;
      font-weight: bold;
      font-size: 30px;
}

.text-small {
      font-size: 18px;
```

```css
}

.btn {
      padding: 8px 20px;
      margin: 7px 0;
      border: 2px solid white;
      border-radius: 8px;
      background:  none;
      color: white;
      cursor: pointer;
}

.btn-sm {
      padding: 6px 10px;
      vertical-align: middle;
}

.section {
      height: 400px;
      display: flex;
      max-width: 90%;
      margin: auto;
      margin-bottom:0px;
}

.section-Left {
      flex-direction: row-reverse;
}

.paras {
      padding: 0px 65px;
      padding-bottom:0px;
}

.thumbnail img {
      width: 300px;
      border: 2px solid black;
      border-radius: 26px;
      height:300px;
      margin-top: 19px;
      margin-bottom:0px;
}

.center {
      text-align: center;
}
```

```
            .text-footer {
                  text-align: center;
                  padding: 30px 0;
                  font-family:cursive;
                  display: flex;
                  justify-content: center;
                  color: white;
            }
            a{
            font-family:cursive;
            }
      </style>
</head>

<body style ="background: radial-gradient(circle at 0.8% 3.1%, rgb(255, 
188, 224) 0%, rgb(170, 165, 255) 46%, rgb(165, 255, 205) 100.2%);">
      <nav class="navbar background">
            <ul class="nav-list">
                  <div class="logo">
                        <img src="{{ url_for('static', 
filename='css/one.png') }}" style="height:50px;width:50px;margin-
left:0px;margin-top:10px">
                  </div>
                  <li><a href="/profile">Profile</a></li>
                  <li><a href="/expenses">Expenses</a></li>
                  <li><a href="/display">Display</a></li>
                  <li><a href="/addBalance">Add Balance</a></li>
                  <li><a href="/login">Logout</a></li>
            </ul>


      </nav>

      <section class="firstsection">
            <div class="box-main">
                  <div class="firstHalf">
                        <h1 class="text-big" id="web" style="line-
height:100px;">Personal Expense Tracker</h1>
                        <p class="text-small" style="line-
height:30px;color:black">
                              Expense management refers to the systems 
deployed by a business to process, pay, and audit employee-initiated 
expenses. These costs include, but are not limited to, expenses incurred 
for travel and entertainment. Expense management includes the policies and 
procedures that govern such spending, as well as the technologies and
```

```
services utilized to process and analyze the data associated with it.
            Software to manage the expense claim, authorization, audit and
repayment processes can be obtained from organizations that provide a
licensed software, implementation and support service, or alternatively,
from software as a service (SaaS) providers. SaaS providers offer on-demand
web-based applications managed by a third party to improve the
productivity of expense management.
                       </p>



                 </div>
           </div>
     </section>
     <section class="section">
           <div class="paras">
                 <h1 class="sectionTag text-big"
style="color:#EF32D9;line-height:100px;">Services that we provide:</h1>

            <ul style="line-height:30px;">
                <li><P>Medical Expense</P></li>
                <li><p>Educational Expense</p></li>
                <li><p>Rent Expense</p></li>
                <li><p>Travel Expense</p></li>
                <li><p>Food Expense</p></li>
              </ul>
                 <br>
                  <p style="width:550px;line-height:30px;margin-left:-
30px;">
           Personal Income-Expense Tracker is an excel template display
in to easily manage your finance by recording your monthly incomes and
expenses.Our expense trackers allow you to build a personal budget & see
your cash balance. Getting on top of your finances comes with huge pay offs
& peace of mind.
          </p>

           </div>


           <div class="thumbnail">
                 <img src="{{ url_for('static', filename='css/simple.png')
}}" alt="laptop image">
           </div>
     </section>
     <div class="box-main">
                 <div class="firstHalf">
                       <h1 class="text-big" id="program">
                             "Beware of little expenses. A small leak will
```

```
sink a Great ship"
                        </h1>
                        <p class="text-small">

                        </p>

                  </div>
            </div>



      <footer class="background">
            <p class="text-footer">
                  Copyright ©-All rights are reserved
            </p>


      </footer>
</body>

</html>
```

7.2. FEATURE 2

Back-end development means working on server-side software, which focuses on everything you can't see on a website. Back-end developers ensure the website performs correctly, focusing on databases, back-end logic, application programming interface (APIs), architecture, and servers.

we used the sendgrid software to send the emails to the customers and service providers.

SendGrid:

SendGrid is a cloud-based SMTP provider that allows you to send email without having to maintain email servers. SendGrid manages all of the technical details, from scaling the infrastructure to ISP outreach and reputation monitoring to whitelist services and real time analytics.

sendgrid.py

```
import os

import mail
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import *


def send_email():
    from_email = Email('m.arunkumarmar12@gmail.com')
    to_email = To('bensonruban2001@gmail.com')
    subject = 'Personal expense tracker'
    content = Content("text/plain", "your balance is less than 1000")
    mail = Mail(from_email, to_email, subject, content)

    try:
        sg = SendGridAPIClient('apikey')
        response = sg.send(mail)
        print(response.status_code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e)


send_email()
```

CHAPTER 8
TESTING

8.TESTING

8.1 TEST CASES

TEST CASE: 1

PRODUCT: PERSONAL EXPENSES TRACKER APPLICATON

USECASE: SIGNUP AND LOGIN

TEST CASE ID	**TESTCASE/ ACTION TO BE PERFORMED**	**EXPECTED RESULT**	**ACTUAL RESULT**	**PASS/FAIL**
1	Fill all the appropriate details in the displayed form to enrol as new user and click submit button	Registered successfully	As expected	PASS
2	Enter username and password in order to login	Home page is displayed after successful login	As expected	PASS

TEST CASE: 2

PRODUCT: PERSONAL EXPENSES TRACKER APPLICATION

USECASE: DASHBOARD

TEST CASE ID	**TESTCASE/ ACTION TO BE PERFORMED**	**EXPECTED RESULT**	**ACTUAL RESULT**	**PASS/FAIL**
1	View the profile details	Viewed successfully	As expected	PASS
2	Add balances	Added Successfully	As expected	PASS

TEST CASE: 3

PRODUCT: PERSONAL EXPENSES TRACKER APPLICATION

USECASE: TRACK AND ADD EXPENSES

TEST CASE ID	**TESTCASE/ ACTION TO BE PERFORMED**	**EXPECTED RESULT**	**ACTUAL RESULT**	**PASS/FAIL**
1	Add expenses	Added successfully	As expected	PASS
2	Track expenses	Tracked Successfully	As expected	PASS

8.2 USER ACCEPTANCE TESTING

1. Final Stage, before handling over to the customer which is usually carried out by the customer where the test cases are executed with actual data.
2. The system under consideration is tested for user acceptance and constantly keeping touch with the prospective system user at the time of developing and making changes whenever required.
3. It involves planning and execution of various types of tests in order to demonstrate that the implemented software system satisfies the requirements stated in the requirement document.
4. Two set of acceptance test to be run:
    - i. Those developed by quality assurance group
    - ii. Those developed by customer

CHAPTER 9
RESULTS

9.RESULTS

9.1. PERFORMANCE METRICES

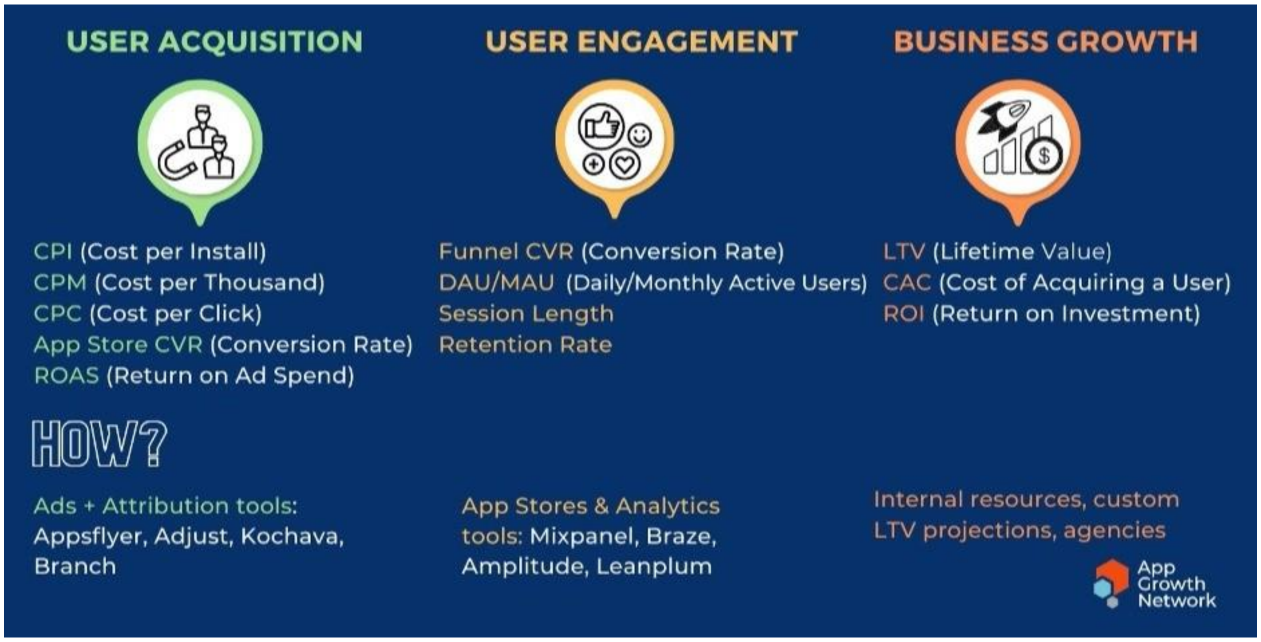

USER ACQUISITION
USER ENGAGEMENT
BUSINESS GROWTH
CPI (Cost per Install)
CPM (Cost per Thousand)
CPC (Cost per Click)
App Store CVR (Conversion Rate)
ROAS (Return on Ad Spend)
Funnel CVR (Conversion Rate)
DAU/MAU (Daily/Monthly Active Users)
Session Length
Retention Rate
LTV (Lifetime Value)
CAC (Cost of Acquiring a User)
ROI (Return on Investment)
HOW?
Ads + Attribution tools: Appsflyer, Adjust, Kochava, Branch
App Stores & Analytics tools: Mixpanel, Braze, Amplitude, Leanplum
Internal resources, custom LTV projections, agencies
App Growth Network

fig 9.1 performance metrices-1

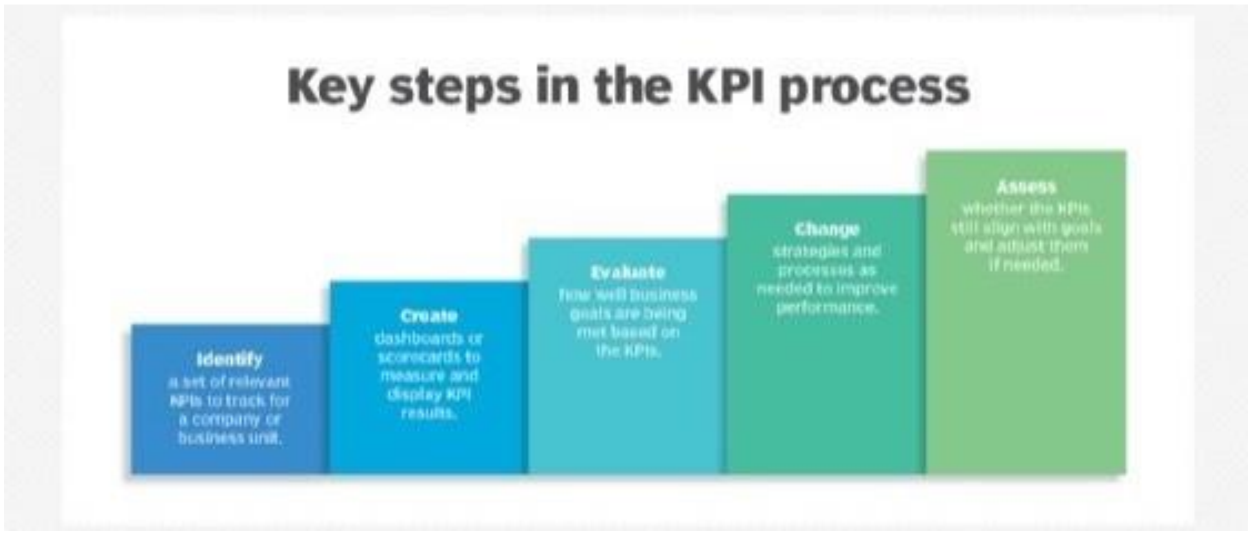

Key steps in the KPI process
Identify a set of relevant KPIs to track for a company or business unit.
Create dashboards or scorecards to measure and display KPI results.
Evaluate how well business goals are being met based on the KPIs.
Change strategies and processes as needed to improve performance.
Assess whether the KPIs still align with goals and adjust them if needed.

fig 9.2 performance metrices-1

CHAPTER 10

ADVANTAGES & DISADVANTAGES

10. ADVANTAGES AND DISADVANTAGES

10.1. ADVANTAGES

When you don’t keep a watch on your spending, you will be short of money, always. This will stress you out. With a daily expense manager, you will be able to allocate money to different priorities and this will also help you cut down on unnecessary spending. As a result, you will be able to save and be able to keep worry at bay.

A daily money tracker helps you budget your money so that you use it wisely. If you find that every month your expenses are more than what you earn, it is time to put your house in order and get a money manager app that keeps track of your money without any problem.

10.2. DISADVANTAGES

It is possible for there to be some budgeting disadvantages when deciding where to allocate your company's or department's money. While budgeting is a helpful tool, it can also come with major challenges. Along with taking a lot of time and resources to create, budgeting may make some departments within a company feel constricted with the funding they get.

When creating a budget, it's important to consider the wants and needs of everyone it may involve. This way you can learn about a team's daily expenses along with the larger costs of operating the company. Another way to limit budgeting disadvantages is having clear policies on spending. Rather than giving employees an incentive to spend their entire budget, try to come up with a system where they spend what they need. This can help your company cut costs while also giving departments the funding they need.

CHAPTER 11
CONCLUSION

11. CONCLUSION

In conclusion, the selection of accurate technique is very important to make sure that the system successfully implemented and achieved the objective. The selected techniqueis artificial neural networks that can be able to predict the financial forecast correctly. Based on the research study, it can be conclude that the cloud computing works is suitable for Expenses Tracker System.

Tracking the daily expenses can not only help in saving money but also help in setting financial goals for the future. If we know where our money is being spent every day, it is easy to set some cutbacks and such to help reduce expenditure. This project is developed to work more efficiently in comparison to other trackers and avoid manual calculation. It is developed to be efficient and look attractive at the same time.

We have developed a mobile application that Keeps track of all of your daily transactions, keeps track of your money lent or borrowed ,suggests you with the most effective investment options, offers your discountsin popular categories , view exchangeand to read latest authenticated financial news. This paper's main aim to eliminate the use of sticky notes, spreadsheets and handlingof large chunks of data is successful, the new experience is hasslefree and very handy. Now, with our application user can manage his expenses more effectively. This application can also help digital marketing agencies in rolling out their advertising campaigns more effectively.

As a part offurther research, we considered adding certain features to create more enhanced experience to the user .We are also going to link this profile with their mobile number, email account, social networks so that the application offers portability, other features to be added are discussed above below within the future enhancement section. The application delivered efficiently in calculating split expenses and recording the expenses together accurately with date and time

CHAPTER 12

FUTURE SCOPE

12. FUTURE SCOPE

All the limitations discussed in Chapter 5 should be addressed in the next version of the application so that it can be more enhanced and user friendly. Provision to add different currencies will be added so that this application is not just limited to USA but also can be used worldwide and the currency converters will be designed and added in order to convert the different currency rates. In order to make it more user friendly and less user intensive, when the user tries to add the same category or vendor to an expense/income record, a duplicate alert will be presented showing the same category/vendor which the user entered previously for some expense/income and then he can tap on it and the entries will be automatically filled for the current record. For example: the user spends US$ 10 at Starbucks (vendor) on drinks (category) on a particular date and the next day he spends some money at the same place on the same category, then when he tries to write that on the expense details view, a duplicate pop up will be presented showing Starbucks as the vendor and drinks as the category. The 54 user taps it and it automatically fills up the detail view making the application less user intensive. 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.

CHAPTER 13
APPENDIX

13. SOURCE CODE

app.py

```
from flask import Flask, render_template, request, session
import ibm_db
import os

import mail
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import *
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=19af6446-6171-4641-8aba-
9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30699;SECUR
ITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=xst23649;PWD=bKig
kwntEUnt56mj",'','')

app = Flask(__name__)
app.secret_key = "arun"
@app.route("/")
def home():
    return render_template("login.html")

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

    email = request.form['email']
    password = request.form['password']
    sql ="SELECT * FROM registration WHERE email=? and password=?";
    stmt =ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,email)
    ibm_db.bind_param(stmt,2,password)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    if account:
      session['response']= account['EMAIL']
      session['amount']=account['INTIAL_AMOUNT']
      session['name'] = account['NAME']
      return render_template('dashboard.html',account =
session['response'])
    else:
        return render_template('login.html',msg="Incorrect Email and
Password")
  return render_template('login.html')
@app.route('/registration')
```

```
def registration():
  return render_template('registration.html')
@app.route('/logging')
def logging():
  return render_template('login.html')
def registration():
  return render_template('registration.html')
@app.route('/register',methods = ['POST', 'GET'])
def register():
  if request.method=='POST':
    name = request.form['name']
    dob = request.form['dob']
    occupation = request.form['occupation']
    gender = request.form['gender']
    email = request.form['email']
    password = request.form['password']
    intial_amount = request.form['intial_amount']
    address = request.form['address']
    phone = request.form['number']
    sql = "SELECT * FROM registration WHERE email=?";
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, email)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    if account:
      return render_template('login.html', msg="You are already a member, 
please login using your details")
    else:
      insert_sql = "INSERT INTO 
registration(name,dob,occupation,gender,email,password,intial_amount,addre
ss,phone_number) VALUES (?,?,?,?,?,?,?,?,?)"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt, 1, name)
      ibm_db.bind_param(prep_stmt, 2, dob)
      ibm_db.bind_param(prep_stmt, 3, occupation)
      ibm_db.bind_param(prep_stmt, 4, gender)
      ibm_db.bind_param(prep_stmt, 5, email)
      ibm_db.bind_param(prep_stmt, 6, password)
      ibm_db.bind_param(prep_stmt, 7, intial_amount)
      ibm_db.bind_param(prep_stmt,8,address)
      ibm_db.bind_param(prep_stmt,9,phone)
      ibm_db.execute(prep_stmt)
      insert_sql = "INSERT INTO expenses VALUES (?,0,0,0,0,0,0,0,0,?)"
      prep_stmt = ibm_db.prepare(conn,insert_sql)
      ibm_db.bind_param(prep_stmt,1,email)
      ibm_db.bind_param(prep_stmt, 2, intial_amount)
      ibm_db.execute(prep_stmt)
      insert_sql = "UPDATE expenses set balance=? WHERE email=?"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt, 1, intial_amount)
      ibm_db.bind_param(prep_stmt, 2, email)
      ibm_db.execute(prep_stmt)
      return render_template('login.html',msg1="registered
successfully",color="green")
@app.route('/profile')
def profile():
  sql = "SELECT * FROM registration WHERE email=?";
  stmt = ibm_db.prepare(conn, sql)
  ibm_db.bind_param(stmt, 1, session['response'])
  ibm_db.execute(stmt)
  account = ibm_db.fetch_assoc(stmt)
  name = account['NAME']
  dob = account['DOB']
  occupation =account['OCCUPATION']
  gender = account['GENDER']
  email = account['EMAIL']
  password = account['PASSWORD']
  intial =account['INTIAL_AMOUNT']
  address = account['ADDRESS']
  phone =account['PHONE_NUMBER']
  return
render_template('profile.html',name=name,dob=dob,occupation=occupation,gend
er=gender,email=email,password =
password,intial=intial,address=address,phone=phone)
@app.route('/expenses')
def expenses():
  sql = "SELECT * FROM expenses WHERE email=?";
  stmt = ibm_db.prepare(conn, sql)
  ibm_db.bind_param(stmt, 1, session['response'])
  ibm_db.execute(stmt)
  account = ibm_db.fetch_assoc(stmt)
  medical = account['MEDICAL']
  education = account['EDUCATION']
  rent = account['RENT']
  food = account['FOOD']
  travel = account['TRAVEL']
  others = account['OTHERS']
  spend =account['TOTAL']
  balance = account['BALANCE']
  credit = account['CREDIT']
  return
render_template('expenses.html',medical=medical,education=education,rent=r
ent,travel=travel,others=others,food=food,spend=spend,balance=balance,cred
it = credit)
@app.route('/expenditure',methods = ['POST','GET'])
def expenditure():
  def send_email(email,amount):
    from_email = Email('m.arunkumarmar12@gmail.com')
    to_email = To(email)
    subject = 'Personal expense tracker'
    content = Content("text/plain", f"your balance is
{balance}".format(balance=amount))
    mail = Mail(from_email, to_email, subject, content)

    try:
      sg = SendGridAPIClient('SG.Obu-XaKdSsmAfnQh6c772Q.2XPa1lUppzUqF9gd-
_k8f0--aSfl8KswNKPy9C4GQxA')
      response = sg.send(mail)
      print(response.status_code)
      print(response.body)
      print(response.headers)
    except Exception as e:
      print(e)
  def check_balance(amount):
      send_email(session['response'],amount)
  if request.method =='POST':
    amount = request.form['amount']
    type = request.form['type']
    sql = "SELECT * FROM expenses WHERE email=?";
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, session['response'])
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    total =
int(account['MEDICAL'])+int(account['EDUCATION'])+int(account['RENT'])+int
(account['TRAVEL'])+int(account['OTHERS'])+int(account['FOOD'])
    if type=="medical":
      balance = int(account['BALANCE']) - int(amount)
      amount =int (account['MEDICAL']) +int(amount)
      total = amount + int(account['EDUCATION']) +int(account['RENT'])
+int(account['TRAVEL']) +int(account['OTHERS']) + \
              int(account['FOOD'])
      insert_sql = "UPDATE expenses SET medical=?,total=?,balance =? WHERE
email=?"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt,1,amount)
      ibm_db.bind_param(prep_stmt,2,total)
      ibm_db.bind_param(prep_stmt,3,balance)
      ibm_db.bind_param(prep_stmt, 4, session['response'])
      ibm_db.execute(prep_stmt)
      sql = "SELECT * FROM expenses WHERE email=?";
      stmt = ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt, 1, session['response'])
      ibm_db.execute(stmt)
      account = ibm_db.fetch_assoc(stmt)
      medical = account['MEDICAL']
      education = account['EDUCATION']
      rent = account['RENT']
      food = account['FOOD']
      travel = account['TRAVEL']
      others = account['OTHERS']
      spend = account['TOTAL']
      balance = account['BALANCE']
      credit = account['CREDIT']
      check_balance(int(balance))
      return render_template('expenses.html', medical=medical,
education=education, rent=rent, travel=travel,
                             others=others, food=food, spend=spend,
balance=balance, credit=credit,msg="Updated successfully!")
    if type=="education":
      balance = int(account['BALANCE']) - int(amount)
      amount =int (account['EDUCATION']) +int(amount)
      balance = int(account['BALANCE']) - amount
      total = amount +int(account['MEDICAL']) + int(account['RENT'])
+int(account['TRAVEL']) +int(account['OTHERS']) + \
              int(account['FOOD'])
      insert_sql = "UPDATE expenses SET education=?,total=?,balance =?
WHERE email=?"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt,1,amount)
      ibm_db.bind_param(prep_stmt,2,total)
      ibm_db.bind_param(prep_stmt, 3, balance)
      ibm_db.bind_param(prep_stmt, 4, session['response'])
      ibm_db.execute(prep_stmt)
      sql = "SELECT * FROM expenses WHERE email=?";
      stmt = ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt, 1, session['response'])
      ibm_db.execute(stmt)
      account = ibm_db.fetch_assoc(stmt)
      medical = account['MEDICAL']
      education = account['EDUCATION']
      rent = account['RENT']
      food = account['FOOD']
      travel = account['TRAVEL']
      others = account['OTHERS']
      spend = account['TOTAL']
      balance = account['BALANCE']
      credit = account['CREDIT']
      check_balance(int(balance))
      return render_template('expenses.html', medical=medical,
education=education, rent=rent, travel=travel,
                             others=others, food=food, spend=spend,
balance=balance, credit=credit,msg="Updated successfully!")
    if type=="rent":
      balance = int(account['BALANCE']) - int(amount)
      amount =int (account['RENT'])  +int(amount)
      total = amount + int(account['EDUCATION']) + int(account['MEDICAL'])
+ int(account['TRAVEL']) + int(account['OTHERS'] )+ \
              int(account['FOOD'])
      insert_sql = "UPDATE expenses SET rent=?,total=?,balance =? WHERE
email=?"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt,1,amount)
      ibm_db.bind_param(prep_stmt,2,total)
      ibm_db.bind_param(prep_stmt, 3, balance)
      ibm_db.bind_param(prep_stmt, 4, session['response'])
      ibm_db.execute(prep_stmt)
      sql = "SELECT * FROM expenses WHERE email=?";
      stmt = ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt, 1, session['response'])
      ibm_db.execute(stmt)
      account = ibm_db.fetch_assoc(stmt)
      medical = account['MEDICAL']
      education = account['EDUCATION']
      rent = account['RENT']
      food = account['FOOD']
      travel = account['TRAVEL']
      others = account['OTHERS']
      spend = account['TOTAL']
      balance = account['BALANCE']
      credit = account['CREDIT']
      check_balance(int(balance))
      return render_template('expenses.html', medical=medical,
education=education, rent=rent, travel=travel,
                             others=others, food=food, spend=spend,
balance=balance, credit=credit,msg="Updated successfully!")
    if type=="travel":
      balance = int(account['BALANCE']) -int(amount)
      amount =int (account['TRAVEL'])+int(amount)
      total = amount + int(account['EDUCATION']) + int(account['RENT'])
```

```
+int(account['MEDICAL']) +int(account['OTHERS']) + \
              int(account['FOOD'])
      insert_sql = "UPDATE expenses SET travel=?,total=?,balance =? WHERE
email=?"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt,1,amount)
      ibm_db.bind_param(prep_stmt,2,total)
      ibm_db.bind_param(prep_stmt, 3, balance)
      ibm_db.bind_param(prep_stmt, 4, session['response'])
      ibm_db.execute(prep_stmt)
      sql = "SELECT * FROM expenses WHERE email=?";
      stmt = ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt, 1, session['response'])
      ibm_db.execute(stmt)
      account = ibm_db.fetch_assoc(stmt)
      medical = account['MEDICAL']
      education = account['EDUCATION']
      rent = account['RENT']
      food = account['FOOD']
      travel = account['TRAVEL']
      others = account['OTHERS']
      spend = account['TOTAL']
      balance = account['BALANCE']
      credit = account['CREDIT']
      check_balance(int(balance))
      return render_template('expenses.html', medical=medical,
education=education, rent=rent, travel=travel,
                             others=others, food=food, spend=spend,
balance=balance, credit=credit,msg="Updated successfully!")
    if type=="food":
      balance = int(account['BALANCE']) -int(amount)
      amount =int (account['FOOD'])+int(amount)
      total = amount + int(account['EDUCATION']) + int(account['RENT'])
+int(account['TRAVEL']) + int(account['OTHERS']) + \
              int(account['MEDICAL'])
      insert_sql = "UPDATE expenses SET food=?,total=?,balance =? WHERE
email=?"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt,1,amount)
      ibm_db.bind_param(prep_stmt,2,total)
      ibm_db.bind_param(prep_stmt, 3, balance)
      ibm_db.bind_param(prep_stmt, 4, session['response'])
      ibm_db.execute(prep_stmt)
      sql = "SELECT * FROM expenses WHERE email=?";
      stmt = ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt, 1, session['response'])
      ibm_db.execute(stmt)
      account = ibm_db.fetch_assoc(stmt)
      medical = account['MEDICAL']
      education = account['EDUCATION']
      rent = account['RENT']
      food = account['FOOD']
      travel = account['TRAVEL']
      others = account['OTHERS']
      spend = account['TOTAL']
      balance = account['BALANCE']
      credit = account['CREDIT']
      check_balance(int(balance))
      return render_template('expenses.html', medical=medical,
education=education, rent=rent, travel=travel,
                             others=others, food=food, spend=spend,
balance=balance, credit=credit,msg="Updated successfully!")
    if type=="others":
      balance = int(account['BALANCE']) -int(amount)
      amount =int (account['OTHERS']) +int(amount)
      total = amount + int(account['EDUCATION']) + int(account['RENT'])
+int( account['TRAVEL']) +int(account['MEDICAL']) + \
              int(account['FOOD'])
      insert_sql = "UPDATE expenses SET others=?,total=?,balance =? WHERE
email=?"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt,1,amount)
      ibm_db.bind_param(prep_stmt,2,total)
      ibm_db.bind_param(prep_stmt, 3, balance)
      ibm_db.bind_param(prep_stmt, 4, session['response'])
      ibm_db.execute(prep_stmt)
      sql = "SELECT * FROM expenses WHERE email=?";
      stmt = ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt, 1, session['response'])
      ibm_db.execute(stmt)
      account = ibm_db.fetch_assoc(stmt)
      medical = account['MEDICAL']
      education = account['EDUCATION']
      rent = account['RENT']
      food = account['FOOD']
      travel = account['TRAVEL']
      others = account['OTHERS']
      spend = account['TOTAL']
      balance = account['BALANCE']
      credit = account['CREDIT']
      check_balance(int(balance))
      return render_template('expenses.html', medical=medical,
education=education, rent=rent, travel=travel,
                             others=others, food=food, spend=spend,
balance=balance, credit=credit,msg="Updated successfully!")
  return render_template('expenses.html')
@app.route('/display')
def display():
  sql ="SELECT * FROM expenses WHERE email=?"
  prep_stmt = ibm_db.prepare(conn,sql)
  ibm_db.bind_param(prep_stmt,1,session['response'])
  ibm_db.execute(prep_stmt)
  account = ibm_db.fetch_assoc(prep_stmt)
  print(account)
  medical = int(int(account['MEDICAL']) / int(account['CREDIT']) * 100)
  education = int(int(account['EDUCATION']) / int(account['CREDIT']) *
100)
  rent = int(int(account['RENT']) / int(account['CREDIT']) * 100)
  travel = int(int(account['TRAVEL']) / int(account['CREDIT']) * 100)
  food = int(int(account['FOOD']) / int(account['CREDIT']) * 100)
  others = int(int(account['OTHERS']) / int(account['CREDIT']) * 100)
  spend = int(int(account['TOTAL']) / int(account['CREDIT']) * 100)
  balance = int(int(account['BALANCE']) / int(account['CREDIT']) * 100)
  print(medical,education,rent,travel,food,others,spend,balance)
  def color(var):
    if(var>70):
      return "red"
    elif(var>30):
      return "yellow"
    else:
      return "green"
  def fake(var):
    if(var>70):
      return "green"
    elif(var>30):
      return "yellow"
    else:
      return  "red"
  medical_color = color(medical)
  education_color=color(education)
  rent_color=color(rent)
  travel_color = color(travel)
  food_color =color(food)
  others_color=color(others)
  spend_color=color(spend)
  balance_color = fake(balance)
  return render_template("display.html",medical_color =medical_color
,education_color=education_color,rent_color=rent_color,travel_color
```

```
=travel_color ,food_color
=food_color,others_color=others_color,spend_color=spend_color,balance_col
or =balance_color,account
=account,medical=medical,education=education,rent=rent,travel=travel,other
s=others,food=food,spend=spend,balance=balance)
@app.route('/addBalance')
def addBalance():
  sql = "SELECT * FROM expenses WHERE email=?"
  prep_stmt = ibm_db.prepare(conn, sql)
  ibm_db.bind_param(prep_stmt, 1, session['response'])
  ibm_db.execute(prep_stmt)
  account = ibm_db.fetch_assoc(prep_stmt)
  return
render_template('addBalance.html',balance=account['BALANCE'],total=account
['TOTAL'],credit=account['CREDIT'])
@app.route('/adder',methods = ['POST','GET'])
def adder():
  if request.method=="POST":
    balance1 = request.form['money']
    sql = "SELECT * FROM expenses WHERE email =?";
    prep_stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(prep_stmt,1,session['response'])
    ibm_db.execute(prep_stmt)
    account = ibm_db.fetch_assoc(prep_stmt)
    balance = int(balance1) +int(account['BALANCE'])
    credit=int(balance1)+int(account['CREDIT'])
    insert_sql = "UPDATE expenses SET balance =?,credit=? WHERE email=?"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, balance)
    ibm_db.bind_param(prep_stmt, 2, credit)
    ibm_db.bind_param(prep_stmt,3,session['response'])
    ibm_db.execute(prep_stmt)
    return render_template("thanks.html",msg="Balance added successfully")
@app.route('/dashboard')
def dashboard():
  return render_template('dashboard.html')
if___name__=="__main__":
    app.run(debug=True)
```

GITHUB & PROJECT DEMO LINK

https://github.com/IBM-EPBL/IBM-Project-3686-1658590110.git