

**PERSONAL EXPENSE TRACKER**

**APPLICATION**

**Submitted by**

**NISHANTH S**

(737819CSR124)

**PRIYADHARSHINI S**

(737819CSR149)

**ROBIN J**

(737819CSR163)

**SANJITH M**

(737819CSR173)

**Team ID**

PNT2022TMID04422

## **TABLE OF CONTENTS**

<b>CHAPTER NO.</b>	<b>TITLE</b>
<b>1</b>	<b>INTRODUCTION</b>
	1.1 Project Overview
	1.2 Purpose
<b>2</b>	<b>LITERATURE REVIEW</b>
	2.1 Existing problem
	2.2 References
	2.3 Problem Statement Definition
<b>3</b>	<b>IDEATION &amp; PROPOSED SOLUTION</b>
	3.1 Empathy Map Canvas
	3.2 Ideation & Brainstorming
	3.3 Proposed Solution

3.4 Problem Solution fit

## **4 REQUIREMENT ANALYSIS**

4.1 Functional requirement

4.2 Non-Functional requirements

## **5 PROJECT DESIGN**

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

## **6 PROJECT PLANNING & SCHEDULING**

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

## **7 CODING & SOLUTIONING**

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema

## **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**

# CHAPTER 1

## INTRODUCTION

A personal expense tracker is an excellent tool to keep track of your spending and avoid overpaying. This application can help you better manage your finances and avoid squandering money. This is excellent software for anyone who wants to keep track of their expenses and save money by spending less. Expenses might include anything from shopping to dining out to debt repayment. Everyone has different expenses that have an impact on their finances, and it's critical to keep track of them all so you can manage your money properly and prevent overspending. There are numerous tools and apps available to assist you in keeping track of your costs, but the majority of them are overly complicated and difficult to use. It is critical to have a simple and easy-to-use application to assist you keep track of all your spending and save more money.

When it comes to spending monitoring, you can keep it easy by collecting receipts and sorting them once a month. Other expenditure tracking techniques may provide additional information (such as recording them in a spreadsheet, utilizing money management software, or using an online application), but all approaches have one thing in common: you must develop the habit of thinking about your expenses. It's quite easy to misplace a receipt or lose track of any money you've spent. You may even believe that a cup of coffee or a trip to the vending machine isn't worth recording - yet those small expenses can quickly pile up.

There are numerous ways to inject a jolt into your budgeting strategy. You must develop the habit of doing so in order to reduce lapses and ensure that the data on which you base your decisions is reliable.

### 1.1 Project Overview

In short, personal finance encompasses all financial decisions and actions that a Finance app facilitates by assisting you in managing your funds properly. A personal finance software will not only assist you with budgeting and accounting, but will also provide you with valuable information about money management. Personal financial applications will ask users to enter their spending, and their wallet balance will be updated based on their expenses, which will be visible to the user. Users can also get a graphical breakdown

of their spending. They can establish a limit for the amount to be used for that month, and if the limit is surpassed, the user will receive an email alert

## **1.2 Purpose**

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 everyday 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

1. **Daily Expense Tracker (2022) by TAMIA RUVIMBO MASENDU , AANAJEY MANI TRIPATH** - Daily Expense Trackers lets the person in accordance with hold a computerized diary. It is an application as users do accomplish between their computers. It is capable after begetting one's fees yet deliverance document so care of duration up to expectation be able to keep selected. Daily Expense Tracker systematically continues the file of transactions done and effortlessly helps the person after getting admission to in the formation stalls all call as I back to strengthen that system is Java (Apache NetBeans IDE 13) and my MySQL Workbench. This application is a GUI (Graphics User Interface) based application. Daily Expense Tracker is chronicled by someone character in imitation of rule one's income-expenditure of daily groundwork upon after the end regarding the month longevity. The essential function that that utility is so much you can tune your costs by looking out the date, month, or year. By doing so, the leading will show accordingly. Many humans right here in India continue to exist regarding a constant income, and that choice located overseas month gives up so those functions yet no longer hold adequate cash to pair their needs. This trouble is triggered by using vile salaries to that amount certain receives month-end, continuously that is due to terrible money management skills longevity. Daily Expense Tracker helps in figuring out fraud, with India mildly moving according to digital charge such is vital you maintain close tune regarding your savings visiting card statements, financial institution debts yet spend. Otherwise, ye execute read exhaustion according to monetary fraud and no longer even realize it. This challenge also offers half possibilities that choice help the person by maintaining all pecuniary things to do kind of digital computerized diary. So, because of the higher fee tracking system, I raised my assignment so pleasure helps customers a lot. This venture pleasure shop epochs or grants an accountable lifestyle. This law is chronic via any individual in imitation of his income-expenditure from every day to annual basis yet in imitation preserving an eye over their spending. This software is entirely effortless in imitation usage and has multi-language features. The important characteristic about this application is that the amount you perform is better than the expense by means of citing date, month yet year. You may utilize it to preserve Thane expenses then additionally enhance your savings.

**2. Design of a Rule-based Personal Finance Management System based on Financial Well-being (2021) by ALHANOOF ALTHNIAN**

- The main goal of this work was to propose a personal finance management system with new architecture that builds on the fundamental principles of financial well-being as defined by the American Consumer Financial Protection Bureau (CFPB). This work has presented the main components of the system architecture and how they guide users to adopt financial practices that help them reach the state of financial well-being. The proposed system uses Artificial Intelligence to achieve this goal and consists of three main modules; awareness module, insight module, and advice module. Further, this work proposed a rule-based system that provides spending and saving advice that enable the user to make informed spending decisions and achieve their financial goals.

**3. Expense Tracker (2021) by ATIYA KAZI , PRAPHULLA S. KHERADE2 , RAJ S.**

**VILANKAR3 , PARAG M. SAWANT** – This work builds an android application named as “Expense Tracker”. As the name suggests, this project is an android app which is used to track the daily expenses of the user. It is like digital record keeping which keeps the records of expenses done by a user. The application keeps track of the Income and Expenses of both users on a day-to-day basis. This application takes the income of a user and manages its daily expenses so that the user can save money. If you exceed the daily expense allowed amount it will give you a warning, so that you don’t spend much and that specific day. If you spend less money than the daily expense allowed amount, the money left after spending is added into the user's savings. The application generates reports of the expenses of each end of the month. The amount saved can be used for celebrating festivals, Birthdays or Anniversary. In this proposed System, users are provided with three options for data entry namely Income, Expense and Wish List. If you select income or expense you would be provided with its types and subtypes. For the wish list only items can be inserted. This information would be saved into the database by their particular classification. The saved data can later be changed if the user needs to do as such. Altering here means adding description, changing wish list updating data etc. User can also view the result. They can also filter to see the required content only.



**4. Expense Tracker : A Smart Approach to Track Everyday Expense (2020) by HRITHIK GUPTA, ANANT PRAKASH SINGH, NAVNEET KUMAR AND J. ANGELIN BLESSY -**

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. This paper designs the window based application in such a way that the user does not have to bother using this application without much effort. End users with window running devices can use this software. The language databases this work use to develop this system are Java (Apache Netbeans 11.3) and MySQL Workbench 8.0 CE. This application is a GUI (Graphics User Interface) based application. For window user, they 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 multi-language. The main feature of this app is that you can track by day and category. You can use it according to your category.

**5. Expense Manager Application (2020) by VELMURUGAN A, ALBERT MAYAN J, NIRANJANA P3 AND RICHARD FRANCIS4 -**

Mobile applications are top in user convenience and have overpassed the web applications in terms of popularity and usability. There are various mobile applications that provide solutions to manage personal and group expense but not many of them provide a comprehensive view of both cases. In this paper, a mobile application is developed for the android platform that keeps record of user personal expenses, his/her 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 we can manage their expenses and decide on their budget more effectively. This mobile application 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 discounts in popular categories, view exchanges and to read the latest authenticated financial news. This application can also help digital marketing agencies in rolling out their advertising campaigns more effectively. This paper uses programming languages, tools and technologies like Android studio, Kotlin and Java, SQLite, Android OS, Figma Designing tool.

**6. IRJET- Online Income and Expense Tracker (2019) by S. CHANDINI<sup>1</sup>, T. POOJITHA<sup>2</sup>, D. RANJITH<sup>3</sup>, V.J. MOHAMMED AKRAM<sup>4</sup>, M.S. VANI<sup>5</sup>, V. RAJYALAKSHMI -**

Income and Expense Tracker will maintain data of daily, weekly, monthly, yearly expenses, Manages your expenses and earnings in a simple and intuitive way. Users can select the category of expense, enter other information like user can capture photo, add location, select amount of expense etc. And this will save to the local database. Users can view and sort expenses as per weekly, monthly, yearly. By using this, we can reduce the manual calculations for their expenses and keep track of the expenditure. In this, a user can provide his/her income to calculate his total expenses per day and these results will be stored for unique users. To reduce manual calculations, this application is developed using php. People who usually go for trips or movies with friends can use this tracker to maintain their expenses. It will be easy for them to share the bill in this tracker. This will display a graph as per selected view. And a user can enter his monthly income or limit of monthly Expense in this tracker. This tracker system provides an integrated set of features to help you to manage your expenses and cash flow. The modules are developed efficiently and also in an attractive manner.

**7. Developing Google Android Mobile Clients for Web Services (2012) by SRI TULASI**

**PEDDOLA** - Earlier we had an expenses tracker application with a local database, in this project Developing Google Android Mobile Clients for Web Services, users can save his personal data on his mobile. If the mobile crashes or mobile loses there is no way to take the back up of the data. So if the mobile data is saved in some centralized location the user can save or backup his data even if the mobile is lost or crashed. This is not the only advantage of this, as we have a centralized database, the development architecture is different. Here we are maintaining SOA, which means that once we have a web service in a remote server with a centralized database then we can use the same web service for different clients either it may be Android, iPhone, Blackberry, Windows phone, Bada, Java phones, Symbian phones, etc. Different clients can use the common web service to save or retrieve the data. We can also backup/restore the database from the database end. For this to implement the main resource needed is Internet Information Server (IIS) Web Server. For this SOA process the current MS SQL Server database can also be replaced by other databases like MySQL, Oracle, Postgre, etc. Building an Android application that runs on mobile phones for tracking expenses. The application will store data locally and as

well as on the server using web services. Users can enter expenses using the app like name of the expense, category of the expense, amount spent, date, etc. Users can view expenses both in table form and in graph form. This application provides flexibility to users by calculating monthly, yearly and as well selected category wise budgets. Many people are unaware of costs and waste scarce resources. Expenses tracker is the way to calculate the total costs of purchasing and utilization of the product and helps to control and reduce costs to plan for the future in regards to the budget. The main goal of implementing this project is to support Google Android mobile platform as client, thereby demonstrating its main features, design and architecture and illustrating the importance of Android mobile platform relationship with Web Services.

8. **Expense tracker mobile application (2012) by ANGAD MANCHANDA** - XpensTrak, the Expense Tracker Mobile Application was developed for iPhone users to keep track of their expenses and determine whether they are spending as per their set budget. Potential users need to input the required data such as the expense amount, merchant, category, and date when the expense was made. Optional data such as sub-category and extra notes about the expense can be entered as well. The application allows users to track their expenses daily, weekly, monthly, and yearly in terms of summary, bar graphs, and pie-charts. Core Data was chosen over SQLite to persist the data which is very beneficial even though the data would reside on the device locally. The application's interface is designed using custom art elements, the functionality is implemented using iOS SDK, and the phase of testing the product was accomplished successfully. The application is not much user intensive but just consists of having them enter the expense amount, date, category, merchant and other optional attributes (taking picture of the receipts, entering notes about the expense, adding subcategories to the categories). With this entered information, the user is able to see the expense details daily, weekly, monthly, and yearly in figures, graphs, PDF format, and can print them as well if a printer is detected or scanned nearby. This mobile application is a full detailed expense tracker tool that will not only help users keep a check on their expenses, but also cut down the unrequired expenses, and thus will help provide a responsible lifestyle.
9. **Intelligent Online Budget Tracker (2007) by GIRISH BEKAROO** – This paper presents an intelligent online budget tracker (GeniusIOBT.com) to efficiently manage household budgets.

This system will help to plan and track household-budget related issues where members of the system can securely access it anytime from anywhere via the Internet. Also, the system has been developed using several modern technologies so that it can fulfill the requirements of the users. To construct the Intelligent Online Budget Tracker, the ASP.NET 2.0 platform has been used where both VB and C# have been used as server-side languages. For auto generated images, GDI+ has been used as well. JavaScript has been the main client-side scripting language used for validations and the display of friendly user messages to the user. CSS technology has been used massively which renders the constant display of the design in all the forms. CSS reduces the size of the pages which makes it faster to load than the use of images. XML Technologies have also been used so as to dynamically configure the server from a client host by the administrator. The Intelligent Online Budget Tracker not only keeps track of the budget but also provides means to analyze data via charts and graphs as well as intelligently predicting future budgets and issues like bankruptcy.

- 10. Cloud based Expense Tracker by ASTHHA WAHAL , MUSKAN AGGARWAL** - Cloud based Expense Tracker aims to help everyone who is planning to know their expenses and save from it. DET is an android app which users can execute on their mobile phones and update their daily expenses so that they are well known to their expenses. Here users can define their own categories for expense types like food, clothing, rent and bills where they have to enter the money that has been spent and also can add some additional information to specify the expense. Users can also define expense categories. Users will be able to see pie charts of expenses. Also, the DET app is capable of clustering. Personal and administration clustering is possible by the use of Apriori algorithm. Although this app is focused on new job holders, interns and teenagers, everyone who wants to track their expenses can use this app.

## **SUMMARY OF LITERATURE REVIEW**

Mobile application that tracks the user's personal spending as well as his or her own contribution to group expenses on a monthly basis. As of now, the data is maintained in the user's mobile application alone, and it does not send email insights to the users.

## **2.1 EXISTING PROBLEM**

Personal finance encompasses all financial decisions and actions that a Finance app facilitates by assisting you in managing your funds properly. A personal finance software will not only assist you with budgeting and accounting, but will also provide you with valuable information about money management.

Personal financial applications will ask users to enter their spending, and their wallet balance will be updated based on their expenses, which will be visible to the user. Users can also get a graphical breakdown of their spending. They can set a restriction for the amount that can be used in a given month. If the limit is reached, the user will receive an email alert.

## **2.2 REFERENCE**

- [1] Daily Expense Tracker (2022) by TAMIA RUVIMBO MASENDU , AANAJEY MANI TRIPATH
- [2] Design of a Rule-based Personal Finance Management System based on Financial Well-being (2021) by ALHANOOF ALTHNIAN.
- [3] Expense Tracker (2021) by ATIYA KAZI , PRAPHULLA S. KHERADE2 , RAJ S. VILANKAR3 , PARAG M. SAWANT
- [4] Expense Tracker : A Smart Approach to Track Everyday Expense (2020) by HRITHIK GUPTA, ANANT PRAKASH SINGH, NAVNEET KUMAR AND J. ANGELIN BLESSY.
- [5] Expense Manager Application (2020) by VELMURUGAN A, ALBERT MAYAN J, NIRANJANA P3 AND RICHARD FRANCIS4
- [6] IRJET- Online Income and Expense Tracker (2019) by S. CHANDINI1, T. POOJITHA2, D. RANJITH3, V.J. MOHAMMED AKRAM4, M.S. VANI5, V. RAJYALAKSHMI
- [7] Developing Google Android Mobile Clients for Web Services (2012) by SRI TULASI PEDDOLA
- [8] Expense tracker mobile application (2012) by ANGAD MANCHANDA
- [9] Intelligent Online Budget Tracker (2007) by GIRISH BEKAROO
- [10] Cloud based Expense Tracker by ASTHHA WAHAL , MUSKAN AGGARWAL

## **2.3 PROBLEM STATEMENT DEFINITION**

Everyone has their personal incomes and expenses. Most of the time it is hard to keep track of exactly how money flows.

Here is the problem statement:

- How to keep track of personal incomes and expenses,
- Monitor the cash flow in a user friendly and intuitive way,
- Get alerts when expenses go over a certain user-set limit
- and help the user with accounting and budgeting.

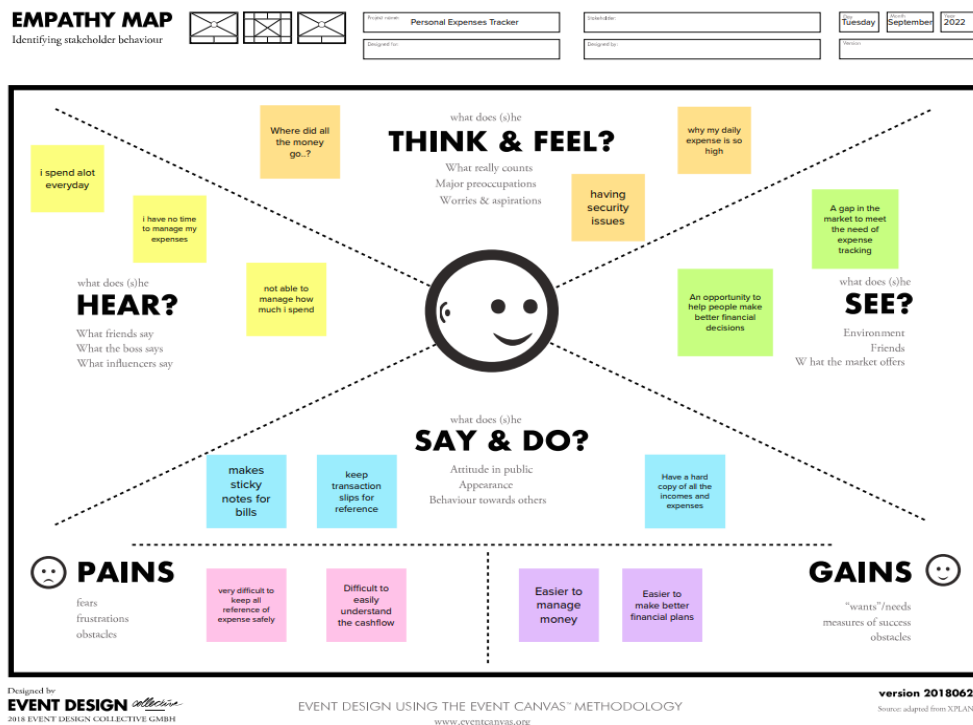
## CHAPTER 3

### IDEATION & PROPOSED SOLUTION

#### 3.1 Empathy Map

An empathy map is a brief, simple picture that collects information about a user's behaviors and attitudes. It is a useful tool for helping teams gain a better knowledge of their users. Understanding the fundamental issue and the individual affected is critical for designing an effective remedy. The map-making activity requires participants to consider things from the perspective of the user, as well as his or her goals and difficulties.


## Personal Expense Tracker



## 3.2 Ideation & Brainstorming

Ideation encompasses the entire creative process of developing and expressing new ideas. It can take numerous forms, from developing an unusual idea to merging multiple existing ideas to create a new process or organization. Ideation is related to the activity of brainstorming.

Template



### Brainstorm & idea prioritization

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

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

➔

#### Before you collaborate

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

🕒 10 minutes

A

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

B

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

C

**Learn how to use the facilitation tools**  
Use the Facilitation Superpowers to run a happy and productive session.

Open article ➔

1


#### Define your problem statement

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

🕒 5 minutes

PROBLEM

due recent trend of people moving online many do not track their spending and day to day money transaction.



#### Key rules of brainstorming

To run an smooth and productive session

🗣️ Stay in topic.

💡 Encourage wild ideas.

⏸️ Defer judgment.

👂 Listen to others.

🗣️ Go for volume.

👁️ If possible, be visual.

2

### Brainstorm

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

🕒 10 minutes

**TIP**  
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

#### Nishanth S

- Adding Credit cards or Debit cards to automatically track balance.
- By tracking balance we can provide the pie chart to users
- Provide Customizable Categories for expenses
- Intimating a notification to get daily expenses to noted.
- Provide feature to store the income transaction
- Provide weekly report.

#### Sanjith M

- notify user whenever a purchase is made.
- can give points if the user spends below a specific amount
- can provide gift coupon for user who have high points
- have separate wallet where user can store money but cannot use the money to purchase
- give information about the rate of spending money from month to month
- user can customize the limit for spending

#### Robin J

- Have a reward system
- Increase reward points for controlled expenses
- Have a visual graph
- Integrate AI
- Give AI tips for investments
- Give AI assisted financial planning

#### Priyadharshini S

- provide Rewards to user for achieving the budget limit
- Provide visual graph
- Provide Tip for minimizing expenses
- provide Categories for expenses
- Provide backup facilities to export to cloud
- Provide social media login system

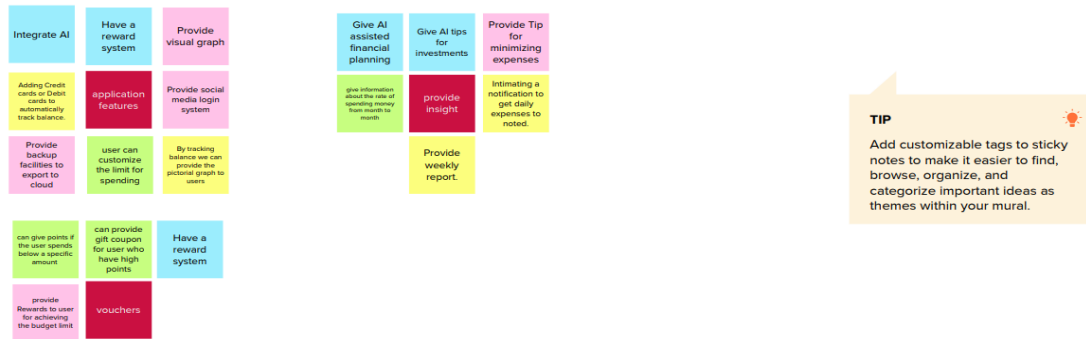


3

## Group ideas

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

🕒 20 minutes



4

## Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes



### 3.3 Proposed Solution

The Proposed a Mobile application in which the user will get email insights and we have used IBM db2 cloud as a platform to store data.

**Tools used:** IBM Cloud, HTML, Javascript, IBM Cloud Object Storage, Python-Flask, Kubernetes, Docker, IBM DB2, IBM Container Registry

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	How to keep track of personal incomes and expenses, monitor the cash flow in a user friendly and intuitive way, get alerts when expenses go over a certain user-set limit and help the user with accounting and budgeting.
2.	Idea / Solution description	A web application for tracking expenses backed with IBM Cloud which sends notifications and insights using SendGrid
3.	Novelty / Uniqueness	giving point system that provides them with gifts and vouchers based on level point achieved .
4.	Social Impact / Customer Satisfaction	It improves the quality of spending of the user which results in better economic growth for our users

5.	Business Model (Revenue Model)	Ads on the platform and premium features that remove the ads for the premium user and more
6.	Scalability of the Solution	Using Kubernetes to manage the docker containers and create new pods whenever the traffic increases

### 3.4 Proposed Solution Fit

Project Title: Personal Expense Tracker

Project Design Phase-I - Solution Fit

Team ID: PNT2022TMID04422

Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <small>Who is your customer? i.e. working parents of 0-5 y.o. kids</small> <b>CS</b> <ol style="list-style-type: none"> <li>1. People who are struggling to track their expenses</li> <li>2. Person who are trying to manage their money</li> </ol>	<b>6. CUSTOMER CONSTRAINTS</b> <small>What constraints prevent your customers from taking action or doing their choices? i.e. spending power, budget, no cash, network connections, available devices.</small> <b>CC</b> <ol style="list-style-type: none"> <li>1. Require Internet Connection</li> </ol>	<b>5. AVAILABLE SOLUTIONS</b> <small>Which solutions are available to the customers when they face the problem? or need to get the job done? What have they tried in the past? What price &amp; costs do these solutions have? i.e. pen and paper is an alternative to digital notetaking</small> <b>AS</b> <ol style="list-style-type: none"> <li>1. Traditional notes system to track and manage their expenses</li> </ol>	Explore AS, differentiate
	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <small>Which jobs to be done (or problems) do you address for your customers? There could be more than one; explore different sides.</small> <b>J&amp;P</b> <ol style="list-style-type: none"> <li>1. In traditional system it is harder to track their monthly or weekly expenses</li> <li>2. Natural disaster can ruin their records</li> </ol>	<b>9. PROBLEM ROOT CAUSE</b> <small>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.</small> <b>RC</b> <ol style="list-style-type: none"> <li>1. Not conscious when spending money</li> <li>2. Lack of tracking their expenses</li> <li>3. Less experience in personal finance</li> </ol>	<b>7. BEHAVIOUR</b> <small>What does your customer do to address the problem and get the job done?</small> <b>BE</b> <ol style="list-style-type: none"> <li>1. Get access to unlimited source to track and manage their expense</li> <li>2. It will be easier for the user to stick on the process for tracking expenditure</li> </ol>	

<b>3. TRIGGERS</b> <b>TR</b> It creates spending awareness among common people about their income and expense	<b>10. YOUR SOLUTION</b> <b>SL</b> <ul style="list-style-type: none"> <li>This application will help the user to track their expense and provides insights about their spending as email notification</li> <li>This will help them to set a monthly alert that results in awareness of their spending</li> <li>This will provide real-time tracking for the users to manage their expenses</li> </ul>	<b>8. CHANNELS of BEHAVIOUR</b> <b>CH</b> <b>8.1 ONLINE</b> <ul style="list-style-type: none"> <li>Real-time tracking of expenses and managing their money</li> <li>Getting insights as notification</li> </ul> <b>8.2 OFFLINE</b> <ul style="list-style-type: none"> <li>As this is a web application, there is no offline facilities</li> </ul>
<b>4. EMOTIONS: BEFORE / AFTER</b> <b>EM</b> <b>Before:</b> Lack of knowledge of personal finance and lack of awareness to track their expense. Results in more spending and less saving culture. <b>After:</b> Track their expenses and gets valuable insights that lead them to create a discipline in saving and reducing their extra spending.		

## CHAPTER 4

### REQUIREMENT ANALYSIS

#### 4.1 Functional requirement

These are the criteria that the end user directly requests as basic system facilities. As part of the contract, all of these features must be implemented into the system. These are depicted or stated in the form of input to the system, operation executed, and intended output. In contrast to non-functional needs, they are essentially the user-specified criteria that can be seen immediately in the final product.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	Forget password	Resetting the password by sending verification code to user's email
FR-3	Dashboard Panel	Insights about the expenses and navigation panel
FR-4	Expense Adder	User can add their expense and get insights
FR-5	Notifications	Send notification insights to the user's email
FR-6	User Confirmation	Confirmation via email

## 4.2 Non-Functional requirements

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to another. They are also called non-behavioral requirements. They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Used to track and manage user's expenses
NFR-2	Security	Encrypt data using TLS protocol and hash the data at the server
NFR-3	Reliability	Technical support by email assistance
NFR-4	Performance	Performance will be constant in any applications it is a web application

NFR-5	Availability	Available all the time as it is backed by IBM Virtual Servers
NFR-6	Scalability	Kubernetes cluster will manage the scalability parts by creating new pods in the cluster whenever required

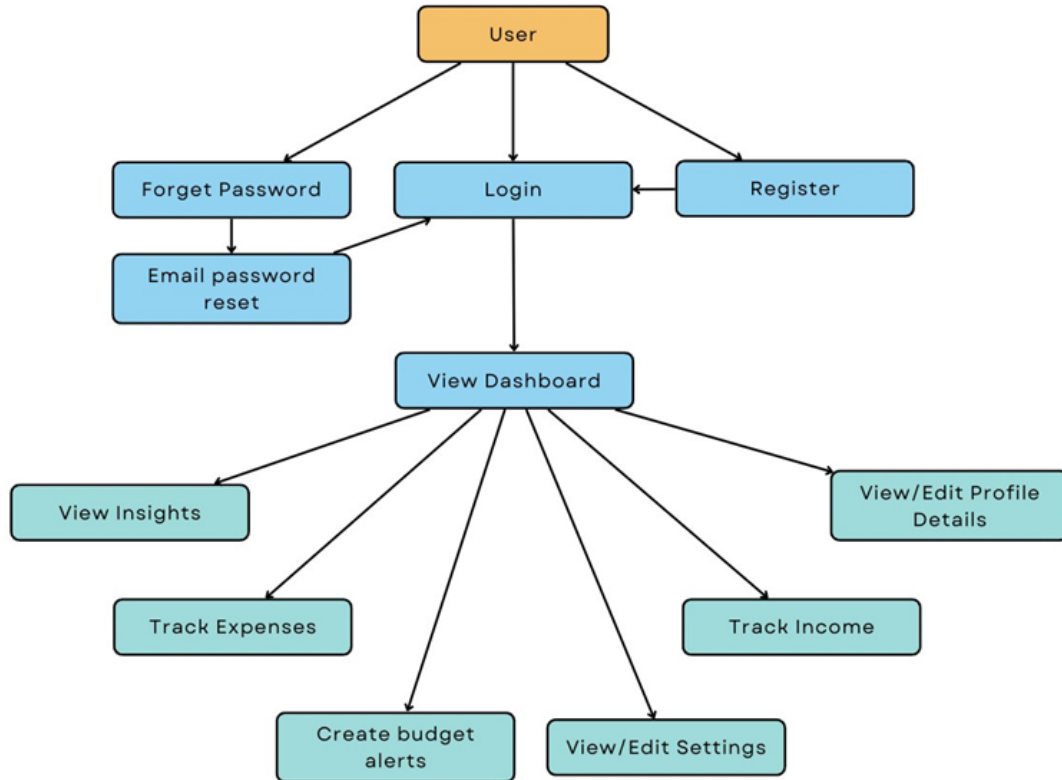
## **CHAPTER 5**

### **PROJECT DESIGN**

#### **5.1 Data Flow Diagrams**

A new user must register with the credentials, and a verification email is issued to the user. Once verified, the page is sent to the login page. If the user forgets his password, he can reset it by selecting the forgot password option and is then redirected to the login page. After successfully logging in, the user can access the dashboard. He has access to the dashboard.

- Add Expense
- Add Income
- View Charts
- Set the budget
- Available balance etc.,



## 5.2 Solution & Technical Architecture

### Components & Technologies

S.No.	Component	Description	Technology
1	User Interface	The user can interact with the application via web browser as a web application	HTML,CSS
2	Application Logic-1	The application contains the register and login services to access the dashboard	HTML,CSS,Python (Flask)
3	Application Logic-2	Dashboard contains insights about income and expenses	HTML,CSS,Python (Flask)
4	Application Logic-3	The user will get the reports weekly and monthly as email notification	HTML,CSS,Python (Flask),SendGrid



5	Database	Income and expenses-related data stored in SQL Database	MySQL
6	Cloud Database	User data are stored in a remote cloud database for high availability and insights	IBM DB2
7	Infrastructure	Kubernetes application with docker pods containing the application that can be deployed in IBM k8s cluster	Local,Cloud Foundry,Kubernetes,etc

### Application Characteristics

S.No.	Characteristics	Description	Technology
1	Open-Source Framework	Flask is used to implement API	Flask
2	Security Implementations	The IBM Container Registry in IBM Cloud provides high security to the user's data	Container Registry,K8s cluster,IBM cloud
3	Scalable Architecture	Kubernetes cluster scales the application based on the traffic	Container Registry,K8s cluster,IBM cloud

4	Availability	The Application is available to the user at any part of the world at any time	IBM cloud
5	Performance	The performance will be high because of the scalability by k8s cluster	Container Registry,K8s cluster,IBM cloud

### 5.3 User Story

In an agile framework, a user story is the smallest unit of work. It's a goal, not a feature, expressed from the perspective of a software user. A user story is an informal, generic explanation of a software feature written from the end user's or customer's point of view.

A user story's objective is to communicate how a piece of work will provide a specific value to the client. Customers do not have to be traditional external end users; they can be internal customers or colleagues within your firm that rely on your team. User stories are a few phrases that define the desired goal in simple terms. They don't go into detail. Requirements are added later, once agreed upon by the team. Stories fit neatly into agile frameworks like scrum and kanban.

User stories are added to sprints and "burned down" over the course of the sprint in scrum. Kanban teams collect user stories and route them through their workflow. This user story work helps scrum teams improve their estimating and sprint planning, resulting in more accurate forecasts and improved agility. Kanban teams may enhance their processes and learn how to manage work-in-progress (WIP) thanks to stories.

User stories are also the foundation of bigger agile frameworks such as epics and initiatives. Epics are major work items that have been broken down into a series of stories, and each endeavour is made up of numerous epics. These broader frameworks ensure that the development team's day-to-day activity (on stories) contributes to the corporate goals outlined in epics and initiatives.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
-----------	-------------------------------	-------------------	-------------------	---------------------	----------	---------

Customer (Mobile/Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Gmail		High	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
		USN-6	As a user, I can log into the application by Gmail		High	Sprint-1
	Forget password	USN-7	As a user, I need to change my password if I forget it		Medium	Sprint-1
	Dashboard	USN-8	As a user, I need to enter my income and expenses	I can view my insights about spending	High	Sprint-2
		USN-9	As a user, I need to set a budget alert for a month or a week	I can set spending alerts for a month or a week	Medium	Sprint-2
		USN-10	As a user, I need to get email notifications about my weekly and monthly spendings and earning	I can see my insights through mail about weekly and monthly insights	Medium	Sprint-2
		USN-11	As a user, I should be provided with some reward for saving money up to some extent	I should receive points for saving money	Low	Sprint-3
Administrator		USN-12	As an administrator, I can update the features and roll out a new version of application	I can fix the bugs and add features as per the request of the user	Low	Sprint-4

## CHAPTER 6

### PROJECT PLANNING & SCHEDULING

#### 6.1 Sprint Planning & Estimation

Estimation in Scrum projects is done by the entire team during the Sprint Planning Meeting. The goal of the estimation would be to prioritize the User Stories for the Sprint and assess the team's ability to deliver inside the Sprint Time Box.

The Product Owner ensures that the prioritized User Stories are clear, can be estimated, and are moved to the top of the Product Backlog.

Because the Scrum Team as a whole is in charge of delivering the product increment, care would be taken to choose the User Stories for the Sprint based on the size of the Product Increment and the effort required.

The size of the Product Increment is estimated in terms of User Story Points. Once the size is determined, the effort is estimated by means of the past data, i.e., effort per User Story Point called Productivity.

#### Sprint Planning

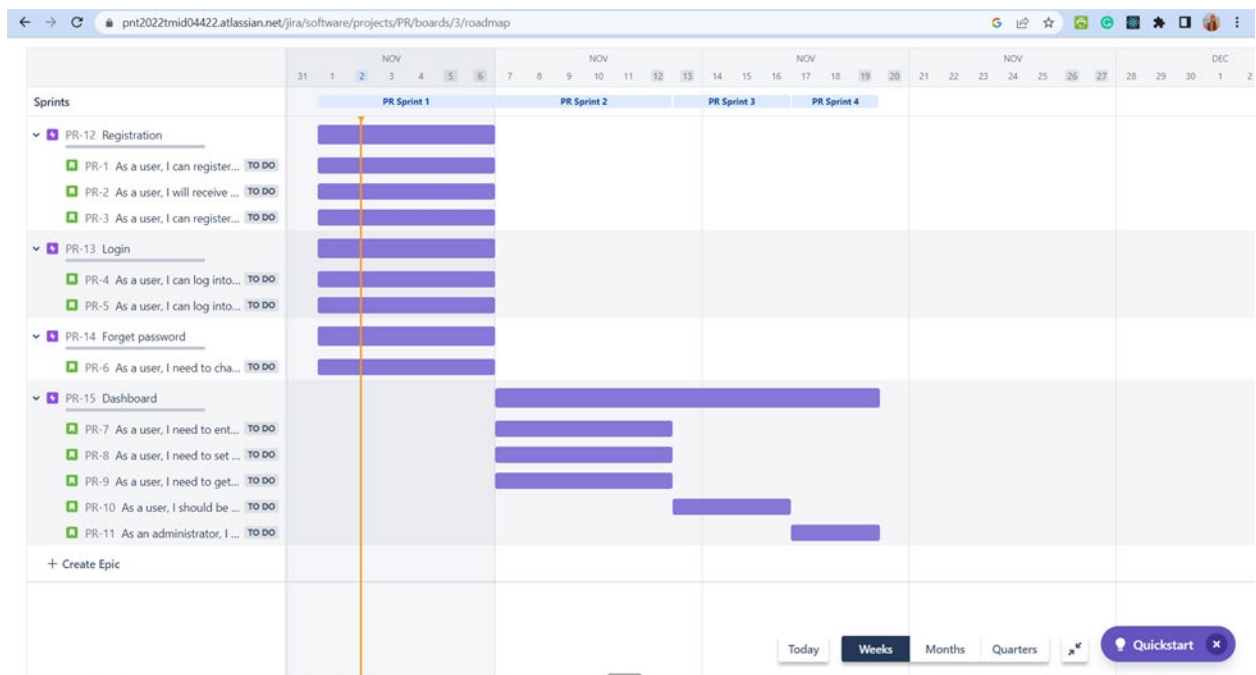
<b>Sprint</b>	<b>Functional Requirement</b>	<b>User Story Number</b>	<b>User Story / Task</b>	<b>Story Points</b>	<b>Priority</b>	<b>Team Members</b>
Sprint - 1	Registration	USN - 1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Nishanth S
Sprint - 1		USN - 2	As a user, I will	3	High	Priyadhars hini S

			receive confirmation email once I have registered for the application			
Sprint - 1		USN - 3	As a user, I can register for the application through Gmail	3	High	Sanjith M
Sprint - 1	Login	USN - 4	As a user, I can log into the application by entering email & password	2	High	Robin J
Sprint - 1		USN - 5	As a user, I can log into the application by Gmail	5	High	Priyadhars hini S
Sprint - 1	Forget password	USN - 6	As a user, I need to change my password if I forget it	5	Medium	Nishanth S
Sprint - 2	Dashboard	USN - 7	As a user, I need to enter my income and expenses	5	High	Priyadhars hini S
Sprint - 2		USN - 8	As a user, I need to set a budget alert for a month or a week	5	Medium	Sanjith M
Sprint - 2		USN - 9	As a user, I need to get email notifications about my weekly and monthly spending	10	Medium	Robin J

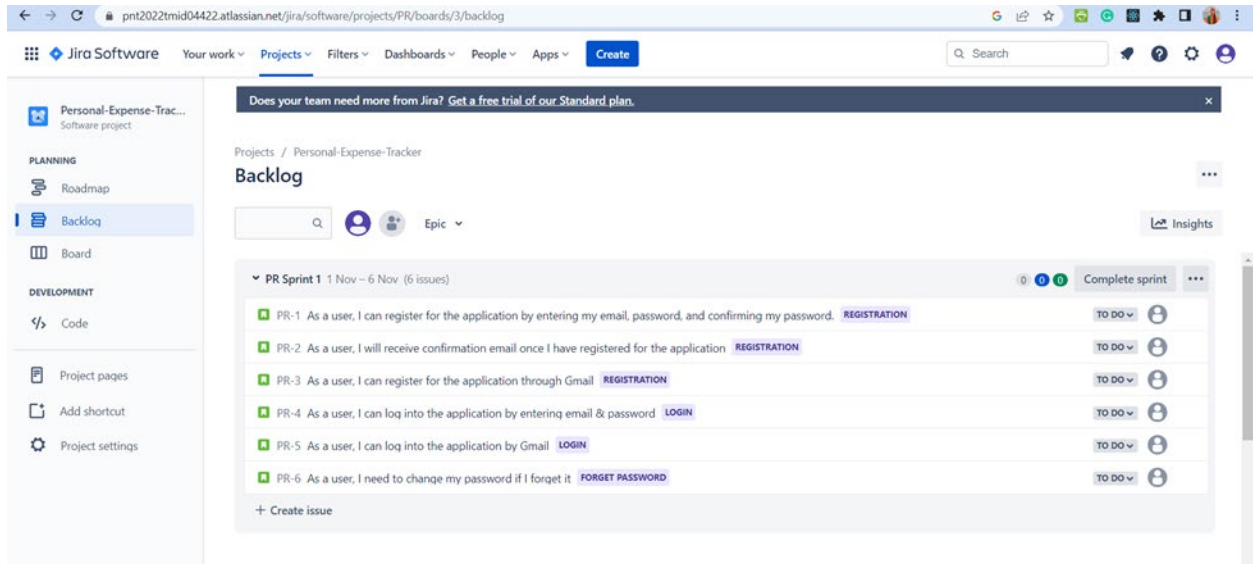
			and earning			
Sprint - 3		USN-10	As a user, I should be provided with some reward for saving money up to some extinct	20	Low	Sanjith M
Sprint - 4		USN-11	As an administrator, I can update the features and roll out a new version of application	20	Low	Nishanth S

## Using JIRA Software

### ROADMAP:

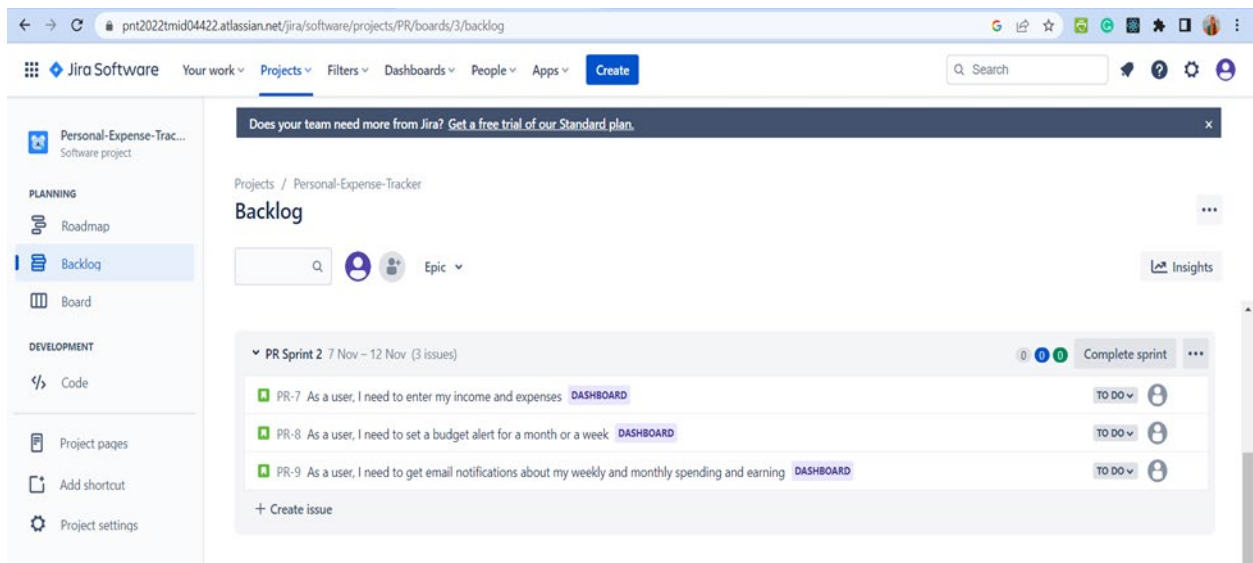


## BACKLOG:



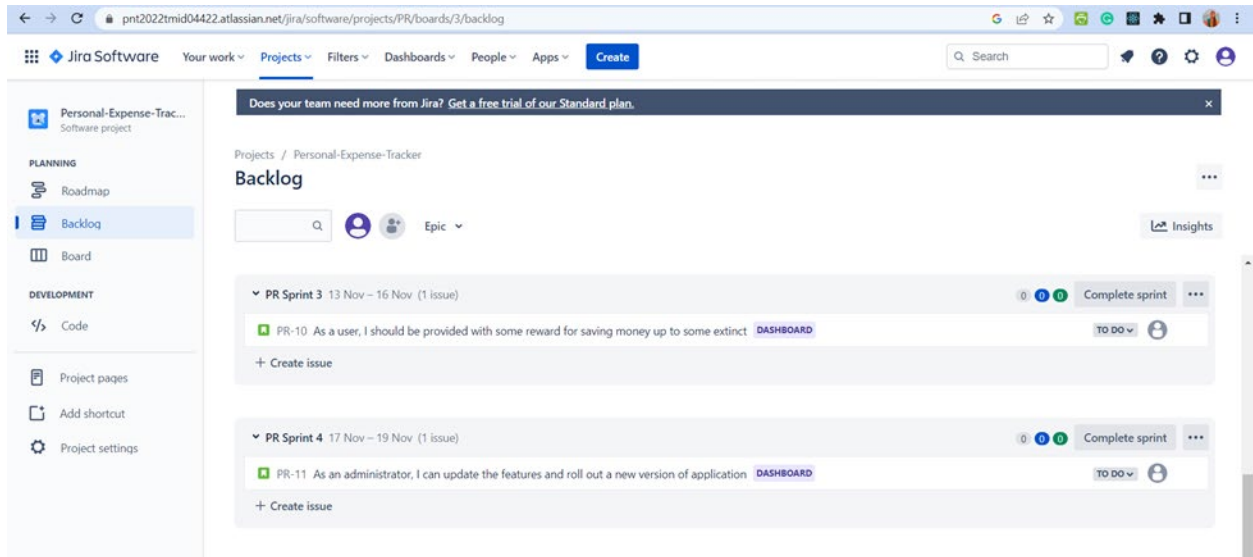
This screenshot shows the Jira Software interface for the 'Personal-Expense-Tracker' project. The left sidebar contains navigation options under 'PLANNING' (Roadmap, Backlog, Board) and 'DEVELOPMENT' (Code, Project pages, Add shortcut, Project settings). The main area displays the 'Backlog' for 'PR Sprint 1' (1 Nov - 6 Nov) with 6 issues. Each issue is a user story with a key, description, category, and status.

Key	Description	Category	Status
PR-1	As a user, I can register for the application by entering my email, password, and confirming my password.	REGISTRATION	TO DO
PR-2	As a user, I will receive confirmation email once I have registered for the application	REGISTRATION	TO DO
PR-3	As a user, I can register for the application through Gmail	REGISTRATION	TO DO
PR-4	As a user, I can log into the application by entering email & password	LOGIN	TO DO
PR-5	As a user, I can log into the application by Gmail	LOGIN	TO DO
PR-6	As a user, I need to change my password if I forget it	FORGET PASSWORD	TO DO



This screenshot shows the Jira Software interface for the 'Personal-Expense-Tracker' project, displaying the 'Backlog' for 'PR Sprint 2' (7 Nov - 12 Nov) with 3 issues. The layout is consistent with the previous screenshot, showing user stories with keys, descriptions, categories, and statuses.

Key	Description	Category	Status
PR-7	As a user, I need to enter my income and expenses	DASHBOARD	TO DO
PR-8	As a user, I need to set a budget alert for a month or a week	DASHBOARD	TO DO
PR-9	As a user, I need to get email notifications about my weekly and monthly spending and earning	DASHBOARD	TO DO



## 6.2 Sprint Delivery Schedule

A sprint is a fixed period of time during which an agile team works to fulfill a defined set of development tasks. In most cases, a major development project includes numerous sprints. Sprints, in the end, provide a framework for breaking down huge, complicated software projects into manageable periods.

When a sprint comes to a close, the team presents their work to the project owner, who reviews it. If the project accomplishes its objectives, the team will move on to the next sprint.

Because sprints are time-limited, it is vital to prevent wasting time throughout planning and development. And this is precisely where sprint scheduling comes into play.

A sprint schedule, for those who are unfamiliar, is a document that details sprint planning from beginning to conclusion. It is one of the initial tasks in the agile sprint planning process, and it necessitates thorough research, planning, and communication.



## Sprint Schedule

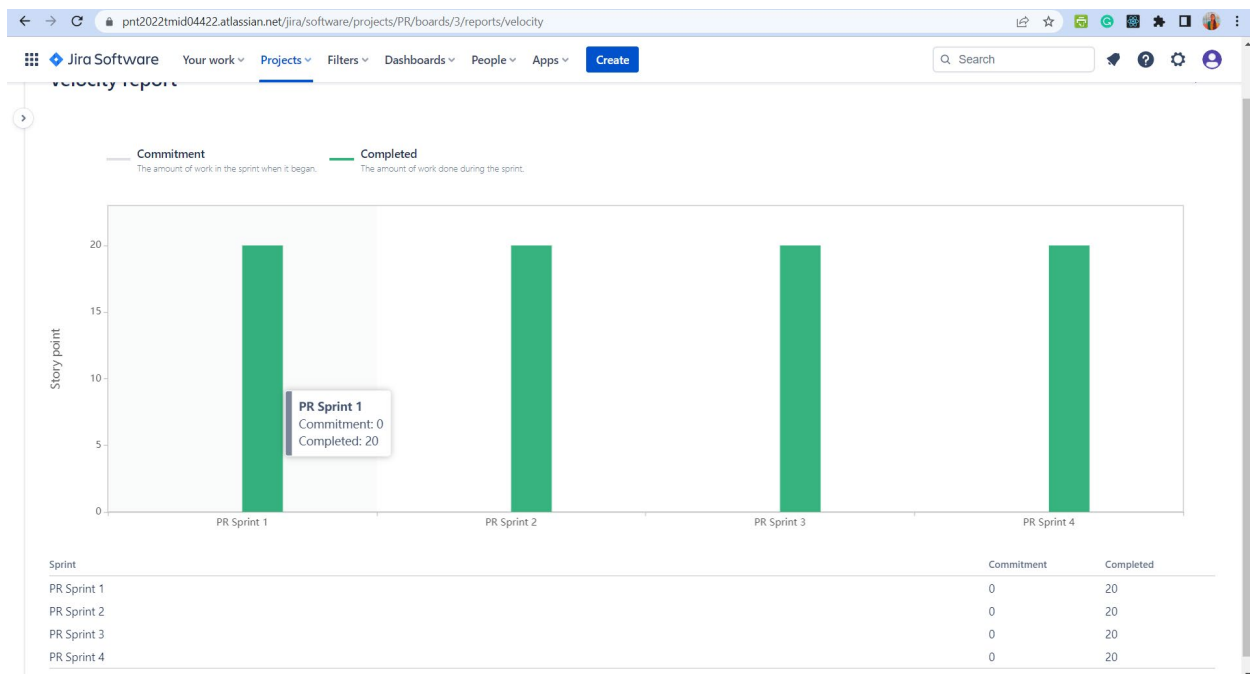
Sprint	Total Story Points	Duration	Sprint Start	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint 1	20	6 days	01 Nov 2022	06 Nov 2022	20	06 Nov 2022
Sprint 2	20	6 days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint 3	20	4 days	13 Nov 2022	16 Nov 2022	20	16 Nov 2022
Sprint 4	20	3 days	17 Nov 2022	19 Nov 2022	20	19 Nov 2022

## 6.3 Reports from JIRA

	A	B	C	D	E	F	G	H	I	J	K	L
1	Issue Type	Key	Summary	Assignee	Reporter	Priority	Status	Resolution	Created	Updated	Due date	
2	Epic	PR-15	Dashboard		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:51	11/24/2022 22:41	11/19/2022	
3	Epic	PR-14	Forget password		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:51	11/24/2022 22:41	11/6/2022	
4	Epic	PR-13	Login		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:51	11/24/2022 22:41	11/6/2022	
5	Epic	PR-12	Registration		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:51	11/24/2022 22:41	11/6/2022	
6	Story	PR-11	As an administrator, I can update the features and roll out a ne		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:50	11/24/2022 21:07		
7	Story	PR-10	As a user, I should be provided with some reward for saving mc		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:50	11/24/2022 21:07		
8	Story	PR-9	As a user, I need to get email notifications about my weekly an		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:50	11/24/2022 21:07		
9	Story	PR-8	As a user, I need to set a budget alert for a month or a week		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:49	11/24/2022 21:07		
10	Story	PR-7	As a user, I need to enter my income and expenses		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:49	11/24/2022 21:06		
11	Story	PR-6	As a user, I need to change my password if I forget it		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:49	11/24/2022 21:06		
12	Story	PR-5	As a user, I can log into the application by Gmail		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:49	11/24/2022 21:06		
13	Story	PR-4	As a user, I can log into the application by entering email & pas		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:48	11/24/2022 21:06		
14	Story	PR-3	As a user, I can register for the application through Gmail		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:48	11/24/2022 21:06		
15	Story	PR-2	As a user, I will receive confirmation email once I have register		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:48	11/24/2022 21:06		
16	Story	PR-1	As a user, I can register for the application by entering my email		PRIYADHARSHINI S 19CSR149	Medium	Done	Done	11/2/2022 10:48	11/24/2022 21:06		
17												
18												
19												

### a.Velocity Report

The average quantity of work completed by a scrum team during a sprint is referred to as velocity. This can be assessed in narrative points or the number of issues in team-managed Jira Software projects. Because the report monitors projected and accomplished work over numerous sprints, teams may use velocity to predict how quickly they can work through the backlog. The more sprints there are, the more accurate the forecast becomes.



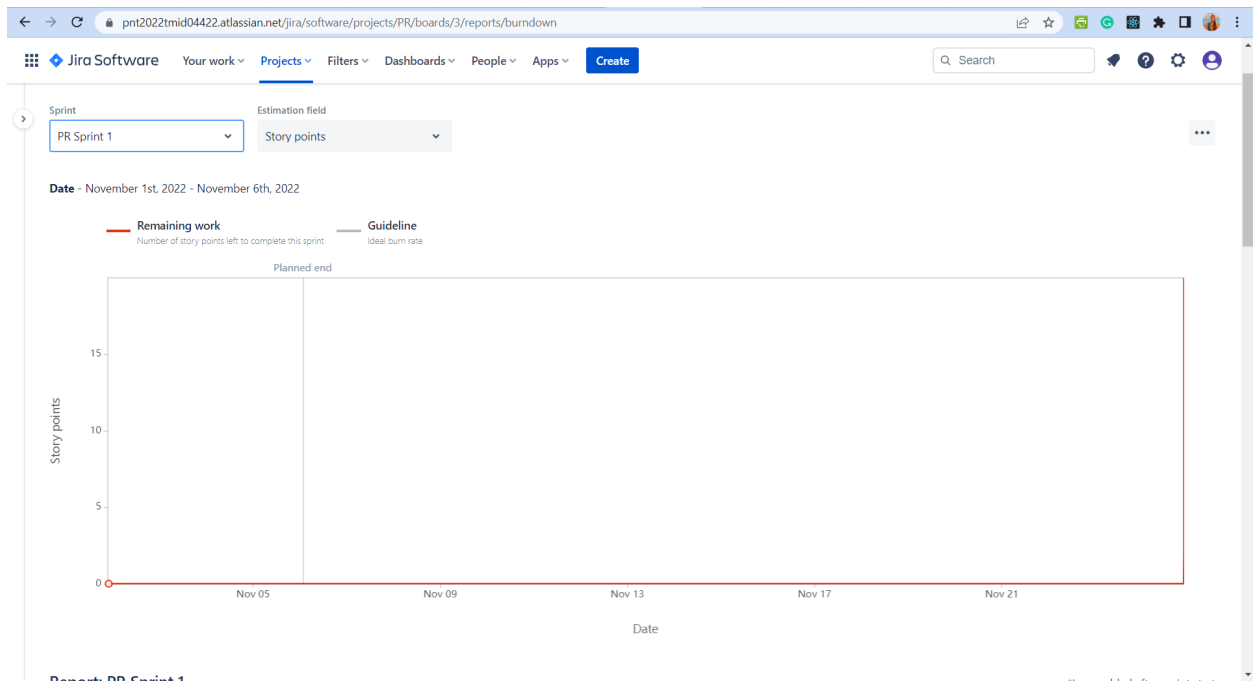
## The above report describes

- The vertical axis displays the statistic used for estimating stories. The horizontal axis displays the last 7 sprints completed by the team; this data is used to calculate velocity.
- The **Commitment** (blue) bar for each sprint shows the total estimate of all issues in the sprint when it begins. After the sprint has started, any stories added to the sprint, or any changes made to estimates, will not be included in this total.
- The **Completed** (green) bar in each sprint shows the total completed estimates when the sprint ends. Any scope changes made after the sprint started are included in this total.

## b.Sprint burndown Chart

A sprint burndown chart displays the amount of work performed in a sprint as well as the total amount of work remaining. Sprint burndown charts are used to forecast your team's chances of finishing their job in the time allotted. A team can manage its progress and respond to trends by tracking the remaining work during the sprint. For example, if the burndown chart indicates that the team is unlikely to meet the sprint objective, the team can take the appropriate steps to stay on track.

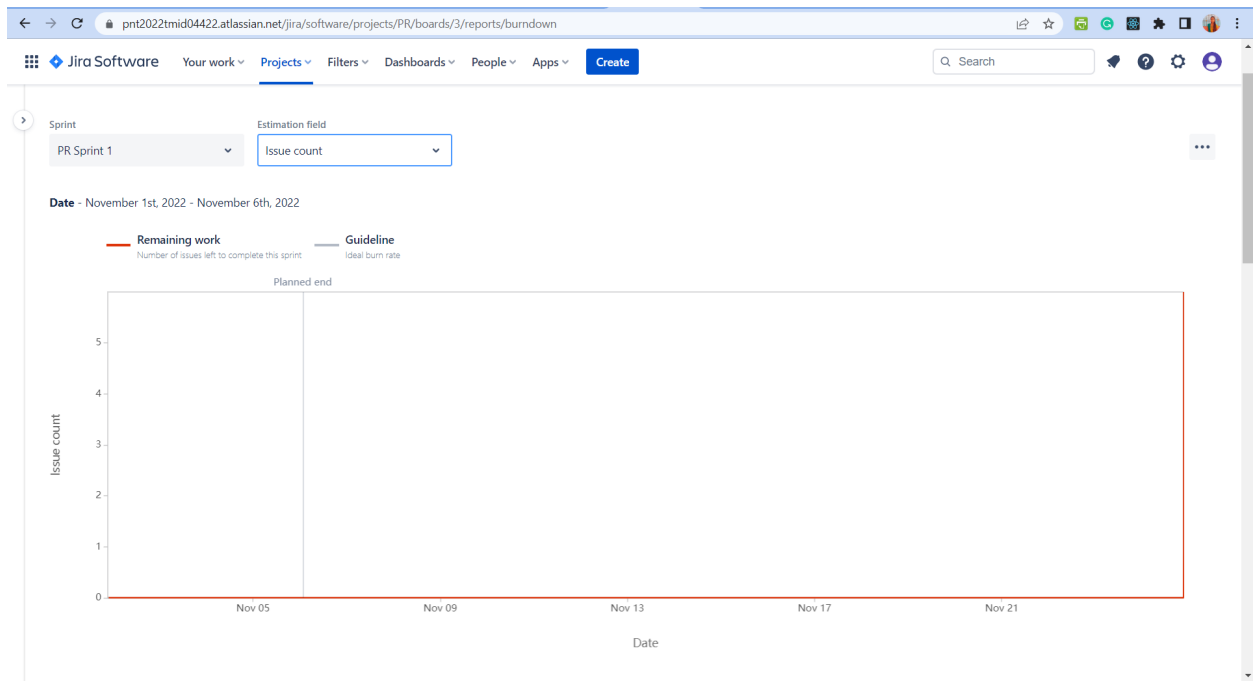
## Sprint-1/Story points



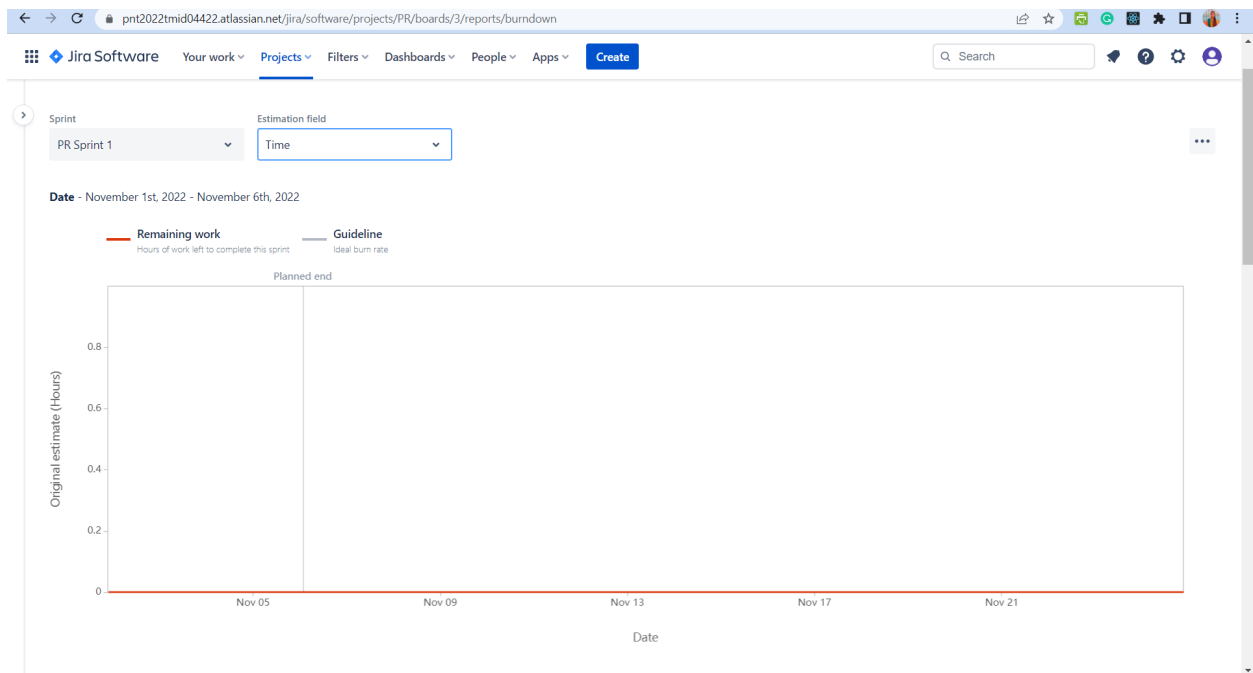
- The vertical axis represents the amount of work; either number of issues or story points (if Estimation is enabled). The horizontal axis represents the timeframe of the sprint.
- The **gray** line shows the ideal progress rate. It trends downwards at a linear rate, because teams should ideally be completing work at a consistent pace.
- The **red** line shows how much work remains in the sprint. The closer this line is to the grey line, the better.

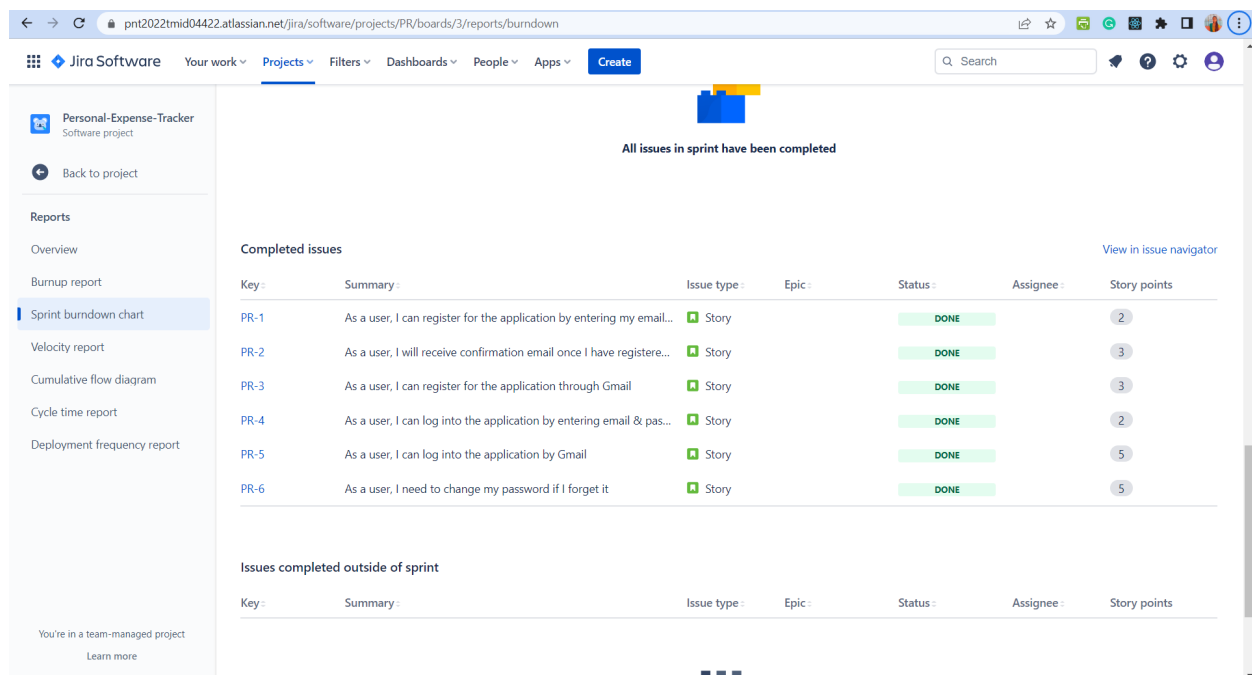
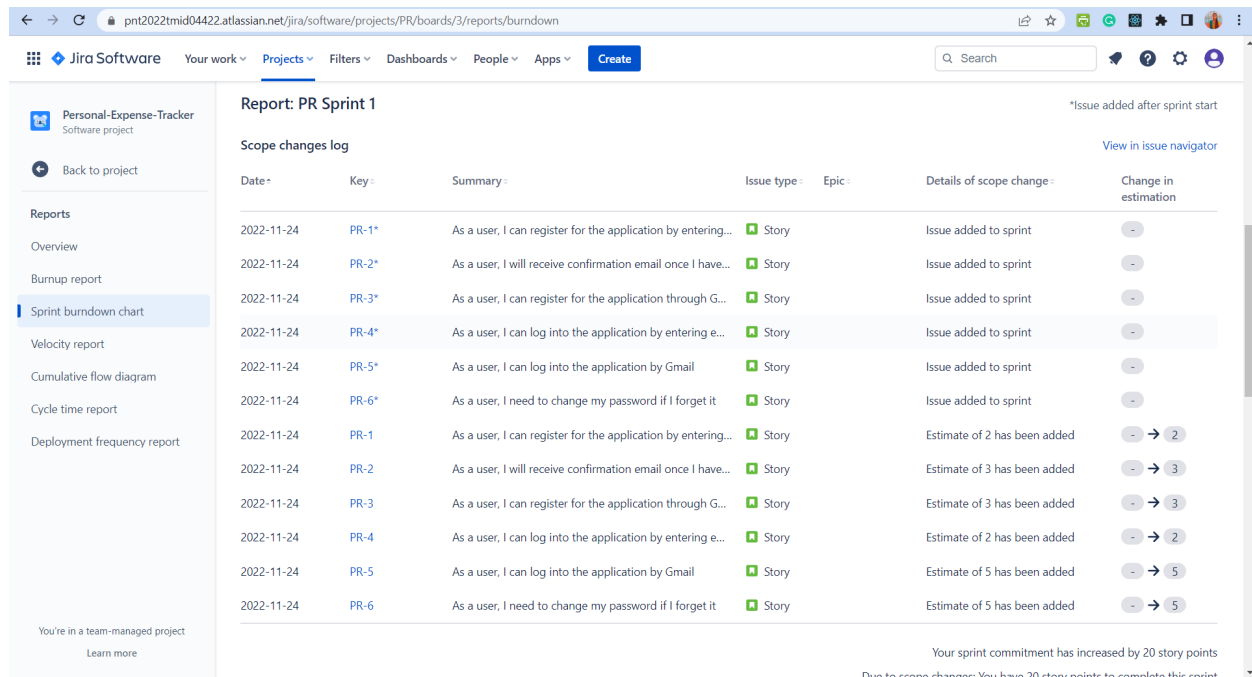
## Sprint-1/Issue

Count



## Sprint-1/Time





- The report also has a number of tables that provide more context for the chart. Table data can be sorted by selecting the column header. The tables are:

- Scope changes log: Issues that were added to the sprint, removed from the sprint, or had estimation changes, while the sprint was in progress.
- Incomplete issues: Issues in the sprint that were never completed (ie: never moved to a *Done* status). This includes issues that were never started.
- Completed issues: Issues that were moved to a *Done* status while the sprint was in progress.
- Issues completed outside of sprint: This includes issues that were:
  - completed and then added to the sprint, either before the sprint started or after the sprint started.
  - added to the sprint, but completed before the sprint started.
- Issues removed from sprint: Issues that were removed from the sprint while the sprint was in progress.

## CHAPTER-7

### CODING & SOLUTIONING

#### 7.1 Feature 1

**Bar chart section;-**

```
<div class="col col-md-6">
```

```
  <canvas id="myChart" width="400" height="400"></canvas>
```

```
  <script>
```

```
    let food = document.getElementById('tfood').innerHTML
```

```
    let entertainment = document.getElementById('tentertainment').innerHTML
```

```
    let business = document.getElementById('tbusiness').innerHTML
```

```
    let rent = document.getElementById('trent').innerHTML
```

```
    let emi = document.getElementById('temi').innerHTML
```

```
    let other = document.getElementById('tother').innerHTML
```

```
    var ctx = document.getElementById('myChart').getContext('2d');
```

```
    console.log("food ", food);
```

```
    var myChart = new Chart(ctx, {
```

```
      type: 'doughnut',
```

```
      data: {
```

```
        labels: ['Food', 'Entertainment', 'Business', 'Rent', 'EMI', 'Other'],
```

```
        datasets: [{
```

```
label: 'Expenses Chart',

data: [food, entertainment, business, rent, emi, other],

backgroundColor: [

    'rgb(255, 99, 132)',

    'rgb(0, 0, 0)',

    'rgb(255, 205, 86)',

    'rgb(201, 203, 207)',

    'rgb(54, 162, 235)',

    'rgb(215, 159, 64)'

],

}]

},

options: {

    responsive: true,

    plugins: {

        legend: {

            position: 'bottom',

        },

        title: {
```



```
        display: true,

        text: 'EXPENSE BREAKDOWN'

    }

}

}

});

</script>

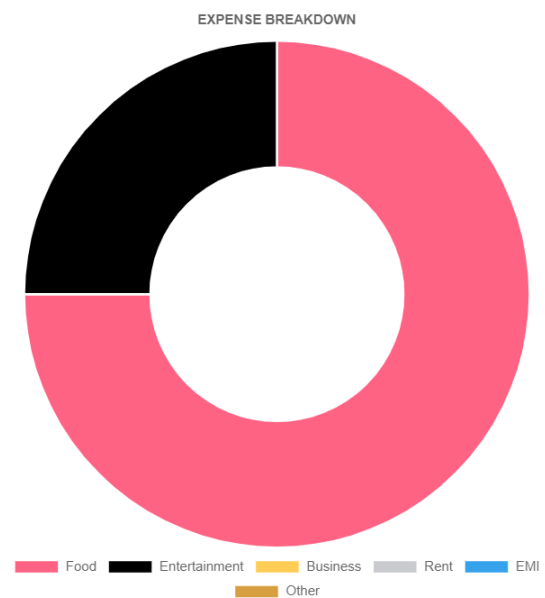
</div>

</div>
```

TIME	AMOUNT
November	4000.0

### Expense Breakdown BY Category

#	Category	Amount
1	Food	3000.0
2	Entertainment	1000.0
3	Business	0
4	Rent	0
5	EMI	0
6	Other	0
7	Total	₹ 4000.0



## 7.2 Feature 2

### Points System:-

```
@app.route("/reward")
```

```
def reward():
```

```
    sql = "SELECT AMOUNT FROM EXPENSES WHERE USERID=? AND  
    MONTH(date)=MONTH(DATE(NOW()))"
```

```
    stmt = ibm_db.prepare(conn, sql)
```

```
    ibm_db.bind_param(stmt, 1, session["id"])
```

```
    ibm_db.execute(stmt)
```

```
    fetchedamount = ibm_db.fetch_tuple(stmt)
```

```
    point = 0
```

```
    for i in fetchedamount:
```

```
        point = point + i
```

```
    sql1 = "SELECT EXPLIMIT FROM LIMITS WHERE USERID=?"
```

```
    stmt1 = ibm_db.prepare(conn, sql1)
```

```
    ibm_db.bind_param(stmt1, 1, session["id"])
```

```
    limit = ibm_db.fetch_tuple(stmt1)
```

```
    for i in limit:
```

```
        print(i)
```

```
        limit = i
```

creditpoint = (point/limit)\*100

creditpoint = 100-creditpoint

return render\_template("reward.html",point=creditpoint)

### 7.3 Database Schema

#### UserTable

Sno	FieldName	Datatype
1	USERID	INT
2	USERNAME	VARCHAR
3	PASSWORD	VARCHAR
4	EMAIL	VARCHAR

#### ExpensesTable

Sno	FieldName	Datatype
1	USERID	INT
2	DATE	DATE
3	EXPENSENAME	VARCHAR
4	PAYMENT MODE	VARCHAR
5	CATEGORY	VARCHAR
6	TIME	VARCHAR

#### ExpenseLimit Table

Sno	FieldName	Datatype
1	USERID	INT
2	EXPLIMIT	VARCHAR

# CHAPTER-8

## TESTING

### 8.1 Test Cases

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)
LoginPage_TC_D001	Functional	Home Page	Verify user is able to see the Login/Signup page		1.Enter URL, and click go 2.Verify login/Signup displayed or not	<a href="http://cloud-object-storage-47-cos-static-web-hosting-aws-s3-web-jp-tok.cloud-object-storage.ap-southeast-1.amazonaws.com/authenticate">http://cloud-object-storage-47-cos-static-web-hosting-aws-s3-web-jp-tok.cloud-object-storage.ap-southeast-1.amazonaws.com/authenticate</a>	Login/Signup page should display	Working as expected	Pass	Initial Load is very less	
LoginPage_TC_D002	UI	Home Page	Verify the UI elements in Login/Signup		1.Enter URL, and click go 2.Verify login/Signup with below UI elements: a.email text box b.password text box c.Login button d.Last password? Forgot password link	<a href="http://cloud-object-storage-47-cos-static-web-hosting-aws-s3-web-jp-tok.cloud-object-storage.ap-southeast-1.amazonaws.com/authenticate">http://cloud-object-storage-47-cos-static-web-hosting-aws-s3-web-jp-tok.cloud-object-storage.ap-southeast-1.amazonaws.com/authenticate</a>	Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.New customer? Create account link e.Last password? Recovery password link	Working as expected	pass	user friendly and attractive	
LoginPage_TC_D003	Functional	Home page	Verify user is able to log into application with Valid credentials		1.Enter URL( <a href="http://cloud-object-storage-47-cos-static-web-hosting-aws-s3-web-jp-tok.cloud-object-storage.ap-southeast-1.amazonaws.com/authenticate">http://cloud-object-storage-47-cos-static-web-hosting-aws-s3-web-jp-tok.cloud-object-storage.ap-southeast-1.amazonaws.com/authenticate</a> ) and click go 2.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: chalam@gmail.com password: Testing123	User should navigate to user account Dashboard		pass	Components are routed	
LoginPage_TC_D004	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL( <a href="http://cloud-object-storage-47-cos-static-web-hosting-aws-s3-web-jp-tok.cloud-object-storage.ap-southeast-1.amazonaws.com/authenticate">http://cloud-object-storage-47-cos-static-web-hosting-aws-s3-web-jp-tok.cloud-object-storage.ap-southeast-1.amazonaws.com/authenticate</a> ) and click go 2.Enter Invalid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: chalam@gmail.com password: Testing123	Application should show 'Incorrect email or password' validation message.		pass	Components are routed	

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)
LoginPage_TC_D004	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL( <a href="http://cloud-object-storage-47-cos-static-web-hosting-aws-s3-web-jp-tok.cloud-object-storage.ap-southeast-1.amazonaws.com/authenticate">http://cloud-object-storage-47-cos-static-web-hosting-aws-s3-web-jp-tok.cloud-object-storage.ap-southeast-1.amazonaws.com/authenticate</a> ) and click go 2.Enter Valid username/email in Email text box 3.Enter Invalid password in password text box 5.Click on login button	Username: chalam@gmail.com password: Testing123678686786876876	Application should show 'Incorrect email or password' validation message.		pass	Its comfortable to use	
LoginPage_TC_D005	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL( <a href="http://cloud-object-storage-47-cos-static-web-hosting-aws-s3-web-jp-tok.cloud-object-storage.ap-southeast-1.amazonaws.com/authenticate">http://cloud-object-storage-47-cos-static-web-hosting-aws-s3-web-jp-tok.cloud-object-storage.ap-southeast-1.amazonaws.com/authenticate</a> ) and click go 2.Enter Invalid username/email in Email text box 3.Enter Invalid password in password text box 4.Click on login button	Username: chalam password: Testing123678686786876876	Application should show 'Incorrect email or password' validation message.		pass	user friendly and attractive	
LoginPage_TC_D006	Functional	Dashboard	Add income and expense and check wheather the insights are changing		1.After login 2.Add Expenses and Income 3.View Charts	Income:Rs in Inr Expense:Rs in Inr	View Charts		pass	Every Events are changing accurately	
LoginPage_TC_D007	Functional	Dashboard	Email Insights		Insights are viewed in mail		Mail Insights		pass	Insights are reached on time	

### 8.2 User Acceptance Testing

### Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	5	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	75

## Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	29	0	0	29
Security	4	0	0	4
Outsource Shipping	6	1	0	5
Exception Reporting	7	0	0	7
Final Report Output	4	0	0	4
Version Control	8	0	0	8

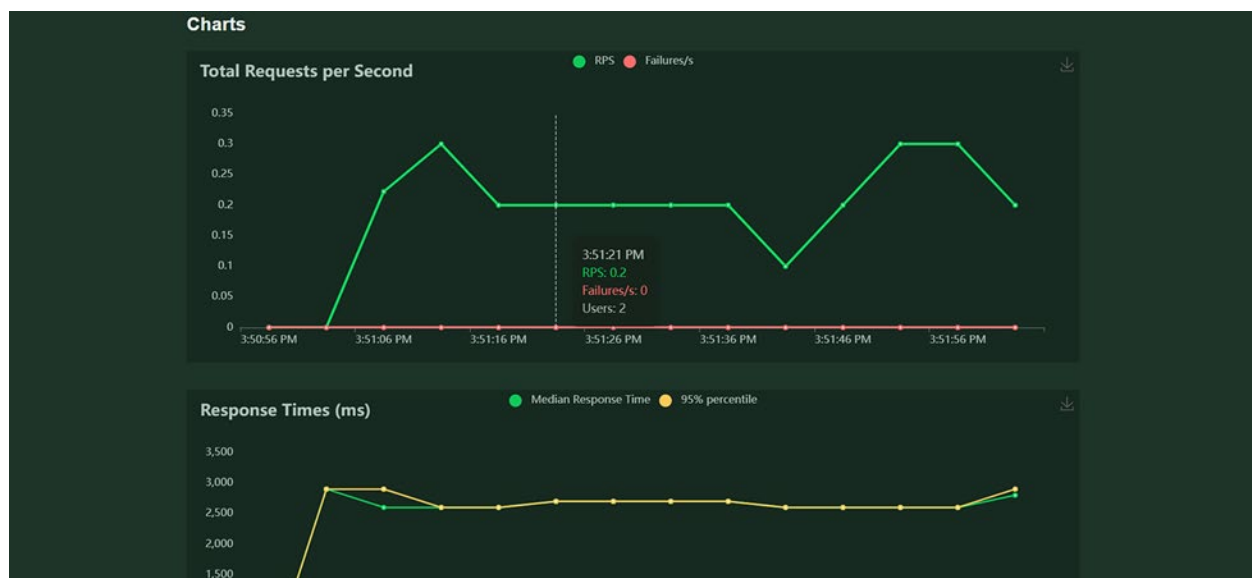
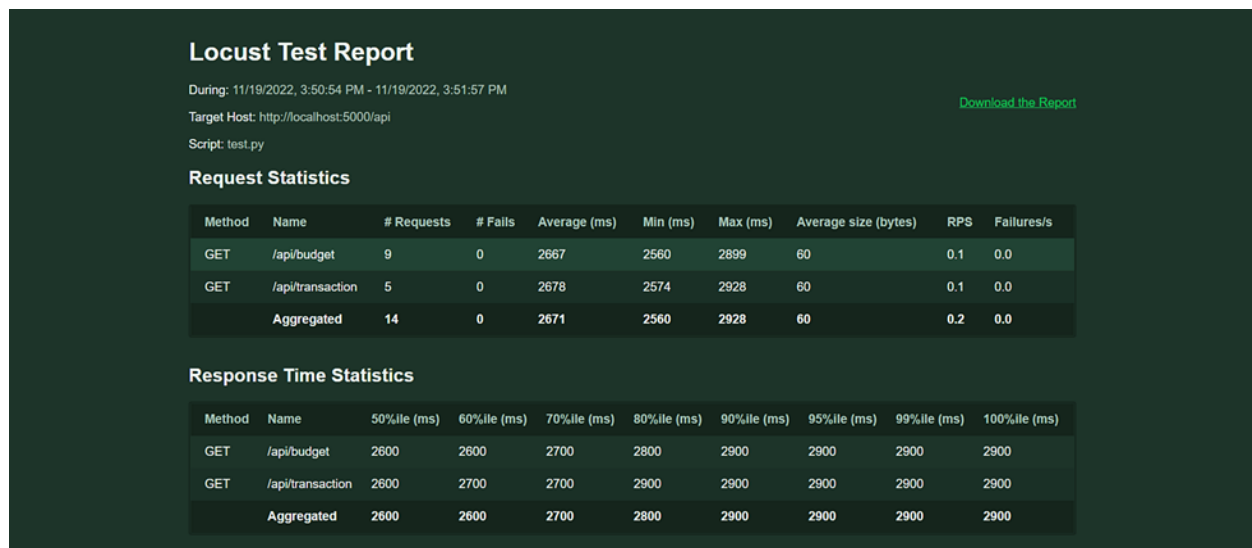
## CHAPTER-9

## RESULTS

### 9.1 Performance Metrics

#### Locust

Locust is a Python-based testing tool for load testing and simulating user behavior. Load testing is the practice of testing a software application with the primary goal of stressing its capabilities. Locust is a software application that generates a set of testing functions





## **CHAPTER-10**

### **ADVANTAGES & DISADVANTAGES**

#### **10.1 Advantages**

There are numerous reasons why you should use a personal spending tracker, and we've highlighted a few of the most significant ones. One of the primary benefits of using a personal spending tracker is that it allows you to keep track of where your money goes each month. This is especially crucial if you're attempting to keep your finances under control; not knowing where your money is going may be a major stress. You'll be able to remain on top of your spending by using a cost tracker, allowing you to spot trouble areas and make the necessary changes to get back on track.

Another reason you might want to utilize a personal cost tracker is because it can be a valuable budgeting tool. This is especially beneficial if you don't have a system in place to keep track of how much money you spend each month - it's impossible to meet your objectives if you don't even know how much money you have available to spend. Using a spending tracker can assist you in creating an accurate budget, giving you a clearer understanding of how much you can spend without going bankrupt.

The following are some of the benefits of using a personal cost tracker:

- 1) You can keep track of the regular expenses you incur, such as groceries and utilities.
- 2) You can create budget categories for your expenses and track how much you spend in each one.
- 3) You can monitor where your money is going each month and make changes to improve your spending habits.
- 4) Your costs are simple to categorize and categorize.
- 5) You can simply see your weekly or monthly spending at a glance, so you always know where you stand financially.



## 10.2 Disadvantages

One of the numerous benefits of using a personal cost tracker is that it allows you to keep track of your spending in an organized and simple fashion. However, there are a few drawbacks to employing this type of spending tracking system. One of the biggest downsides of utilizing a personal spending tracker is that it requires a significant amount of attention to operate properly.

The following are some disadvantages of utilizing a personal cost tracker:

- 1) Keeping track of all of your costs in one spot can be challenging.
- 2) Entering all of the information of your transactions each week or month might be time consuming.
- 3) Seeing how much money you spent during the month might be unpleasant, and it can make you feel guilty about overspending in some areas.
- 4) If you use the app to track your costs, categorizing each transaction can take a long time.
- 5) Finding a category for a transaction can be difficult if you don't know where to put it.

## **CHAPTER-11**

### **CONCLUSION**

As a business owner, you understand how important it is to keep track of your spending in order to keep your company running effectively. Manually performing this procedure, on the other hand, can be time-consuming and complicated. As a result, having an automated expenditure monitoring system that records all of your business expenses and allows you to track them quickly is critical. Once you've established such a system, you can use it to examine your spending and discover areas where you might be able to cut costs. This will allow you to save money in the long term while also improving your company's performance. Below is a summary of some of the advantages that your company can expect from deploying a fully automated expense management system.

1. It makes it simple to detect and eliminate wasteful spending. If your company has no way of identifying and eliminating unnecessary expenses, it may not be getting the most out of its expenses. However, by implementing a system that automatically records and analyzes your expenses, you can quickly and easily identify any areas that are not producing positive results for your business. You can then put strategies in place to eliminate these costs and get your business back on track.

2. It enables you to examine your spending habits and identify opportunities for improvement. A strategy for tracking your expenses allows you to identify areas where you may be overspending and make the required modifications and cost cuts. These changes will assist you in lowering your overall costs and increasing the efficiency of your organization

## **CHAPTER-12**

### **FUTURE SCOPE**

Individuals are increasingly looking for quick and effective solutions to manage their finances as the world gets more computerized. Personal cost trackers are a common way to accomplish this. Personal cost trackers allow you to track all of your expenses in real time, making it simple to stay on top of your money and stay on track with your budget. When it comes to personal spending trackers, there are a number of excellent solutions on the market, but what truly distinguishes them is their extensive feature set. In this post, we'll look at some of the top personal cost trackers on the market today and see what differentiates them from the competition. Let's begin by looking at some of the essential elements that a decent personal spending tracker should have...

#### **12.1 Key features of a personal expense tracker**

##### **12.1.1 Helps you stick to your budget**

While being able to watch all of your transactions in real time is excellent, it can be difficult to figure out where all of the money goes. This is where a personal cost tracker can come in handy. The majority of personal cost trackers feature budget tracking capability, which allows you to track your spending on a daily basis. Setting a budget allows you to keep track of your expenditures and avoid spending more than you can afford. This might keep you from going into significant debt and help you stay on top of your finances.

##### **12.1.2 Monitors your spending in real time**

It's more vital than ever to keep track of where your money is going when it comes to personal finance. A decent personal spending tracker will assist you with this by continuously monitoring all of your transactions. This assists you in identifying areas where you may be overspending or where you may save money. It also allows you to keep track of unexpected costs and plan for the future. When you watch all of your transactions in real time, you can immediately take corrective action to ensure your financial health in the future.

### **12.1.3 Works on any device**

Most individuals nowadays access the internet using mobile devices such as smartphones and tablets. This means you'll need a tool that can be accessed from any device so you can track your spending on the fly. Most personal spending trackers allow you to access your account from any smartphone, tablet, or computer and examine real-time account information.

## CHAPTER-13

### APPENDIX

#### SOURCE CODE:

##### App.py

```
from flask import Flask, render_template, request, redirect, session

import ibm_db

import re

import os

import sendgrid

from sendgrid.helpers.mail import Mail, Email, To, Content

app = Flask(__name__)

DB_URL = str(os.environ.get('DB_URL'))

SENDGRID_API_KEY = str(os.environ.get('SENDGRID_API_KEY'))

app.secret_key = "a"

conn = ibm_db.connect(

    DB_URL,
```

```
    "" ,

    "" ,

)

# 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 = ""

    if request.method == "POST":

        username = request.form["username"]

        email = request.form["email"]

        password = request.form["password"]

        sql = "SELECT * FROM REGISTER WHERE USERNAME =?"

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt, 1, username)

        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        print(account)

        if account:

            msg = "Account 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:
```

```

        sql1 = "INSERT INTO REGISTER (USERNAME, PASSWORD, EMAIL)
VALUES (?, ?, ?) "

        stmt1 = ibm_db.prepare(conn, sql1)

        ibm_db.bind_param(stmt1, 1, username)

        ibm_db.bind_param(stmt1, 2, password)

        ibm_db.bind_param(stmt1, 3, email)

        ibm_db.execute(stmt1)

        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"]

    sql = "SELECT * FROM REGISTER WHERE USERNAME =? AND PASSWORD =?"

    stmt = ibm_db.prepare(conn, sql)

    ibm_db.bind_param(stmt, 1, username)

    ibm_db.bind_param(stmt, 2, password)

    ibm_db.execute(stmt)

    account = ibm_db.fetch_assoc(stmt)

    if account:

        session["loggedin"] = True

        session["id"] = account["ID"]

        userid = account["ID"]

        session["username"] = account["USERNAME"]

        session["email"] = account["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"]

    time = request.form["time"]

    sql = "INSERT INTO EXPENSES (USERID, DATE, EXPENSENAME, AMOUNT, PAYMENTMODE, CATEGORY, TIME) VALUES (?, ?, ?, ?, ?, ?, ?) "

    creditpoint = (int(amount)/10000)*100
```

```
# sql2 = "INSERT INTO CREDITS(USERID,CREDIT) VALUES (?,?) "

# sql3 = "SELECT CREDIT FROM CREDITS WHERE USERID=?"

stmt = ibm_db.prepare(conn, sql)

# stmt2 = ibm_db.prepare(conn, sql2)

# stmt3 = ibm_db.prepare(conn, sql3)

# ibm_db.bind_param(stmt3,1,session["id"])

# ibm_db.execute(stmt3)

# fetchedamount = ibm_db.fetch_tuple(stmt3)

# point = 0

# for i in fetchedamount:

#     point = point + i

# creditpoint = point+creditpoint

# print(creditpoint)

ibm_db.bind_param(stmt, 1, session["id"])

ibm_db.bind_param(stmt, 2, date)

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.bind_param(stmt, 7, time)

# ibm_db.bind_param(stmt2,1,session["id"])

# ibm_db.bind_param(stmt2,2,creditpoint)

ibm_db.execute(stmt)
```

```

# ibm_db.execute(stmt2)

print(date + " " + expensename + " " +

        amount + " " + paymode + " " + category)

sql1 = "SELECT * FROM EXPENSES WHERE USERID=? AND
MONTH(date)=MONTH(DATE(NOW()))"

stmt1 = ibm_db.prepare(conn, sql1)

ibm_db.bind_param(stmt1, 1, session["id"])

ibm_db.execute(stmt1)

list2 = []

expense1 = ibm_db.fetch_tuple(stmt1)

while expense1:

    list2.append(expense1)

    expense1 = ibm_db.fetch_tuple(stmt1)

total = 0

for x in list2:

    total += x[4]


sql2 = "SELECT EXPLIMIT FROM LIMITS ORDER BY LIMITS.ID DESC LIMIT 1"

stmt2 = ibm_db.prepare(conn, sql2)

ibm_db.execute(stmt2)

limit = ibm_db.fetch_tuple(stmt2)

if len(limit) > 0 and total < limit[0]:

    sendEmail(session["email"])

```

```
return redirect("/display")

def sendEmail(reciver):

    sg = sendgrid.SendGridAPIClient(api_key=SENDGRID_API_KEY)

    # Change to your verified sender
    from_email = Email("nishanth.19cse@kongu.edu")

    # to_email = To(session["email"]) # Change to your recipient
    to_email = To(reciver) # Change to your recipient

    subject = "Expense Alert Limit"

    content = Content(

        "text/plain", "Dear User, You have exceeded the specified monthly
expense Limit!!!!")

    mail = Mail(from_email, to_email, subject, content)

    # Get a JSON-ready representation of the Mail object
    mail_json = mail.get()

    # Send an HTTP POST request to /mail/send
    response = sg.client.mail.send.post(request_body=mail_json)

    print(response.status_code)

    print(response.headers)

@app.route("/reward")
```

```

def reward():

    sql = "SELECT AMOUNT FROM EXPENSES WHERE USERID=? AND
MONTH(date)=MONTH (DATE (NOW ())) "

    stmt = ibm_db.prepare(conn, sql)

    ibm_db.bind_param(stmt, 1, session["id"])

    ibm_db.execute(stmt)

    fetchedamount = ibm_db.fetch_tuple(stmt)

    point = 0

    for i in fetchedamount:

        point = point + i

    sql1 = "SELECT EXPLIMIT FROM LIMITS WHERE USERID=?"

    stmt1 = ibm_db.prepare(conn, sql)

    ibm_db.bind_param(stmt1, 1, session["id"])

    limit = ibm_db.fetch_tuple(stmt)

    for i in limit:

        print(i)

        limit = i

    creditpoint = (point/10000)*100

    creditpoint = 100-creditpoint

    return render_template("reward.html",point=creditpoint)

@app.route("/display")

def display():

    print(session["username"], session["id"])

    sql = "SELECT * FROM EXPENSES WHERE USERID=?"

```

```
stmt = ibm_db.prepare(conn, sql)

ibm_db.bind_param(stmt, 1, session["id"])

ibm_db.execute(stmt)

list1 = []

row = ibm_db.fetch_tuple(stmt)

while row:

    list1.append(row)

    row = ibm_db.fetch_tuple(stmt)

print(*list1, sep="\n")

total = 0

t_food = 0

t_entertainment = 0

t_business = 0

t_rent = 0

t_EMI = 0

t_other = 0


for x in list1:

    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]


return render_template(

    "display.html",

    expense=list1,

    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,

)
```

```
# delete---the--data
```



```
@app.route("/delete/<string:id>", methods=["POST", "GET"])
```

```
def delete(id):
```

```
    print(id)
```

```
    sql = "DELETE FROM expenses WHERE id =?"
```

```
    stmt = ibm_db.prepare(conn, sql)
```

```
    ibm_db.bind_param(stmt, 1, id)
```

```
    ibm_db.execute(stmt)
```

```
    return redirect("/display")
```

```
# UPDATE---DATA
```

```
@app.route("/edit/<id>", methods=["POST", "GET"])
```

```
def edit(id):
```

```
    sql = "SELECT * FROM expenses WHERE id =?"
```

```
    stmt = ibm_db.prepare(conn, sql)
```

```
    ibm_db.bind_param(stmt, 1, id)
```

```
    ibm_db.execute(stmt)
```

```

row = ibm_db.fetch_tuple(stmt)

print(row)

return render_template("edit.html", expenses=row)

@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"]

        time = request.form["time"]

        sql = "UPDATE expenses SET date =? , expensename =? , amount =?,
paymentmode =?, category =?, time=? WHERE expenses.id =? "

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt, 1, date)

        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, time)

        ibm_db.bind_param(stmt, 7, 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"]

        sql = "INSERT INTO LIMITS (USERID, EXPLIMIT) VALUES (?, ?) "
```

```
stmt = ibm_db.prepare(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():
```

```
    sql = "SELECT EXPLIMIT FROM LIMITS ORDER BY LIMITS.ID DESC LIMIT 1"
```

```
    stmt = ibm_db.prepare(conn, sql)
```

```
    ibm_db.execute(stmt)
```

```
    row = ibm_db.fetch_tuple(stmt)
```

```
    return render_template("limit.html", y=row)
```

```
# REPORT
```

```
@app.route("/today")
```

```
def today():
```

```
sql = "SELECT * FROM expenses WHERE userid =? AND date = DATE(NOW())"

stmt = ibm_db.prepare(conn, sql)

ibm_db.bind_param(stmt, 1, session["id"])

ibm_db.execute(stmt)

list2 = []

texpanse = ibm_db.fetch_tuple(stmt)

print(texpanse)
```

```
sql = "SELECT * FROM EXPENSES WHERE USERID=? AND DATE(date) =  
DATE(NOW())"
```

```
stmt = ibm_db.prepare(conn, sql)

ibm_db.bind_param(stmt, 1, session["id"])

ibm_db.execute(stmt)

list1 = []

expense = ibm_db.fetch_tuple(stmt)

while expense:

    list1.append(expense)

    expense = ibm_db.fetch_tuple(stmt)
```

```
total = 0
```

```
t_food = 0
```

```
t_entertainment = 0
```

```
t_business = 0
```

```
t_rent = 0

t_EMI = 0

t_other = 0


for x in list1:

    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]
```

```
return render_template(

    "today.html",

    texpanse=list1,

    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():
```

```
    sql = "SELECT MONTHNAME (DATE),SUM(AMOUNT) FROM EXPENSES WHERE USERID=?  
GROUP BY MONTHNAME (DATE) "
```

```
    stmt = ibm_db.prepare(conn, sql)
```

```
    ibm_db.bind_param(stmt, 1, session["id"])
```

```
    ibm_db.execute(stmt)
```

```
    list2 = []
```

```
    texpanse = ibm_db.fetch_tuple(stmt)
```

```

while texpanse:

    list2.append(texpanse)

    texpanse = ibm_db.fetch_tuple(stmt)

print(list2)


sql      =      "SELECT      *      FROM      EXPENSES      WHERE      USERID=?      AND
MONTH(date)=MONTH (DATE (NOW ( ) ) ) "

stmt = ibm_db.prepare(conn, sql)

ibm_db.bind_param(stmt, 1, session["id"])

ibm_db.execute(stmt)

list1 = []

expense = ibm_db.fetch_tuple(stmt)

while expense:

    list1.append(expense)

    expense = ibm_db.fetch_tuple(stmt)


total = 0

t_food = 0

t_entertainment = 0

t_business = 0

t_rent = 0

t_EMI = 0

t_other = 0

```



```
for x in list1:

    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(

    "month.html",

    texpanse=list2,

    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():

    sql = (

        "SELECT YEAR (DATE),SUM (AMOUNT) FROM EXPENSES WHERE USERID=? GROUP

BY YEAR (DATE) "
```

```

)

stmt = ibm_db.prepare(conn, sql)

ibm_db.bind_param(stmt, 1, session["id"])

ibm_db.execute(stmt)

list2 = []

texpanse = ibm_db.fetch_tuple(stmt)

while texpanse:

    list2.append(texpanse)

    texpanse = ibm_db.fetch_tuple(stmt)

print(list2)


sql      =      "SELECT      *      FROM      EXPENSES      WHERE      USERID=?      AND
YEAR(date)=YEAR (DATE (NOW ())) "

stmt = ibm_db.prepare(conn, sql)

ibm_db.bind_param(stmt, 1, session["id"])

ibm_db.execute(stmt)

list1 = []

expense = ibm_db.fetch_tuple(stmt)

while expense:

    list1.append(expense)

    expense = ibm_db.fetch_tuple(stmt)


total = 0

t_food = 0

```

```
t_entertainment = 0

t_business = 0

t_rent = 0

t_EMI = 0

t_other = 0


for x in list1:

    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(

        "year.html",

        texpense=list2,

        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")


if __name__ == "__main__":

    app.run(debug=True)
```

## login.html

```
<!DOCTYPE html>

<html>

<head>

    <title>Login Form</title>
```

```
<link rel="stylesheet" type="text/css" href="../static/css/login.css">

<link
href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">

<script src="https://kit.fontawesome.com/a81368914c.js"></script>

<meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body >

    <!--  -->

    <div class="container">

        <div class="img">

            <div id="png"><a href="/" title="HOME"></a></div>

        </div>

        <div class="login-content">

            <form action='/login' method="POST">

                <div class="msg">{{ msg }}</div>

                
```

```
<h2 class="title">Login</h2>

<div class="input-div one">

    <div class="i">

        <i class="fas fa-user"></i>

    </div>

    <div class="div">

        <h5>Username</h5>

        <input type="text" name="username" class="input"
required>

    </div>

</div>

<div class="input-div pass">

    <div class="i">

        <i class="fas fa-lock"></i>

    </div>

    <div class="div">

        <h5>Password</h5>

        <input type="password" name="password"
class="input" required>

    </div>

</div>

<a href="#">Forgot Password?</a>

<input type="submit" class="btn" value="Login">
```



```

        <a href="/signup" class="btn" style="text-decoration: none;
padding-top: 15px; font-size: 12px; transition: none;"><center>I don't have
an account</center></a>

    </form>

</div>

</div>

</div>

<script type="text/javascript" src="../../static/js/login.js"></script>
</body>
</html>

```

## Display.html

```

{% extends 'base.html' %}

{% block body %}

<div class="container m-10">

    <h3 class="mt-3">EXPENSES</h3>

    {% if expense | length > 0 %}

    <table class="table">

        <thead class="thead-dark">

            <tr>

                <th scope="col">DateOfSpending</th>

```

```

        <th scope="col">Name</th>

        <th scope="col">Amount</th>

        <th scope="col">PaymentType</th>

        <th scope="col">Category</th>

        <th scope="col"></th>

        <th scope="col"></th>

    </tr>

</thead>

<tbody>

    {% for row in expense %}

    <tr>

        <td>{{row[2]}}</td>

        <td>{{row[3]}}</td>

        <td>{{row[4]}}</td>

        <td>{{row[5]}}</td>

        <td>{{row[6]}}</td>

        <td>

            <div class="col-md-1 mt-3">

                <a href="/edit/{{row[0]}}" class="btn btn-sm btn-
danger">Edit</a>

            </div>

        </td>

        <td>

            <div class="col-md-1 mt-3">

```

```

                <a href="/delete/{{row[0]}}" class="btn btn-sm btnDelete btn-
danger">Delete</a>

            </div>

        </td>

    </tr>

{% endfor %}

</tbody>

</table>

<!--when no DATA-Found-->

{% else %}

<div class="card shadow-sm mb-2 bg-white rounded"></div>

<div class="card-body">

    <div style="text-align: center ; font-family: monospace; color:red ; ">

        <h5><a href="/add"> ADD-DATA </a> to Display</h5>

    </div>

</div>

{% endif %}

<div class="container mt-10">

    <div class="row">

        <div class="col col-md-6">

            <h3 class="mt-5">Expense Breakdown</h3>

            <table class="table">

```

```
<thead class="thead-dark">

  <tr>

    <th scope="col-sm">#</th>

    <th scope="col-sm">Category</th>

    <th scope="col-sm">Amount</th>

  </tr>

</thead>

<tbody>

  <tr>

    <th scope="row">1</th>

    <td>Food</td>

    <td id="tfoot">{{t_food}}</td>

  </tr>

  <tr>

    <th scope="row">2</th>

    <td>Entertainment</td>

    <td id="tentertainment">{{t_entertainment}}</td>

  </tr>

  <tr>

    <th scope="row">3</th>

    <td>Business</td>

    <td id="tbusiness">{{t_business}}</td>

  </tr>

</tbody>
```

```
<tr>

    <th scope="row">4</th>

    <td>Rent</td>

    <td id="trent">{{t_rent}}</td>

</tr>

<tr>

    <th scope="row">5</th>

    <td>EMI</td>

    <td id="temi">{{ t_EMI }}</td>

</tr>

<tr>

    <th scope="row">6</th>

    <td>Other</td>

    <td id="tother">{{ t_other}}</td>

</tr>

<tr>

    <th scope="row-dark">7</th>

    <td>Total</td>

    <td>₹ {{total}}</td>

</tr>

</tbody>

</table>

</div>
```

```

<div class="col col-md-6">

  <canvas id="myChart" width="400" height="400"></canvas>

  <script>

    let food = document.getElementById('tfood').innerHTML

    let          entertainment          =
document.getElementById('tentertainment').innerHTML

    let business = document.getElementById('tbusiness').innerHTML

    let rent = document.getElementById('trent').innerHTML

    let emi = document.getElementById('temi').innerHTML

    let other = document.getElementById('tother').innerHTML

    var ctx = document.getElementById('myChart').getContext('2d');

    console.log("food ", food);

    var myChart = new Chart(ctx, {

      type: 'doughnut',

      data: {

        labels: ['Food', 'Entertainment', 'Business', 'Rent', 'EMI',
'Other'],

        datasets: [{

          label: 'Expenses Chart',

          data: [food, entertainment, business, rent, emi, other],

          backgroundColor: [

            'rgb(255, 99, 132)',

            'rgb(0, 0, 0)',

            'rgb(255, 205, 86)',

```

```
        'rgb(201, 203, 207)',  
  
        'rgb(54, 162, 235)',  
  
        'rgb(215, 159, 64)'  
    ],  
  
    }]  
  
    },  
  
    options: {  
  
        responsive: true,  
  
        plugins: {  
  
            legend: {  
  
                position: 'bottom',  
  
            },  
  
            title: {  
  
                display: true,  
  
                text: 'EXPENSE BREAKDOWN'  
  
            }  
  
        }  
  
    }  
  
});
```

```
</script>
```

```
        </div>

    </div>

</div>

</div>

{% endblock %}
```

## Add.py

```
{% extends 'base.html' %}

{% block body %}

<style>

    body{

        font-family: 'Lato', sans-serif;

    }

    .add{

        margin-top: 50px;

    }

</style>

<div class="container">

    <div class="row">

        <div class="col-md-6 add">
```



```
<h3>Add Expense</h3>

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

    <div class="form-group">

        <label for="">Date</label>

        <input class="form-control" type="date" name="date"
id="date"></div>

    <div class="form-group">

        <label for="">Time</label>

        <input class="form-control" type="time" name="time"
id="time"></div>

    <div class="form-group"> <label for="">Expense name</label>

        <input class="form-control" type="text"
name="expensename" id="expensename">

    </div>

    <div class="form-group">

        <label for="">Expense Amount</label>

        <input class="form-control" type="number" min="0"
name="amount" id="amount">

    </div>

    <div class="form-group">

        <label for=""></label>

        <select class="form-control" name="paymode"
id="paymode">
```

```

        <option selected hidden>Pay-Mode</option>

        <option name="cash" value="cash">cash</option>

        <option
value="debitcard">debitcard</option>
                                name="debitcard"

        <option
value="creditcard">creditcard</option>
                                name="creditcard"

        <option
value="epayment">epayment</option>
                                name="epayment"

        <option
value="onlinebanking">onlinebanking</option>
                                name="onlinebanking"

    </select>

```

```

    <div class="form-group">

        <label for=""></label>

        <select class="form-control" name="category"
id="category">

            <option selected hidden>Category</option>

            <option name = "food" value="food">food</option>

            <option name = "entertainment"
value="entertainment">Entertainment</option>

```

```

        <option name = "business" value="business">Business</option>

        <option name = "rent" value="rent">Rent</option>

        <option name = "EMI" value="EMI">EMI</option>

        <option name = "other" value="other">other</option>

    </select>

</div>

    <input class="btn btn-danger" type="submit" value="Add"
id="">

</form>

    <div style="position: relative; left: 590px; top: -460px;"
class="imagge">

    </div>

</div>

</div>

```

```
</div>
```

```
{% endblock %}
```

## Base.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
    <link                                rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.
css"                                integrity="sha384-
TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
crossorigin="anonymous">
```

```
    <script
src="https://cdn.jsdelivr.net/npm/chart.js@3.2.1/dist/chart.min.js"></scri
pt>
```

```
    <script                src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
```

```
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKA1Zap3H66lZ81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>
```

```
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>

<link
rel="stylesheet"
href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
integrity="sha384-
AYmEC3Yw5cVb3ZcuHtOA93w35dYTsvhLPVnYs9eStHfGJvOvKxVfELGroGkvsg+p"
crossorigin="anonymous"/>

<title>Personal Expense Tracker</title>

</head>

<body>

  <nav class="navbar sticky-top navbar-expand-lg navbar-dark"
style="background-color: rgba(220, 20, 60, 0.809);">

    <a class="navbar-brand" href="#">Personal Expense Tracker</a>

    <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarSupportedContent"
aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">

      <span class="navbar-toggler-icon"></span>

    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">

      <ul class="navbar-nav mr-auto">
```

```
<li class="nav-item active">

    <a class="nav-link" href="/home">Home <span class="sr-
only">(current)</span></a>

</li>

<li class="nav-item">

    <a class="nav-link" href="/add">Add</a>

</li>

<li class="nav-item">

    <a class="nav-link" href="/display">History</a>

</li>

<li class="nav-item">

    <a class="nav-link" href="/limit">Limits</a>

</li>

<li class="nav-item">

    <a class="nav-link" href="/reward">Points</a>

</li>

<li class="nav-item dropdown">

    <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">

        Report

    </a>

    <div class="dropdown-menu" aria-labelledby="navbarDropdown">

        <a class="dropdown-item" href="/today">Today</a>
```

```
        <a class="dropdown-item" href="/month">Month</a>

        <a class="dropdown-item" href="/year">Year</a>

    </div>

</li>

<li class="nav-item">

    <a class="nav-link" href="/logout">Logout</a>

</li>

</ul>

</div>

</nav>

{% block body %}

{% endblock %}

</body>

</html>
```

**GitHub Link:**

<https://github.com/IBM-EPBL/IBM-Project-11478-1659330364>

**PROJECT DEMO LINK:**

[https://drive.google.com/file/d/1gy\\_n9KI6MjvcXN0foxPsGE\\_kqc1E3eCo/view?usp=share link](https://drive.google.com/file/d/1gy_n9KI6MjvcXN0foxPsGE_kqc1E3eCo/view?usp=share_link)

**LIVE URL:**

<http://169.51.204.142:30675/>



