Personal Expense Tracker Application

Team id:PNT2022TMID08698

SUBMITTED BY

HAGITH.D	727619BEC017
YASHASWINI.C	727619BEC015
THALA MUTHU MANIVEL.A	727619BEC011
AVINASH.G	727620BEC053

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

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

CONTENT

1. INTRODUCTION

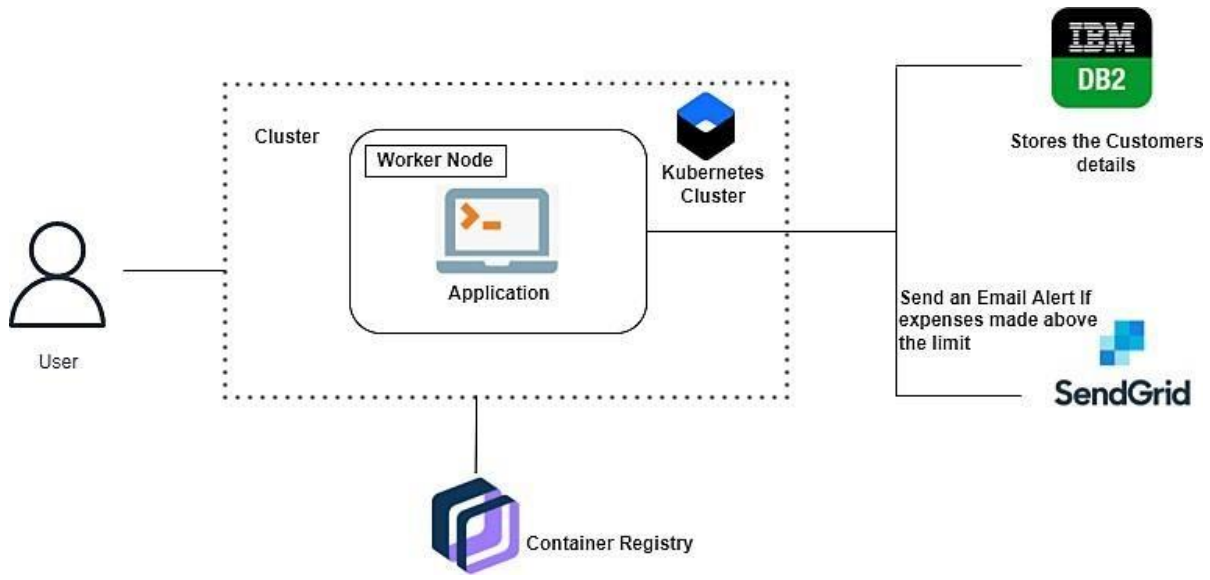
1.1 PROJECT OVERVIEW:

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 a user 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 on track all financial data then we can overcome this problem. The user to 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 for know our day to day life expenses.
- It's helps to regulate our expenditure in daily life.

TECHNOLOGY STACK:



2. LITERATURE SURVEY

2.1 EXISTING PROBLEM:

i.A Novel Expense Tracker using Statistical Analysis School of Computer Science and Engineering, Galgotias University Muskaan Sharma , Ayush Bansal , Dr. Raju Ranjan , Shivam Sethi © June 2021| IJIRT | Volume 8 Issue 1 | ISSN: 2349-6002

• Expense Tracker is used to maintain and manage data of daily expenditure in a more precise way it can give profound knowledge of their expenses

• 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.

• This helps the society to prevent the issues like bankruptcy and save time from manual calculations.

• People when usually go for trips with friends, can use this tracker to maintain their expense.

• If we know where our money is being spent every day, it is easy to set some cutbacks and such to help reduce expenditure.

• It is developed to be efficient and look attractive at the same time.

ii. Expense Tracker Author(s): Praphulla S. Kherade ; Raj S. Vilankar ; Parag M. Sawant ; Atiya Kazi Paper ID: 1702687 Published Date: 04-05-2021 Published In: Iconic Research And Engineering Journals Publisher: IRE Journals e-ISSN: 2456-8880 Volume/Issue: Volume 4 Issue 11 May-2021.

• We are building 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.

• If you exceed 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 user’s savings.

• amount saved can be used for celebrating festivals, Birthdays or Anniversary.

• If you exceed daily expense allowed amount it will give you a warning, so that you don’t spend much and that specific day.

iii. Expense Tracker : A Smart Approach to Track Everyday Expense

• 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.

• We have developed the necessary system to work without internet. We need a database, desktop, application and user to use this system.

• System design is the process of defining architecture, modules, interfaces, and data for a system to meet specific needs. System design can be seen as an application of systems theory to product development.

• Generally, the budget is assembled according to category. Categories vary, for example, food, entertainment, transportation, education, health, clothing, and so on.

• However, spending is limited to budget revenue. For this reason, we need to keep track of our expenses so that they do not exceed our budget.

• Explaining the latest apps built in this category, YNAB is an expense tracker that automatically tracks our expenses through our bank account or credit card. We can also define future costs.

2.2 REFERENCES:

1.A Novel Expense Tracker using Statistical Analysis School of Computer Science and Engineering, Galgotias University Muskaan Sharma , Ayush Bansal , Dr. Raju Ranjan , Shivam Sethi © June 2021| IJIRT | Volume 8 Issue 1 | ISSN: 2349-6002

2. Daily Expense Tracker, Department of Computer Engineering, MIT Polytechnic, Pune, India.Professor, Department of Computer Engineering, Dr. Vishwanath Karad MIT World Peace University, Pune,India.

3. Expense Tracker : A Smart Approach to Track Everyday Expense Hrithik Gupta, Anant Prakash Singh, Navneet Kumar and J. Angelin Blessy December 25, 2020

4. Expense Tracker : A Smart Approach to Track Everyday Expense

2.3 PROBLEM STATEMENT:

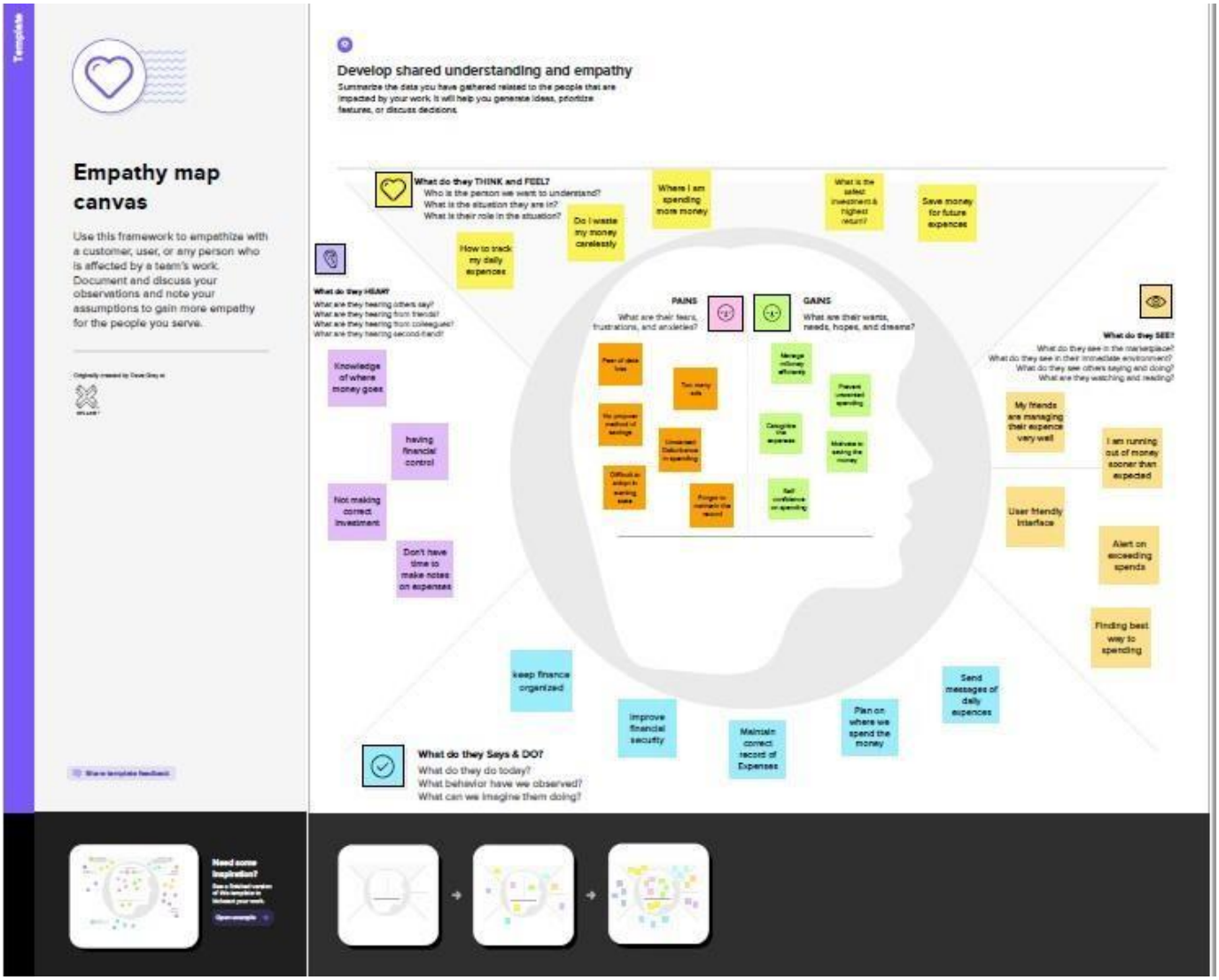
Personal finance entails all the financial decisions and activities that a Finance app makes life easier by helping the user to manage their finances efficiently. A personal finance app will not only help them with budgeting and accounting but also give helpful insights about money management.

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 occuring?	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

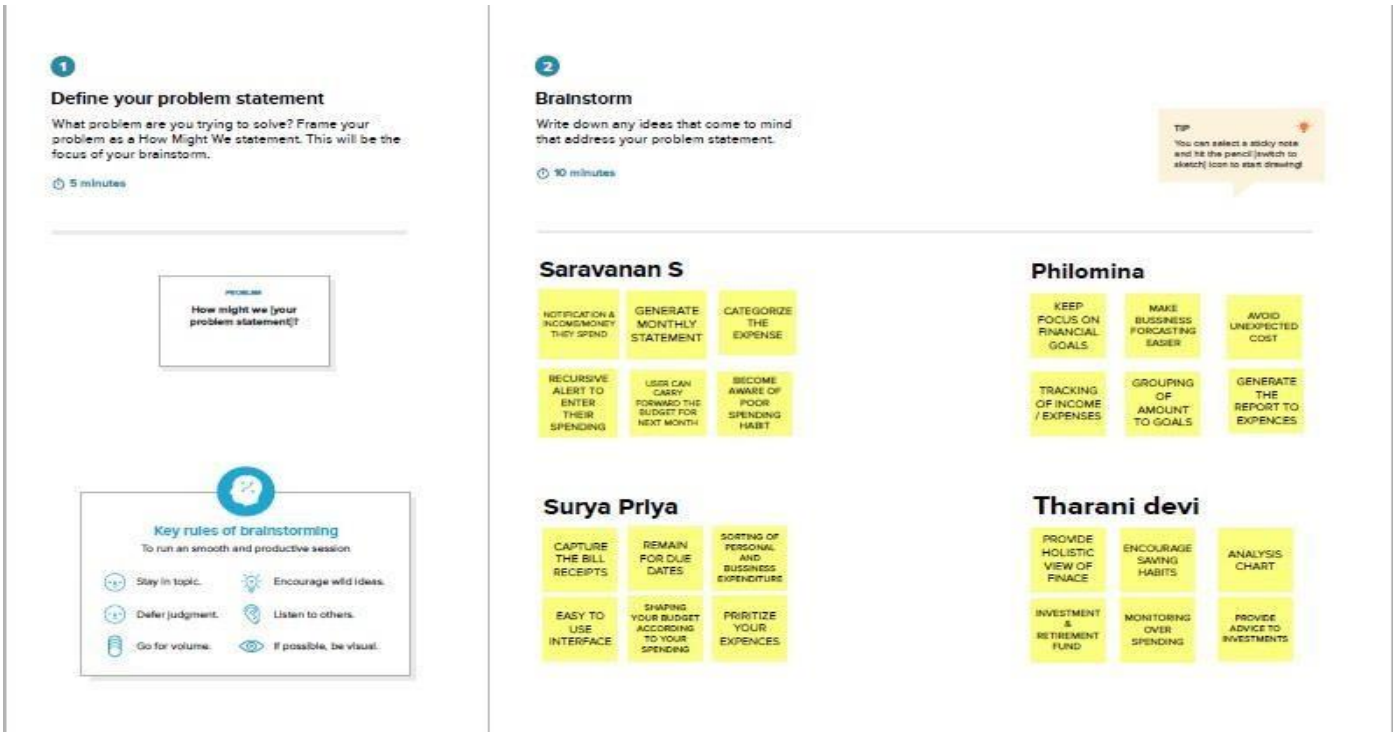
3.1 EMPATHY MAP CANVAS:

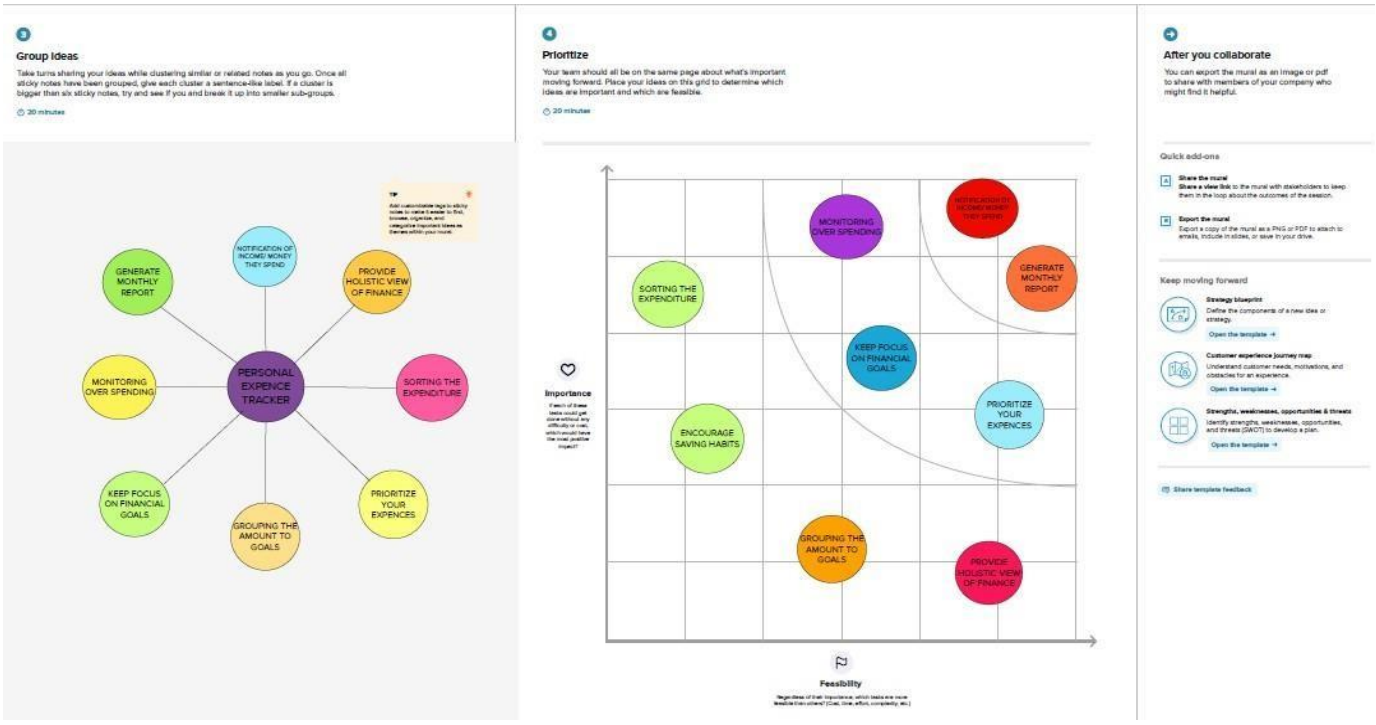
An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. An Empathy Map consists of four quadrants. The four quadrants reflect four key traits, which the user 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.



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.





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 expensewhich 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**

1. CUSTOMER SEGMENT(S) CS

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

Customer who can make sure that cash is used wisely.

Customer are people who want to maintain an correct document in their money.

Customer who desires to categorize the costs which include education,entertainment, food,etc.

6. CUSTOMER CONSTRAINTS CC

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

Adding the costs made each

and every time manually reduces the users.

Internet hosts the plenty of commercials

proscribing the software usability.

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 notetaking

Using Excel spreadsheets to be aware the expenses and making the calculations in which the calculation calls for greater time and no graphical illustration is provided.

2. JOBS-TO-BE-DONE / PROBLEMS JP

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

A fee monitoring facilitates in finance management through understanding the earnings primarily based totally on expenditure made this facilitates to store money.

The goal of the software is to achieve optimal earnings each in lengthy and quick run.

People also can view the costs as a graphical representation and evaluate the costs made.

9. PROBLEM ROOT CAUSE RC

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

Spending lavishly with out with out preserving records cause spend past limit.

It consists of harassed and headaches to stay a economically balanced lifestyles.
Inconvenience to stay a lifestyles with a Standardized monetary expenses.

7. BEHAVIOUR BE

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

User can lessen few charges made unnecessarily.

Sends the Email alert if the price exceeds the limit.

Keep song of charges and consider them in

graphical layout for specific analysis.

Identify strong TR & EM

3. TRIGGERS TR

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

Application permits the purchaser to lessen the lavish costs made.

10. YOUR SOLUTION SL

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

Email alert which notifies the consumer whilst most

quantity is spent the usage of spendgrid framework.

Application lets in to view fees in graphical form.

8. CHANNELS of BEHAVIOUR CH

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

Expense tracker in on-line include loads of commercials that have opportunities of stealing data.

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

User have to be privy to the tax guidelines through studying phrases and conditions.

Identify strong TR & EM

4. EMOTIONS: BEFORE / AFTER EM

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

They have a higher know-how of the earnings and outgoings.

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:

S.No	Functional Requirement	Description
FR-1	User Registration	Registration through Form.
FR-2	Confirmation	Confirmation via OTP.
FR-3	User Login	Login through valid username and password.
FR-4	Forgot password	User can reset the password through mail.
FR-5	User Calender	Allows user to add the data to their expenses.
FR-6	Expense Tracker	Graphically represent the expense in the form of report.
FR-7	Category	Allows user to add categories.
FR-8	Email alert	If amount exceeds, alert through mail.

4.2 NON-FUNTIONAL REQUIREMENT:

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:

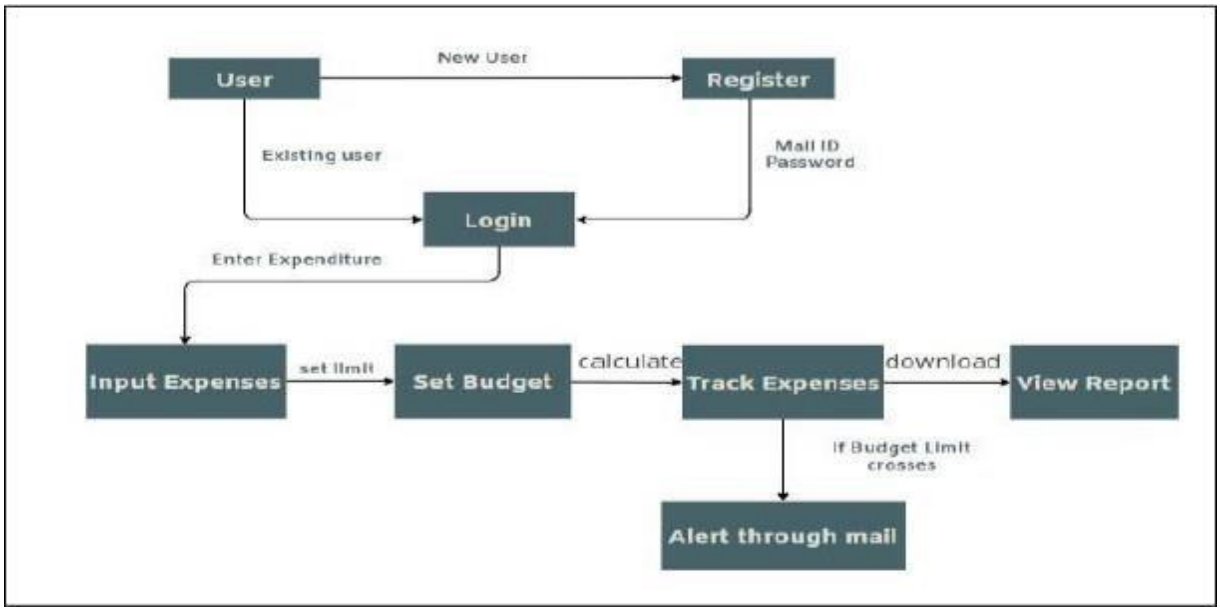
S.No	non-functional Requirement	Description
NFR-1	Usability	Helps to keep an accurate record of the amount.
NFR-2	Security	Secured by the password and prevent the app from the cybercrimes.
NFR-3	Reliability	No risk of data loss because each data record is kept in a well built database schema.
NFR-4	Performance	Categories and options are the types of expense. Because of lightweight database support, the system's throughput increases.
NFR-5	Availability	The app is accesible all the time.
NFR-6	Scalability	The ability to appropriately handle increasing demands.

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.

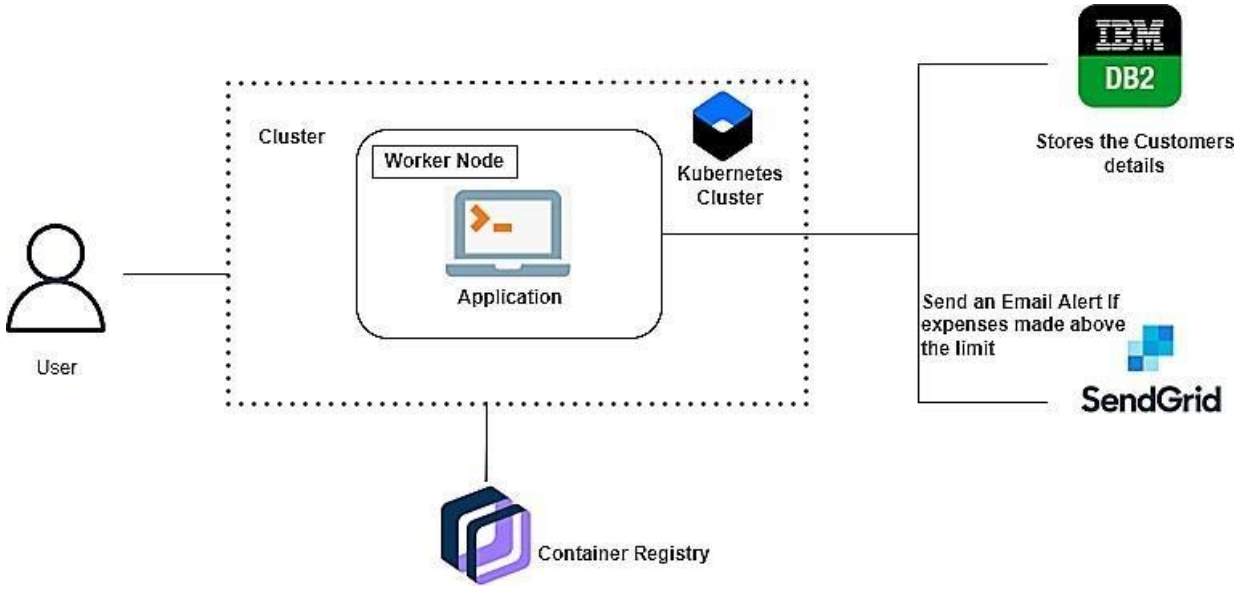
DATAFLOW OF PERSONAL EXPENSE TRACKER APPLICATION:



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.

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard.	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm.	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login.	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register for the app through Gmail login.	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can register & access the dashboard with Gmail Login.	High	Sprint-1
	Dashboard	USN-6	As a user, I can add my day-to-day expenses regularly.	I can track my expenses perfectly.	High	Sprint-2
Customer (Web user)	Dashboard	USN-7	As a user, I can see login page and registration page for which the user logins and input expenses.	I can login through Gmail and register for expense tracking.	Medium	Sprint-2
Customer Care Executive	Dashboard	USN-8	As a customer care executive, I can solve the queries of users.	I can reply to their queries and solve their problems.	High	Sprint-3
Administrator	Registration	USN-9	As an Administrator, I can view the basic details of user.	I can provide the login details.	Medium	Sprint-4
	Dashboard	USN-10	As an administrator, I can able to view the overall progress of a user.	I can give rewards based on their progress.	Low	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.

SPRINTS	FUNCTIONAL REQUIREMENT	USER STORY NO.	USER STORY	STORY POINTS	PRIORITY	TEAM MEMBERS
ST – 1	User Registration	USN-1	Create an account for the users to get access to all the features	10	High	Saravanan, Philomina, Tharani Devi
ST – 1	User Wallets	USN-2	Wallets hold the users money	5	High	Saravanan, Philomina, Tharani Devi
ST – 1	User Category Management	USN-3	Users can customize their income and expense categories	5	Medium	Saravanan, Philomina, Tharani Devi
ST – 2	User Income	USN-4	Users can attach their income to the wallets	7	High	Saravanan, Philomina, Suryapriya
ST – 2	User Expenses	USN-5	Users can deduct their amount from the wallets	7	High	Saravanan, Philomina, Suryapriya
ST – 2	User Budget Alerts	USN-6	Users can set alerts and limit their expenses	6	Medium	Saravanan, Philomina, Suryapriya
ST – 3	Analytics	USN-7	Users can get a visualization on their income and expenses	6	High	Saravanan, Surya Priya, Tharani Devi
ST – 3	Multilingual Support	USN-8	Users should be able to use the application in their languages	4	Low	Saravanan, Surya Priya, Tharani Devi
ST – 3	File Management	USN-9	Users should be able to attach files to their income or expenses	6	Medium	Saravanan, Surya Priya, Tharani Devi
ST – 3	Economic News	USN-10	Users should be alerted with economic news	4	Low	Saravanan, Surya Priya,
ST – 4	Debt and Investment Calc	USN-11	Users can calculate their returns and risks	7	Medium	Saravanan, Philomina Surya Priya, Tharani Devi
ST – 4	Dockerization and Deploy the application	USN-13	Container the application and deploy to the kubernetes cluster	13	High	Saravanan, Philomina Surya Priya, Tharani Devi

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:

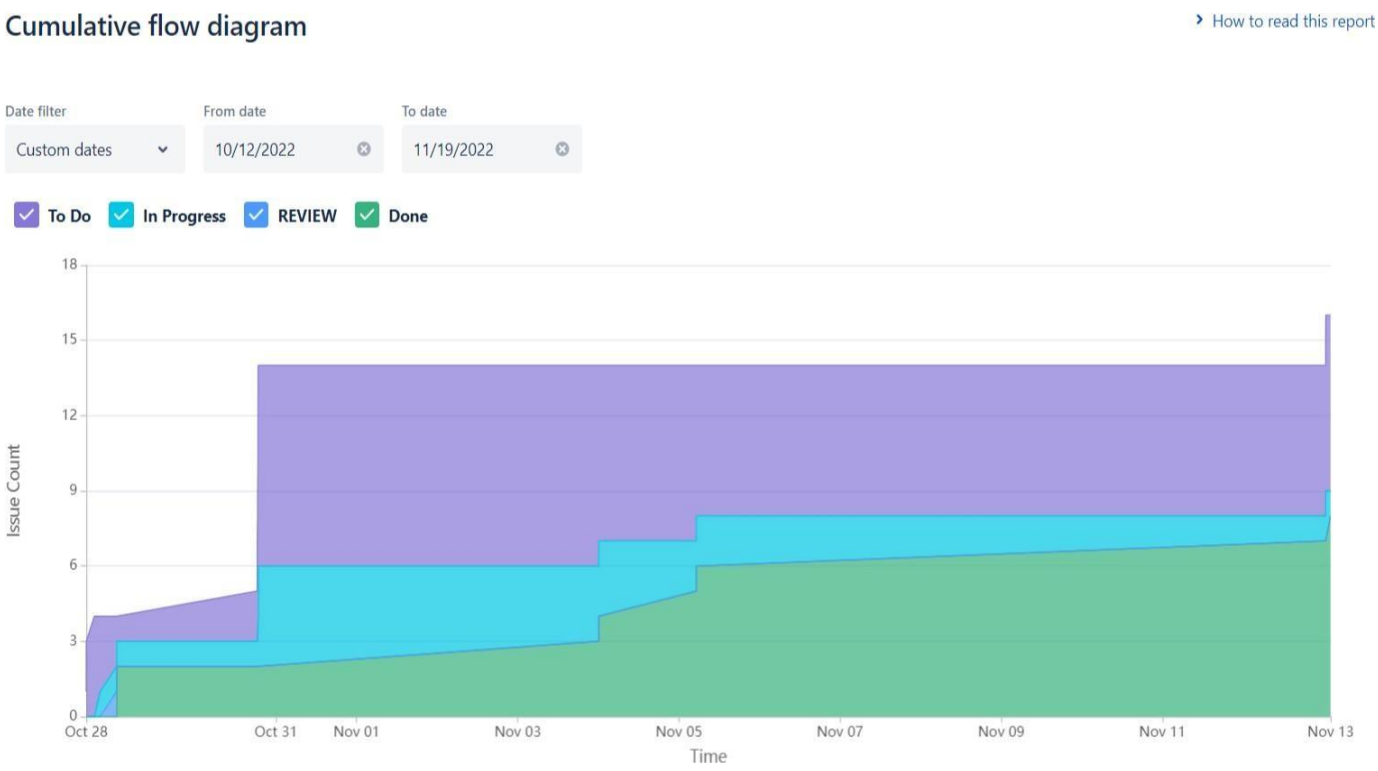
Backlog is usually a list of issues describing what your team is going to do on a project. It's a convenient place for creating, storing, and managing several kinds of issues: issues that you're currently working on (you can also see them on the board and in the current sprint if you're using a Scrum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

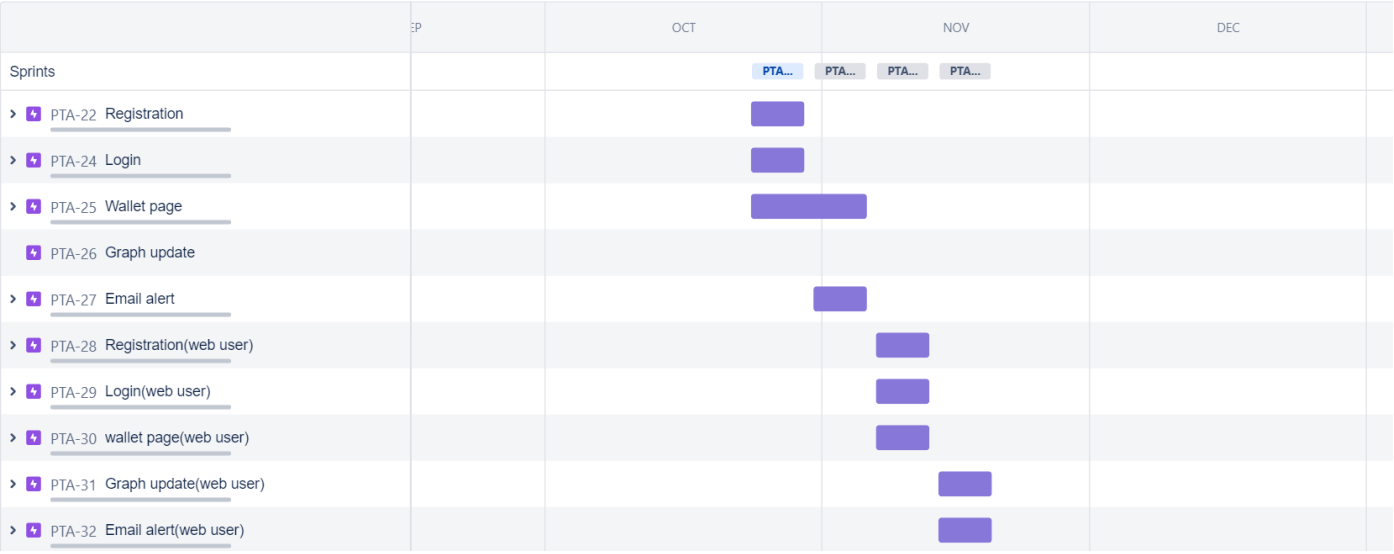
COMMULATIVE FLOW DIAGRAM:

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).



ROAD MAP:

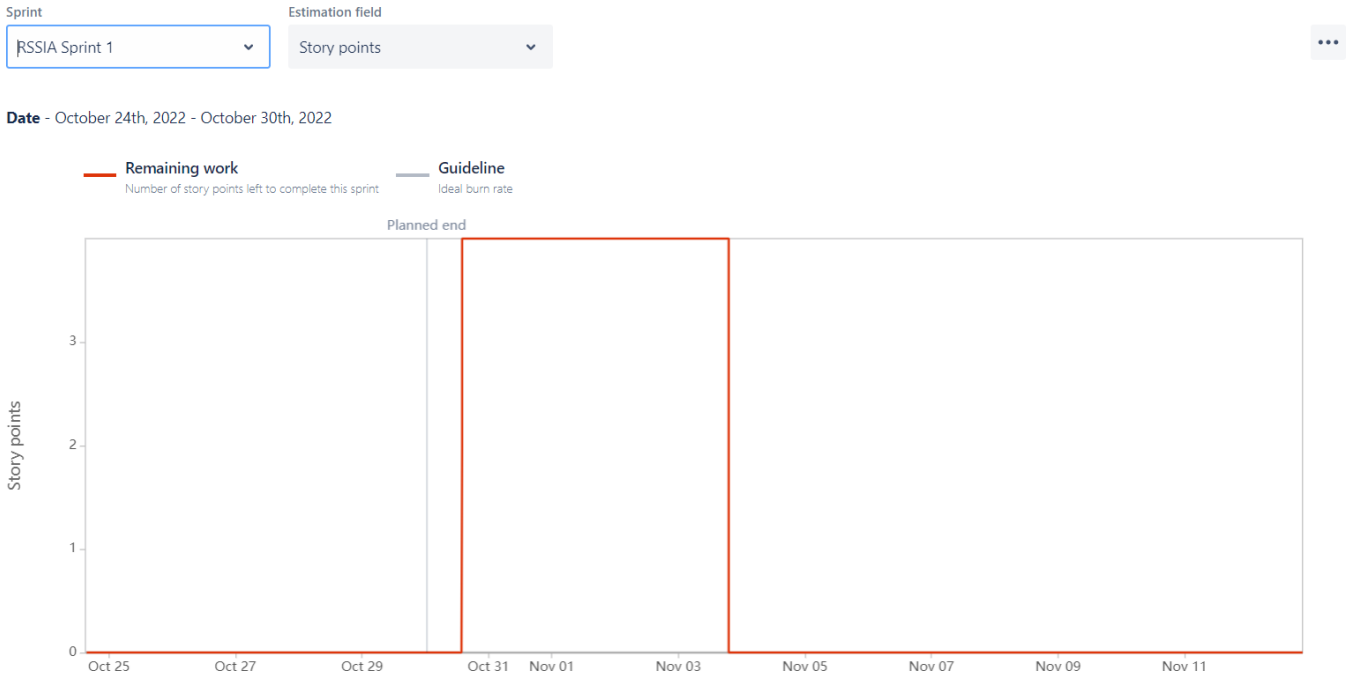
Roadmaps in Jira Software are team-level roadmaps useful for planning large pieces of work several months in advance at the Epic level within a single project. Simple planning and dependency management features help your teams visualize and manage work better together.



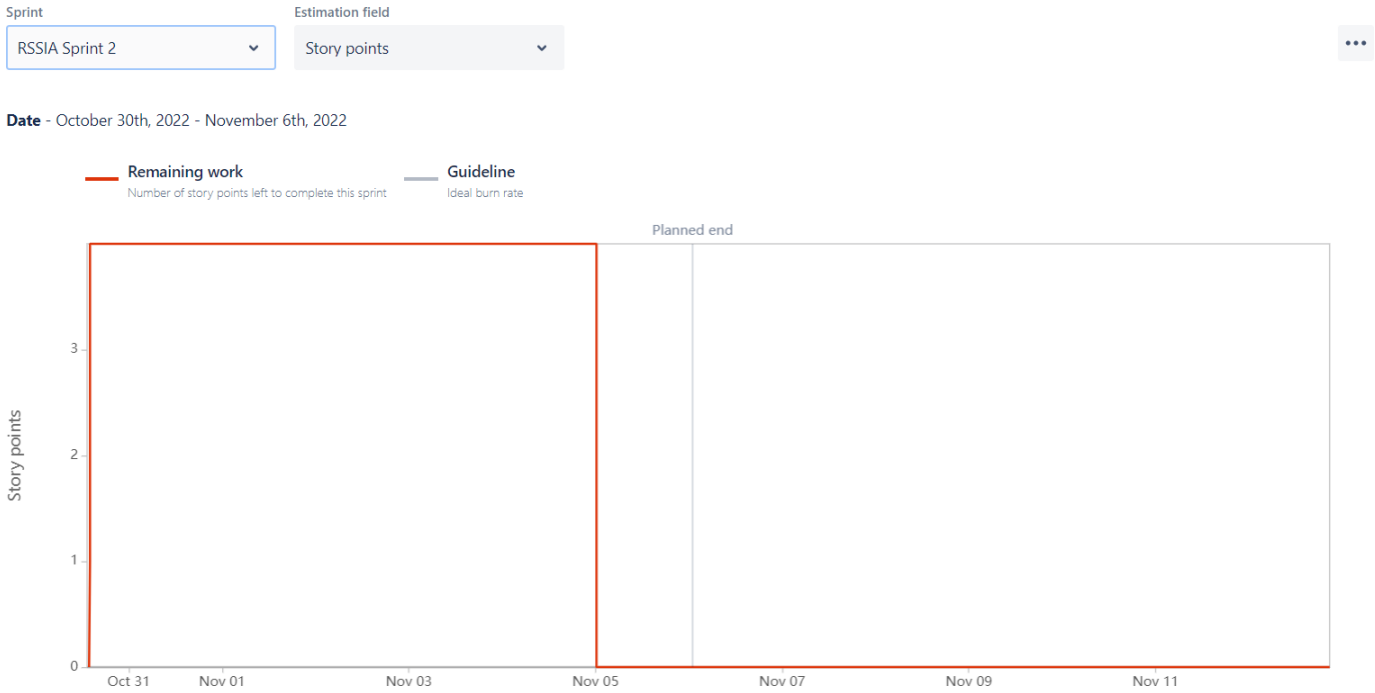
SPRINT BURNDOWN CHART:

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.

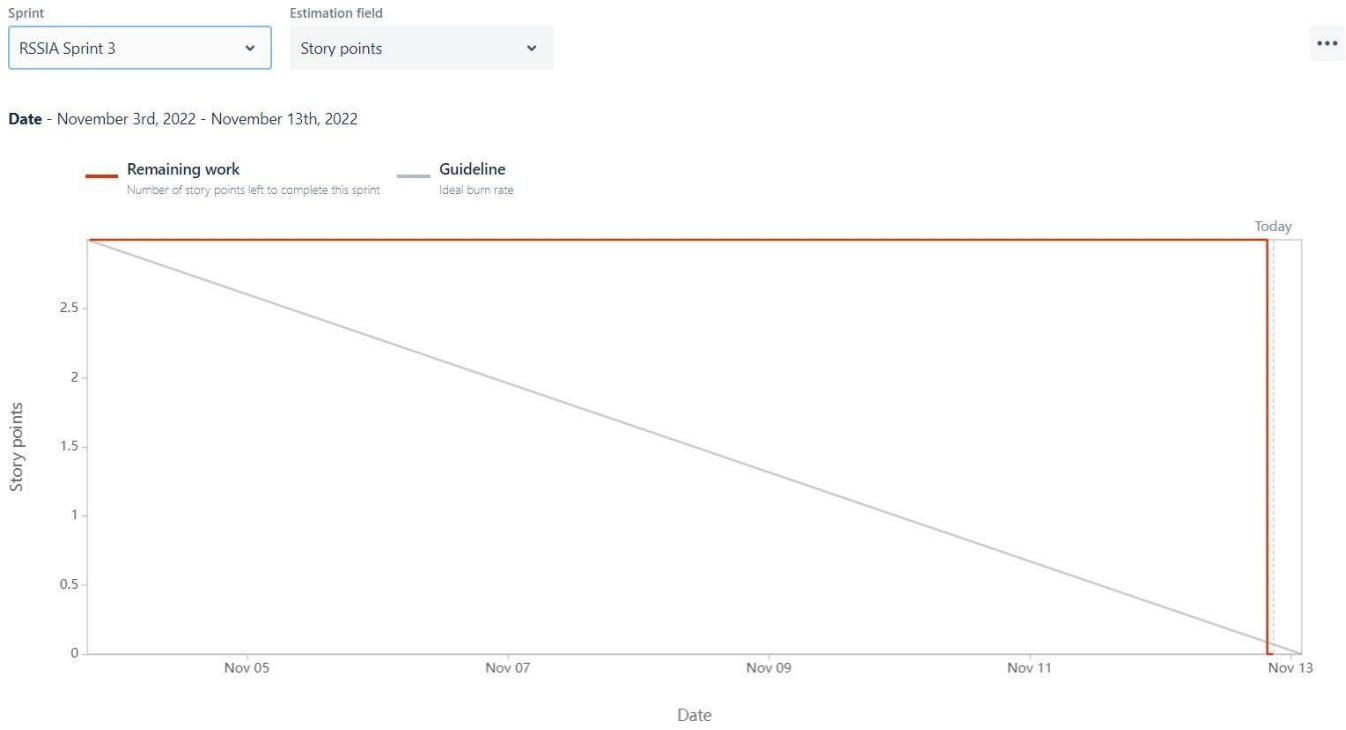
SPRINT 1:



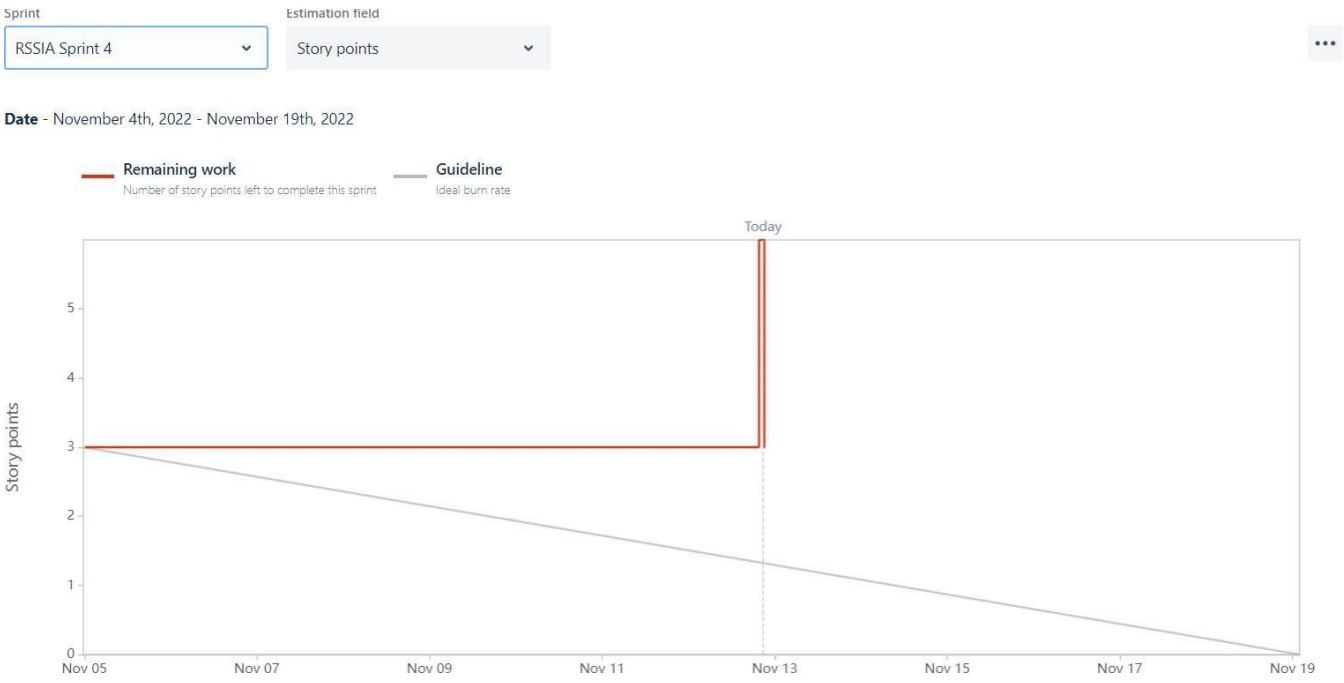
SPRINT 2:



SPRINT 3:



SPRINT 2:


Sprint
RSSIA Sprint 4
Estimation field
Story points
Date - November 4th, 2022 - November 19th, 2022
Remaining work
Number of story points left to complete this sprint
Guideline
Ideal burn rate
Today
Story points
0
1
2
3
4
5
Nov 05
Nov 07
Nov 09
Nov 11
Nov 13
Nov 15
Nov 17
Nov 19

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 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 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 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>
<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)',
```

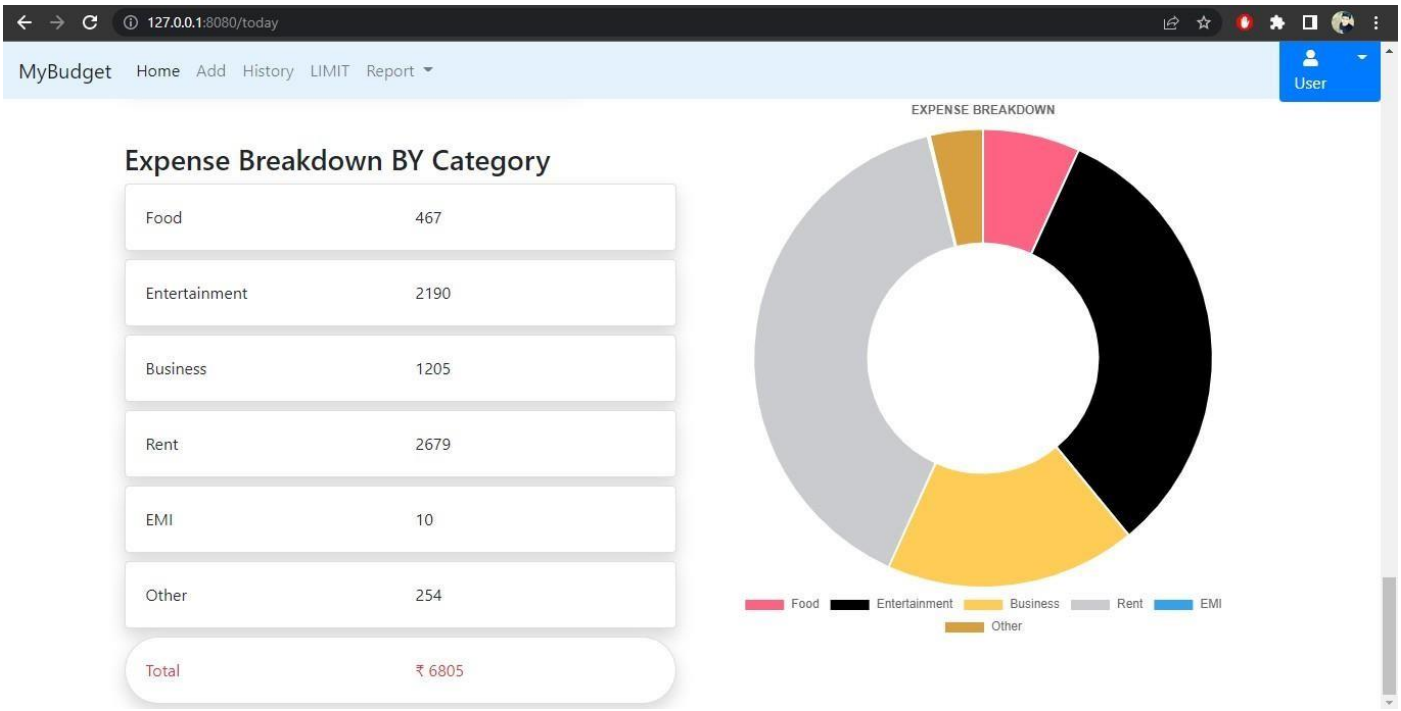
```
                'rgb(201,203,207)',
                 'rgb(54,162,235)',
                  'rgb(215,159,64)'
               ],
            }]
        },
        options: {
            responsive: true,
            plugins: {
    legend: {
      position: 'bottom',
    },
    title: {
 display: true,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

limit.html:

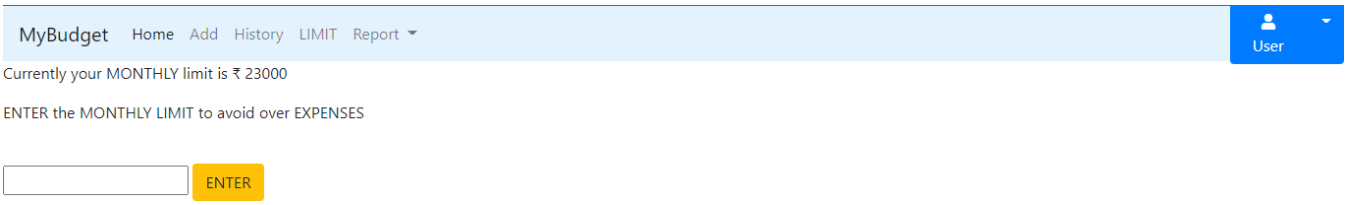
```
{% extends 'base.html' %}
{% block body %}
<p> Currently your MONTHLY limit is ₹ {{y}} </p>
<form action="/limitnum" method="POST">
<p> ENTER the MONTHLY LIMIT to avoid over EXPENSES</p> <br/>
   <input type="number" name="number" required/> <span>
      <button class="btn btn-warning" type="submit">ENTER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

OUTPUT:

1. Daily report:



2. Limit setup:



7.2 FEATURE :

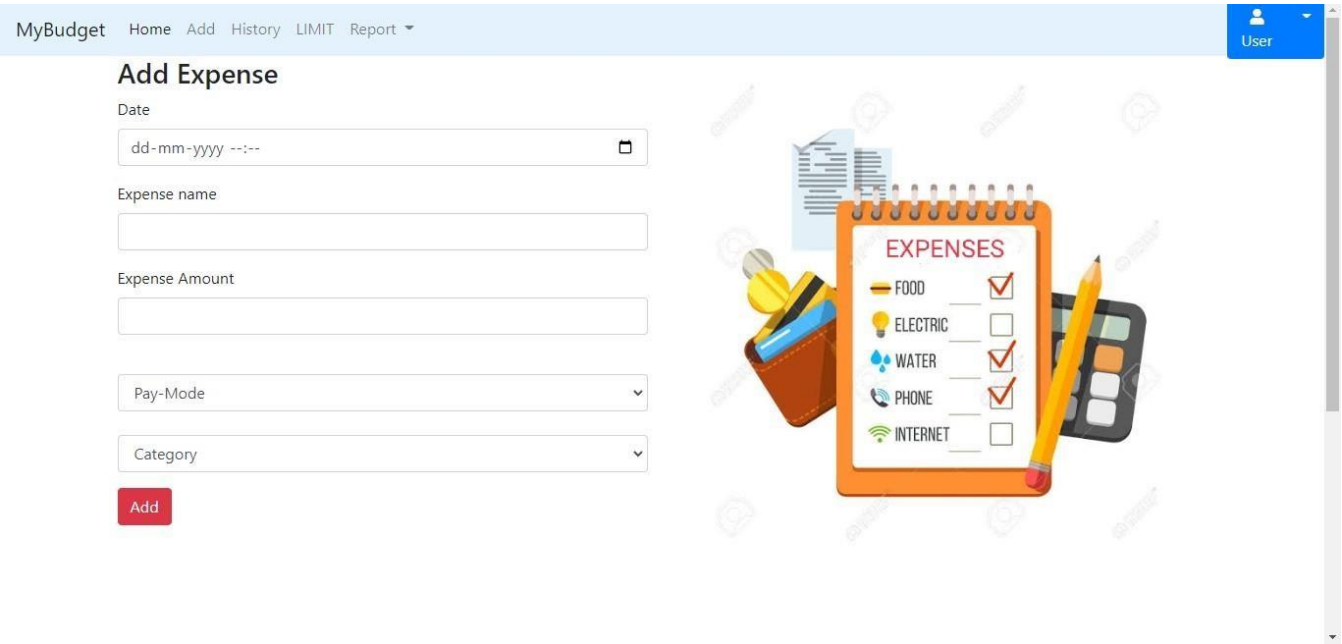
CODE :

addexpense.html:

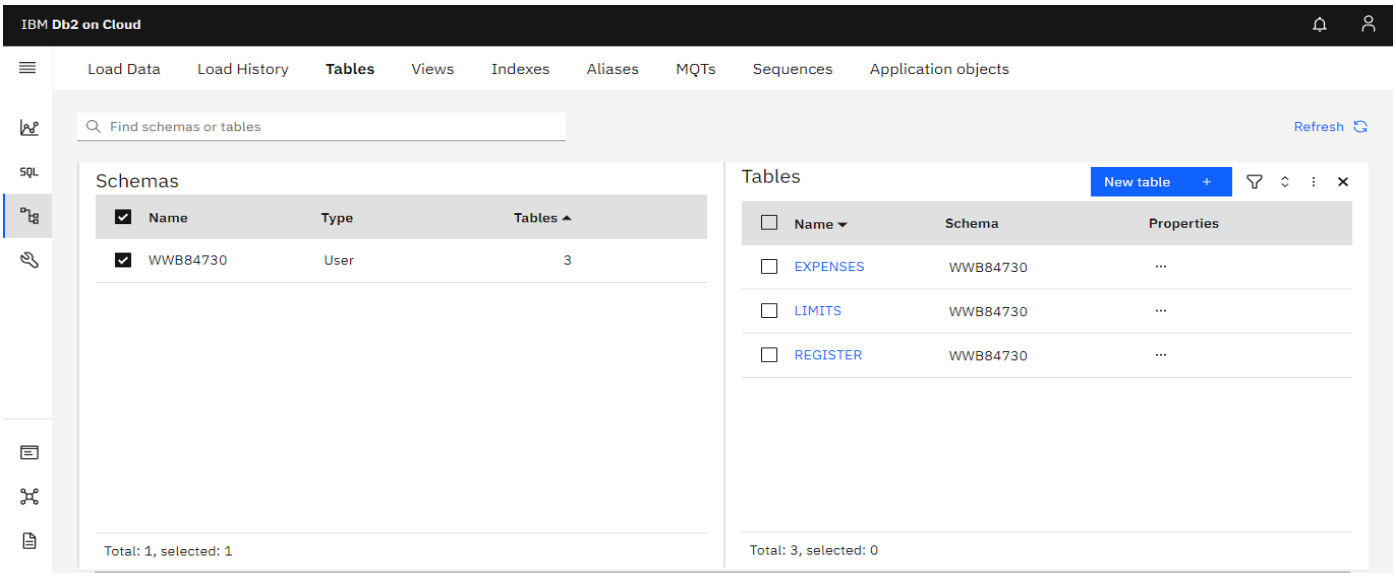
```
{% extends 'base.html' %}
{% block body %}
<div class="container">
  <div class="row">
    <div class="col-md-6">
      <h3>Add Expense</h3>
      <form action="/addexpense" method="POST">
        <div class="form-group">
          <label for="">Date</label>
          <input class="form-control" 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>
```

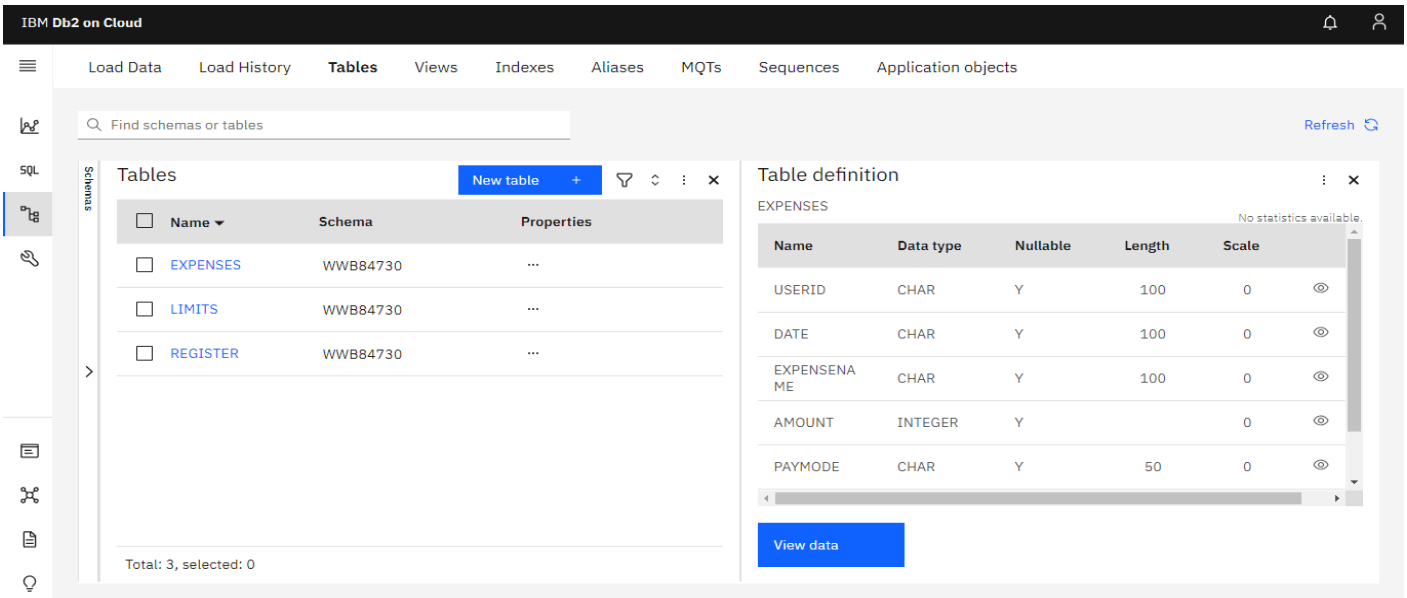
```
                <option name = "EMI" value="EMI">EMI</option>
                <option name = "other" value="other">other</option>
              </select>
            </div>
            <input class="btn btn-danger" type="submit" value="Add" id="">
          </form>
          <div style="position: relative; left: 590px; top: -460px;" class="imagge">
            <img width="90%" height="90%" src="../static/images/addd.jpg" class="rounded float-left" alt="...">
           </div>
        </div>
    </div>
</div>
    {% endblock %}
```

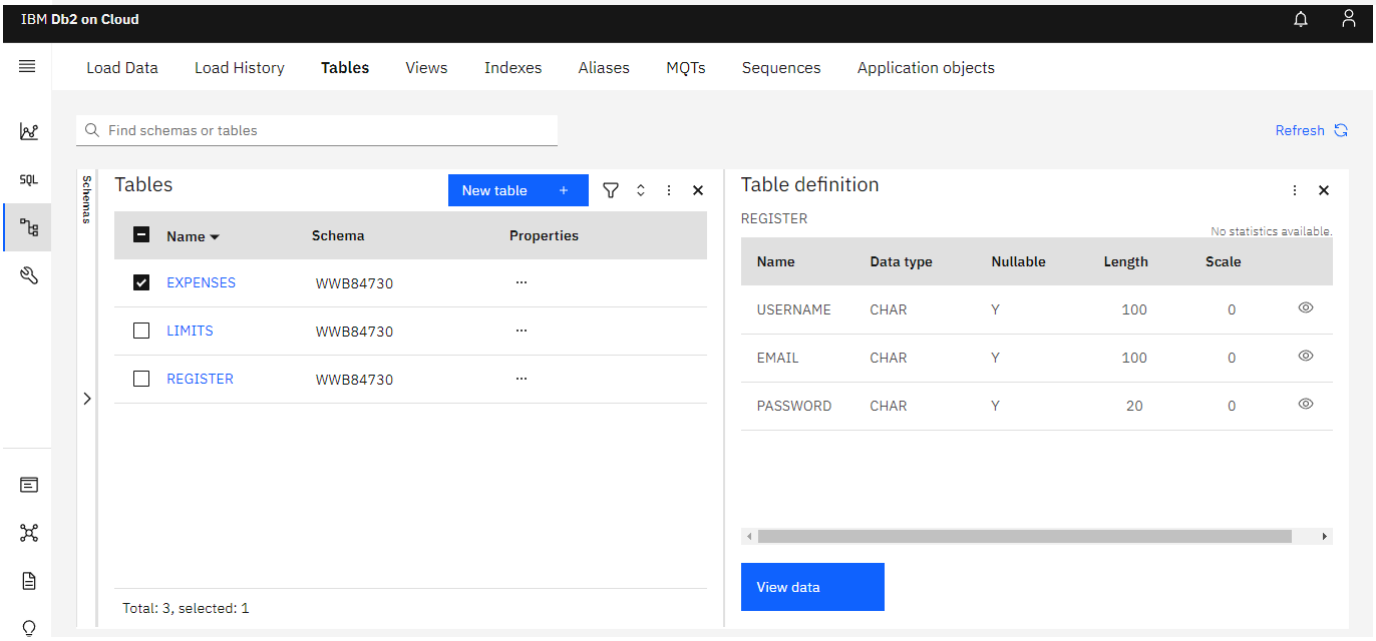
OUTPUT:



7.3 DATABASE SCHEMA :







8. TESTING

8.1 TEST CASES:

- To check weather 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 themself as an 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 catogories are working normal.
- Check that the validations are working normal In input session.

TEST CASE DATA:

Date	13-Nov-22
Team ID	PNT2022TMID07102
Project Name	Personal Expense tracker applica
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
Testcase_001	Functional	Login Page	To check weather 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 passwoed: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 passwoed: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 passwoed: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 passwoed: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.	Retrive 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 passwoed: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 databse with recently added limit	1.Enter your username 2.Enter your password 3.click signin button 4.Go to limit page	username: test passwoed: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 retrive 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 passwoed:test@123	report can be seen	Working as expected	Pass		N		saravanan s
Testcase_009	UI	report page	verify all catogories 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 passwoed: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 normaly	1.Enter your username 2.Enter your password 3.click signin button	username: test passwoed:test@123	smooth running of pages	Working as expected	Pass		N		philomina s

8.2 USER ACCEPTANCE TESTING:

PURPOSE OF DOCUMENT:

The purpose of this document is to briefly explain the test coverage and open issues of the Personal Expense Tracker Application project at the time of the release to User Acceptance Testing (UAT).

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	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 :(FULL SET)

FILE NAME : app.py:

```
import os
import re
import ibm_db
import ibm_db_dbi
from flask import Flask, redirect, render_template, request, session
from flask_db2 import DB2
from gevent.pywsgi import
WSGIServerapp = Flask(__name__)
app.secret_key = 'a'
app.config['database'] = 'bludb'
app.config['hostname'] = '1bbf73c5-d84a-4bb0-85b9-
ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud'
app.config['port'] = '32286'
app.config['protocol'] = 'tcpip' app.config['uid']
= 'wwb84730' app.config['pwd'] =
'O2rrLro2Lm1JVBKx'app.config['security'] =
'SSL'
try:
   mysql = DB2(app)
   conn_str='database=bludb;hostname=1bbf73c5-d84a-4bb0-85b9-
ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;port=32286;protocol=tcpip;\
         uid=wwb84730;pwd=O2rrLro2Lm1JVBKx;security=SSL'
   ibm_db_conn = ibm_db.connect(conn_str,",")
   print("Database connected without any error !!")
except:
   print("IBM DB Connection error   :     " +DB2.conn_errormsg())
#HOME--PAGE
@app.route("/home")
def home():
   return render_template("homepage.html")
@app.route("/")
def add():
   return render_template("home.html")
#SIGN--UP--OR--REGISTER
@app.route("/signup")
```

```
def signup():
  return render_template("signup.html")
@app.route('/register', methods =['GET', 'POST'])
def register():
  msg = ''
  print("Break point1")
  if request.method == 'POST' :
    username= request.form['username']
    email = request.form['email']
    password =request.form['password']
    print("Break point2" + "name: " + username + "------" + email + "  ------- " + password)
    try:
      print("Break point3")
      connectionID = ibm_db_dbi.connect(conn_str, '', '')
      cursor = connectionID.cursor()
      print("Break point4")
    except:
      print("No connection Established")
    sql = "SELECT * FROM register WHERE username = ?"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.execute(stmt)
    result = ibm_db.execute(stmt)
    account = ibm_db.fetch_row(stmt)
    param = "SELECT * FROM register WHERE username = '" + "\'" + username + "\'"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    print(" ----")
    dictionary = ibm_db.fetch_assoc(res)
    while dictionary != False:
      print("The ID is : ", dictionary["USERNAME"])
      dictionary = ibm_db.fetch_assoc(res)
    print("break point 6")
    if account:
      msg = 'Username already exists !'
    elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
      msg = 'Invalid email address !'
 elif not re.match(r'[A-Za-z0-9]+', username):
      msg = 'name must contain only characters and numbers !'
```

```
    else:
        sql2 = "INSERT INTO register (username, email,password) VALUES (?, ?, ?)"
        stmt2 = ibm_db.prepare(ibm_db_conn, sql2)
        ibm_db.bind_param(stmt2, 1, username)
        ibm_db.bind_param(stmt2, 2, email)
        ibm_db.bind_param(stmt2, 3, password)
        ibm_db.execute(stmt2)
        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(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.bind_param(stmt, 2, password)
    result = ibm_db.execute(stmt)
    account = ibm_db.fetch_row(stmt)
    param = "SELECT * FROM register WHERE username = " + "\'" + username + "\'" + " and password = " +
"\'" + password + "\'"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    if account:
        session['loggedin'] = True
        session['id'] = dictionary["USERNAME"]
        userid = dictionary["USERNAME"]
        session['password'] = dictionary["PASSWORD"]
        session['email'] = dictionary["EMAIL"]
        return redirect('/home')
    else:
```

```
        msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)
#ADDING --- DATA
@app.route("/add")
def adding():
    return render_template('add.html')
@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():
    date = request.form['date']
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category  =  request.form['category']
    p1 = date[0:10]
    p2 = date[11:13]
    p3 = date[14:]
    p4 = p1 + "-" + p2 + "." + p3 + ".00"
    sql = "INSERT INTO expenses (userid, date, expensename, amount, paymode, category) VALUES (?, ?, ?,
?, ?, ?)"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, session['id'])
    ibm_db.bind_param(stmt, 2, p4)
    ibm_db.bind_param(stmt, 3, expensename)
    ibm_db.bind_param(stmt, 4, amount)
    ibm_db.bind_param(stmt, 5, paymode)
    ibm_db.bind_param(stmt, 6, category)
    ibm_db.execute(stmt)
    # email part
    param = "SELECT * FROM expenses WHERE userid = '' + "\'" + session['id'] + "\' AND MONTH(date) =
MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
```

```
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)
        print(dictionary)
    total= 0
    for x in expense:
         total= total + x[3]
    param = "SELECT userid, limitss FROM limits WHERE userid =  " + "\'" + session['id'] + "\' ORDER BY limitss
DESC LIMIT 1"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = 0
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMITSS"])
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[0]
    if total > int(s):
        msg = "Hello " + session['username'] + " , " + "you have crossed the monthly limit of Rs. " + s + "/- !!!" +
"\n" + "Thank you, " + "\n" + "Team Personal Expense Tracker."
    return redirect("/display")
#DISPLAY---graph
@app.route("/display")
def display():
    print(session['email'],session['id'])
 param = "SELECT * FROM expenses WHERE userid = " + "\'" + session['id'] + "\' ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
```

```
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
    return render_template('display.html' ,expense = expense)
#delete---the--data
@app.route('/delete/<string:userid>', methods = ['POST', 'GET' ])
def delete(id):
    param = "DELETE FROM expenses WHERE userid = " + userid
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    print('deleted successfully')
    return redirect("/display")
#UPDATE---DATA
@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    param = "SELECT * FROM expenses WHERE userid = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        row.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)
 print(row[0])
    return render_template('edit.html', expenses = row[0])
@app.route('/update/<id>', methods = ['POST'])
def update(id):
  if request.method == 'POST' :
      date = request.form['date']
```

```
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category  =  request.form['category']
    p1 = date[0:10]
    p2 = date[11:13]
    p3 = date[14:]
    p4 = p1 + "-" + p2 + "." + p3 + ".00"
    sql = "UPDATE expenses SET date = ? , expensename = ? , amount = ?, paymode = ?, category = ?
WHERE id = ?"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, p4)
    ibm_db.bind_param(stmt, 2, expensename)
    ibm_db.bind_param(stmt, 3, amount)
    ibm_db.bind_param(stmt, 4, paymode)
    ibm_db.bind_param(stmt, 5,category)
    ibm_db.bind_param(stmt, 6, id)
    ibm_db.execute(stmt)
    print('successfully updated')
    return redirect("/display")
 #limit
@app.route("/limit" )
def limit():
     return redirect('/limitn')
@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
   if request.method == "POST":
      number= request.form['number']
      sql = "INSERT INTO limits (userid, limitss) VALUES (?, ?)"
      stmt = ibm_db.prepare(ibm_db_conn, sql)
      ibm_db.bind_param(stmt, 1, session['id'])
      ibm_db.bind_param(stmt, 2, number)
      ibm_db.execute(stmt)
      return redirect('/limitn')
@app.route("/limitn")
def limitn():
   param = "SELECT userid, limitss FROM limits WHERE userid = " + "\'" + session['id'] + "\'ORDER BY limitss
DESC LIMIT 1"
```

```
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = " /-"
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMITSS"])
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[0]
    return render_template("limit.html" , y= s)
#REPORT
@app.route("/today")
def today():
    param1 = "SELECT TIME(date) as tn, amount FROM expenses WHERE userid = " + "\'" + session['id'] +
"\'" #AND DATE(date) = DATE(current timestamp) ORDER BY date DESC"
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpense = []
    while dictionary1 != False:
        temp = []
        temp.append(dictionary1["TN"])
        temp.append(dictionary1["AMOUNT"])
        texpense.append(temp)
        print(temp)
        dictionary1 = ibm_db.fetch_assoc(res1)
    param = "SELECT * FROM expenses WHERE userid = " + "\'" + session['id'] + "\'"  # AND DATE(date) =
DATE(now()) ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    print(dictionary)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
```

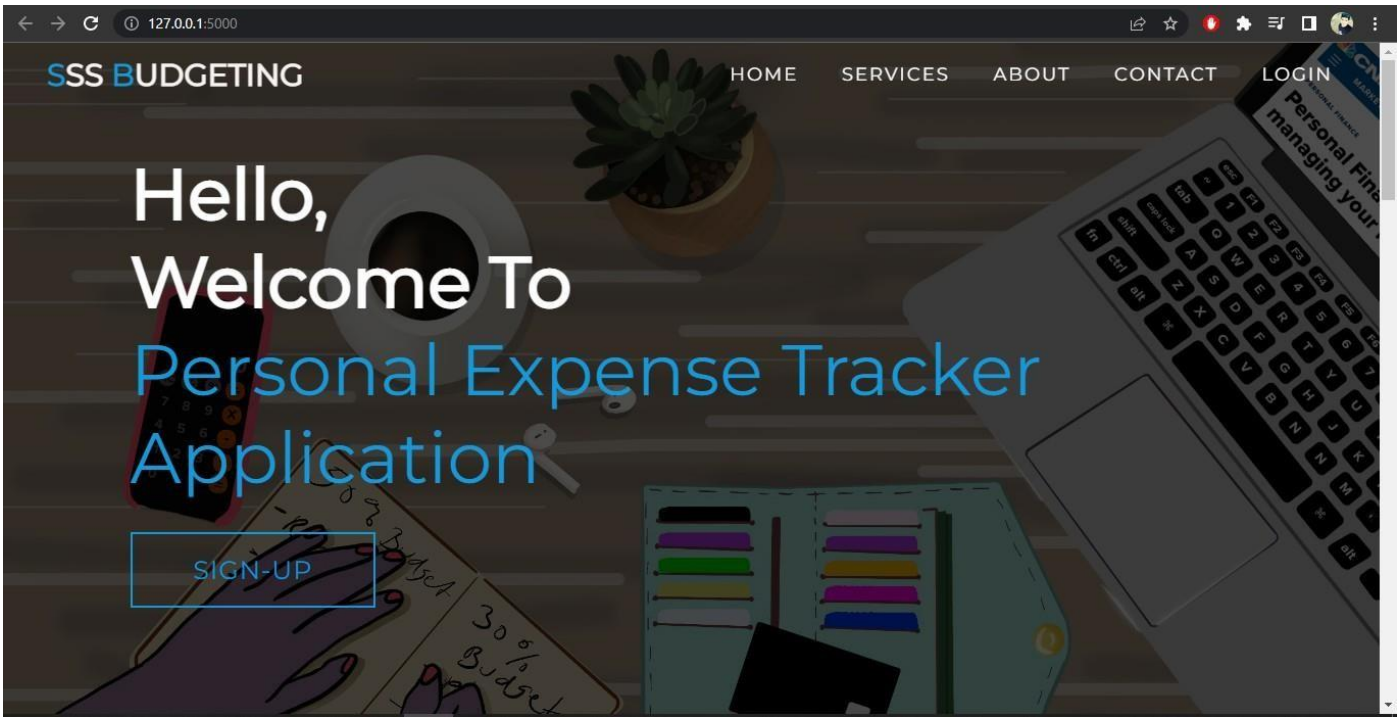
```
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0
    for x in expense:
      print(x[3])
      total += x[3]
      if x[5] == "food                                    ":
         t_food += x[3]
      elif x[5] == "entertainment                                  ":
         t_entertainment += x[3]
      elif x[5] == "business                                  ":
         t_business += x[3]
      elif x[5] == "rent                                    ":
        t_rent += x[3]
      elif x[5] == "EMI                                    ":
       t_EMI += x[3]
      elif x[5] == "other                                    ":
       t_other += x[3]
    return render_template("today.html", texpense = texpense, expense = expense, total = total ,
                    t_food = t_food,t_entertainment = t_entertainment,
                    t_business = t_business, t_rent = t_rent,
                    t_EMI = t_EMI, t_other = t_other )
@app.route("/month")
def month():
    param1 = "SELECT DATE(date) as dt, SUM(amount) as tot FROM expenses WHERE userid =  " + "\'" +
session['id'] + "\' AND MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current
timestamp) GROUP BY DATE(date) ORDER BY DATE(date)"
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpense = []
```

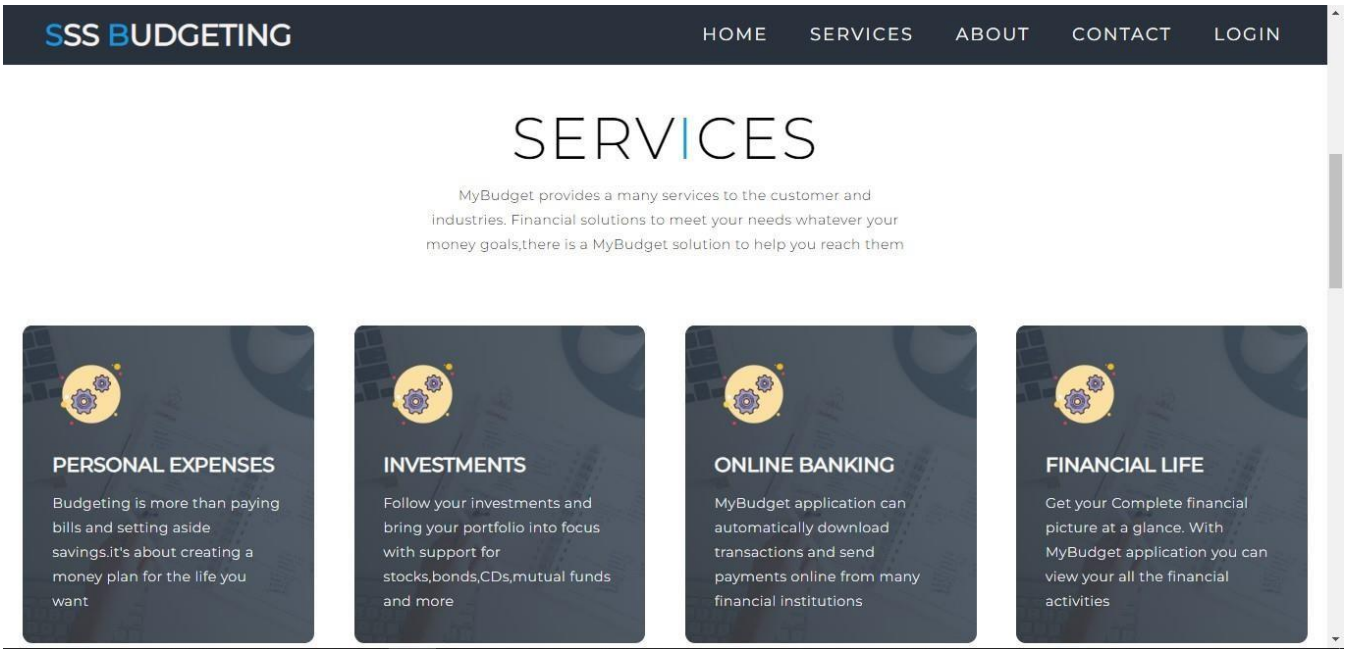
```
    while dictionary1 != False:
        temp = []
        temp.append(dictionary1["DT"])
        temp.append(dictionary1["TOT"])
        texpense.append(temp)
        print(temp)
        dictionary1 = ibm_db.fetch_assoc(res1)
    param = "SELECT * FROM expenses WHERE userid =  " + "\'" + session['id'] + "\'" #AND MONTH =
MONTH(current timestamp) AND YEAR = YEAR(current timestamp) ORDER BY date
    DESCres = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    print(dictionary)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0
    for x in expense:
      print(x[3])
      total += x[3]
      if x[5] == "food                                    ":
          t_food += x[3]
      elif x[5] == "entertainment                                  ":
          t_entertainment += x[3]
      elif x[5] == "business                                  ":
```

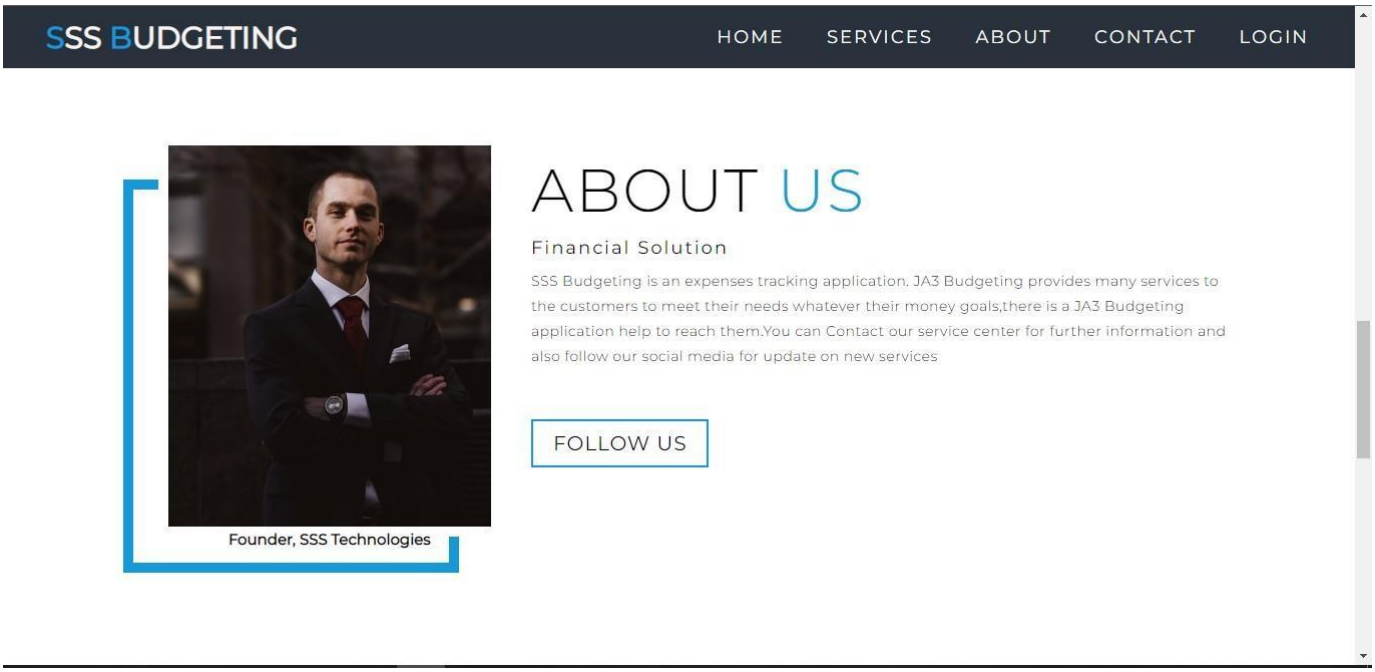
```
        t_business += x[3]
      elif x[5] == "rent                    ":
        t_rent += x[3]
      elif x[5] == "EMI                     ":
        t_EMI += x[3]
      elif x[5] == "other                   ":
        t_other += x[3]
    return render_template("today.html", texpense = texpense, expense = expense, total = total ,
                    t_food = t_food,t_entertainment = t_entertainment,
                    t_business = t_business, t_rent = t_rent,
                    t_EMI = t_EMI, t_other = t_other )
@app.route("/year")
def year():
    param1 = "SELECT * FROM expenses WHERE userid = " + "\'" + session['id'] + "\'" #AND YEAR(date) =
YEAR(current timestamp) GROUP BY MONTH(date) ORDER BY MONTH(date)
    res = ibm_db.exec_immediate(ibm_db_conn,
    param1)dictionary = ibm_db.fetch_assoc(res)
    print(dictionary)
    expense = []
    while dictionary != False:
      temp = []
      temp.append(dictionary["USERID"])
      temp.append(dictionary["DATE"])
      temp.append(dictionary["EXPENSENAME"])
      temp.append(dictionary["AMOUNT"])
      temp.append(dictionary["PAYMODE"])
      temp.append(dictionary["CATEGORY"])
      expense.append(temp)
      dictionary = ibm_db.fetch_assoc(res)
    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0
    for x in expense:
      print(x[3])
```

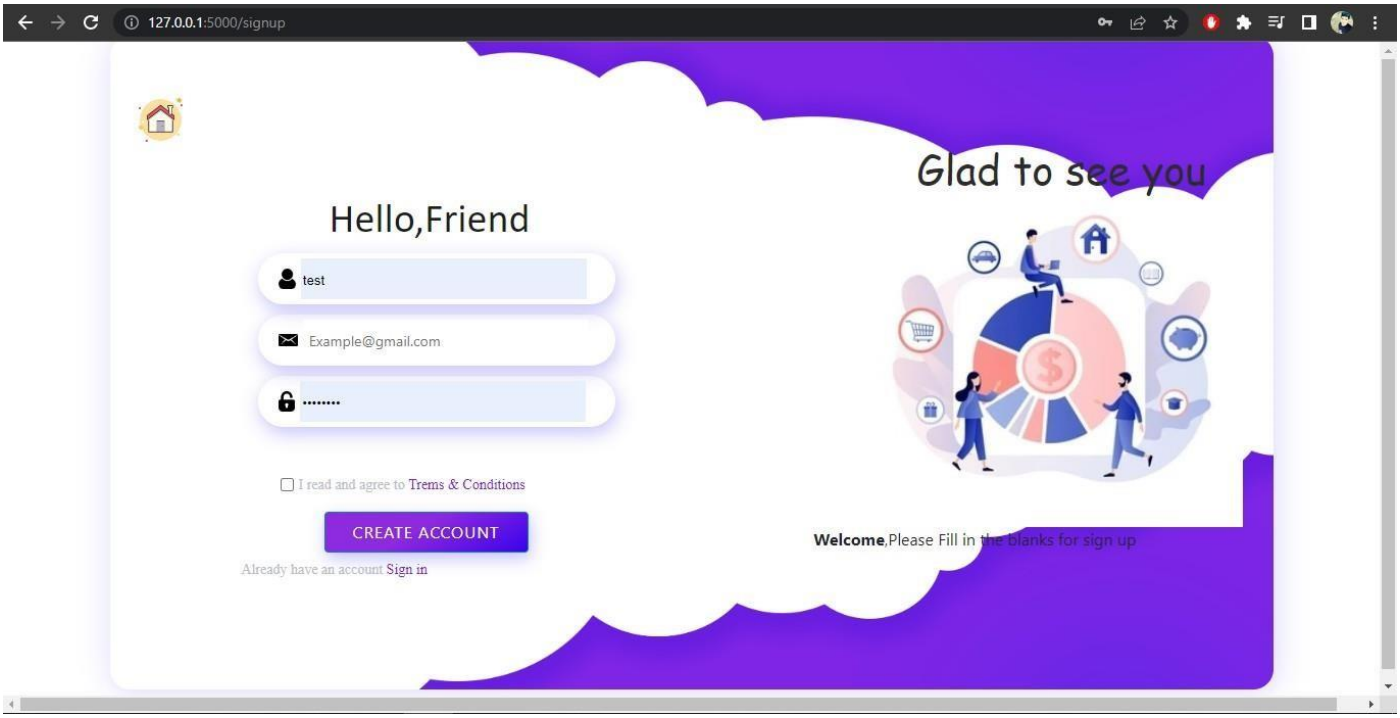
```
        total += x[3]
        if x[5] == "food                                    ":
            t_food += x[3]
        elif x[5] == "entertainment                                  ":
            t_entertainment += x[3]
        elif x[5] == "business                                    ":
            t_business += x[3]
        elif x[5] == "rent                                    ":
            t_rent += x[3]
        elif x[5] == "EMI                                    ":
            t_EMI += x[3]
        elif x[5] == "other                                    ":
            t_other += x[3]
    return render_template("today.html", texpense = total, expense = expense, total = total ,
                    t_food = t_food,t_entertainment = t_entertainment,
                    t_business = t_business, t_rent = t_rent,
                    t_EMI = t_EMI, t_other = t_other )
#log-out
@app.route('/logout')
def logout():
  session.pop('loggedin', None)
  session.pop('id', None)
  session.pop('username', None)
  session.pop('email', None)
  return render_template('home.html')
port = os.getenv('VCAP_APP_PORT', '8080')
if __name__ == "__main__":
   app.secret_key = os.urandom(12)
   app.run(debug=True, host='0.0.0.0', port=port)
```

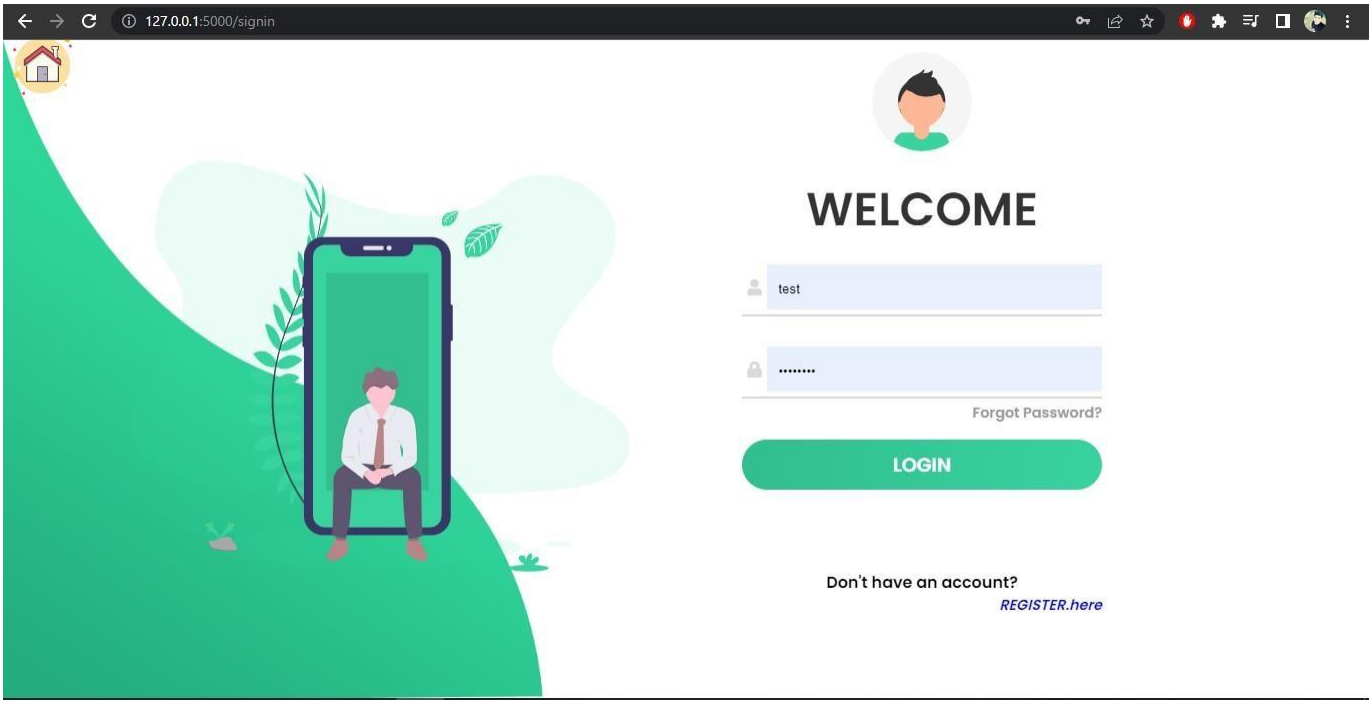
OUTPUT SCREENSHOTS:

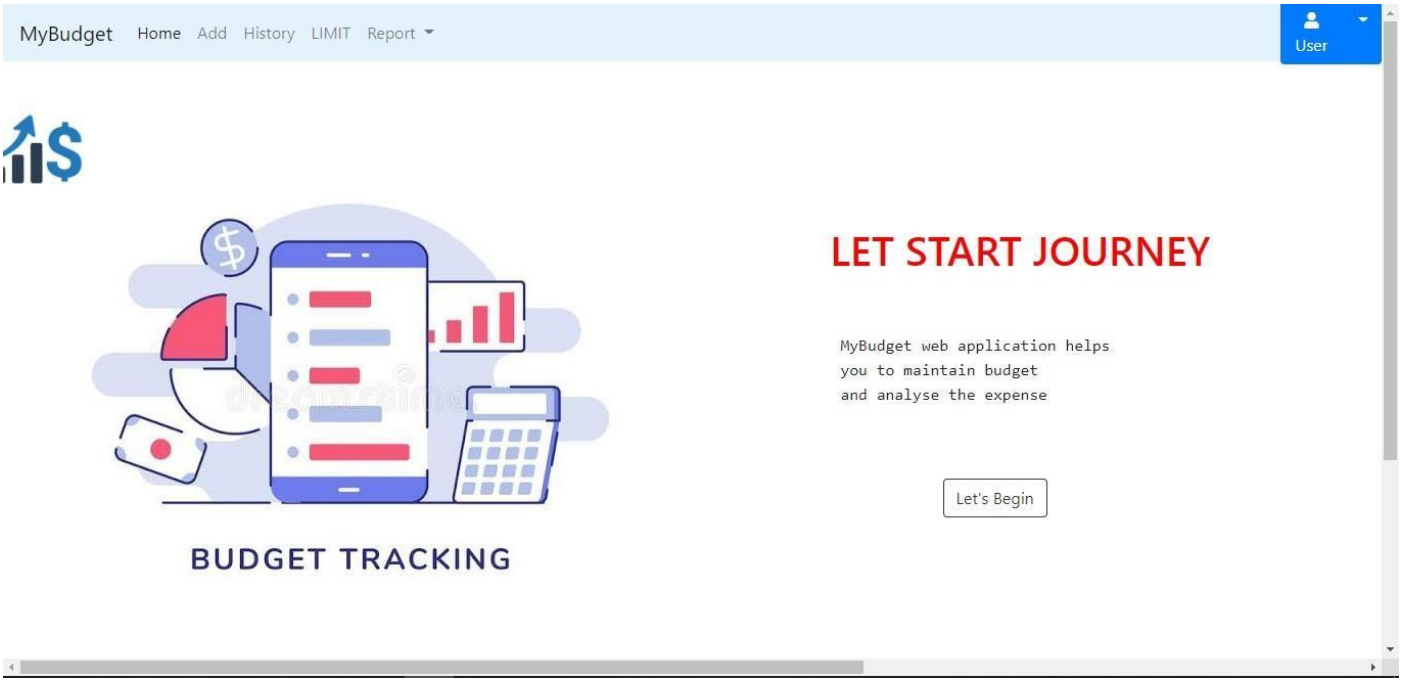


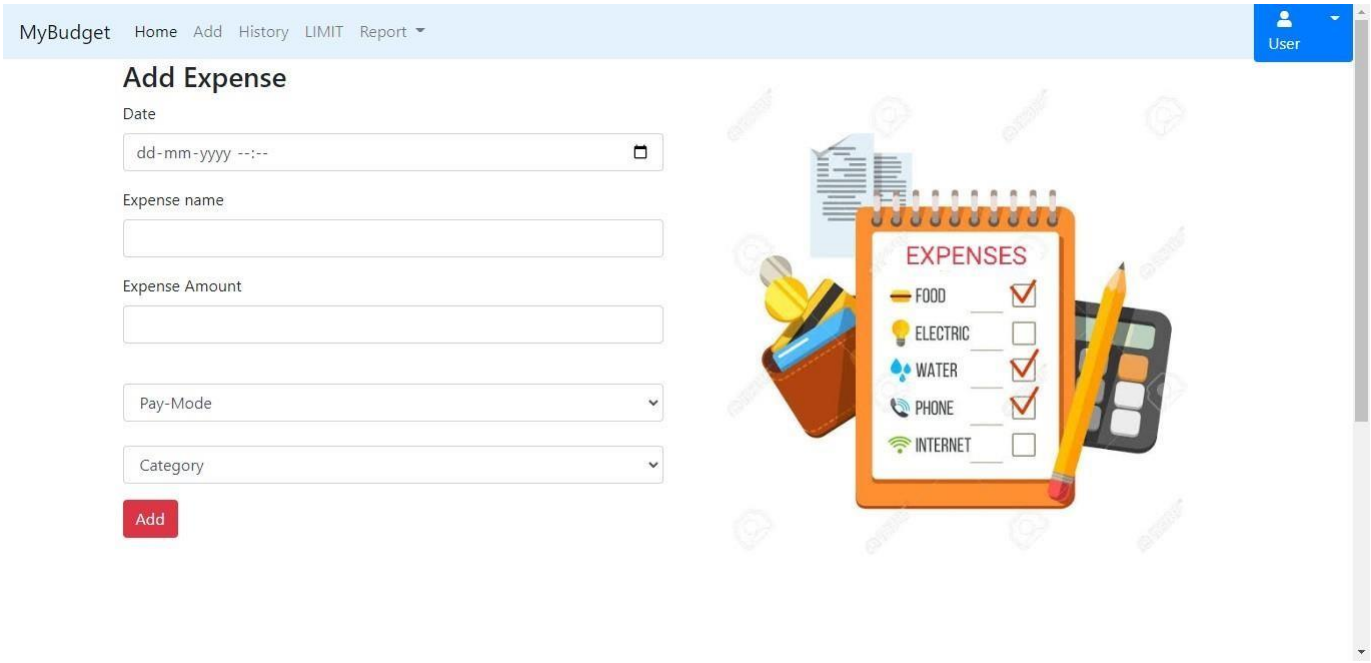


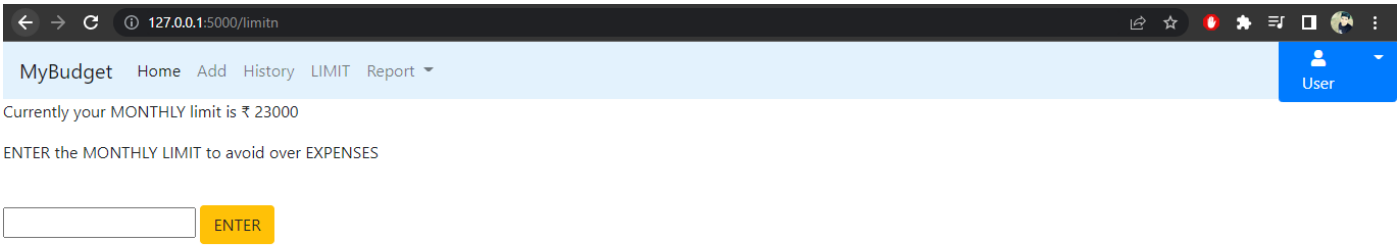

SSS BUDGETING
HOME
SERVICES
ABOUT
CONTACT
LOGIN
Founder, SSS Technologies
ABOUT US
Financial Solution
SSS Budgeting is an expenses tracking application. JA3 Budgeting provides many services to the customers to meet their needs whatever their money goals,there is a JA3 Budgeting application help to reach them.You can Contact our service center for further information and also follow our social media for update on new services
FOLLOW US

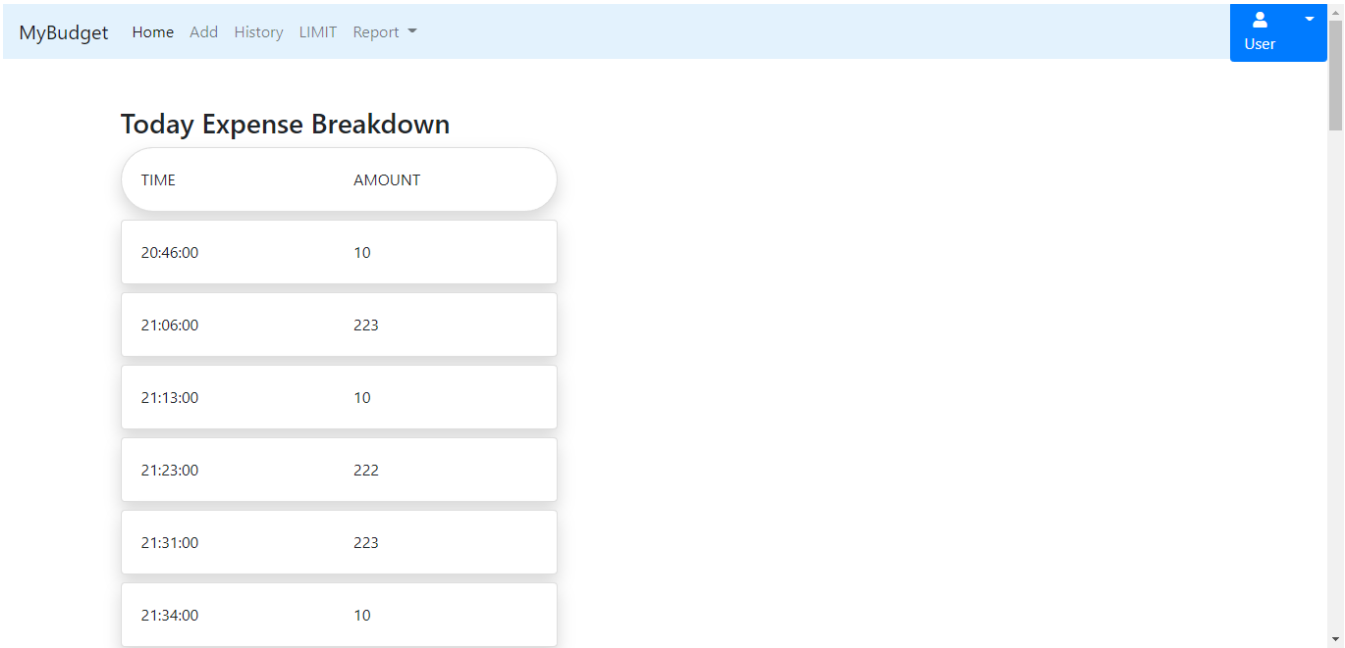

127.0.0.1:5000/signup
Hello,Friend
test
Example@gmail.com
I read and agree to Trems & Conditions
CREATE ACCOUNT
Already have an account Sign in
Glad to see you
Welcome,Please Fill in the blanks for sign up

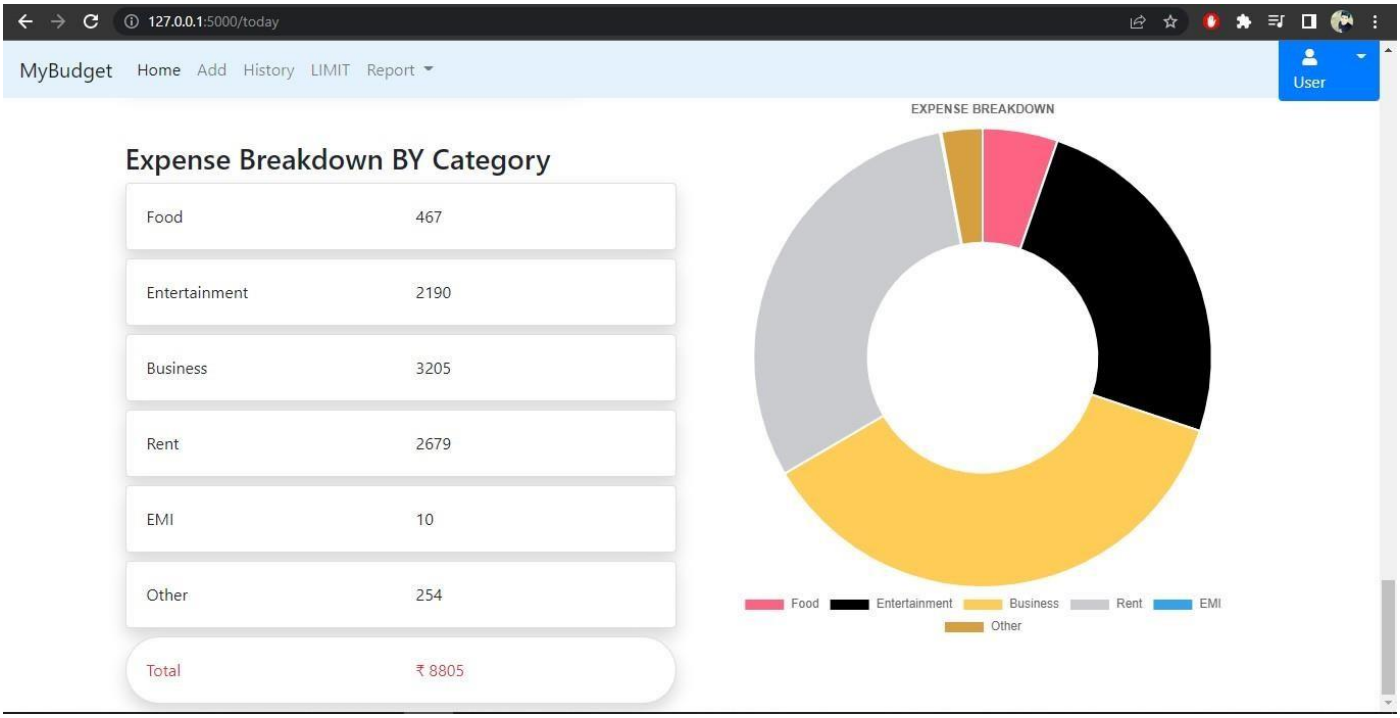

127.0.0.1:5000/signin
WELCOME
test
Forgot Password?
LOGIN
Don't have an account?
REGISTER.here

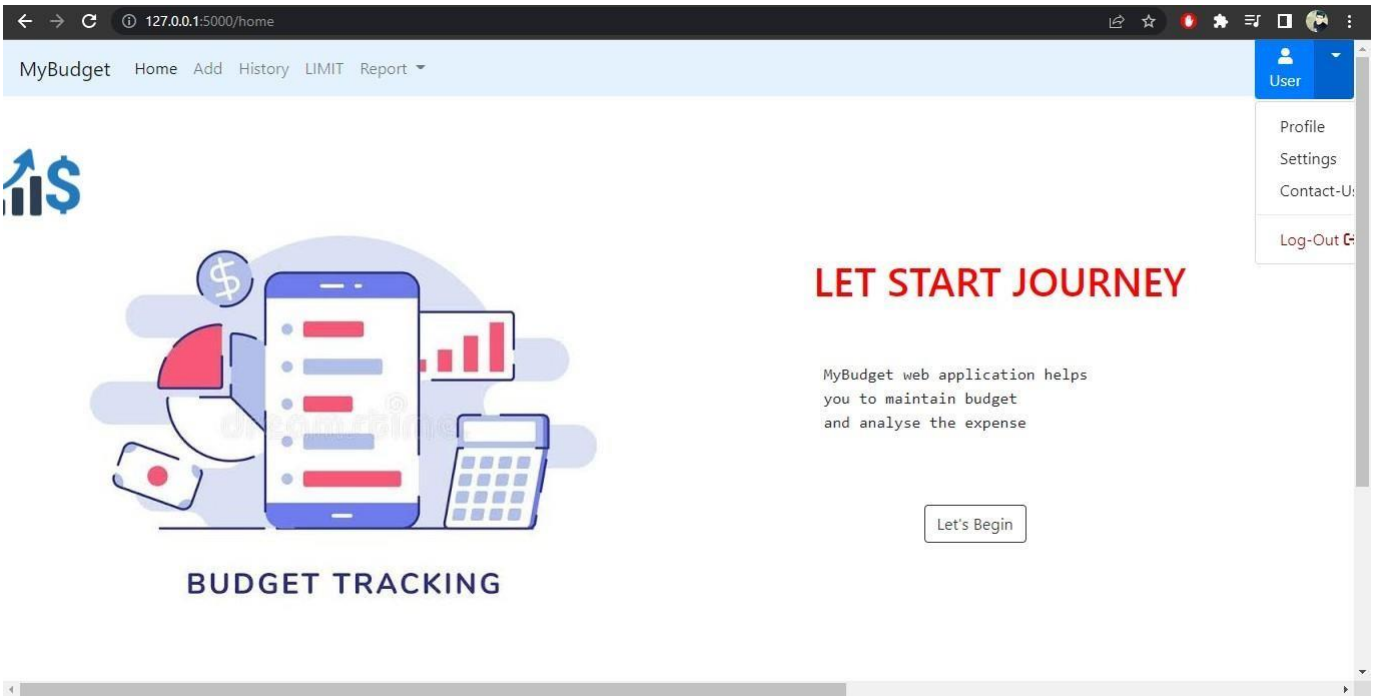

MyBudget
Home
Add
History
LIMIT
Report
User
LET START JOURNEY
MyBudget web application helps
you to maintain budget
and analyse the expense
Let's Begin
BUDGET TRACKING


MyBudget
Home
Add
History
LIMIT
Report
User
Add Expense
Date
dd-mm-yyyy --:--
Expense name
Expense Amount
Pay-Mode
Category
Add
EXPENSES
FOOD
ELECTRIC
WATER
PHONE
INTERNET


127.0.0.1:5000/limitn
MyBudget
Home
Add
History
LIMIT
Report
User
Currently your MONTHLY limit is ₹ 23000
ENTER the MONTHLY LIMIT to avoid over EXPENSES
ENTER


MyBudget
Home
Add
History
LIMIT
Report
User
Today Expense Breakdown
TIME
AMOUNT
20:46:00
10
21:06:00
223
21:13:00
10
21:23:00
222
21:31:00
223
21:34:00
10


127.0.0.1:5000/today
MyBudget
Home
Add
History
LIMIT
Report
User
Expense Breakdown BY Category
Food 467
Entertainment 2190
Business 3205
Rent 2679
EMI 10
Other 254
Total ₹ 8805
EXPENSE BREAKDOWN
Food
Entertainment
Business
Rent
EMI
Other


127.0.0.1:5000/home
MyBudget
Home
Add
History
LIMIT
Report
User
Profile
Settings
Log-Out
LET START JOURNEY
MyBudget web application helps
you to maintain budget
and analyse the expense
Let's Begin
BUDGET TRACKING

13.2 GITHUB LINK:

LINK : https://github.com/IBM-EPBL/IBM-Project-7368-1658853592

13.3 PROJECT DEMO

LINK:Demo