IBM - Project
Project Documentation Report

Project Name: Finance-Tracker-IBM

Team ID: PNT2022TMID27267

Team Members:

- Kiridharan [Team Lead]
- Edwin Binu
- Kiran Babu
- Kurinjilan

Github Link : https://github.com/IBM-EPBL/IBM-Project-20186-1659714254.git

1. INTRODUCTION:

1.1 Project Overview:

Personal expense tracking application. Persona expense tracker is required to maintain budget & get useful insights about the expenses. By understanding what you spend money on and how much spend, you can see exactly where your cash is going and areas where you can cut back. The app categorize your expense as needs/ wants to helps you get a good idea of your purchasing behavior

1.2 Purpose:

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

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

2. LITERATURE SURVEY:

2.1 Existing Solutions:

Aman Garg, Mukul Goel , Sagar Mittal , Mr. Shekhar Singh : This Expense Tracker is a web application that facilitates the users to keep track and manage their personal as well as business expenses. Using paper is not easy to manage. It is common to delete files accidentally or misplace files. This expense tracker provides a complete digital solution to this problem. Excel sheets do very little to help in tracking expenses. Furthermore, they don't have the advanced functionality of preparing graphical visuals automatically. Not only will it save the time of the people but also it will assure error-free calculations. The user just has to enter the income and expenditures and everything else will be performed by the system. Keywords: Expense Tracker, budget, planning, savings, graphical visualization of expenditure

Girish Bekaroo and Sameer Sunhaloo
School of Business Informatics and Software Engineering,
University of Technology, Mauritius:

We present an intelligent online budget tracker (GeniusIOBT.com) to efficiently

manage house-
hold a budget. Our system will help to plan and track household-budget related issues where mem-
bers of the system can securely access it anytime from anywhere via the Internet. The Intelligent
Online Budget Tracker not only keeps track of the budget but also provides means to analyze data
via charts and graphs as well as intelligently predicting future budgets and issues like bankruptcy.
Keywords: intelligent online budget tracker, household budget, data analysis

S. Chandini, T. Poojitha, D. Ranjith, V.J. Mohammed Akram, M.S. Vani, V. Rajyalakshmi - Income and Expense Tracker will maintain data of daily, weekly, monthly, yearly expenses, Manage your expenses and earnings in a simple and intuitive way. User can select category of expense, enter other information like user can
capture photos, 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, we can reduce the manual calculations for their expenses and keep the track of the
expenditure. And user can enter his monthly income or limit of monthly
Expense in this tr. This tracker system provides an integrated set of features to help you to manage your expenses and cash flow.

2.2 References:

[1] Rami M. Mohammad, Fadi Thabtah, Lee McCluskey: An Assessment of Features Related to Phishing Websites using an Automated Technique:In The 7th International Conference for Internet Technology and Secured Transactions,IEEE,2012

[2] Ahmad Abunadi, Anazida Zainal ,Oluwatobi Akanb: Feature Extraction Process: A Phishing Detection Approach :In IEEE,2013.

[3] Mustafa AYDIN, Nazife BAYKAL : Feature Extraction and Classification Phishing Websites Based on URL : IEEE,2015

[4] Chunlin Liu, Bo Lang : Finding effective classifier for malicious URL detection : InACM,2018

2.3 Problem Statement Definition:

1. User Registration and Creation:

This program will feature a user login screen and different options for enlisting, just like the vast majority of applications. When a user is using something for the first time, they should sign up for this application. However, the customer who has now registered can access the application using the login credentials they created at the time of registration.

2. Adding Income and Expenses:

This application will let you select the different types or categories of income or expenses. Every application user has the opportunity to input incomes and expenses in the appropriate amounts. Each record should include information such as the date the item occurred and its specifics.

3. Category Master:

This module fundamentally relies upon the SQLite for putting away classification details and expense subtleties and income. The class exchange is put away in a SQlite database.

4. Management View- Date Wise:

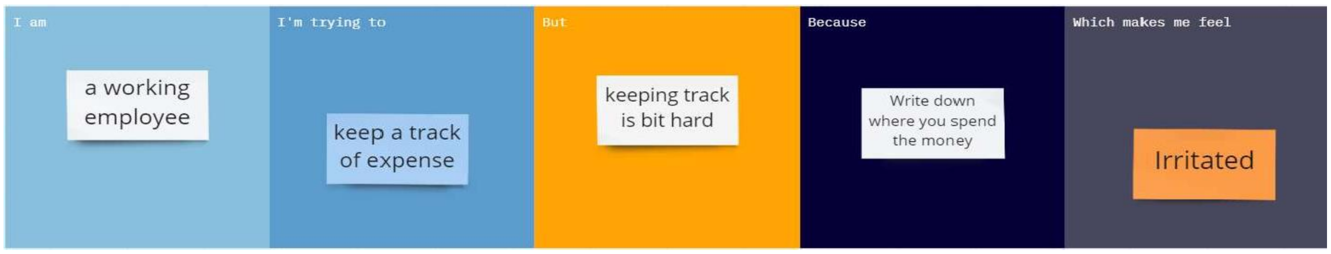
According to the predetermined date insightful in this module, the expenses are recorded. Our varied expenses are seen as a breakdown of exchanges classes by recovering all the income and spending nuances. By using SQL lite queries and the saw in the advanced cell, the income and expenses are recovered.

5. Management View- Category Wise:

According to the predetermined classification used in this module, the expenses are recorded. By gathering all the financial and geographical nuances, seen by our system as a list of exchange classes varied costs. The earnings and costs are using SQLite queries, retrieved, and observed in modern cells.

6. Remainder

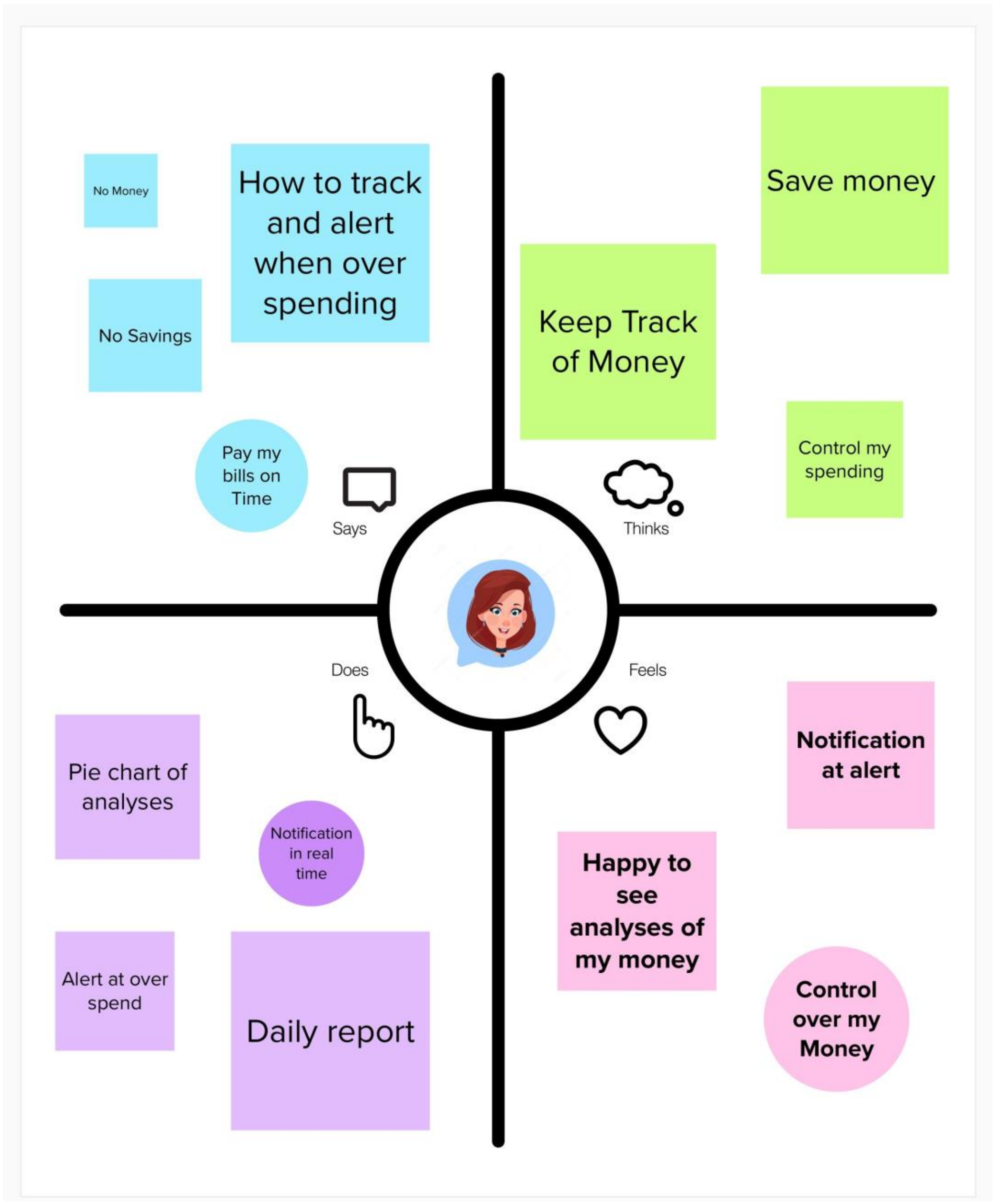
The remainder is a warning module that, upon user recognition, will prompt the user to enter income or expenses on a daily or periodic basis, depending on their requirements.



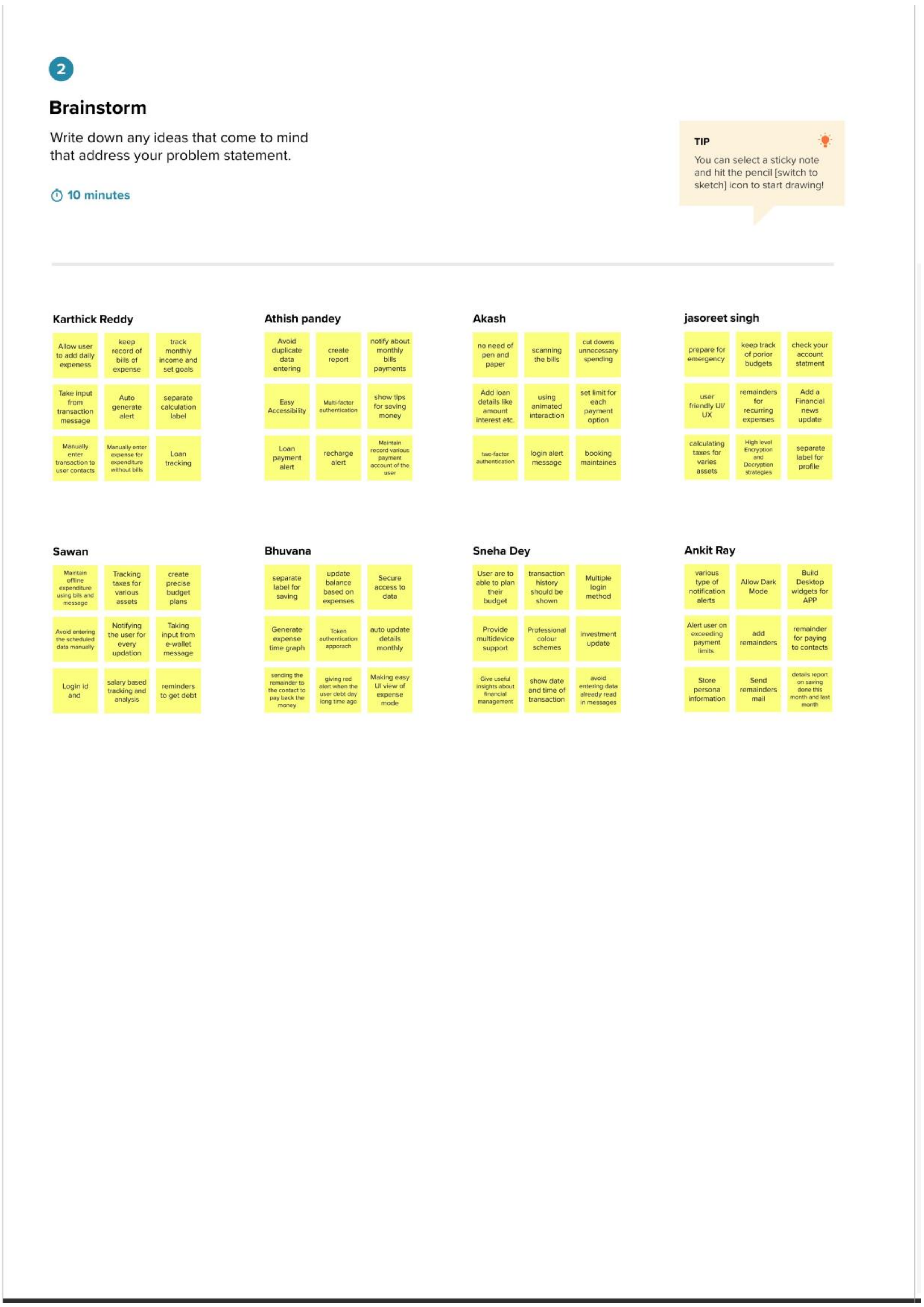
Problem Statement (PS)	**Iam (Customer)**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	A working employee	Save my money	I cant get myself to limit my spending	Write down, where you spend the money	Irritated
PS-2	Business	Organizing the expense	I spend money lavishly	Making calculation	Frustrated

3. IDEATION & PROPOSED SOLUTION:

3.1 Empathy Map:



3.2 Ideation and Brainstorming:



2

Brainstorm

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

10 minutes

TIP

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

Karthick Reddy

Allow user to add daily expeness	keep record of bills of expense	track monthly income and set goals
Take input from transaction message	Auto generate alert	separate calculation label
Manually enter transaction to user contacts	Manually enter expense for expenditure without bills	Loan tracking

Athish pandey

Avoid duplicate data entering	create report	notify about monthly bills payments
Easy Accessibility	Multi-factor authentication	show tips for saving money
Loan payment alert	recharge alert	Maintain record various payment account of the user

Akash

no need of pen and paper	scanning the bills	cut downs unnecessary spending
Add loan details like amount interest etc.	using animated interaction	set limit for each payment option
two-factor authentication	login alert message	booking maintaines

jasoreet singh

prepare for emergency	keep track of porior budgets	check your account statment
user friendly UI/ UX	remainders for recurring expenses	Add a Financial news update
calculating taxes for varies assets	High level Encryption and Decryption strategies	separate label for profile

Sawan

Maintain offline expenditure using bils and message	Tracking taxes for various assets	create precise budget plans
Avoid entering the scheduled data manually	Notifying the user for every updation	Taking input from e-wallet message
Login id and	salary based tracking and analysis	reminders to get debt

Bhuvana

separate label for saving	update balance based on expenses	Secure access to data
Generate expense time graph	Token authentication apporach	auto update details monthly
sending the remainder to the contact to pay back the money	giving red alert when the user debt day long time ago	Making easy UI view of expense mode

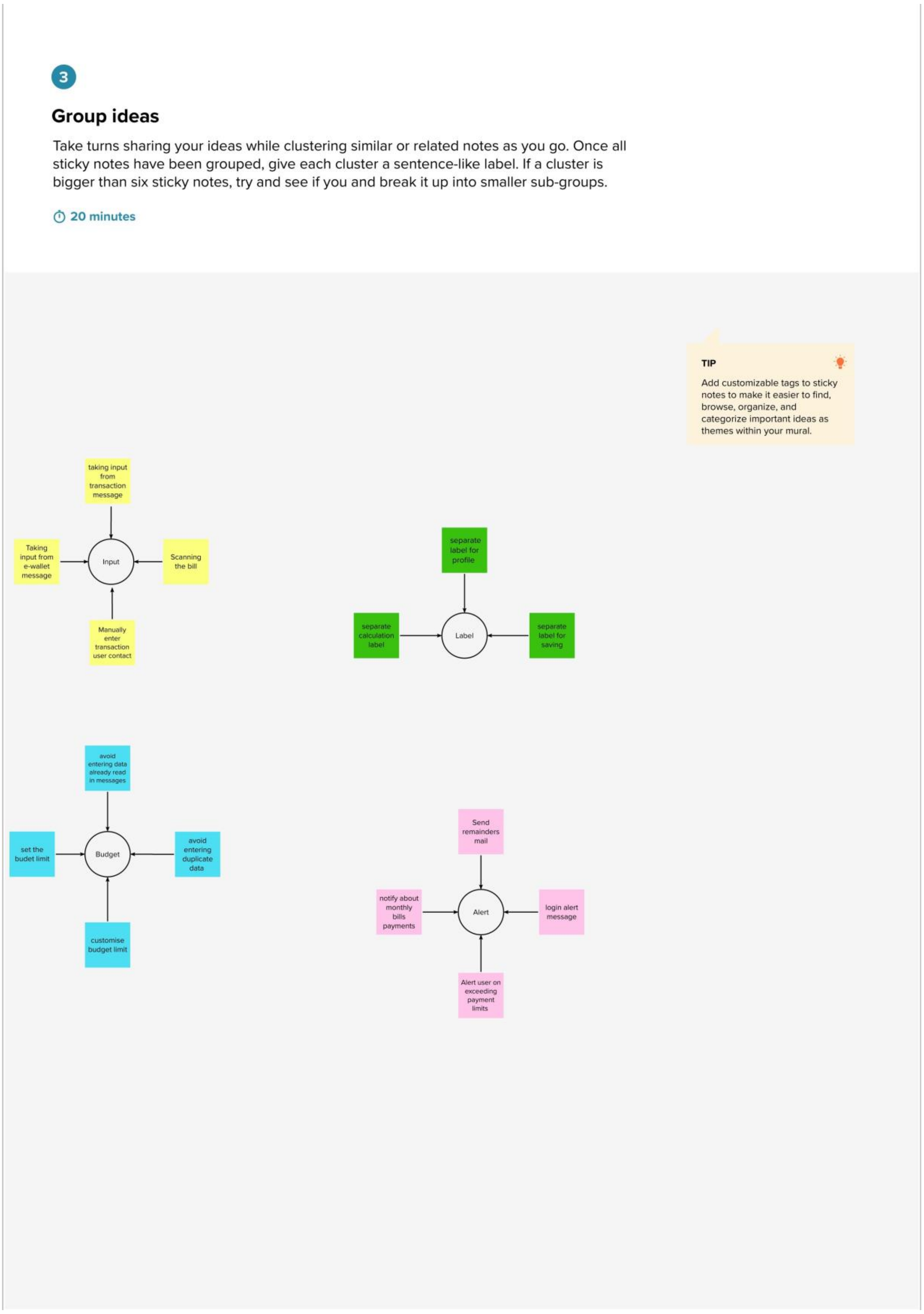
Sneha Dey

User are to able to plan their budget	transaction history should be shown	Multiple login method
Provide multidevice support	Professional colour schemes	investment update
Give useful insights about financial management	show date and time of transaction	avoid entering data already read in messages

Ankit Ray

various type of notification alerts	Allow Dark Mode	Build Desktop widgets for APP
Alert user on exceeding payment limits	add remainders	remainder for paying to contacts
Store persona information	Send remainders mail	details report on saving done this month and last month

Step-2: Brainstorm, Idea Listing and Grouping



3

Group ideas

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

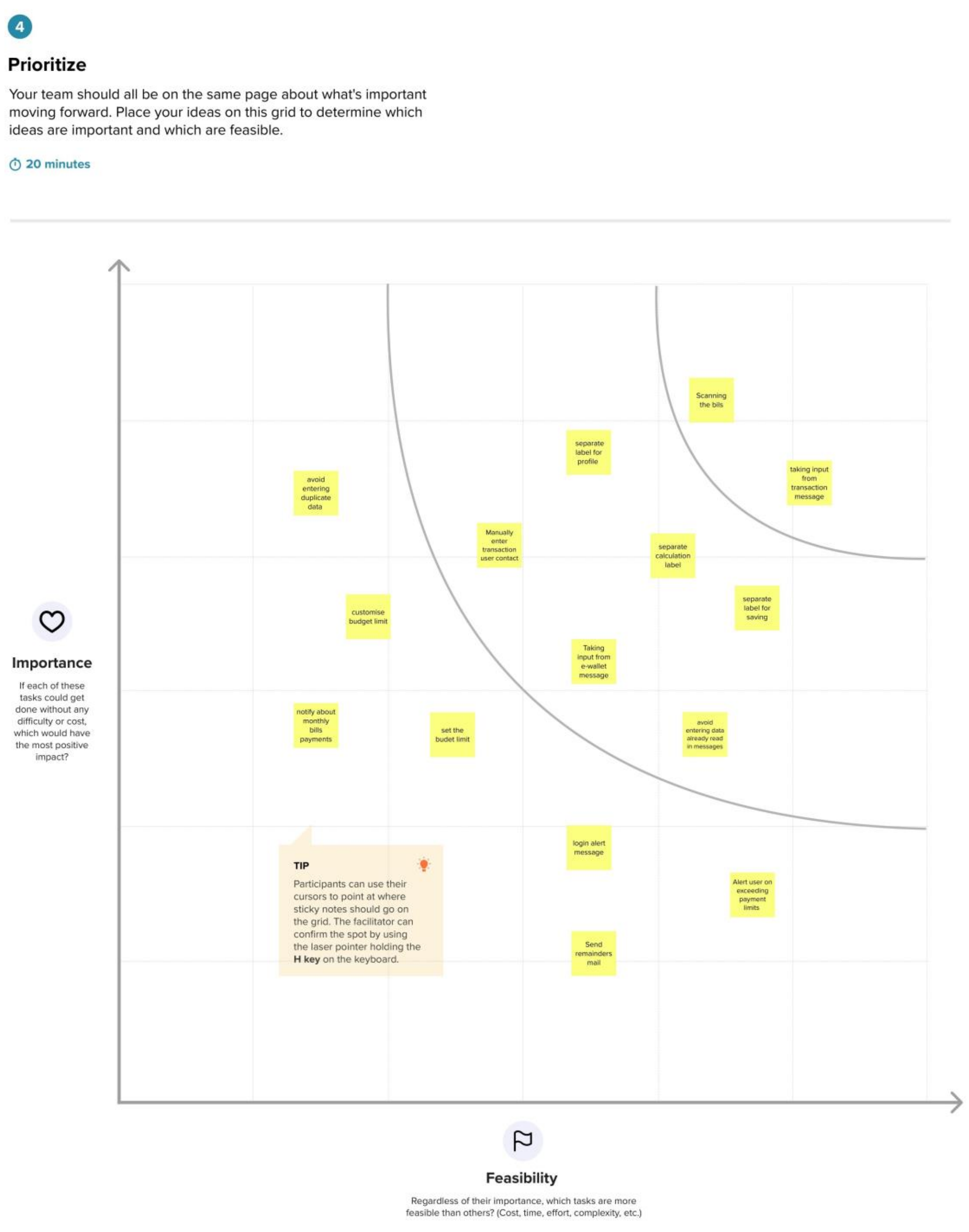
20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

taking input from transaction message
Taking input from e-wallet message
Input
Scanning the bill
Manually enter transaction user contact
separate label for profile
separate calculation label
Label
separate label for saving
avoid entering data already read in messages
set the budet limit
Budget
avoid entering duplicate data
customise budget limit
Send remainders mail
notify about monthly bills payments
Alert
login alert message
Alert user on exceeding payment limits

Step-3: Idea Prioritization


4

Prioritize

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

20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Feasibility

Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

Scanning the bils
separate label for profile
taking input from transaction message
avoid entering duplicate data
Manually enter transaction user contact
separate calculation label
separate label for saving
customise budget limit
Taking input from e-wallet message
notify about monthly bills payments
set the budet limit
avoid entering data already read in messages
login alert message
Alert user on exceeding payment limits
Send remainders mail

3.3 Proposed Solution

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To track daily spending more effectively and conveniently, the online application "Expense Tracker" was created. This tool helps us keep track of our spending and reduces the need for manual daily expense calculations. This application allows the user to enter their income to determine their daily expenses, and the results are saved for each user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert
2.	Idea / Solution description	By avoiding duplicate entries and allowing user to set a customized budget limit so that the application will avoid calculation errors.
3.	Novelty / Uniqueness	Expense Tracker System is a system that will keep a track of Income-Expense on a day-to-day basis, This System takes Income from your House-Wife and divides it into daily expenses allowed If you exceed that day’s expense it will cut it from your income and give new daily expense allowed Amount, and if that day’s expense is less it will add it in savings. The daily expense tracking System will generate the report at the end of the month to show Income-Expense Curve. It will let you add the savings at which you had saved for some particular Festivals or days like Birthdays or Anniversaries.

4.	Social Impact / Customer Satisfaction	• Reducing unwanted expense • Helping people to save money • Maintain budget
5.	Business Model (Revenue Model)	Think of a subscription as a contract between you and the customer. The customer agrees to pay for a service for a period and the business fulfils that offer as long as the customer completes their recurring payments. When the contract is up, the customer has the option to renew or cancel their subscription. So they can subscribe to the plan and get insight of their expense and where they can cut the expense.
6.	Scalability of the Solution	Scalability can be increased by integrating the model within related apps for complex and efficient expense tracking.

3.4 **Problem Solution Fit:**

Define CS, fit into CC

1. CUSTO MER SEGME NT(S)

CS

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

-person who make budget
-person who plan for trip
-person who makes weekly or monthly budget
-persons who makes more expensive than salary

6. CUSTOMER

CC

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

-customers have an account
-customers have a cell phone
-subscriptions

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 dothese solutions have? i.e. pen and paper is an alternative to digital notetaking

Explore AS, differentiate

Budget Bakers: it can be used only in android. QuickBooks: No chat supports

E

What is the real reason that this problem exists? What is

RC

J&P

understand RC

2. JOBS-TO-BE-DONE / PROBLEMS

Which jobs-to-be-done (or problems) do you address for your

customers? There

9. PROB LEM ROOT CAUS

7. BEHAVIOUR

What does your customer do to address the problem and get the job done?

could be more than one; explore different sides.

BE

-People need to have their recorded on the pen and paper

-People can keep the data on single device only and they cannot share the budget with the family members

the back story behind the need to do this job?

i.e. customers have to do it because of the change in regulations.

-People need to have their manually enter the data everything they make a payment.

-People are not going open the app everything they make a payment

While they enter the data, they must remember all the payment they made for the day

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)

-Half of the people make use of pen and paper to keep track.

-people skip half of the payment they made for day

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

3. TRIGGERS

TR

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

-Half of the people make use of pen and paper to keep track.

-people skip half of the payment they made for day

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.

-People don't feel attracted by the expensive tracker

-People will feel free to use the app on their use only they check how make they have spent today

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 inthe canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

-our solution will make the user need not to enter the data by their hand's

-we will provide the two modes

-manual mode were the user needs to enter the data by their hand.

-automatic were the data will be entersystem itself

8. CHANNELS of BEHAVIOUR

CH

Extract online & offline CH of BE

ONLINE

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

-Their daily expense get update to thecloud

-Their can share their budget plan withfriends and family members

OFFLINE

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

-Their daily expense budget calculationwill be done

-Their graphical representation will beshow in offline

4. REQUIREMENT ANALYSIS:

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through Phone number
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Expense planner	Application show the graphical representation of user daily expense
FR-4	Category	Application allow the user to add new categories on their expense
FR-5	Expense tracker	Application show the report on user expense
FR-6	Calendar	Personal expense tracker allow users to add the data to their expense

4.2 Non-functional Requirements:

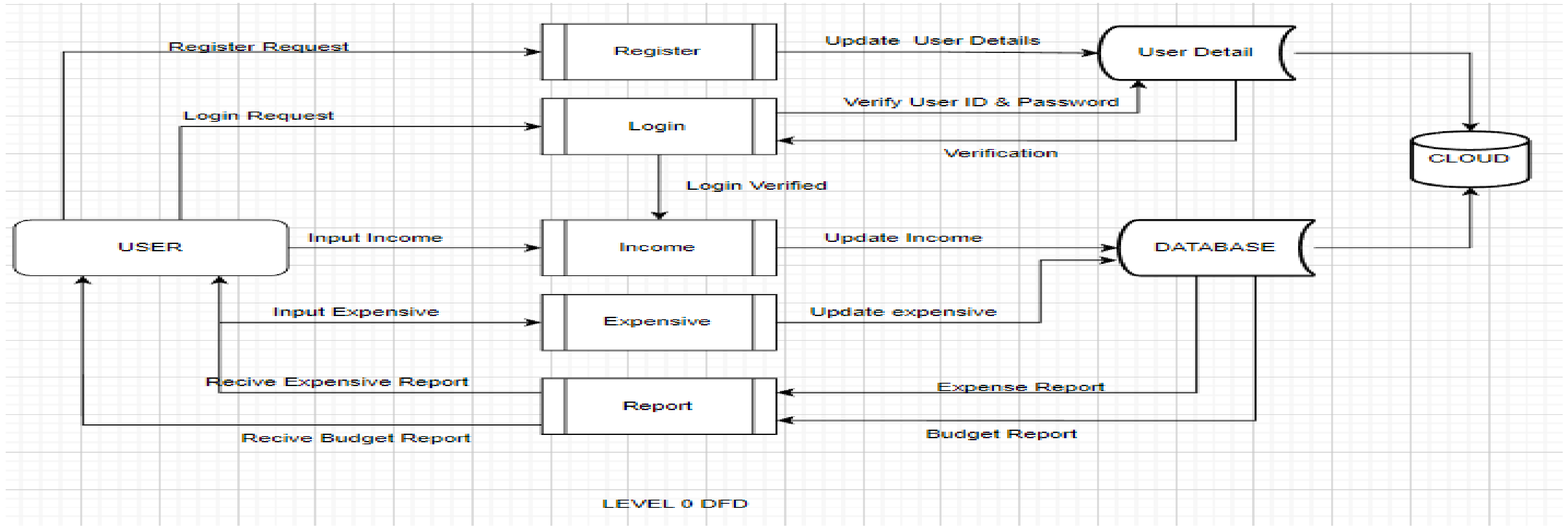
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	To ease the navigation there is a back tab to provide access to pervious page
NFR-2	**Security**	More security of the customer data and bank account details
NFR-3	**Reliability**	Each data record is stored on a effective and secure database. There wouldn’t be any data loss
NFR-4	**Performance**	There are different type of expense are stored in categories along with different option. Faster the database and high throughput of the system is increased due light weight interface
NFR-5	**Availability**	It is available 24*7

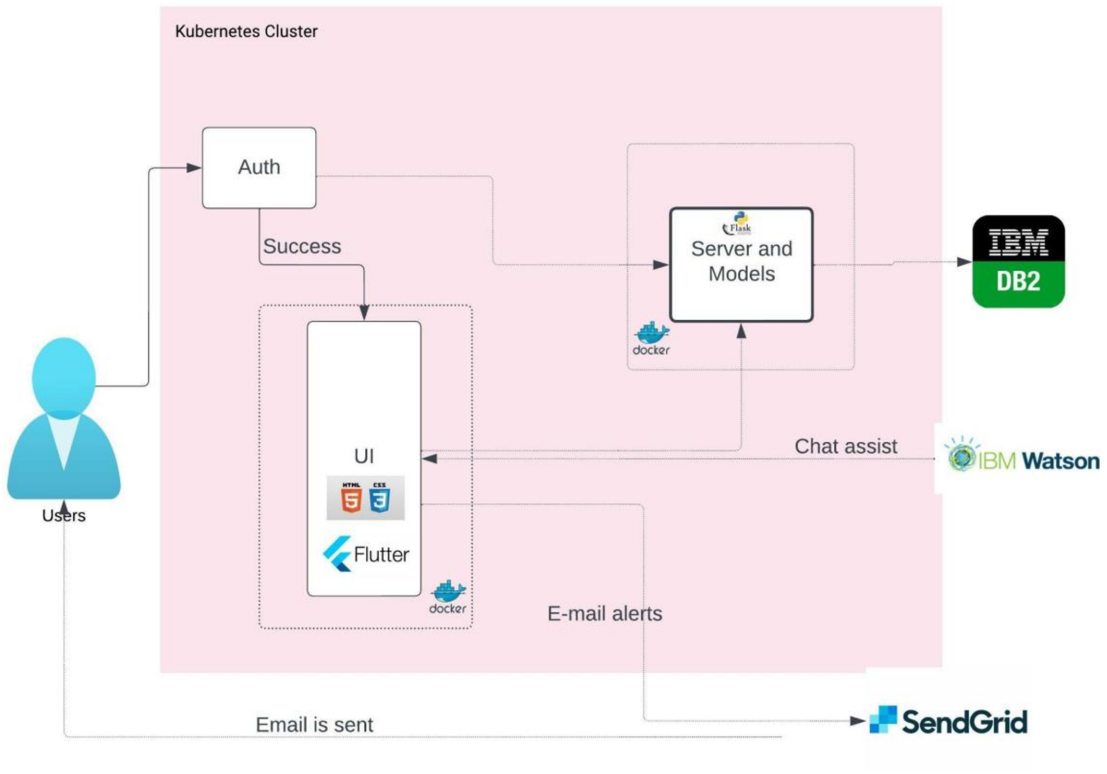
5. PROJECT DESIGN:

5.1 Data Flow Diagram:

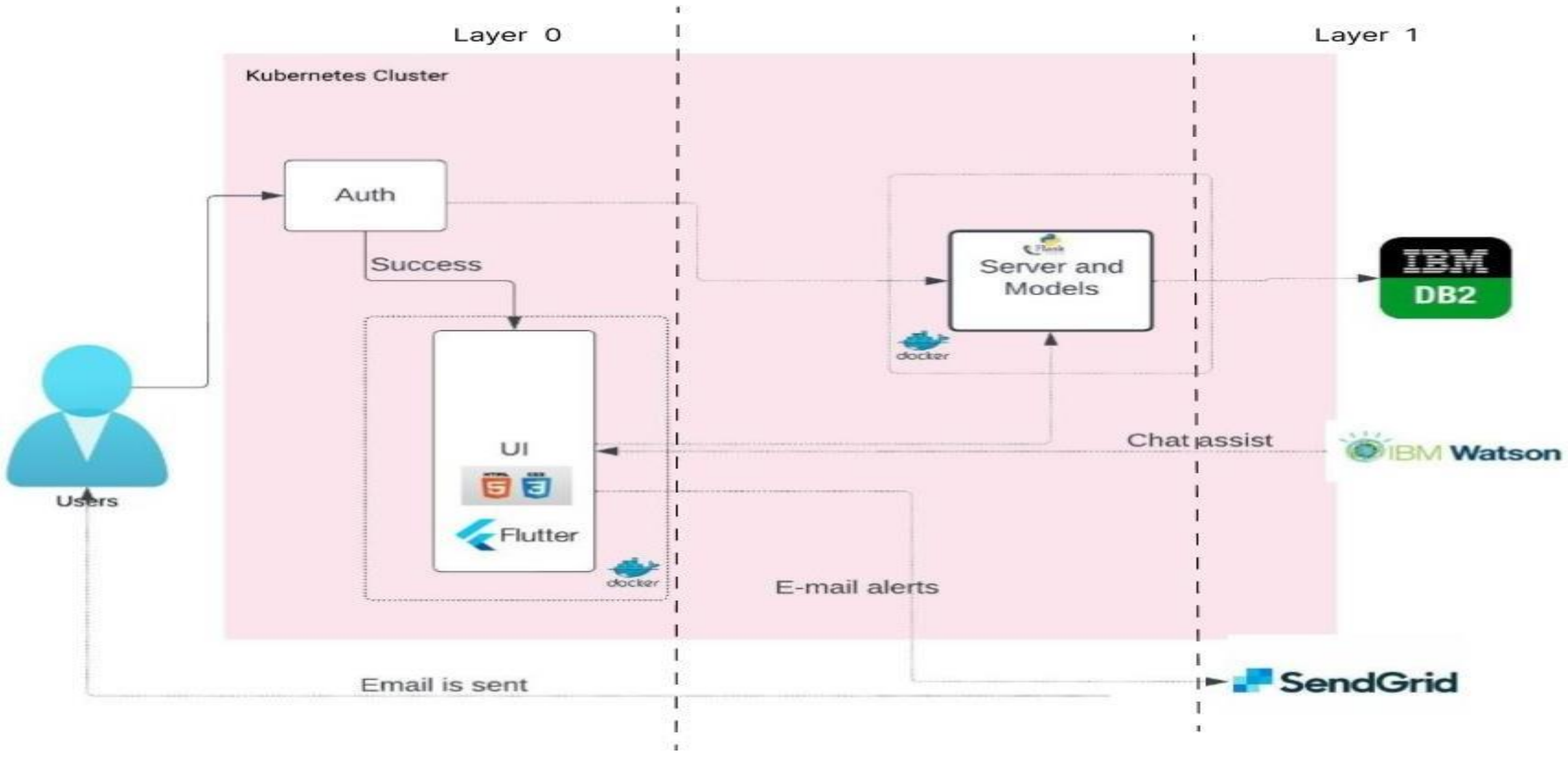
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the rightamount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



5.2 Solution and Technical Architecture Diagram:



Technical Architecture:



5.3 User Stories:

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	Registration	USN-1	As a user, I can register for the application byentering 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 emailonce I have registered for the application	I can receive confirmationemail & click confirm	High	Sprint-1
Customer	Login	USN-1	As a user, I used my Mail id and password forlogin	I can access my account /dashboard	High	Sprint-1
		USN-2	As a user, I forget my password. Used forgetpassword ,	I got verification mail andchanged my password	High	Sprint-2
Customer	Dashboard	USN-1	As a user, there is profile tab	Where I can update/editmy personal details	High	Sprint-3
		USN-2	As a user, there is budget tab	Where I can update/edit/set budget	High	Sprint-2
Customer	Profile	USN-1	As a user, I can change my phone no, mail,name	It get updated	Low	Sprint-2
	Budget	USN-1	As a user, I create a budget, update thebudget.	It get create, update	High	Sprint-4
		USN-2	As a user, I can enter my expense intocategory	It get update to budget	High	Sprint-4
Customer	Report	USN-1	As a user, I get a expense report anytimeI need	It show the report	High	Sprint-5
Customer	Logout	USN-1	As a user, I click on the logout button	It logout the user account	Low	Sprint-5
Customer care	Chat bot	ADMIN-1	As a admin, chat bot help to get familiar with application	It teach the user for the first time	High	Sprint-5

6. PROJECT PLANNING &SCHEDULING:

6.1 Sprint planning and Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Kiridharan
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Kurinjilan
Sprint-1	Login	USN-3	As a user, I used my Mail id and password for login	2	High	Kiran babu
Sprint-2		USN-4	As a user, I forget my password. Used forget password ,	2	High	Edwin
Sprint-2	Dashboard	USN-5	As a user, there is profile tab	1	High	Kiridharan
Sprint-2		USN-6	As a user, there is budget tab	2	High	Kurinjilan
Sprint-3	Budget	USN-7	As a user, I create a budget, update the budget	1	Low	Kiran babu
Sprint-3		USN-9	As a user, I can enter my expense into category	2	High	Edwin
Sprint-4	Report	USN-10	As a user, I get a expense report anytime I need	2	High	Kiran babu, Edwin
Sprint-4	Chat bot	USN-12	As a admin, chat bot help to get familiar with application	2	High	Kiridharan, Kurinjilan

6.2 Sprint Delivery schedule:

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

Velocity:
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

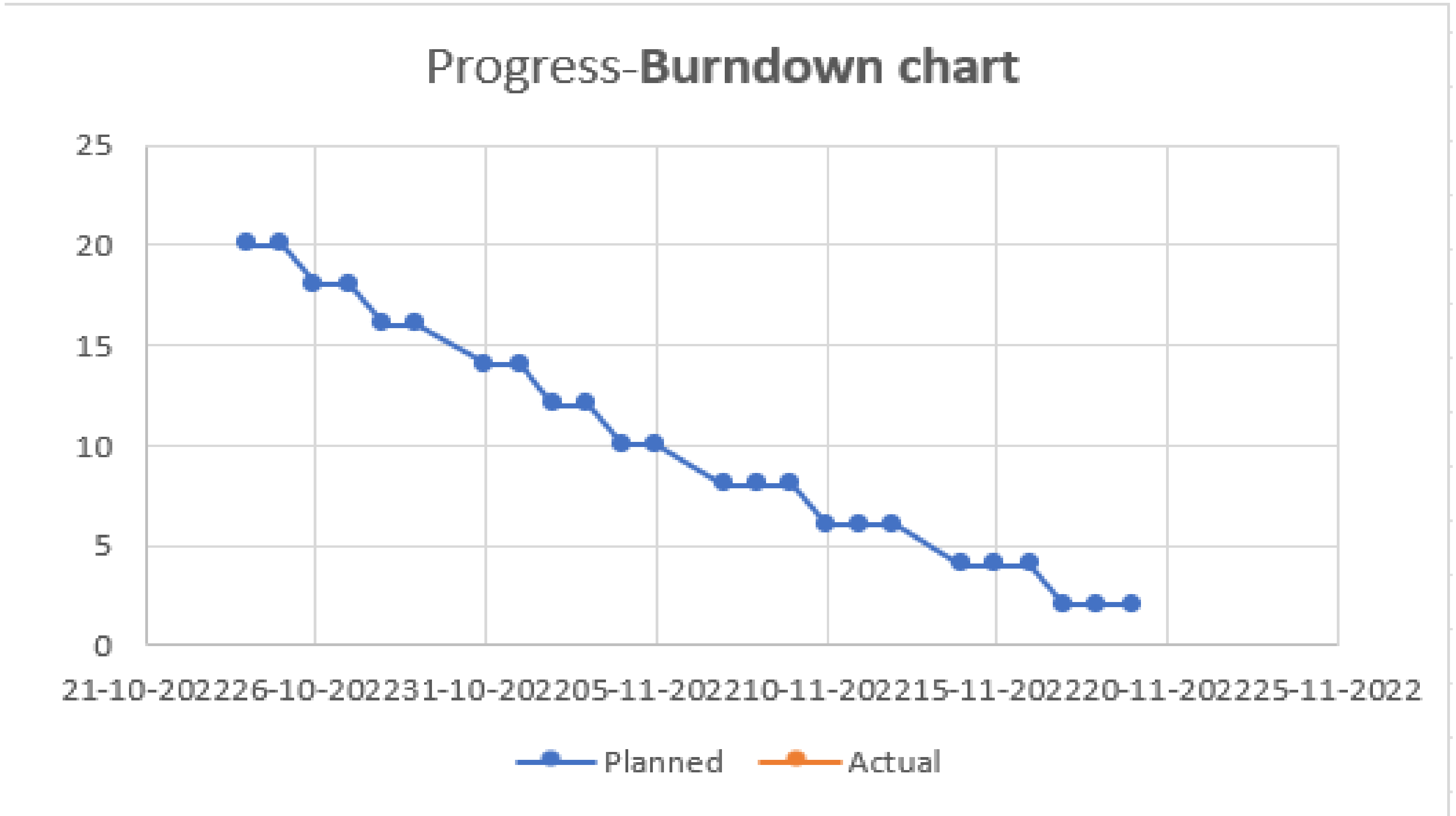
$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

AV=20/6

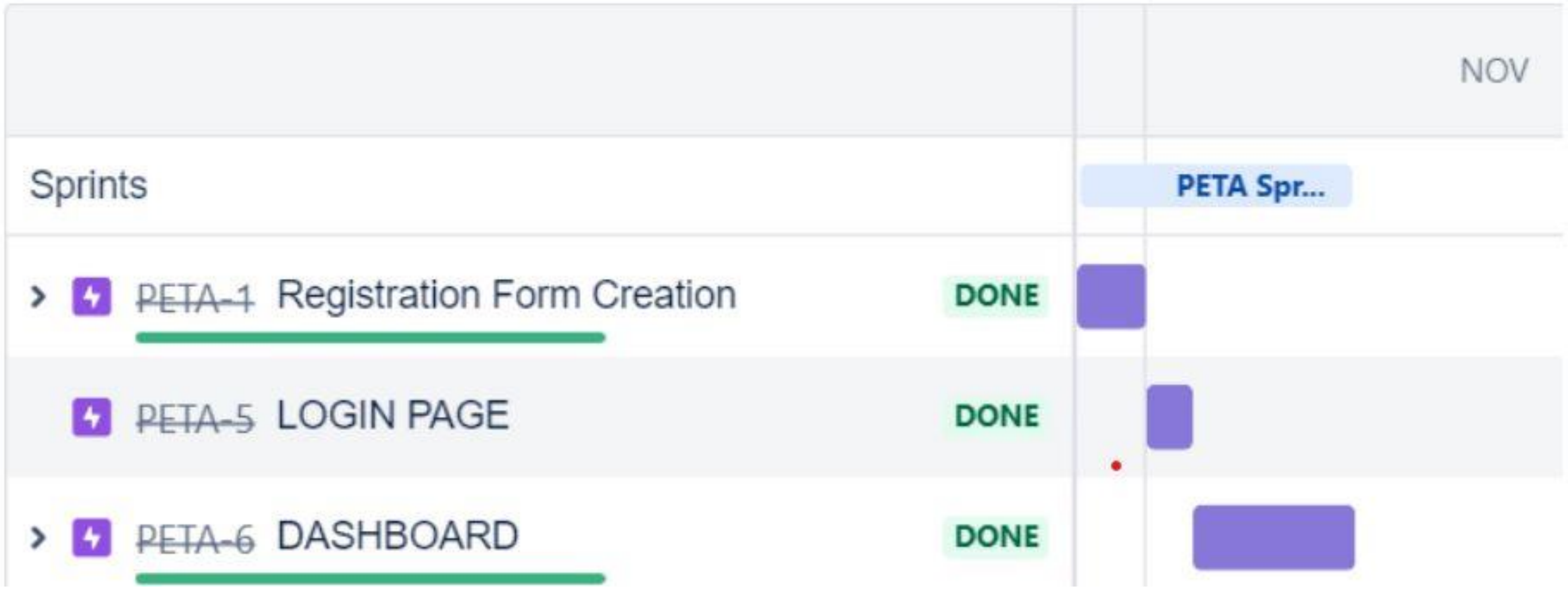
AV=3.33

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies suchas Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



6.3. Report from JIRA:



NOV

Sprints

PETA Spr...

~~PETA-1~~ Registration Form Creation DONE

~~PETA-5~~ LOGIN PAGE DONE

~~PETA-6~~ DASHBOARD DONE

7. CODING &SOLUTIONING:

7.1 Weekly Report:

```
{%  load static %}
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
        <meta name="description" content="" />
        <meta name="author" content="" />
        <title>Dashboard - SB Admin</title>
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bulma@0.9.4/css/bulma.min.css">

        <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.min.js" integrity="sha512-
d9xgZrVZpmmQlfonhQUvTR7lMPtO7NkZMkA0ABN3PHCbKA5nqylQ/yWlFAyY6hYgdF1Qh6nYiuADWwKB4C2WSw=="
crossorigin="anonymous"></script>
        <link href="https://cdn.datatables.net/1.10.20/css/dataTables.bootstrap4.min.css" rel="stylesheet" crossorigin="anonymous"
/>
    </head>
    <body class="sb-nav-fixed">
        <div class="messages" id="alert-message">
       <nav class="navbar" role="navigation" aria-label="main navigation">
                    <div class="navbar-brand">
                      <a class="navbar-item" href="#">
                        <img src="https://is5-ssl.mzstatic.com/image/thumb/Purple123/v4/bc/a7/84/bca78422-ac30-5125-2daa-
2f0be5e5a757/AppIcon-0-1x_U007emarketing-0-0-GLES2_U002c0-512MB-sRGB-0-0-0-85-220-0-0-0-4.png/1200x630wa.png"
width="100%" height="100%">
                      </a>
```

```
           <a role="button" class="navbar-burger" aria-label="menu" aria-expanded="false" data-
target="navbarBasicExample">
            <span aria-hidden="true"></span>
            <span aria-hidden="true"></span>
            <span aria-hidden="true"></span>
           </a>
          </div>

          <div id="navbarBasicExample" class="navbar-menu">
           <div class="navbar-start">
            <a class="navbar-item" href="/index">
                <div class="sb-nav-link-icon"><i class=""></i></div>Home</a>

            <a class="navbar-item" href="/profile">
              <i class="glyphicon glyphicon-user"></i>
                 PROFILE</a>


            <div class="navbar-item has-dropdown is-hoverable">
             <a class="navbar-link">
              More
             </a>

             <div class="navbar-dropdown">
              <a class="navbar-item">
               WEEKLY RECORD(need to wrok)
              </a>
             <a class="navbar-item" >
               MONTHLY RECORD(need to wrok)
             </a>
              <a class="navbar-item" href="/tables">
                HISTORY
```

```
        </a>
        <a class="navbar-item" href="/info">
          Yearly Record
        </a>
        <hr class="navbar-divider">
        <a class="navbar-item">
         Report an issue
        </a>
       </div>
      </div>
     </div>

     <div class="navbar-end">
      <div class="navbar-item">
       <div class="buttons">
          <a class="button is-primary" href="/handleLogout">Logout</a>
          <div class="button is-secondary">
            <div class="small">Logged as:</div>
            {{request.user.username}}
          </div>
       </div>
      </div>
     </div>
    </div>
   </nav>
</div>
<div id="layoutSidenav_content">
  <main>
    <div class="container-fluid">
      </ol>
            </div>
        <div class="row">
```

```
              <div class="col-lg-6">
                <div class="card mb-4">
                  <div class="card-header">
                    <i class="fas fa-chart-pie mr-1"></i>
                    Weekly Expense
                  </div>
                  <div class="container"style="width:100%;">
                  <div class="card-body"><canvas id="myChart" width="400" height="400"></canvas></div>
                </div>
                </div>
                </div>
                <div class="col-lg-6">
                <div class="card mb-4">
                  <div class="card-header">
                    <i class="fas fa-chart-pie mr-1"></i>
                    Weekly Expense
                  </div>
                  <div class="card-body">
                 <p> Amount spent this week : {{addmoney_info.sum}}</p>
                 <p> Amount saved this week : {{addmoney_info.x}}</p>
              </div>
            </div>
          </div>
          </div>
        </main>
{% comment %} <div class="container"style="width:30%;">
<canvas id="myChart" width="400" height="400" ></canvas>
</div> {% endcomment %}
<script src="{% static 'javascript/weekly.js'%}"></script>
<script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>
   <script type="text/javascript">
   </script>
```

```
        <script src="https://code.jquery.com/jquery-3.5.1.min.js" crossorigin="anonymous"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.bundle.min.js"
crossorigin="anonymous"></script>
        <script src="js/scripts.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.8.0/Chart.min.js" crossorigin="anonymous"></script>
        <script src="assets/demo/chart-area-demo.js"></script>
        <script src="assets/demo/chart-bar-demo.js"></script>
        <script src="https://cdn.datatables.net/1.10.20/js/jquery.dataTables.min.js" crossorigin="anonymous"></script>
        <script src="https://cdn.datatables.net/1.10.20/js/dataTables.bootstrap4.min.js" crossorigin="anonymous"></script>
        <script src="assets/demo/datatables-demo.js"></script>
    </body>
</html>
</body>
</html>
```

7.2 Expense Edit:

```
{% load static%}
<!DOCTYPE html>
<html lang="en">
  <head>

    <link
      rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/bulma@0.9.4/css/bulma.min.css"
    />
    <link
      href="~bulma-calendar/dist/css/bulma-calendar.min.css"
      rel="stylesheet"
    />
```

```html
  <title>My Wallet</title>
</head>

<body>

  <main
    class="container mt-6 is-centered is-max-widescreen has-text-centered is-half"
  >

    <div class="row card is-half">
      <div class="">
        <div class="box is-centered has-text-centered">

          <h1 class="has-text-centered is-centered">My Wallet</h1>
        </div>
        <form
          method="post"
          action="/addmoney_submission/"
          class="card-content"
        >
          {% csrf_token %}

          <div class="checkbox">
            What you want to add?<br />
            <label class="radio" for="add_money">
              <input
                type="radio"
                name="add_money"
                id="add_money"
                value="Expense"
                checked
          required
        />
        Expense
      </label>

      <label class="radio" for="add_money">
        <input
          class="radio"
          type="radio"
          name="add_money"
          id="add_money"
          value="Income"
          checked
          required
        />
        Income</label
      >
    </div>

    <div>
      <label class="label" for="quantity">Amount:</label>
      <input
        class="input is-link"
        type="number"
        name="quantity"
        value="{{addmoney_info.quantity}}"
        required
      /><br />
    </div>
    <br />
    <br />
    <div>
```

```html
        <label class="label">
          Expense Date:
          <input
            type="date"
            name="Date"
            value="{{addmoney_info.Date}}" /></label
        ><br />
      </div>
      <div>
        <div class="f1 dropdown is-active">
          Select category-
          <select
            class="Category"
            name="Category"
            value="{{addmoney_info.Category}}"
            required
          >
            <br />
            <br />
            <option value="Food">Food</option>
            <option value="Entertainment">Entertainment</option>
            <option value="Travel">Travel</option>
            <option value="Shopping">shopping</option>
            <option value="Necessities">Necessities</option>
            <option value="Others">Others</option></select
          ><br />
        </div>
      </div>
      <br />
      <br />
      <div class="btn">
        <button type="submit" class="button is-link" href="/addmoney">
```

```
                Submit
              </button>
            </div>
          </form>
        </div>
      </div>
    </main>
  </body>
</html>
```

7.3. Database schema:

```
from django.db import models
from django.utils.timezone import now
from django.contrib.auth.models import User
from django.conf import settings
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.db.models import Sum
#Create your models here.
SELECT_CATEGORY_CHOICES = [
    ("Food","Food"),
    ("Travel","Travel"),
    ("Shopping","Shopping"),
    ("Necessities","Necessities"),
    ("Entertainment","Entertainment"),
    ("Other","Other")
]
ADD_EXPENSE_CHOICES = [
     ("Expense","Expense"),
     ("Income","Income")
]
```

```
PROFESSION_CHOICES =[
    ("Employee","Employee"),
    ("Business","Business"),
    ("Student","Student"),
    ("Other","Other")
]
class Addmoney_info(models.Model):
    user = models.ForeignKey(User,default = 1, on_delete=models.CASCADE)
    add_money = models.CharField(max_length = 10 , choices = ADD_EXPENSE_CHOICES )
    quantity = models.BigIntegerField()
    Date = models.DateField(default = now)
    Category = models.CharField( max_length = 20, choices = SELECT_CATEGORY_CHOICES , default ='Food')
    class Meta:
        db_table:'addmoney'

class UserProfile(models.Model):
    user = models.OneToOneField(User,on_delete=models.CASCADE)
    profession = models.CharField(max_length = 10, choices=PROFESSION_CHOICES)
    Savings = models.IntegerField( null=True, blank=True)
    income = models.BigIntegerField(null=True, blank=True)
    # image = models.ImageField(upload_to='profile_image',blank=True)
    def __str__(self):
        return self.user.username
```

8.Testing:

8.1 Test Case:

Test case ID	Feature Type	Component	Test Scenario	Prerequisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO 1	Functional	Home Page	Verify user is able to see the Landing Page when user can type the URL in the box		1.Enter URL and click go; 2.Type the URL; 3.Verify whether it is processing or not.	https://pipeline-silly-tiger.mybluemix.net/	Should Display the Webpage	Working as expected	Pass		N		Kiran babu
LoginPage_TC_OO 2	UI	Home Page	Verify the UI elements is Responsive		1.Enter URL and click go 2. Type or copy paste the URL 3. Check whether the button is responsive or not 4. Reload and Test Simultaneously	https://pipeline-silly-tiger.mybluemix.net/	Should Wait for Response and then gets Acknowledge	Working as expected	Pass		N		kurinjilan
LoginPage_TC_OO 3	Functional	Home page	Verify whether the link is legitimate or not		1.Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimate or not 4. Observe the results	https://pipeline-silly-tiger.mybluemix.net/	User should observe whether the website is legitimate or not.	Working as expected	Pass		N		Kiridharan
LoginPage_TC_OO 4	Functional	Home Page	Verify user is able to access the legitimate website or not		1.Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimate or not 4. Continue if the website is legitimate or be cautious if it is not legitimate.	https://pipeline-silly-tiger.mybluemix.net/	Application should show that Safe Webpage or Unsafe.	Working as expected	Pass		N		Edwin dinu
LoginPage_TC_OO 5	Functional	Home Page	Testing the website with multiple URLs		1.Enter URL (https://phishing-shield.herokuapp.com/) and click go 2. Type or copy paste the URL to test 3. Check the website is legitimate or not 4. Continue if the website is secure or be cautious if it is not secure	https://pipeline-silly-tiger.mybluemix.net/	User can able to identify the websites whether it is secure or not	Working as expected	Pass		N		Aakash b

8.2User Acceptance Testing:

1. *Defect Analysis:*

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

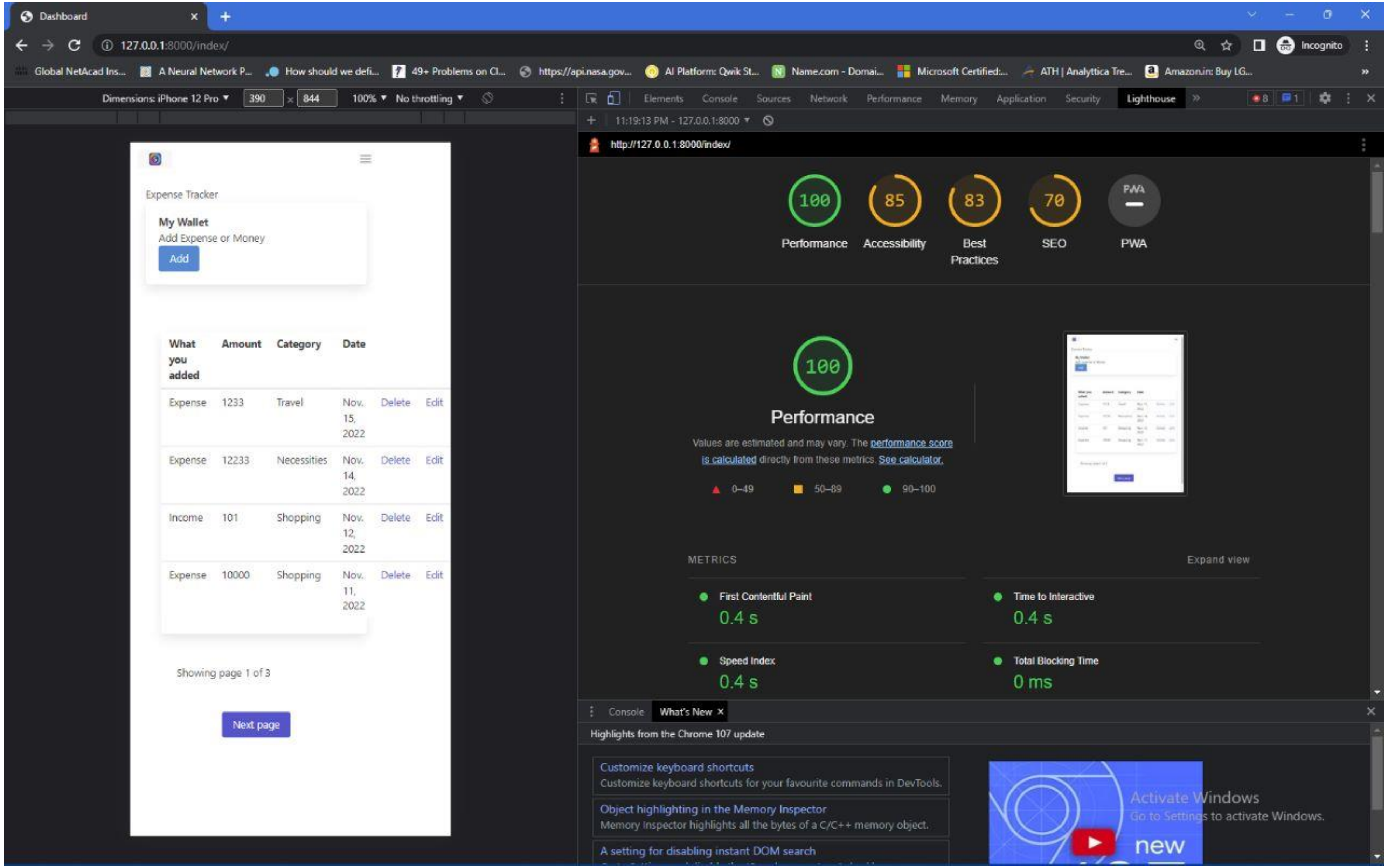
2. *Test Case Analysis:*

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3

Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9. **Result:**

9.1 Performance Metrics:


Dashboard
127.0.0.1:8000/index/
Dimensions: iPhone 12 Pro 390 × 844 100% No throttling
Elements Console Sources Network Performance Memory Application Security Lighthouse
11:19:13 PM - 127.0.0.1:8000
http://127.0.0.1:8000/index/
Expense Tracker
My Wallet
Add Expense or Money
Add
Showing page 1 of 3
Next page
Console What's New

What you added	Amount	Category	Date		
Expense	1233	Travel	Nov. 15, 2022	Delete	Edit
Expense	12233	Necessities	Nov. 14, 2022	Delete	Edit
Income	101	Shopping	Nov. 12, 2022	Delete	Edit
Expense	10000	Shopping	Nov. 11, 2022	Delete	Edit

Performance	Accessibility	Best Practices	SEO	PWA
100	85	83	70	—

100
Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

METRICS Expand view

Metric	Value
First Contentful Paint	0.4 s
Time to Interactive	0.4 s
Speed Index	0.4 s
Total Blocking Time	0 ms

Highlights from the Chrome 107 update

Customize keyboard shortcuts
Customize keyboard shortcuts for your favourite commands in DevTools.

Object highlighting in the Memory Inspector
Memory Inspector highlights all the bytes of a C/C++ memory object.

A setting for disabling instant DOM search

Activate Windows
Go to Settings to activate Windows.

10. ADVANTTAGES & DISADVANTAGES:

Advantage:

- Easy to add the daily Expense and user friendly
- Easy for the user modify expense
- Easy to view the pervious history

Disadvantage:

- Notification can be send only mail
- User have to enter the data manually

11.CONCLUSION:

This personal expense tracker can be used for reviewing your monthly budget, keep tracker of your saving and to limite lavish spending

12. FUTURE SCOPE:

- Notification can even be send through the sms
- Automatic payment dection from your sms

13.APPENDIX:

Source code:

View.py

```
from django.shortcuts import render,HttpResponse,redirect
from django.contrib import messages
from django.contrib.auth import authenticate ,logout
from django.contrib.auth import login as dj_login
from django.contrib.auth.models import User
from .models import Addmoney_info,UserProfile
from django.contrib.sessions.models import Session
from django.core.paginator import Paginator, EmptyPage , PageNotAnInteger
```

```python
from django.db.models import Sum
from django.http import JsonResponse
import datetime
from django.utils import timezone
from django.core.mail import send_mail
import os
# Create your views here.
def home(request):
    if request.session.has_key('is_logged'):
        return redirect('/index')
    return render(request,'home/login.html')
   # return HttpResponse('This is home')
def index(request):
    if request.session.has_key('is_logged'):
        user_id = request.session["user_id"]
        user = User.objects.get(id=user_id)
        addmoney_info = Addmoney_info.objects.filter(user=user).order_by('-Date')
        paginator = Paginator(addmoney_info , 4)
        page_number = request.GET.get('page')
        page_obj = Paginator.get_page(paginator,page_number)
        context = {
            # 'add_info' : addmoney_info,
           'page_obj' : page_obj
        }
    #if request.session.has_key('is_logged'):
        return render(request,'home/index.html',context)
    return redirect('home')
    #return HttpResponse('This is blog')
def register(request):
    return render(request,'home/register.html')
    #return HttpResponse('This is blog')
def password(request):
    return render(request,'home/password.html')

def charts(request):
    return render(request,'home/charts.html')
def search(request):
    if request.session.has_key('is_logged'):
        user_id = request.session["user_id"]
        user = User.objects.get(id=user_id)
        fromdate = request.GET['fromdate']
        todate = request.GET['todate']
        addmoney = Addmoney_info.objects.filter(user=user, Date__range=[fromdate,todate]).order_by('-Date')
        return render(request,'home/tables.html',{'addmoney':addmoney})
    return redirect('home')
def tables(request):
    if request.session.has_key('is_logged'):
        user_id = request.session["user_id"]
        user = User.objects.get(id=user_id)
        fromdate = request.POST.get('fromdate')
        todate = request.POST.get('todate')
        addmoney = Addmoney_info.objects.filter(user=user).order_by('-Date')
        return render(request,'home/tables.html',{'addmoney':addmoney})
    return redirect('home')
def addmoney(request):
    return render(request,'home/addmoney.html')

def profile(request):
    if request.session.has_key('is_logged'):
        return render(request,'home/profile.html')
    return redirect('/home')

def profile_edit(request,id):
    if request.session.has_key('is_logged'):
        add = User.objects.get(id=id)
        return render(request,'home/profile_edit.html',{'add':add})
    return redirect("/home")

def profile_update(request,id):
    if request.session.has_key('is_logged'):
        if request.method == "POST":
            user = User.objects.get(id=id)
            user.first_name = request.POST["fname"]
            user.last_name = request.POST["lname"]
            user.email = request.POST["email"]
            Savings = request.POST["Savings"]
            income = request.POST["income"]
            profession = request.POST["profession"]
            UserProfile.objects.filter(user=user).update(Savings=Savings, income=income, profession=profession)

            # user.userprofile.save()
            user.save()
            return redirect("/profile")
    return redirect("/home")

def handleSignup(request):
    if request.method =='POST':
            # get the post parameters
            uname = request.POST["uname"]
            fname=request.POST["fname"]
            lname=request.POST["lname"]
            email = request.POST["email"]
            profession = request.POST['profession']
            Savings = request.POST['Savings']
            income = request.POST['income']
            pass1 = request.POST["pass1"]
    pass2 = request.POST["pass2"]
    profile = UserProfile(Savings = Savings,profession=profession,income=income)
    # check for errors in input
    if request.method == 'POST':
        try:
            user_exists = User.objects.get(username=request.POST['uname'])
            messages.error(request," Username already taken, Try something else!!!")
            return redirect("/register")
        except User.DoesNotExist:
            if len(uname)>15:
                messages.error(request," Username must be max 15 characters, Please try again")
                return redirect("/register")

            if not uname.isalnum():
                messages.error(request," Username should only contain letters and numbers, Please try again")
                return redirect("/register")

            if pass1 != pass2:
                messages.error(request," Password do not match, Please try again")
                return redirect("/register")

    # create the user
    user = User.objects.create_user(uname, email, pass1)
    user.first_name=fname
    user.last_name=lname
    user.email = email
    # profile = UserProfile.objects.all()

    user.save()
    # p1=profile.save(commit=False)
    profile.user = user
    profile.save()
```

```python
            messages.success(request," Your account has been successfully created")
            return redirect("/")
    else:
        return HttpResponse('404 - NOT FOUND ')
    return redirect('/login')

def handlelogin(request):
    if request.method =='POST':
        # get the post parameters
        loginuname = request.POST["loginuname"]
        loginpassword1=request.POST["loginpassword1"]
        user = authenticate(username=loginuname, password=loginpassword1)
        if user is not None:
            dj_login(request, user)
            request.session['is_logged'] = True
            user = request.user.id
            request.session["user_id"] = user
            messages.success(request, " Successfully logged in")
            return redirect('/index')
        else:
            messages.error(request," Invalid Credentials, Please try again")
            return redirect("/")
    return HttpResponse('404-not found')
def handleLogout(request):
        del request.session['is_logged']
        del request.session["user_id"]
        logout(request)
        messages.success(request, " Successfully logged out")
        return redirect('home')

#add money form
def addmoney_submission(request):
    if request.session.has_key('is_logged'):
        if request.method == "POST":
            user_id = request.session["user_id"]
            user1 = User.objects.get(id=user_id)
            addmoney_info1 = Addmoney_info.objects.filter(user=user1).order_by('-Date')
            add_money = request.POST["add_money"]
            quantity = request.POST["quantity"]
            Date = request.POST["Date"]
            Category = request.POST["Category"]
            add = Addmoney_info(user = user1,add_money=add_money,quantity=quantity,Date = Date,Category= Category)
            add.save()
            paginator = Paginator(addmoney_info1, 4)
            page_number = request.GET.get('page')
            page_obj = Paginator.get_page(paginator,page_number)
            context = {
                'page_obj' : page_obj
                }
            return render(request,'home/index.html',context)
    return redirect('/index')
def addmoney_update(request,id):
    if request.session.has_key('is_logged'):
        if request.method == "POST":
            add  = Addmoney_info.objects.get(id=id)
            add .add_money = request.POST["add_money"]
            add.quantity = request.POST["quantity"]
            add.Date = request.POST["Date"]
            add.Category = request.POST["Category"]
            add .save()
            return redirect("/index")
    return redirect("/home")

def expense_edit(request,id):
    if request.session.has_key('is_logged'):
        addmoney_info = Addmoney_info.objects.get(id=id)
        user_id = request.session["user_id"]
        user1 = User.objects.get(id=user_id)
        return render(request,'home/expense_edit.html',{'addmoney_info':addmoney_info})
    return redirect("/home")

def expense_delete(request,id):
    if request.session.has_key('is_logged'):
        addmoney_info = Addmoney_info.objects.get(id=id)
        addmoney_info.delete()
        return redirect("/index")
    return redirect("/home")

def expense_month(request):
    todays_date = datetime.date.today()
    one_month_ago = todays_date-datetime.timedelta(days=30)
    user_id = request.session["user_id"]
    user1 = User.objects.get(id=user_id)
    addmoney = Addmoney_info.objects.filter(user = user1,Date__gte=one_month_ago,Date__lte=todays_date)
    finalrep ={}

    def get_Category(addmoney_info):
        return addmoney_info.Category
    Category_list = list(set(map(get_Category,addmoney)))

    def get_expense_category_amount(Category,add_money):
        quantity = 0
        filtered_by_category = addmoney.filter(Category = Category,add_money="Expense")
        for item in filtered_by_category:
            quantity+=item.quantity
        return quantity
    for x in addmoney:
        for y in Category_list:
            finalrep[y]= get_expense_category_amount(y,"Expense")

    return JsonResponse({'expense_category_data': finalrep}, safe=False)


def stats(request):
    if request.session.has_key('is_logged') :
        todays_date = datetime.date.today()
        one_month_ago = todays_date-datetime.timedelta(days=30)
        user_id = request.session["user_id"]
        user1 = User.objects.get(id=user_id)
        addmoney_info = Addmoney_info.objects.filter(user = user1,Date__gte=one_month_ago,Date__lte=todays_date)
        sum = 0
        for i in addmoney_info:
            if i.add_money == 'Expense':
                sum=sum+i.quantity
        addmoney_info.sum = sum
        sum1 = 0
        for i in addmoney_info:
            if i.add_money == 'Income':
                sum1 =sum1+i.quantity
        addmoney_info.sum1 = sum1
        x= user1.userprofile.Savings+addmoney_info.sum1 - addmoney_info.sum
        y= user1.userprofile.Savings+addmoney_info.sum1 - addmoney_info.sum
        if x<0:
            # added Logs here
            logger.error("You are in debt")

            messages.warning(request,'Your expenses exceeded your savings')
```

```python
            x = 0
        if x>0:
            y = 0
        addmoney_info.x = abs(x)
        addmoney_info.y = abs(y)
        return render(request,'home/stats.html',{'addmoney':addmoney_info})

def expense_week(request):
    todays_date = datetime.date.today()
    one_week_ago = todays_date-datetime.timedelta(days=7)
    user_id = request.session["user_id"]
    user1 = User.objects.get(id=user_id)
    addmoney = Addmoney_info.objects.filter(user = user1,Date__gte=one_week_ago,Date__lte=todays_date)
    finalrep ={}

    def get_Category(addmoney_info):
        return addmoney_info.Category
    Category_list = list(set(map(get_Category,addmoney)))


    def get_expense_category_amount(Category,add_money):
        quantity = 0
        filtered_by_category = addmoney.filter(Category = Category,add_money="Expense")
        for item in filtered_by_category:
            quantity+=item.quantity
        return quantity

    for x in addmoney:
        for y in Category_list:
            finalrep[y]= get_expense_category_amount(y,"Expense")

    return JsonResponse({'expense_category_data': finalrep}, safe=False)
```

```
def weekly(request):
    if request.session.has_key('is_logged') :
        todays_date = datetime.date.today()
        one_week_ago = todays_date-datetime.timedelta(days=7)
        user_id = request.session["user_id"]
        user1 = User.objects.get(id=user_id)
        addmoney_info = Addmoney_info.objects.filter(user = user1,Date__gte=one_week_ago,Date__lte=todays_date)
        sum = 0
        for i in addmoney_info:
            if i.add_money == 'Expense':
                sum=sum+i.quantity
        addmoney_info.sum = sum
        sum1 = 0
        for i in addmoney_info:
            if i.add_money == 'Income':
                sum1 =sum1+i.quantity
        addmoney_info.sum1 = sum1
        x= user1.userprofile.Savings+addmoney_info.sum1 - addmoney_info.sum
        y= user1.userprofile.Savings+addmoney_info.sum1 - addmoney_info.sum
        if x<0:

            messages.warning(request,'Your expenses exceeded your savings')
            x = 0
        if x>0:
            y = 0
        addmoney_info.x = abs(x)
        addmoney_info.y = abs(y)
        return render(request,'home/weekly.html',{'addmoney_info':addmoney_info})

def check(request):
    if request.method == 'POST':
```

```python
        user_exists = User.objects.filter(email=request.POST['email'])
        messages.error(request,"Email not registered, TRY AGAIN!!!")
        return redirect("/reset_password")

def info_year(request):
    todays_date = datetime.date.today()
    one_week_ago = todays_date-datetime.timedelta(days=30*12)
    user_id = request.session["user_id"]
    user1 = User.objects.get(id=user_id)
    addmoney = Addmoney_info.objects.filter(user = user1,Date__gte=one_week_ago,Date__lte=todays_date)
    finalrep ={}

    def get_Category(addmoney_info):
        return addmoney_info.Category
    Category_list = list(set(map(get_Category,addmoney)))


    def get_expense_category_amount(Category,add_money):
        quantity = 0
        filtered_by_category = addmoney.filter(Category = Category,add_money="Expense")
        for item in filtered_by_category:
            quantity+=item.quantity
        return quantity

    for x in addmoney:
        for y in Category_list:
            finalrep[y]= get_expense_category_amount(y,"Expense")

    return JsonResponse({'expense_category_data': finalrep}, safe=False)

def info(request):
    return render(request,'home/info.html')
```

Model.py

```
from django.db import models
from django.utils.timezone import now
from django.contrib.auth.models import User
from django.conf import settings
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.db.models import Sum
#Create your models here.
SELECT_CATEGORY_CHOICES = [
    ("Food","Food"),
    ("Travel","Travel"),
    ("Shopping","Shopping"),
    ("Necessities","Necessities"),
    ("Entertainment","Entertainment"),
    ("Other","Other")
 ]
ADD_EXPENSE_CHOICES = [
     ("Expense","Expense"),
     ("Income","Income")
 ]
PROFESSION_CHOICES =[
    ("Employee","Employee"),
    ("Business","Business"),
    ("Student","Student"),
    ("Other","Other")
]
class Addmoney_info(models.Model):
    user = models.ForeignKey(User,default = 1, on_delete=models.CASCADE)
    add_money = models.CharField(max_length = 10 , choices = ADD_EXPENSE_CHOICES )
    quantity = models.BigIntegerField()
    Date = models.DateField(default = now)
    Category = models.CharField( max_length = 20, choices = SELECT_CATEGORY_CHOICES , default ='Food')
    class Meta:
        db_table:'addmoney'

class UserProfile(models.Model):
    user = models.OneToOneField(User,on_delete=models.CASCADE)
    profession = models.CharField(max_length = 10, choices=PROFESSION_CHOICES)
    Savings = models.IntegerField( null=True, blank=True)
    income = models.BigIntegerField(null=True, blank=True)
    def __str__(self):
       return self.user.username
```

Urls.py

```
from django.contrib import admin
from django.urls import path
from django.urls import include
from . import views
from django.contrib.auth import views as auth_views

urlpatterns = [
     path('', views.home, name='home'),
    path('index/', views.index, name='index'),
    path('register/',views.register,name='register'),
    path('handleSignup/',views.handleSignup,name='handleSignup'),
    path('handlelogin/',views.handlelogin,name='handlelogin'),
    path('handleLogout/',views.handleLogout,name='handleLogout'),
    path('reset_password/',auth_views.PasswordResetView.as_view(template_name =
"home/reset_password.html"),name='reset_password'),
path('reset_password_sent/',auth_views.PasswordResetDoneView.as_view(template_name="home/reset_password_sent.html"),name='password_reset_done'),
    path('reset/<uidb64>/<token>/',auth_views.PasswordResetConfirmView.as_view(template_name="home/password_reset_form.html"),name='password_reset_confirm'),
    path('reset_password_complete/',auth_views.PasswordResetView.as_view(template_name="home/password_reset_done.html"),name='password_reset_complete'),
    path('addmoney/',views.addmoney,name='addmoney'),
    path('addmoney_submission/',views.addmoney_submission,name='addmoney_submission'),
    path('charts/',views.charts,name='charts'),
    path('tables/',views.tables,name='tables'),
    path('expense_edit/<int:id>',views.expense_edit,name='expense_edit'),
    path('<int:id>/addmoney_update/', views.addmoney_update, name="addmoney_update") ,
    path('expense_delete/<int:id>',views.expense_delete,name='expense_delete'),
    path('profile/',views.profile,name = 'profile'),
    path('expense_month/',views.expense_month, name = 'expense_month'),
    path('stats/',views.stats, name = 'stats'),
    path('expense_week/',views.expense_week, name = 'expense_week'),
    path('weekly/',views.weekly, name = 'weekly'),
    path('check/',views.check,name="check"),
    path('search/',views.search,name="search"),
    path('<int:id>/profile_edit/',views.profile_edit,name="profile_edit"),
    path('<int:id>/profile_update/',views.profile_update,name="profile_update"),
    path('info/',views.info,name="info"),
    path('info_year/',views.info_year,name="info_year"),

]
```

GitHub Links: https://github.com/IBM-EPBL/IBM-Project-20186-1659714254.git

Demo Links: https://drive.google.com/file/d/1j2XymZpYW7hsaXJaEovXglOz0gDXE-9t/view?usp=share_link