

Personal Expense Tracker Application

Team id:PNT2022TMID08585

SUBMITTED BY

VASANTH T S	727619BEC017
SURYA R	727619BEC015
ASHWINI S	727619BEC011
SRI ABINAYA S	727620BEC053

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

**Dr . MAHALINGAM COLLEGE OF ENGINEERING AND
TECHNOLOGY An Autonomous Institution Affiliated to
ANNA UNIVERSITY CHENNAI – 600 025**

CONTENT

- 1. INTRODUCTION**
 - 1.1 Project Overview
 - 1.2 Purpose
- 2. LITERATURE SURVEY**
 - 2.1 Existing problem
 - 2.2 References
 - 2.3 Problem Statement Definition
- 3. IDEATION & PROPOSED SOLUTION**
 - 3.1 Empathy Map Canvas
 - 3.2 Ideation & Brainstorming
 - 3.3 Proposed Solution
 - 3.4 Problem Solution fit
- 4. REQUIREMENT ANALYSIS**
 - 4.1 Functional requirement
 - 4.2 Non-Functional requirements
- 5. PROJECT DESIGN**
 - 5.1 Data Flow Diagrams
 - 5.2 Solution & Technical Architecture
 - 5.3 User Stories
- 6. PROJECT PLANNING & SCHEDULING**
 - 6.1 Sprint Planning & Estimation
 - 6.2 Sprint Delivery Schedule
 - 6.3 Reports from JIRA
- 7. CODING & SOLUTIONING**
 - 7.1 Feature 1
 - 7.2 Feature 2
 - 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**
- 13. APPENDIX**
 - 13.1 Source Code
 - 13.2 GitHub link
 - 13.3 Project Demo Link

1. INTRODUCTION

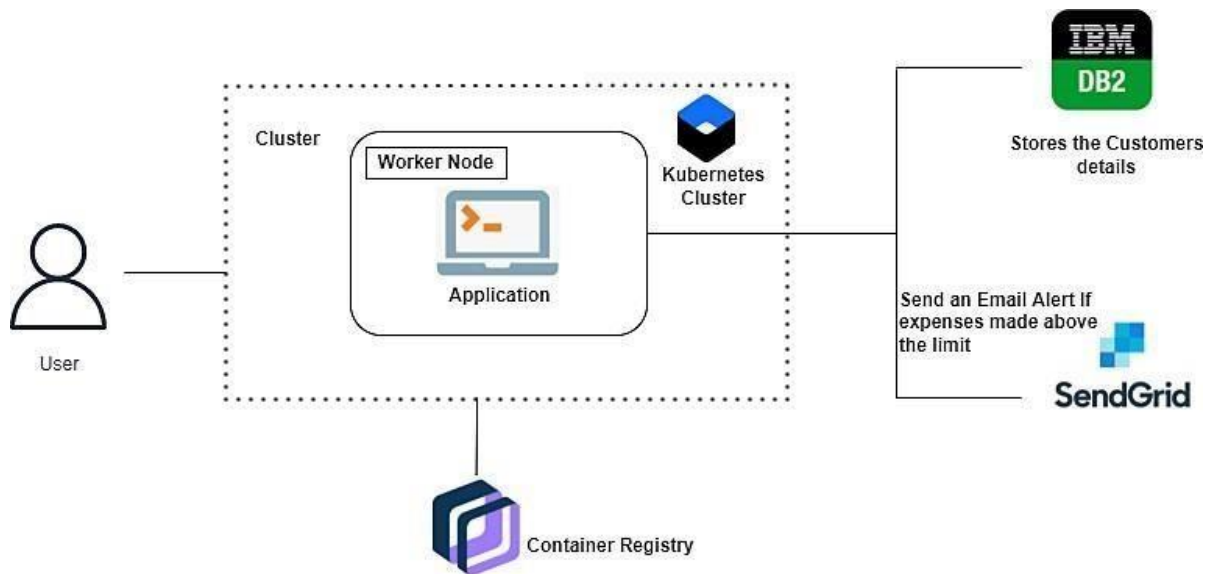
1.1 PROJECT OVERVIEW:

Personal Expense Tracker Application is a one kind of digital diary that helps to keep an eye on all of our money related transitions and also provides all financial activities report daily, weekly, monthly and yearly. As humans, we face many difficulties in our daily file. In our daily life, money is the most important portion and without it we cannot last one day on earth. But if we keep track on all financial data, then we can overcome this problem. The user can sustain all financial activities like digital automated dairy.

1.2 PROJECT PURPOSE:

- To find our daily expense and monthly expense and yearly expense.
- Maintain our money and save money by avoiding unwanted expenditures.
- Report is used to know our day-to-day life expenses.
- It helps to regulate our expenditures in daily life.

TECHNOLOGY STACK:



2. LITERATURE SURVEY

2.1 EXISTING PROBLEM:

2.1.1 MOBIWIK EXPENSE TRACKING APPLICATION:

Mobikwik came up with a new feature in their app called Expense Manager. With this feature, you can track and manage your expenditures (expenses), savings, reminders and bill payments. This is a personal budget management app that tracks your expenditures and income and gives you recommendations to make you economically strong. The main idea of developing this feature for giving users a clear picture that how much they are spending and where they are spending and when. We remind them to pay their utilities and card bills before the due date by using the same platform in just one tap, instead of going any other way. Also serving them by giving saving tips for their good future investment.

Reference : <https://blog.mobikwik.com/expense-manager-smart-manger/>

2.1.2 EXPENSE TRACKER:

Expense Tracker is a web application that allows you to track the daily expense of the user and help them to better manage their resources. It creates a digital record of the income and expense of the user. It inputs from the user an income, source of this income and the date of earning that income and creates a transaction entry under income category sums to the total amount of income and making real time changes. The various sources of income can be added and thus the distribution of your income is also illustrated by real time functioning charts that will keep updating as per your transactions. Similarly, it will also have an expense category where you can make similar transaction about the source of your expense, amount and date. On creating such transaction a different chart for distribution of expense will also be made in real time.

Reference : <https://nevonprojects.com/daily-expense-tracker-system/>

2.1.3 DAILY EXPENSE TRACKER:

Author : Shivam Mehra, Prabhat Parashar.

UG Student, Department of Computer Science and Engineering HMR Institute of Technology and Management, Delhi, India.

Daily Expense Tracker System is a system which will keep a track of Income-Expense of a House-Wife on a day-to-day basics, This System takes Income from House-Wife and divides in daily expense allowed, If you exceed that days' expense, it will cut it from your income and give new daily expense allowed amount, and if that days' expense is less it will add it in savings. Daily expense tracking System will generate report at the end of month to show Income-Expense Curve. It will let you add the savings amount, which you had saved for some particular festivals or day like birthday or anniversary.

Reference : <https://www.ijres.org/v9-i12.html>

2.1.4 A NOVEL EXPENSE TRACKER USING STATISTICAL ANALYSIS:

Author : Muskaan Sharma , Ayush Bansal , Dr. Raju Ranjan , Shivam Sethi

School of Computer Science and Engineering, Galgotias University.

In this system user can actually have a knowledge about their expenditure on their daily basis, weekly as well as monthly basis. This systematic way of storing your information related to your expenses would help you to keep a track of your expenditure and further you do not have to do the manual stuff. Some statistical analysis has to be done to be able to give users correct information on their expenses and help them spend better. This helps the society to prevent the issues like bankruptcy and save time from manual calculations. User can provide his/her income to calculate the total expense per day and the results will be stored for each individual user. People when usually go for trips with friends, can use this tracker to maintain their expense.

Reference : https://www.academia.edu/82849383/Daily_Expense_Tracker?f_r=483

2.1.5 EXPENDITURE MANAGEMENT SYSTEM

Author : V Geetha, G Nikhitha, H Sri Lasya³ Dr CK Gomathy, 2019

Expense Tracker is an everyday expense control application designed to track effortlessly and efficiently each day's costs. This helps us to get rid of the need of paper responsibilities that systematically maintains information. This device can be utilized by any individual to govern their income expenditure from each day to annual basis and to hold an eye on their spending, including the person to whom the payments were made and the purpose for the payment. On a weekly, monthly, and yearly basis, details of expenses will be displayed in the form of a pie chart. It aids us in remembering and adding information about what money we receive from others and what costs or payments we must make on a given date or month. We have categories in the expense tracker such as add expense, monthly expenses, add new expense, and so on. It gives the daily remainder about the savings we need to do.

Reference : <http://www.journaleca.com/gallery/jeca%20-%202654.pdf>

2.1.6 DAILY EXPENSE TRACKER:

Author : Karim, Md. Abdul; Orin, Taslima Yesmin

This project aims to create an easy, faster and smooth tracking system between the expense and the income. This project also offers some opportunities that will help the user to sustain all financial activities like digital automated diary. Most of the people cannot track their expense and income one way they face in money crisis, in this case daily expense tracker can help the people to tracking income-expense day to day and making life tension free.

Reference : <http://dspace.daffodilvarsity.edu.bd:8080/handle/123456789/4026>

2.1.7 DET – CLOUD BASED EXPENSE TRACKER

Author : Asthha Wahl, Muskan Aggarwal Galgotias University, 2021.

Cloud based Expense Tracker aims to help everyone who are planning to know their expenses and save from it. DET is an android app which users can execute in their mobile phones and update their daily expenses so that they are well known to their expenses. Here user can define their own categories for expense type like food, clothing, rent and bills where they have to enter the money that has been spent and also can add some information in additional information to specify the expense. User can also define expense categories. User will be able to see pie chart of expense.

Reference:

https://orbi.uliege.be/bitstream/2268/289183/1/WORLD%20WOMEN%20CONFERENCE-IV_Kad%C4%B1n-Kitap.pdf

2.1.8 ONLINE INCOME AND EXPENSE TRACKER:

Author : S Chandini, T Poojitha, D Ranjith, VJ Mohammed Akram, MS Vani, V Rajyalakshmi

International Research Journal of Engineering and Technology (IRJET) 6 (3), 2395-0056, 2019

Income and Expense Tracker will maintain data of daily, weekly, monthly, yearly expenses, Manages your expenses and earnings in a simple and intuitive way. User can select 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. User can view and sort expense as per weekly, monthly, yearly. By using this, it can reduce the manual calculations for their expenses and keep the track of the expenditure. This will display graph as per selected view.

Reference : <https://www.irjet.net/archives/V6/i3/IRJET-V6I31110.pdf>

2.1.9 PYTHON BASED EXPENSE TRACKERS:

In this python 5curr project, we will create an expense tracker that will take details of our expenses. While filling the signup form a person will also need to fill in the details about the income and the amount he/she wants to save. Some people earn on a daily basis, so their income can also be added on a regular basis. Details of expenses will be shown in the form of a pie chart on a weekly, monthly, and yearly basis. Installation of django is a must to start with the Expense Tracker project.

Reference : <https://data-flair.training/blogs/expense-tracker-python>

2.1.10 XPENSTRAK:

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. 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. An analysis comparing existing expense tracking software with the one being introduced is provided.

Reference: <https://digitallibrary.sdsu.edu/islandora/object/sdsu%3A3676/datastream/OBJ/view>

2.2 PROBLEM STATEMENT:

Personal finance entails all the financial decisions and activities that a Finance app makes life easier by helping the user to manage their finances efficiently. A personal finance app will not only help them with budgeting and accounting but also give helpful insights about money management.

Who does the problem affect?	User, Working people.
What are the boundaries of the problem?	Tracking Budget
What is the issue?	In this world, people spend lots of money than earning it. Most of the time they spend more money on unwanted things. By this activity they are facing so much struggles to run their family at the end of the month or year. By solving this issue, an application which is used to add the expenses of a user and spend money according to that plan. If a user spend additional money, this application notify them through their mail. Also by developing this application financial issue in a family will not be arise anymore.

When does the issue occur?	Financial issue occur when people spend lots of money on buying unwanted stuffs and things. They do not have any plan to spend it. This is the problem which will affect all types of people and their surrounding as well. When they do not have a plan, they did not spend the money in a proper way.
Where is the issue occurring?	This issue occur among people who earns money.
Why is it important that we fix the problem?	When people have an efficient plan to spend money in a proper way, the financial problem issue will be solve.

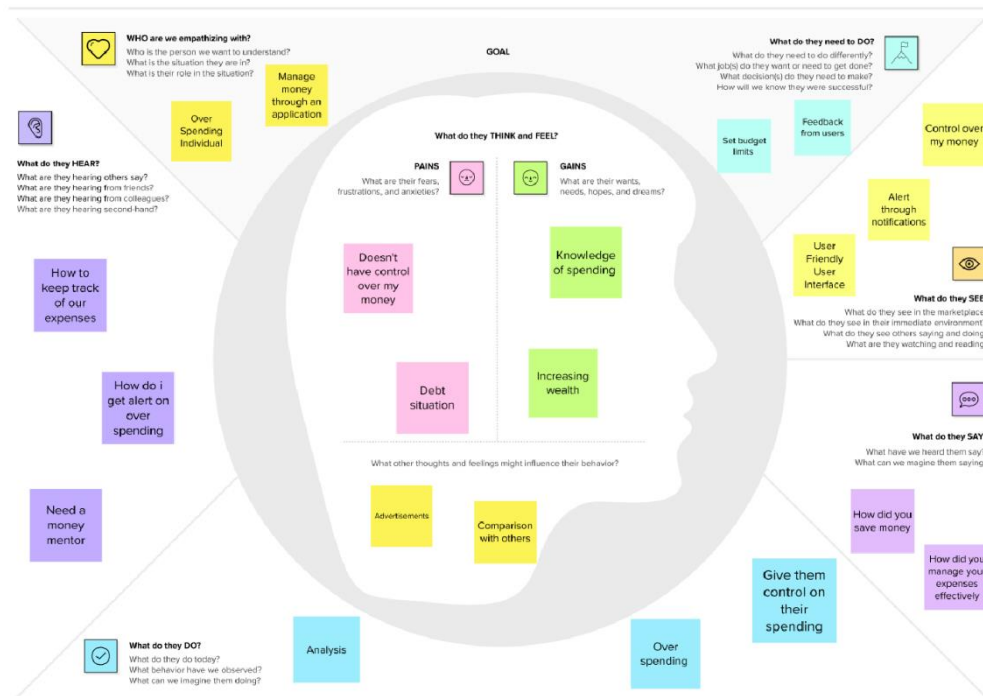
3.

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS:

An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. An Empathy Map consists of four quadrants. The four quadrants reflect four key traits, which the user demonstrated/possessed during the observation/research stage. The four quadrants refer to what the user: Said, Did, Thought, and Felt. It's fairly easy to determine what the user said and did.

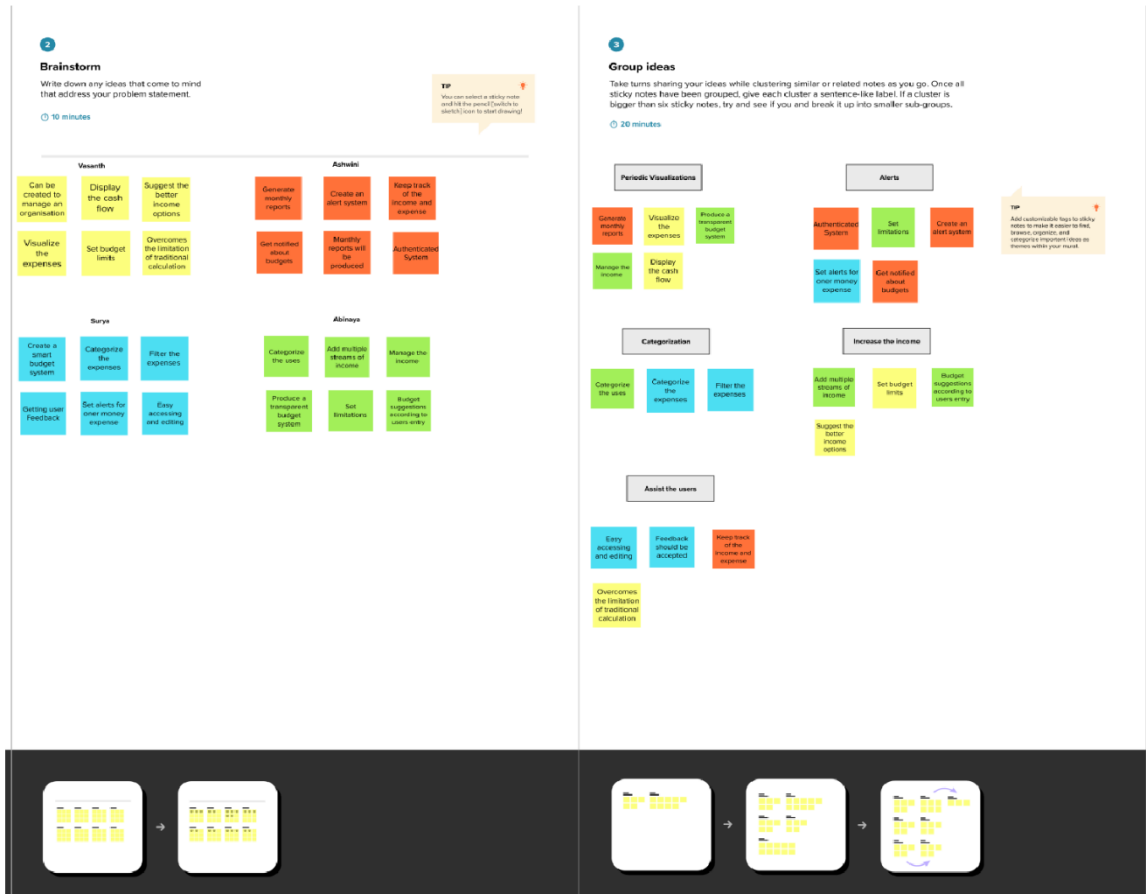
Personal Expense Tracker Application:



3.3 IDEATION & BRAINSTROMING:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Step-2: Brainstorm, Idea Listing and Grouping



3.3 PROPOSED SOLUTION:

Proposed Solution means the technical solution to be provided by the Implementation agency in response to the requirements and the objectives of the Project.

The main goal of presenting a business proposal is to provide solution to a problem faced by a potential buyer. This section should be as comprehensive as possible, and able to address all the needs that you have pointed in the first section.

S.No	Parameter	Description
------	-----------	-------------

1	Problem Statement (Problem to be solved)	Keep track of personal incomes and expenses. In an user friendly way, monitor the cash flow get alerts when expenses exceeds.
2	Idea / Solution Description	App backed with IBM Cloud sends notifications and alert if the expense exceeds. Users Login verified and based on the users expense it will show the graph for the users.
3	Novelty / Uniqueness	Splits are made for the expenses of the users like food,education,personal etc. Giving points based on their maintainance of their expenses.
4	Social Impact / Customer Satisfaction	Improves the quality of users spending expense which results in better economic growth.
5	Business Model (Revenue Model)	Record of transactions are securely maintained and advertisements on the premium features that removes the advertisement for the premium users.
6	Scalability of the Solution	The app provides optimized result. Using Kubernetes the docker manages the containers and create new pods whenever the traffic increases.

3.4 PROPOSED SOLUTION FIT:

Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem. The Problem-Solution Fit is an important step towards the Product-Market Fit, but often an underestimated one. Problem-Solution canvas is a tool for entrepreneurs, marketers and corporate innovators, which helps them identify solutions with higher chances for solution adoption, reduce time spent on solution testing and get a better overview of current situation.

Project Title: Personal Expense Tracker Application

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMID08585

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) CS Who is your customer? i.e. working parents or 2-3 y.o. kids 1. People who wants to maintain their expense 2. People who needs more savings 3. Persons who have small scale industries	4. CUSTOMER CONSTRAINTS CC What concerns prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, weather, etc. 1. Doubts in reliability of the solution due to lots of scam 2. Insufficient knowledge.	5. AVAILABLE SOLUTIONS AS Which solutions are available to the customers when they face the problem? or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notebook. 1. Easy evaluation of the budget. 2. Control over the budget. 3. Alerts when exceeds the limits.	Explore AS, differentiate
	2. JOBS-TO-BE-DONE / PROBLEMS JB/P Which jobs-to-be-done (or problems) do you address for your customer? There could be more than one, explore different sides. 1. User data security 2. Alert at right time 3. Comprehensive User Interface	9. PROBLEM ROOT CAUSE RC What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in requirements. 1. Skepticism about the privacy. 2. Lack of proper input. 3. Lack of knowledge	7. BEHAVIOUR BE What does your customer do to address the problem and get the job done? i.e. directly related: find the right color panel settings, consider usage and benefits; indirectly associated: customers spend less time on consuming work (i.e. Greenpeace) 1. Give proper input in the application. 2. Gain sufficient knowledge.	
Identify strong TR & EM	3. TRIGGERS TR What triggers customer to act? i.e. seeing their neighbor's smartphone upgrade, reading about a more efficient solution on the net. 1. Comparison With Others 2. Lack of Budgeting Knowledge	10. YOUR SOLUTION SI If you are working on existing features, write down your current solution first. Fill in the matrix, and think how much it fits reality. If you are working on a new business proposition, then keep it blank until you fill other corners and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour. The main purpose of the application is to, <ul style="list-style-type: none"> Track the expense and savings of the user on monthly basis. Based on the analysis, improve the budget management of the user. 	8. CHANNELS of BEHAVIOUR CH ONLINE What kind of actions do customers take online? Extract online channels from #7. OFFLINE What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development. 1. User can share the reports to others. 2. Inspect the expenses and plan for future	Identify strong TR & EM
	4. EMOTIONS: BEFORE / AFTER EM When do customers feel what they feel a problem or a job and afterwards? i.e. lost, accurate > confident, as needed, use it as your communication strategy & design. • Before using this application user don't have any budget control and get frustrated when he/she realize his/her mistake • After using this application user have the knowledge to manage budget so he/she feels happy and confident.			

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT:

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in use cases. Functional requirements drive the application architecture of a system, while non- functional requirements drive the technical architecture of a system.

FUNCTIONAL REQUIREMENT ANALYSIS:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	User Confirmation	Confirmation via Email/ Confirmation via OTP
FR-3	User Income Data(Monthly)	Data to be entered in the app
FR-4	User Expense Data(Monthly)	Data to be entered in the app
FR-5	User Budget limits	Budget Limit data set by the user
FR-6	Alerts	Alert in the application/ Alert through E-mail

4.2 NON-FUNCTIONAL REQUIREMENTS:

Non-functional requirements are often mistakenly called the "quality attributes" of a system, however there is a distinction between the two. Non-functional requirements are the criteria for evaluating how a software system should perform and a software system must have certain quality attributes in order to meet non-functional requirements.

NON-FUNCTIONAL REQUIREMENT ANALYSIS:

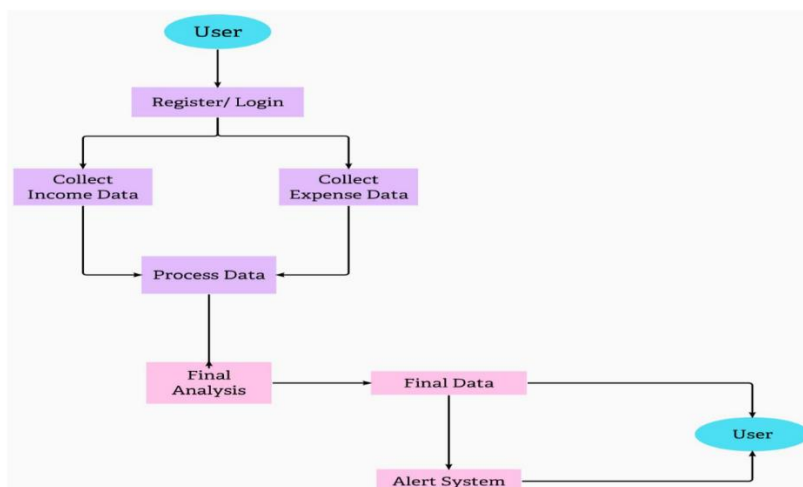
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Efficiency, Effective User Interface, Productivity, User Satisfaction.
NFR-2	Security	Authentication, Verification, Data security.
NFR-3	Reliability	Product will perform its intended function for a specified time period in all conditions.
NFR-4	Performance	Smooth Functioning of the application and effective response to the user.
NFR-5	Availability	Application will function properly under various circumstances.
NFR-6	Scalability	Handle multiple users.

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAM:

A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops.

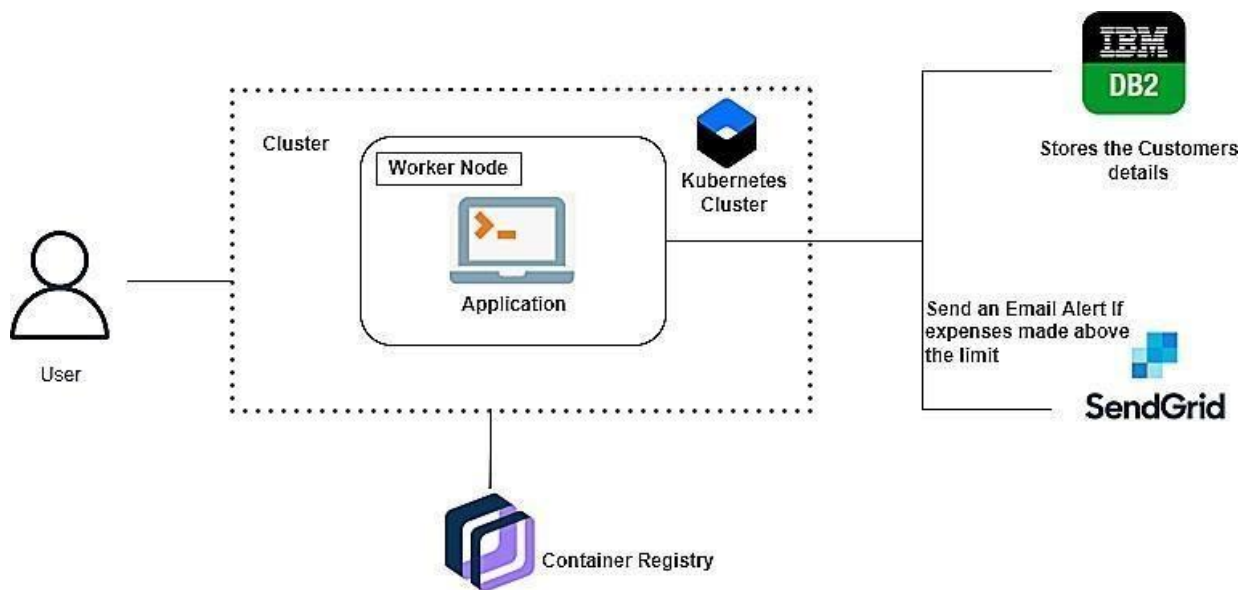
DATAFLOW OF PERSONAL EXPENSE TRACKER APPLICATION:



5.2 SOLUTION & TECHNICAL ARCHITECTURE:

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

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



5.3 USER STORIES:

A user story is an informal, general explanation of a software feature written from the perspective of the end user or customer. The purpose of a user story is to articulate how a piece of work will deliver a particular value back to the customer.

In software development and product management, a user story is an informal, natural language description of features of a software system.

User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (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 create my account in the application	High	Sprint-1
	Login	USN-2	As a user, I can log into the application by entering email & password	I can access my dashboard	High	Sprint-1
	Add/ Remove Income Details	USN-3	As a user, I can add/delete my income details	I can modify the income details	High	Sprint-2
	Add/ Remove Expense Details	USN-4	As a user, I can add/delete my expense details	I can modify the expense details	High	Sprint-2
	View Budgets	USN-5	As a user, I can view my expenses	Visualization of budget details	Medium	Sprint-3
	Alert System	USN-6	As a user, I get notified about budget limits if I exceed.	Receiving alerts about budgets and act accordingly	Medium	Sprint-4

6.**PROJECT PLANNING & SCHEDULING****6.1 SPRINT PLANNING AND ESTIMATION:**

In Scrum Projects, Estimation is done by the entire team during Sprint Planning Meeting. The objective of the Estimation would be to consider the User Stories for the Sprint by Priority and by the Ability of the team to deliver during the Time Box of the Sprint.

Sprint	Functional Requirement (Epic)	User story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Panel	USN-1	The user shall access the website and view the products list provided after registration	20	High	Vasanth T S Surya R Ashwini S Sri Abinaya S
Sprint-2	Admin panel	USN-2	The Administrator's task is to look over the user entries and monitor the people's information	20	High	Vasanth T S Surya R Ashwini S Sri Abinaya S
Sprint-3	Application features	USN-3	The user shall experience the additional features such as over expense alert system and budget recommendation system	20	High	Vasanth T S Surya R Ashwini S Sri Abinaya S
Sprint-4	Final Delivery	USN-4	Container of applications using docker Kubernetes and deploy the application. Create the documentation and finally submit the application	20	High	Vasanth T S Surya R Ashwini S Sri Abinaya S

6.2 SPRINT DELIVERY SCHEDULE :

Project Tracker, Velocity & Burndown Chart: (4 Marks)

SPRINTS	TOTAL STORY POINTS	DURATION	SPRINT START DATE	SPRINT END DATE	STORY POINTS COMPLETED	SPRINT RELEASE DATE
SPRINT – 1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
SPRINT – 2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
SPRINT – 3	20	6 Days	07 Oct 2022	12 Nov 2022	20	12 Nov 2022
SPRINT – 4	20	6 Days	14 Nov 2022	19 Oct 2022	20	19 Nov 2022

Average Velocity = $20 / 6 = 3.33$

6.3 REPORTS FROM JIRA:

The reports in jira has been denoted below:

BACKLOG:

Backlog is usually a list of issues describing what your team is going to do on a project. It's a convenient place for creating, storing, and managing several kinds of issues: issues that you're currently working on (you can also see them on the board and in the current sprint if you're using a Scrum project).

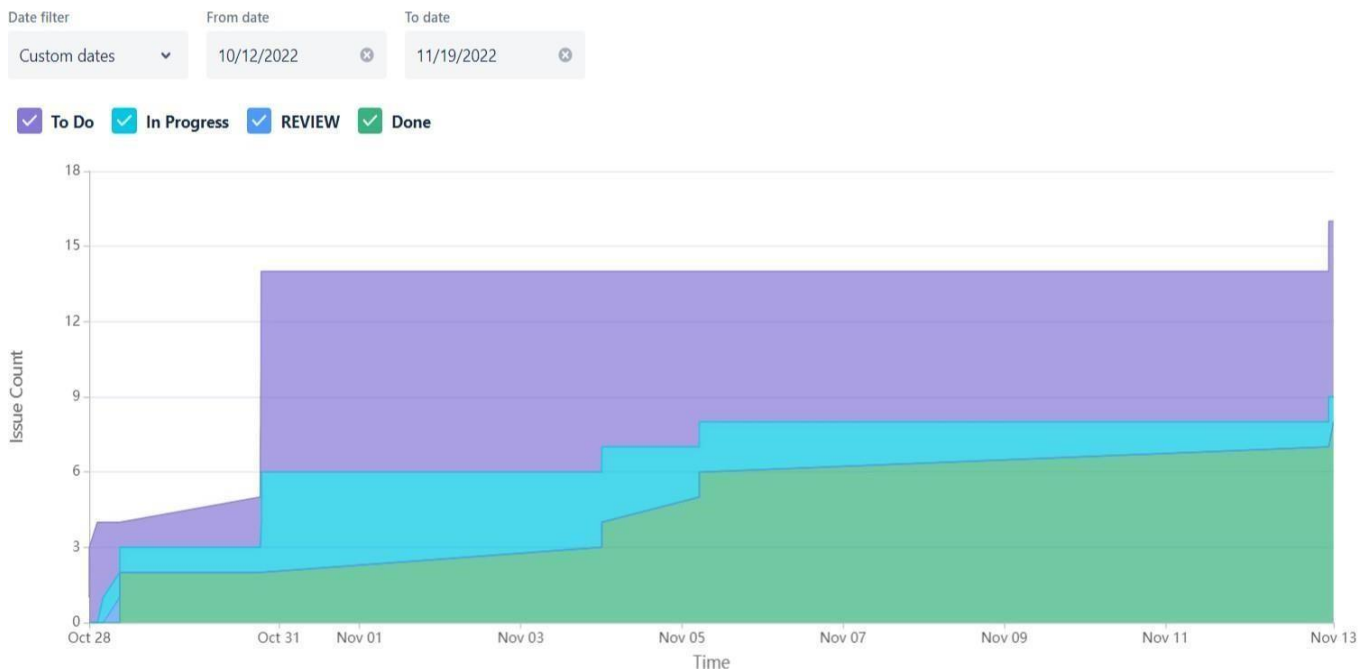
BOARD:

A board displays your team's work as cards you can move between columns. In Jira Software, cards and the tasks they represent are called "issues". Usually, your board reflects your team's process, tracking the status of work as it makes its way through your team's process.

CUMMULATIVE FLOW DIAGRAM:

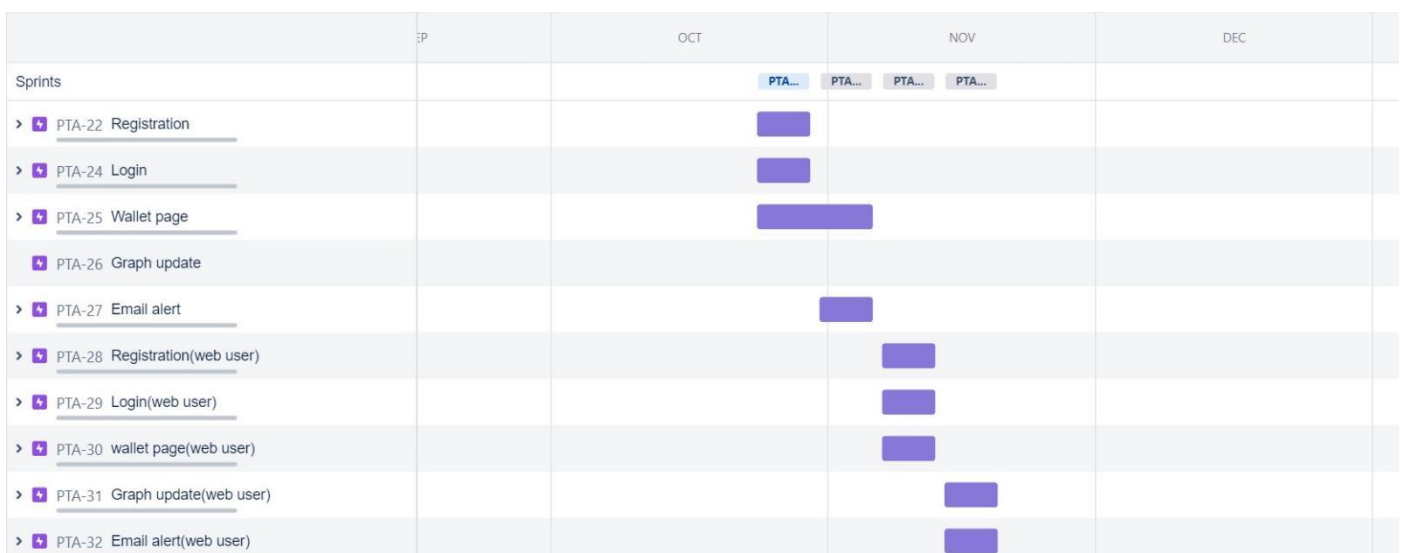
A Cumulative Flow Diagram (CFD) is an area chart that shows the various statuses of work items for an application, version, or sprint. The horizontal x-axis in a CFD indicates time, and the vertical y-axis indicates cards (issues).

Cumulative flow diagram

[How to read this report](#)


ROAD MAP:

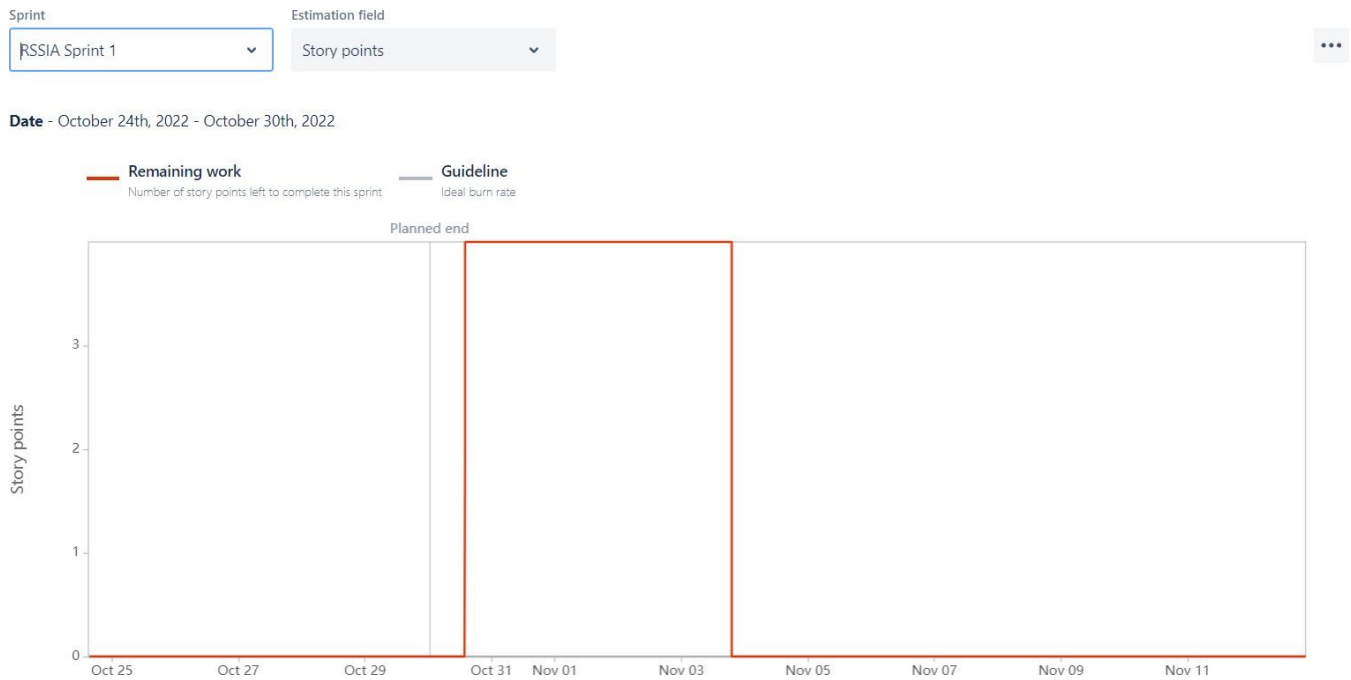
Roadmaps in Jira Software are team-level roadmaps useful for planning large pieces of work several months in advance at the Epic level within a single project. Simple planning and dependency management features help your teams visualize and manage work better together.



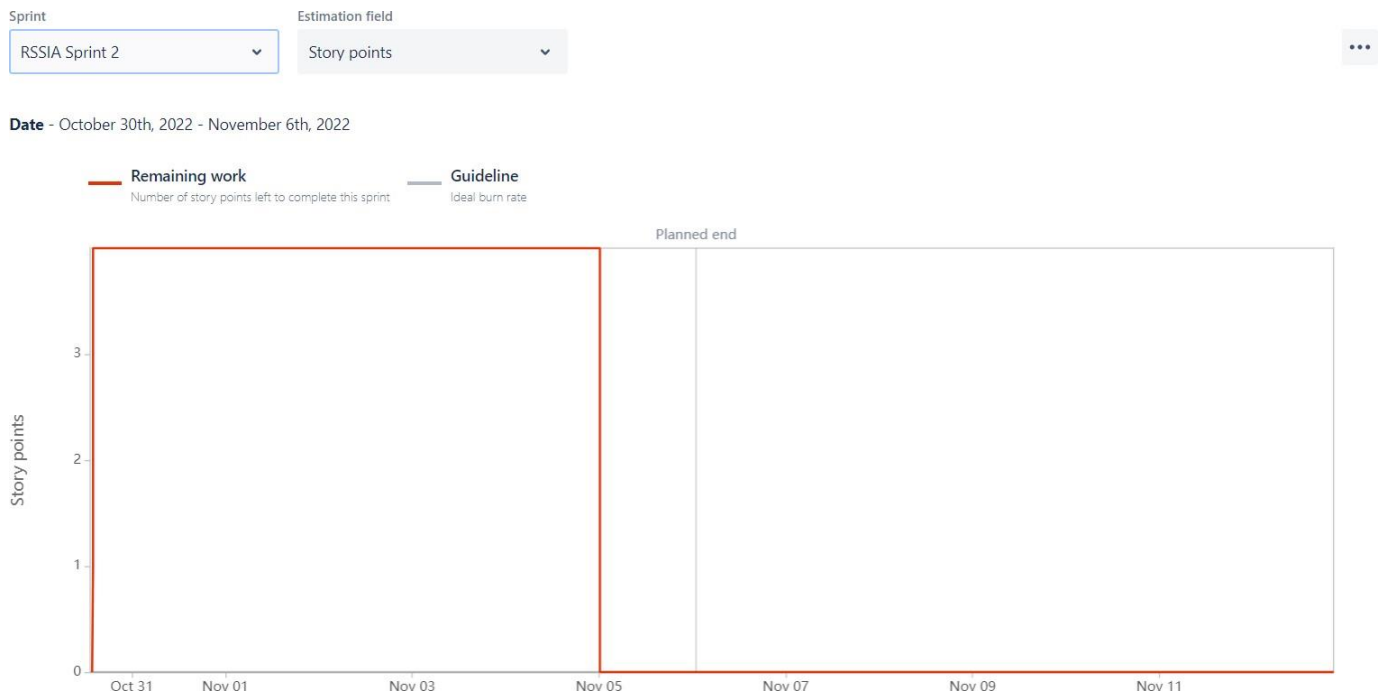
SPRINT BURNDOWN CHART:

A burndown chart shows the amount of work that has been completed in an epic or sprint, and the total work remaining. Burndown charts are used to predict your team's likelihood of completing their work in the time available.

SPRINT 1:



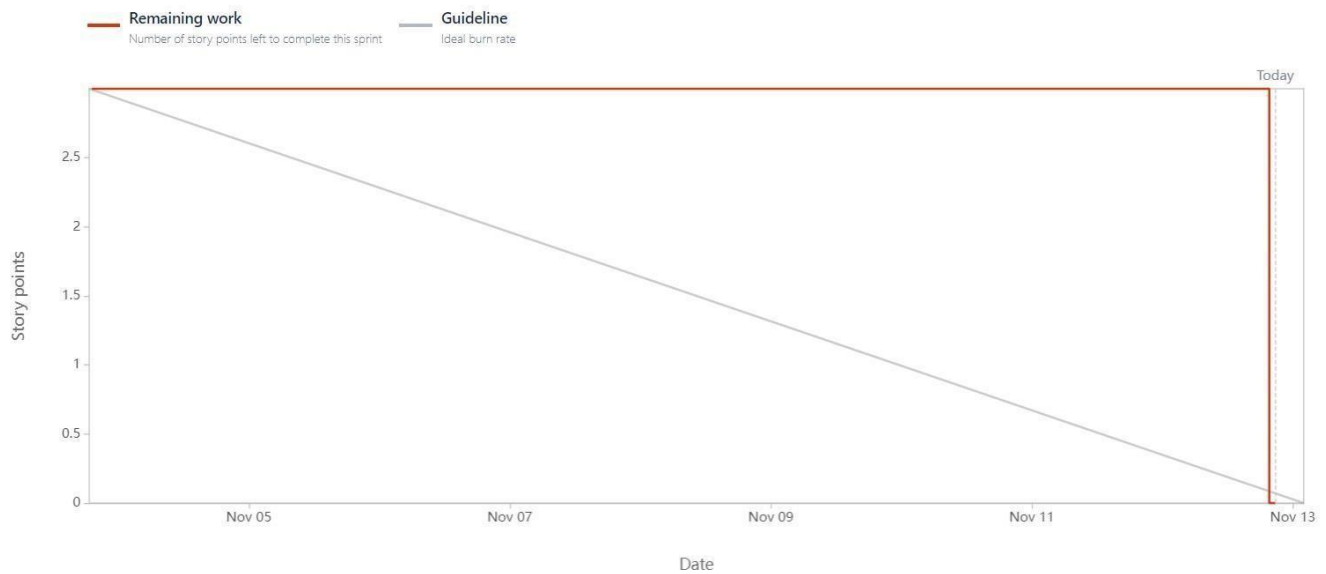
SPRINT 2:



SPRINT 3:

Sprint Estimation field ...

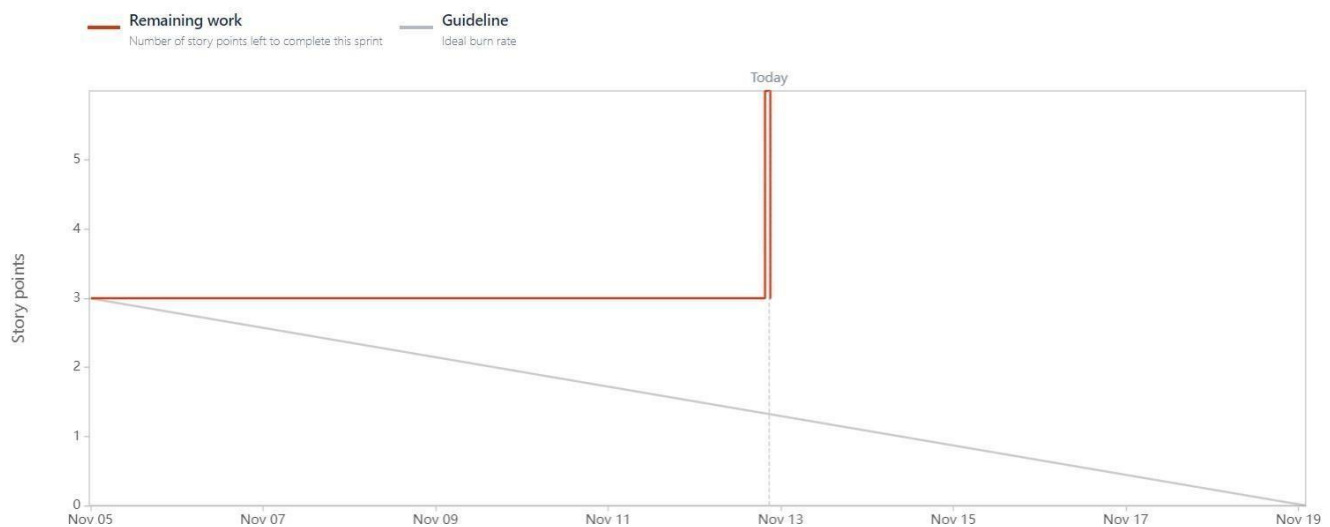
Date - November 3rd, 2022 - November 13th, 2022



SPRINT 2:

Sprint Estimation field ...

Date - November 4th, 2022 - November 19th, 2022



7. CODING & SOLUTIONING

7.1 FEATURES:

DAILY REPORT GENERATOR:

1.today.html

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

```
{% block body %}
```

```
<div class="container ">
```

```
<div class="row">
```

```
<div class="col-md-5">
```

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

```
<div class="card shadow mb-2 bg-white rounded-pill">
```

```
<div class="card-body ">
```

```
<div class="row">
```

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

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

```
</div>
```

```
</div>
```

```
</div>
```

```
{% for row in texpense %}
```

```
<div class="card shadow mb-2 bg-white rounded-bottom">
```

```
<div class="card-body ">
```

```
<div class="row">
```

```
<div id ="ttime" class="col-md-6">{{row [0]}}</div>
```

```
<div id="tamount" class="col-md-6"> {{row[1] }} </div> </div>
```

```
</div>
```

```
</div>
```

```
{% endfor %}
```

```
</div>
```

```
</div>
```

```
<section>
```

```
<div class="row">
```

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

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

```
<div class="card shadow mb-2 bg-white rounded-bottom">
```

```
<div class="card-body ">
```

```
<div class="row">
```

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

```
<div id="tfood" class="col-md-6"> {{ t_food}} </div> </div>
```

```

    </div>
</div>
<div class="card shadow mb-2 bg-white rounded">
  <div class="card-body">
    <div class="row">
      <div class="col-md-6">Entertainment</div>
      <div id="tentertainment" class="col-md-6"> {{ t_entertainment }} </div> </div>
    </div>
  </div>
</div>
<div class="card shadow mb-2 bg-white rounded">
  <div class="card-body">
    <div class="row">
      <div class="col-md-6">Business</div>
      <div id="tbusiness" class="col-md-6"> {{ t_business }} </div> </div>
    </div>
  </div>
</div>
<div class="card shadow mb-2 bg-white rounded">
  <div class="card-body">
    <div class="row">
      <div class="col-md-6">Rent</div>
      <div id="trent" class="col-md-6"> {{ t_rent }} </div>
    </div>
  </div>
</div>
<div class="card shadow mb-2 bg-white rounded">
  <div class="card-body">
    <div class="row">
      <div class="col-md-6">EMI</div>
      <div id="temi" class="col-md-6"> {{ t_EMI }} </div>
    </div>
  </div>
</div>
<div class="card shadow mb-2 bg-white rounded">
  <div class="card-body">
    <div class="row">
      <div class="col-md-6">Other</div>
      <div id="tother" class="col-md-6"> {{ t_other }} </div> </div>
    </div>
  </div>
</div>
<div class="card shadow mb-2 btn-outline-danger rounded-pill">

```

```

<div class="card-body">
  <div class="row">
    <div class="col-md-6">Total</div>
    <div class="col-md-6">₹ {{total}} </div>
  </div>
</div>
</div>
<div class="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'); 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,

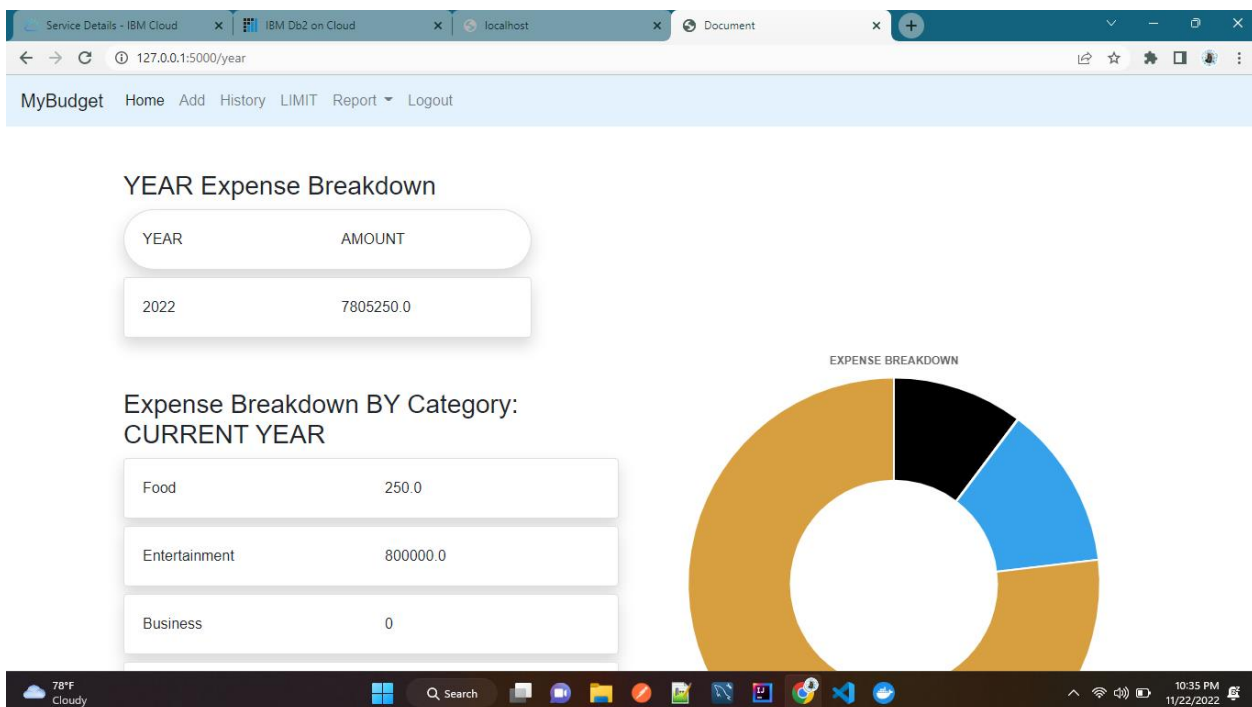
```

```

    text: 'EXPENSE BREAKDOWN'
  }
}
});
</script>
</div>
</div>
</div>
</section>
</div> {%
endblock %}

limit.html:
{% extends 'base.html' %}
{% block body %}
<p> Currently your MONTHLY limit is ₹ {{y}} </p>
<form action="/limitnum" method="POST">
<p> ENTER the MONTHLY LIMIT to avoid over EXPENSES</p> <br/>
  <input type="number" name="number" required/> <span>
    <button class="btn btn-warning" type="submit">ENTER</button>
  </span></form>{% endblock %}

```

OUTPUT:**1. Yearly report:****2. Limit setup:**

7.2 FEATURE :

CODE :

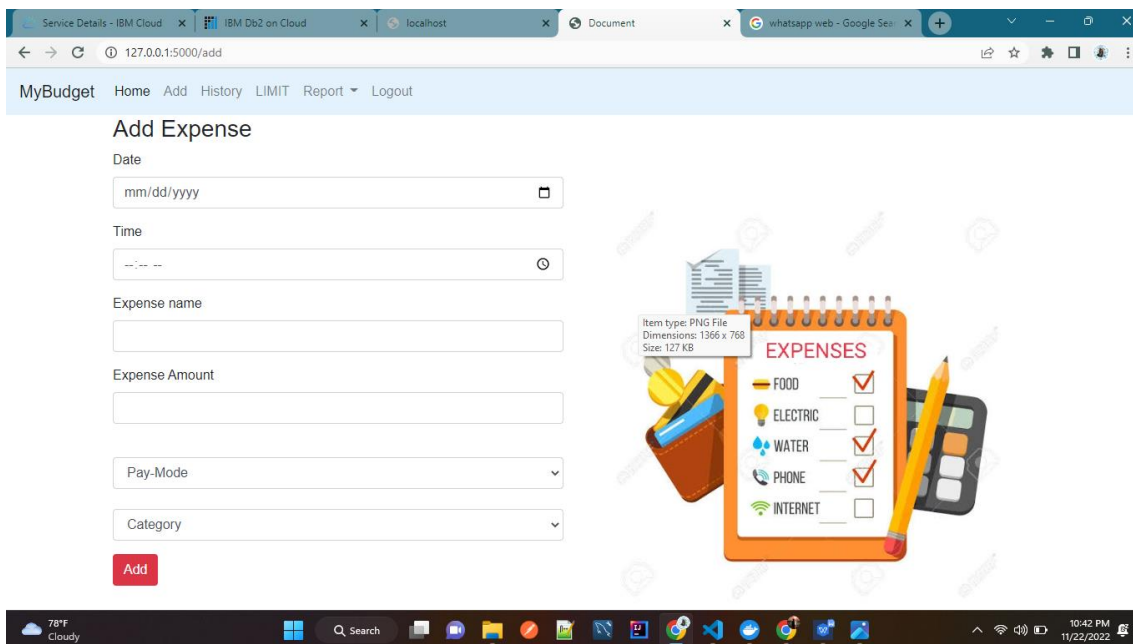
```
addexpense.html:      {%
extends 'base.html' %}
{% block body %}
<div class="container">
  <div class="row">
    <div class="col-md-6">
      <h3>Add Expense</h3>
      <form action="/addexpense" method="POST"> <div
        class="form-group">
          <label for="">Date</label>
          <input class="form-control" type="datetime-local" name="date" id="date"></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 name="debitcard" value="debitcard">debitcard</option>
              <option name="creditcard" value="creditcard">creditcard</option>
              <option name="epayment" value="epayment">epayment</option>
              <option name="onlinebanking" value="onlinebanking">onlinebanking</option> </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="image">
    
</div>
</div>
</div>
</div>
{% endblock %}

```

OUTPUT:


The screenshot displays the 'Add Expense' form in the MyBudget application. The form is located on the left side of the page, with a navigation bar at the top containing 'MyBudget', 'Home', 'Add', 'History', 'LIMIT', 'Report', and 'Logout'. The form fields are: Date (mm/dd/yyyy), Time (hh:mm), Expense name, Expense Amount, Pay-Mode (dropdown), and Category (dropdown). A red 'Add' button is at the bottom of the form. On the right side, there is a decorative illustration of a notepad titled 'EXPENSES' with a checklist of categories: FOOD, ELECTRIC, WATER, PHONE, and INTERNET. The notepad also shows a pencil and a calculator.

7.3 DATABASE SCHEMA :

Service Details - IBM Cloud | IBM Db2 on Cloud | localhost | Register

bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2Fd15e170551874da5bed665c72dff6...

IBM Db2 on Cloud

Data objects

KTZ09482

Tables

EXPENSES

LIMITS

REGISTERUSER

Views

MQTs

Aliases

Nicknames

* bludb

History

Results

Result set 1

Filter table

ID	USERID	DATE	EXPENSENAME	AMOUNT	PAYMENTMODE	CATEGORY	TIME
2	1	2022-11-19	Dinner	250.0	epayment	food	23:37:00
4	3	2022-11-20	Breakfast	55.0	cash	food	09:10:00

78°F Cloudy 10:17 PM 11/22/2022

Service Details - IBM Cloud | IBM Db2 on Cloud | localhost | Register

bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2Fd15e170551874da5bed665c72dff6...

IBM Db2 on Cloud

Data objects

KTZ09482

Tables

EXPENSES

LIMITS

REGISTERUSER

Views

MQTs

Aliases

Nicknames

* bludb

History

Results

Result set 1

Filter table

1

select * from limits;

Run selected

ID	USERID	EXPLIMIT
2	1	10000.0
3	3	4000.0

Service Details - IBM Cloud | IBM Db2 on Cloud | localhost | Register

bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2Fd15e170551874da5bed665c72dff6...

IBM Db2 on Cloud

Data objects

KTZ09482

Tables

EXPENSES

LIMITS

REGISTERUSER

Views

MQTs

Aliases

Nicknames

* bludb

History

Results

Result set 1

Filter table

Run selected

ID	USERNAME	PASSWORD	EMAIL
1	Surya	Surya@2003	jayasuryar2003@gmail.com
2	Vasanth	tsv007	vasanthvvb0710@gmail.com
3	Nandhish	nandhish0012	nandhishkumar0012@gmail.com

78°F Mostly cloudy 10:16 PM 11/22/2022

- To check whether the user is registered or not.
- To check the login whether its login only if the data is correct.
- To check the username is already exist or not.
- To check whether an register user cannot register themselves as a new user.
- To check the user can add their expense in the add session.
- All users can see their expense history and the data is correct or not.
- Verify that user can see their expense report with pie chart.
- Verify all categories are working normal.
- Check that the validations are working normal In input session.

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)	BUG ID	Executed By
Testcase_001	Functional	Login Page	To check whether the user is registered or not	Checks whether the logged in username is registered in backend.	1.Enter your username 2.Enter your password 3.click signin button	username: test password:test@123	Homepage should display	Working as expected	Pass		N		saravanan s	
Testcase_002	Functional	Login Page	To check the login whether its login only if the data is correct	Checks whether the logged in username is not registered in backend.	1.Enter your username 2.Enter your password 3.click signin button	username: test password:test@123	Homepage will not display	Working as expected	pass		N		Philomina s	
testcase_003	Functional	register page	To check the username is already exist or not.	The details given by the user is stored in backend	1.Enter your username 2.Enter your email 3.Enter your password 4.Enter your confirm password 5.Click on signup button	User Input	If it already present it popup message	working as expected	pass		N		Saravanan S	
testcase_004	Functional	Register page	to check whether an register user cannot register themself as an new user.	checks the user name is present in the database.	1.Enter your username 2.Enter your email 3.Enter your password 4.Enter your confirm password 5.Click on signup button	username: test password:test@123	User will not be able to access to login page	working as expected	pass		N		Tharani devi R	
Testcase_005	Functional	Add Page	To check the user can add their expense in the add session	whether it's add or not.	1.Enter your username 2.Enter your password 3.click signin button 4.add button	username: test password:test@123	successfully add and go to history page.	Working as expected	Pass		N		surya priya s	
Testcase_006	Functional	history Page	All users can see their expense history and the data is correct or not.	Retrieve data from database that suitable for particular user or not.	1.Enter your username 2.Enter your password 3.click signin button 4.add button	username: test password:test@123	To display all the expense in history tab.	Working as expected	Pass		N		saravanan s	
Testcase_007	Functional	limit Page	Verify that user can update their limit for monthly expense.	Update database with recently added limit	1.Enter your username 2.Enter your password 3.click signin button 4.Go to limit page	username: test password:test@123	user can update there limit.	working as expected	Pass		N		tharani devi r	
Testcase_008	Functional	Report page	verify that user can see their expense report with pie chart	updated database for retrieve data from expense	1.Enter your username 2.Enter your password 3.click signin button 4.Go to limit page 5. click report and today	username: test password:test@123	report can be seen	Working as expected	Pass		N		saravanan s	
Testcase_009	UI	report page	verify all categories are working normal	check the results are in correct state	1.Enter your username 2.Enter your password 3.click signin button 4.Go to limit page 5. click report and today	username: test password:test@123	give accurate image	Working as expected	Pass		N		suryapriya s	
Testcase_0010	Functional	Home page	Check that the validations are working normal in input session.	check that working normally	1.Enter your username 2.Enter your password 3.click signin button	username: test password:test@123	smooth running of pages	Working as expected	Pass		N		philomina s	

26

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	9	4	4	2	20
Duplicate	0	0	0	0	0
External	5	2	0	4	11
Fixed	2	0	1	0	3
Not Reproduced	0	0	4	0	4
Skipped	0	0	0	0	0
Won't Fix	0	0	1	0	1
Totals	16	8	10	6	39

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	10	0	0	10
Client Application	45	0	0	45
Security	2	0	0	2
Outsource Shipping	4	0	0	4
Exception Reporting	5	0	0	5
Final Report Output	7	0	0	7
Version Control	3	0	0	3

9. RESULTS

9.1 PERFORMANCE METRICS:

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No	Parameters	Values
1	Dashboard design	Dashboard consists of details, services, contact us page
2	Data Responsiveness	Data was responsive to register and login. The data was retrieve fast in IBM cloud.
3	Amount Data to Rendered (DB2 Metrics)	100+ Users login and access their expense and add their expense in day-to-day life
4	Utilization of Data Filters	Data filters was used to get the exact data from users for good environment and security.
5	Effective User Story	Story consists of 6 user stories and all are good for maintenance.
6	Descriptive Reports	Create more report for better understand and good case study.

10. ADVANTAGES & DISADVANTAGES

10.1 ADVANTAGES:

- Tracking income and expenses: Monitoring the income and tracking all expenditures.
- The expense tracking app generates and sends reports to give a detailed insight about profits, losses, budgets, income, balance sheets, etc.,
- Determine project profitability by tracking labor costs, payroll, expenses, etc., of your ongoing project.
- Automated approvals allow you to cut down on the time spent by approvers verifying claims, allowing them to focus on non-compliant or out of the ordinary items.
- With a mobile expense management app, the digital database and integrated corporate policies do the work for you. Indeed, it provides you with business insights on which you can make data-driven decisions.
- Smartphones and apps are ubiquitous nowadays, making it easy for your employees to adopt a mobile expense management solution, especially if it improves their experience of submitting expenses
- Tracking Your Expenses Can Reveal Spending Issues.
- If it's your highest priority to pay down high-interest debt, for example, include debt repayment as a fixed expense in your budget.
- The activity shouldn't take more than a few minutes each day if you adopt an expense-tracking approach that works for you, but if you consistently track your expenses, you will be able to save more, spend less, and make other necessary changes to finances that will allow you to build wealth and go after the things you want in life.

10.2 DISADVANTAGES:

- Internet is need for access the data.
- It reduce the momory power in human.
- It create laziness to maintain in notebook or in memory.
- It takes time to put all the details and see our daily expenses.

11. CONCLUSION

Tracking your expenses daily can save your amount, but it can also help you set financial goals for the future. If you know exactly where your amount is going every month, you can easily see where some cutbacks and compromises can be made. The project what we have developed is work more efficient than the other income and expense tracker. The project successfully avoids the manual calculation for avoiding calculating the income and expense per month. The modules are developed with efficient and also in an attractive manner. The developed systems dispense the problem and meet the needs of by providing reliable and comprehensive information. All the requirements projected by the user have been met by the system. Since the screen provides online help messages and is very userfriendly, any user will get familiarized with its usage. Module s are designed to be highly flexible so that any

failure requirements can be easily added to the modules without facing many problems. The best organizations have a way of tracking and handling these reimbursements.

12.**FUTURE SCOPE**

Provision to add different currencies will be added so that this application is not just limited to USA but also can be used worldwide and the currency converters will be designed and added in order to convert the different currency rates.

In order to make it more user friendly and less user intensive, when the user tries to add the same category or vendor to an expense/income record, a duplicate alert will be presented showing the same category/vendor which the user entered previously for some expense/income and then he can tap on it and the entries will be automatically filled for the current record.

A new tab named "Search" will be implemented so that if the user searches for any vendor, category or subcategory by name, he can see the expenses made on that particular search in a table view list with the total number of transactions made and the total expense amount for that search. the graph reports show the expenses and income graphs separately in the current version. In the future, a comparison between the income made and expense will be shown graphically providing the user more options to see what they are making and what they are spending accordingly.

13.APPENDIX 13.1 SOURCE CODE

FILE NAME : app.py:

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

```
import ibm_db
import re
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
```

```
app = Flask(__name__)
```

```
app.secret_key = 'a'
```

```
conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=32536;SECURITY=SSL;SSLServiceCertificate=DigiCertGlobalRootCA.crt;UID=ktz09482;PWD=MzbDztm2ZeEUSKuG", ' ','')
```

```
#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 REGISTERUSER 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 REGISTERUSER(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 REGISTERUSER 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)
        print(account)

    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(?,?,?,?,?,?,?,?)"

```

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

```
    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.execute(stmt)

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(total>limit[0]):

    mail_from = '19i304@psgtech.ac.in'
    mail_to = session['email']

```

```

msg = MIMEMultipart()

```

```

msg['From'] = mail_from
msg['To'] = mail_to
msg['Subject'] = 'Expense Alert Limit'
mail_body = ""

Dear User, You have exceeded the specified monthly expense Limit!!!!

```

```

"""

msg.attach(MIMEText(mail_body))

```

```

try:
    server = smtplib.SMTP_SSL('smtp.sendgrid.net', 465)
    server.ehlo()
    server.login('apikey',
'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')
    server.sendmail(mail_from, mail_to, msg.as_string())
    server.close()
    print("mail sent")
except:
    print("issue")

```

```

return redirect("/display")

```

```

#DISPLAY---graph

```

```

@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)
```

```
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", 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 )

```

```
#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')

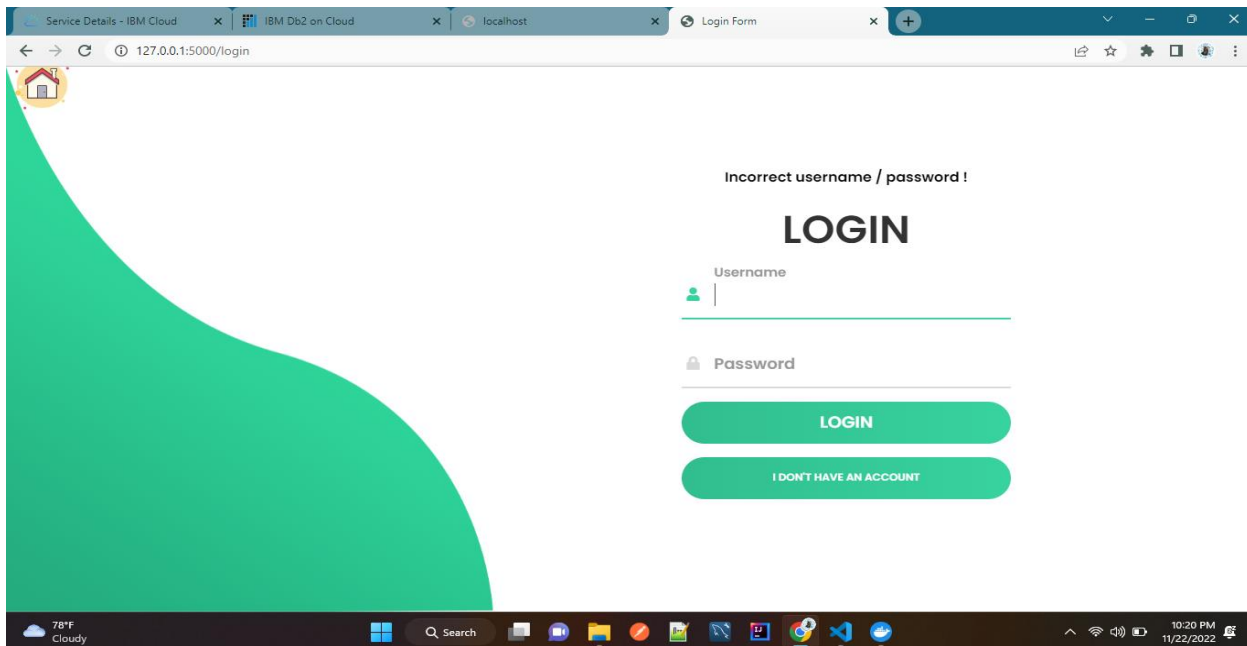
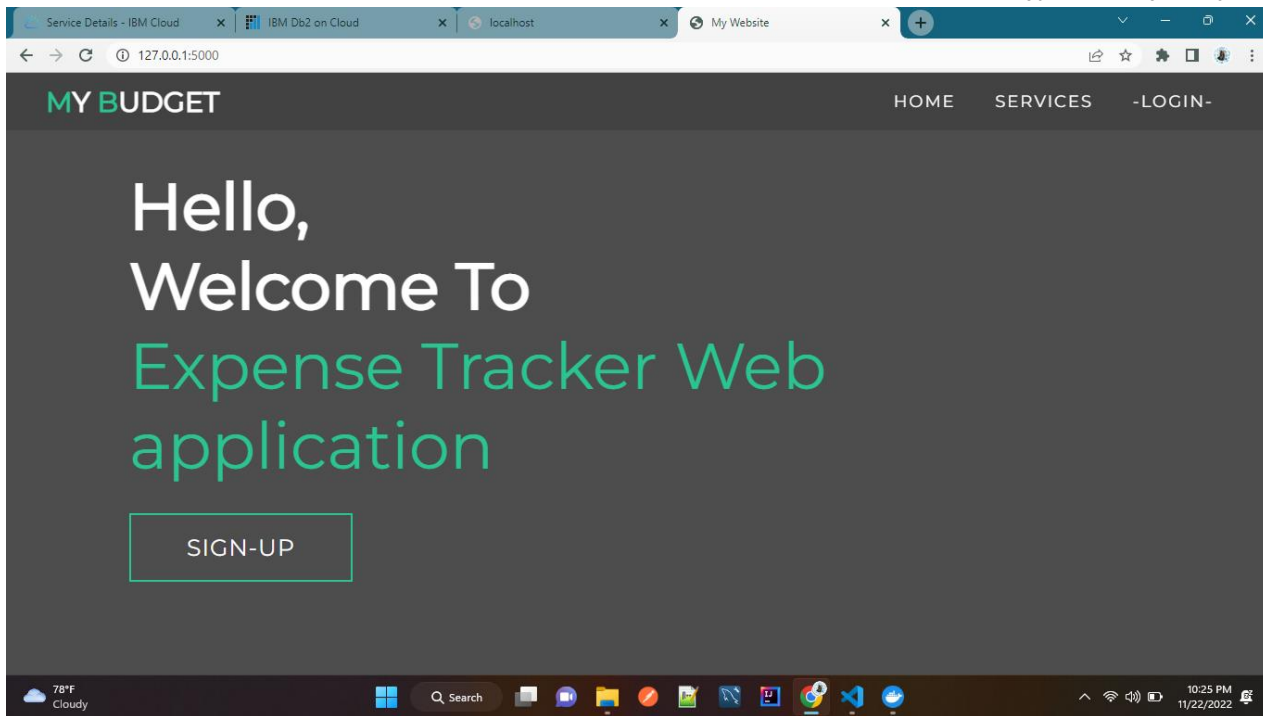
```

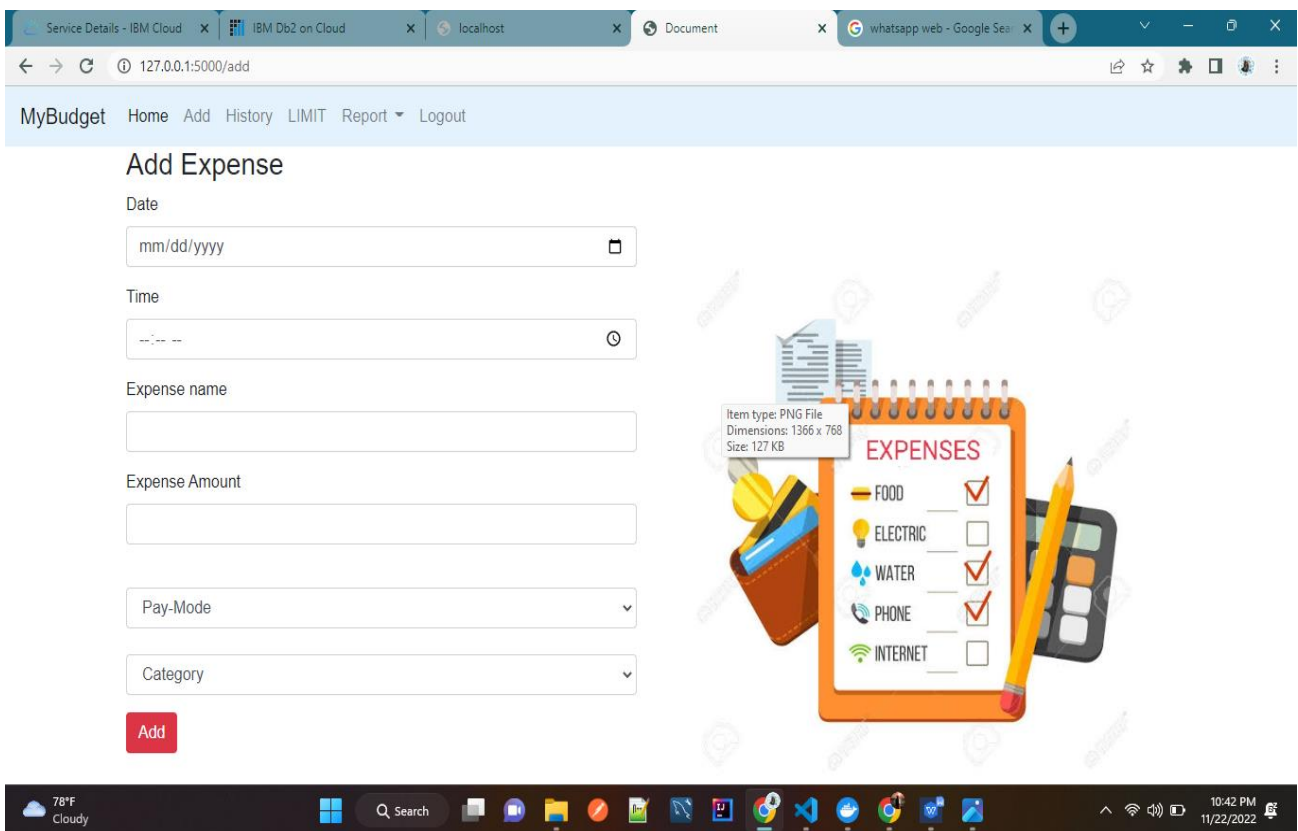
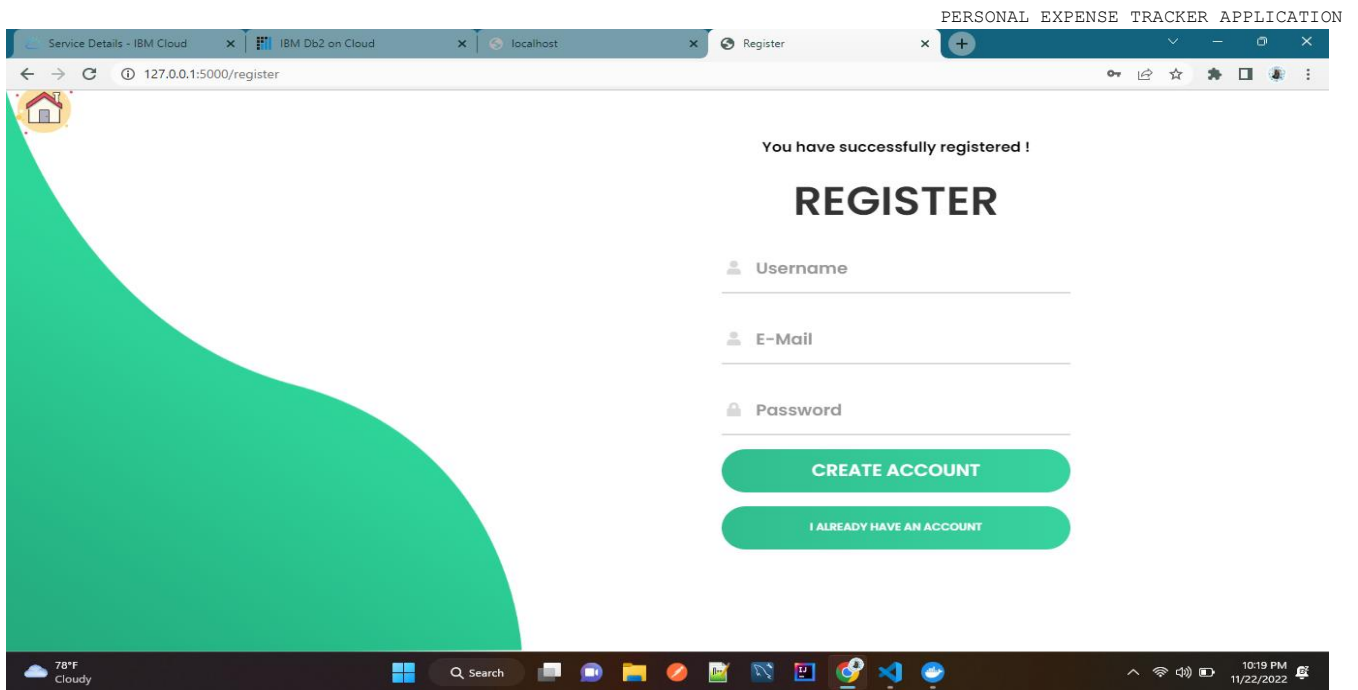
```

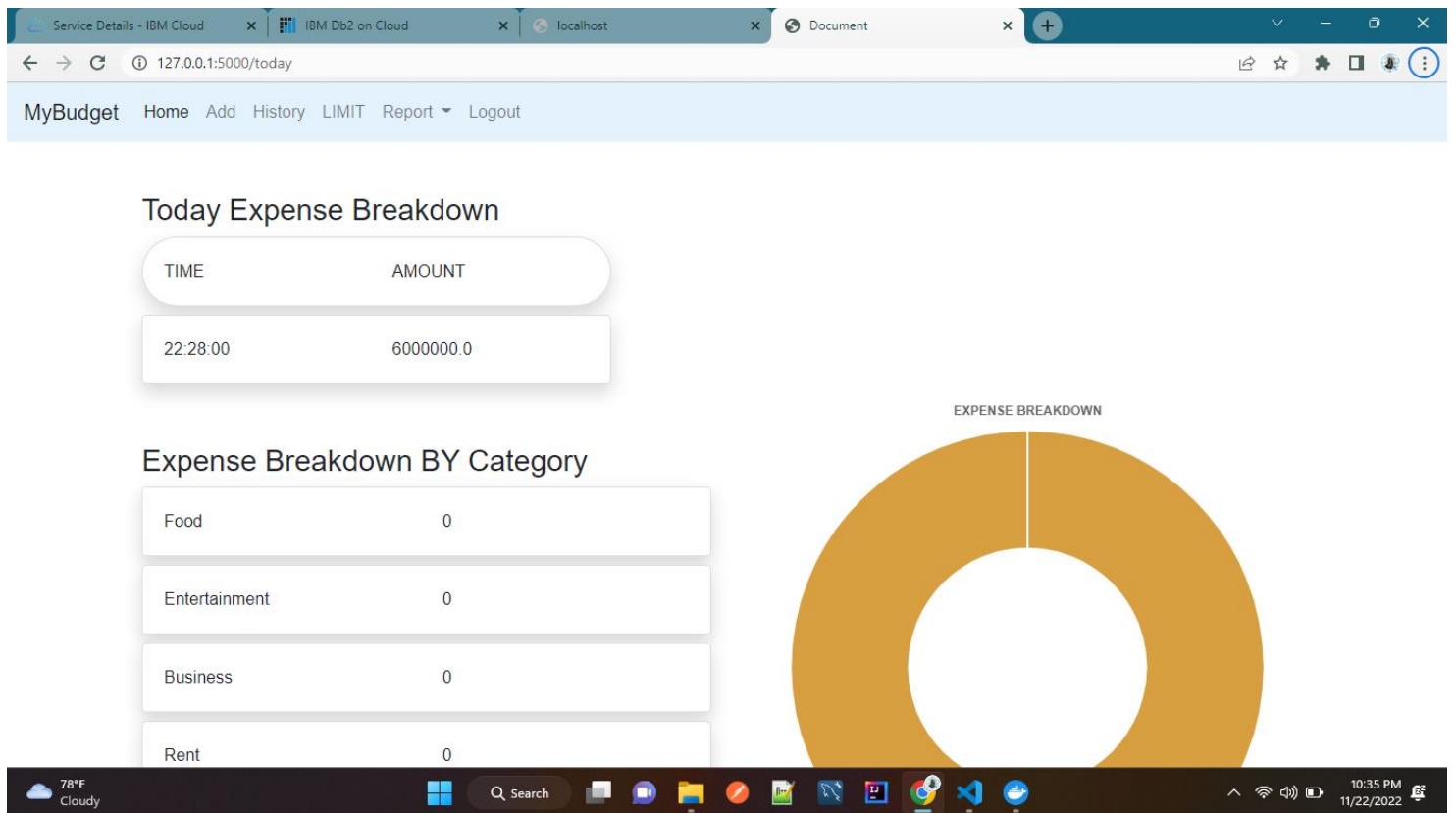
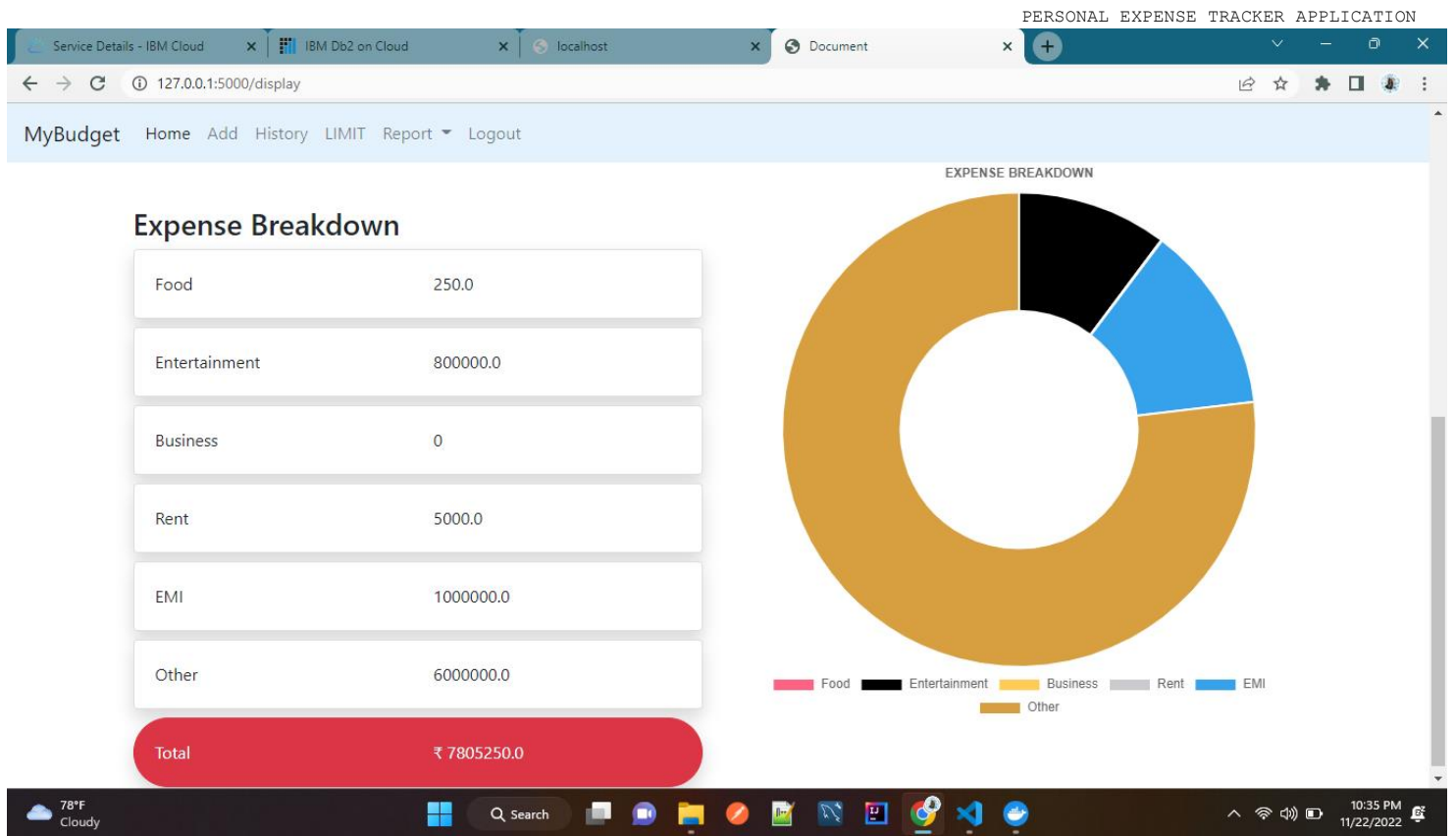
if __name__ == "__main__":
    app.run(debug=True)

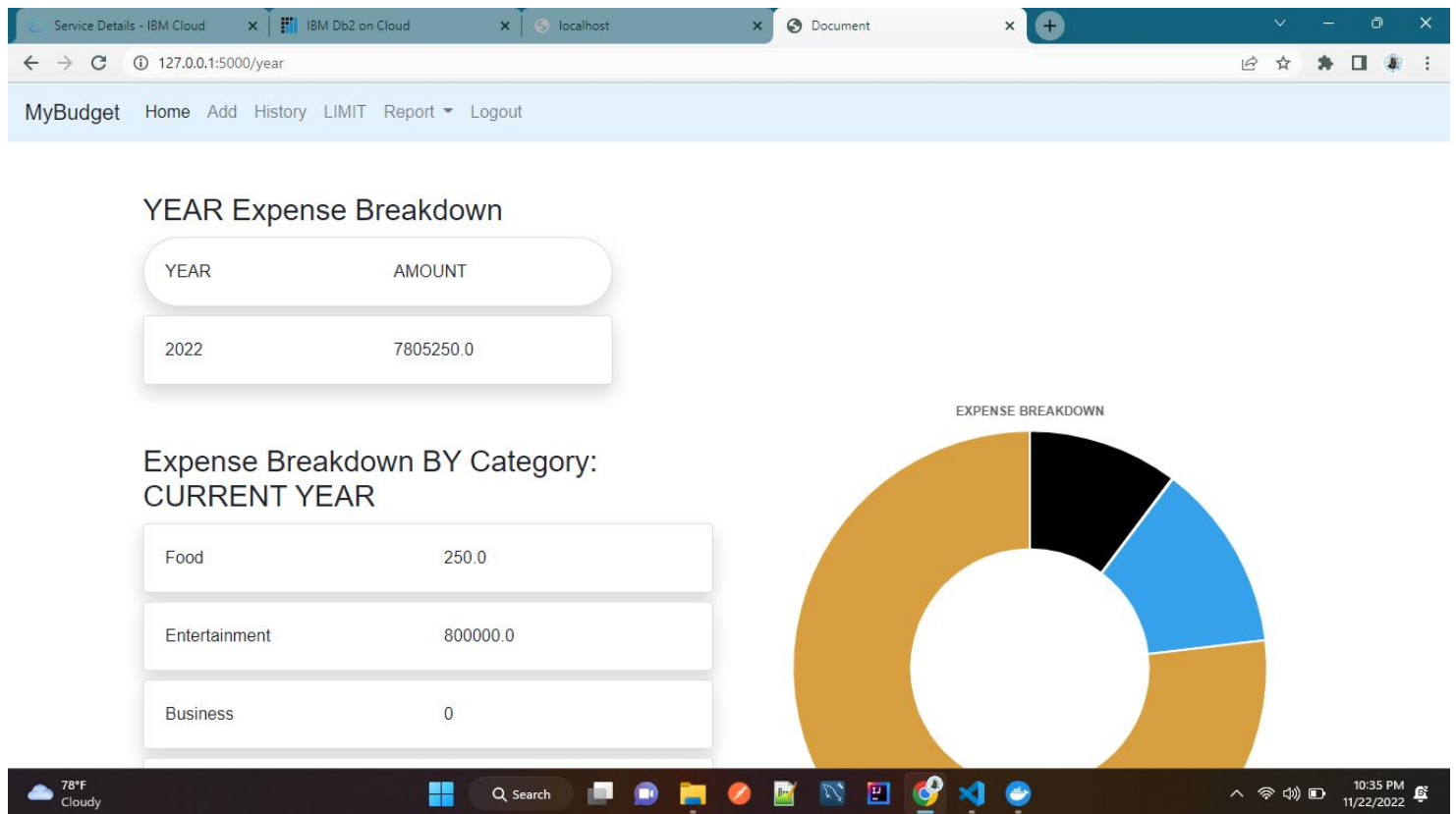
```

OUTPUT SCREENSHOTS:









13.2 GITHUB LINK:

LINK :<https://github.com/IBM-EPBL/IBM-Project-9585-1659023553>

13.3 PROJECT DEMO

LINK:[Demo](#)

