IBM - Project

Project Documentation Report

**Project Name: PERSONAL EXPENSE**

**TRACKER APPLICATION-IBM**

**Team ID:** PNT2022TMID0319

# Team Members:

* Madhna Raj S [Team Lead]
* Lalithraju A N
* Makeshvar S
* Vethaprasath M

**Github Link :** [**https://github.com/IBM-EPBL/IBM-Project-26203-1660020960**](https://github.com/IBM-EPBL/IBM-Project-26203-1660020960)

# INTRODUCTION:

## 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

## 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

# LITERATURE SURVEY:

## 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.

## 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

## 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.

* + 1. **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.

* + 1. **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.

* + 1. **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.

* + 1. **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.

* + 1. **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.



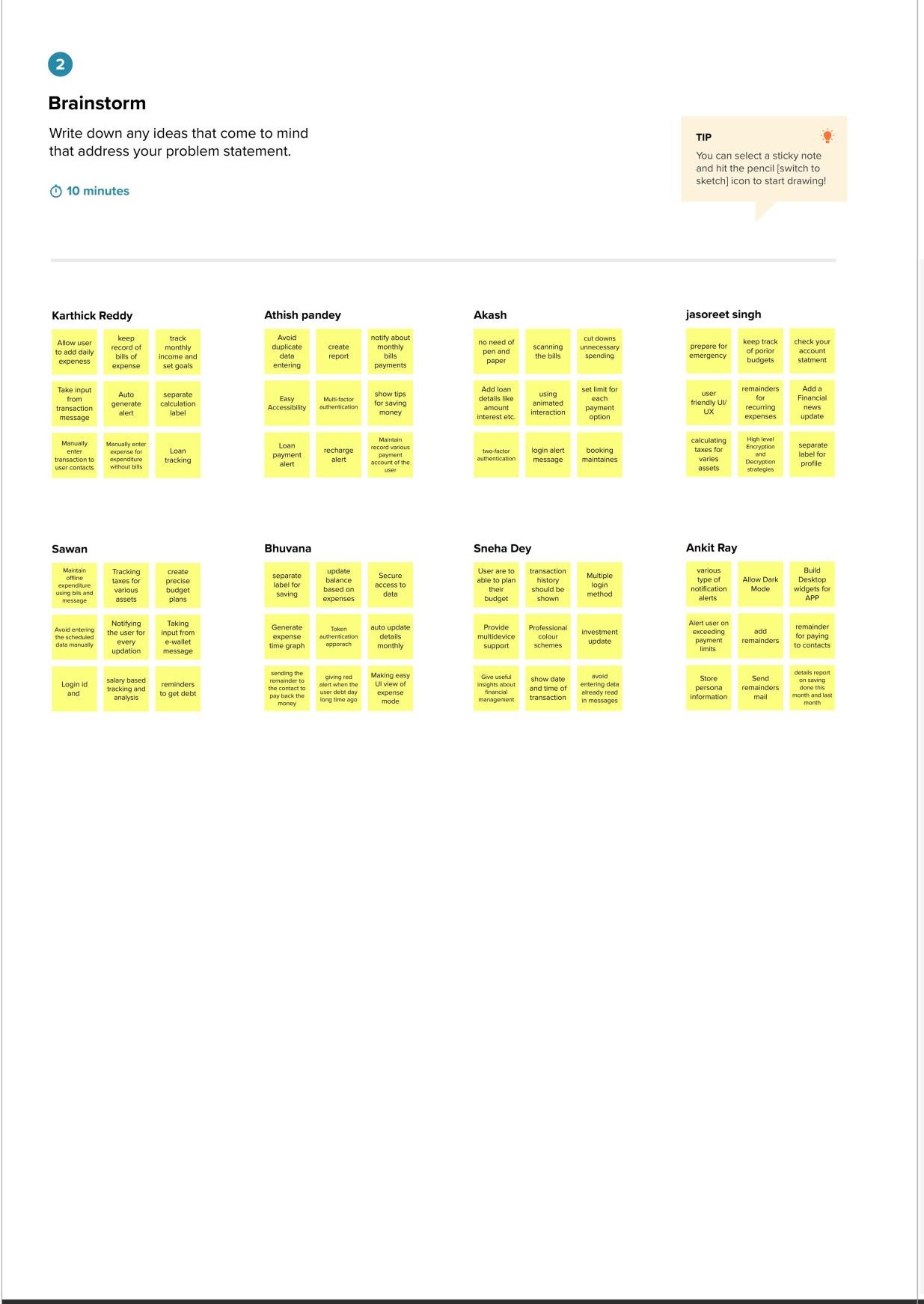
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem**  **Statement (PS)** | **Iam (Customer**  **)** | **I’m trying to** | **But** | **Because** | **Which makes me feel** |
| PS-1 | A  working employe e | Save my money | I cant get myself to limit  my spending | Write down, where you spend the money | Irritated |
| PS-2 | Business | Organizin g the expense | I  spend money lavishl y | Making calculati on | Frustrated |

# IDEATION & PROPOSED SOLUTION:

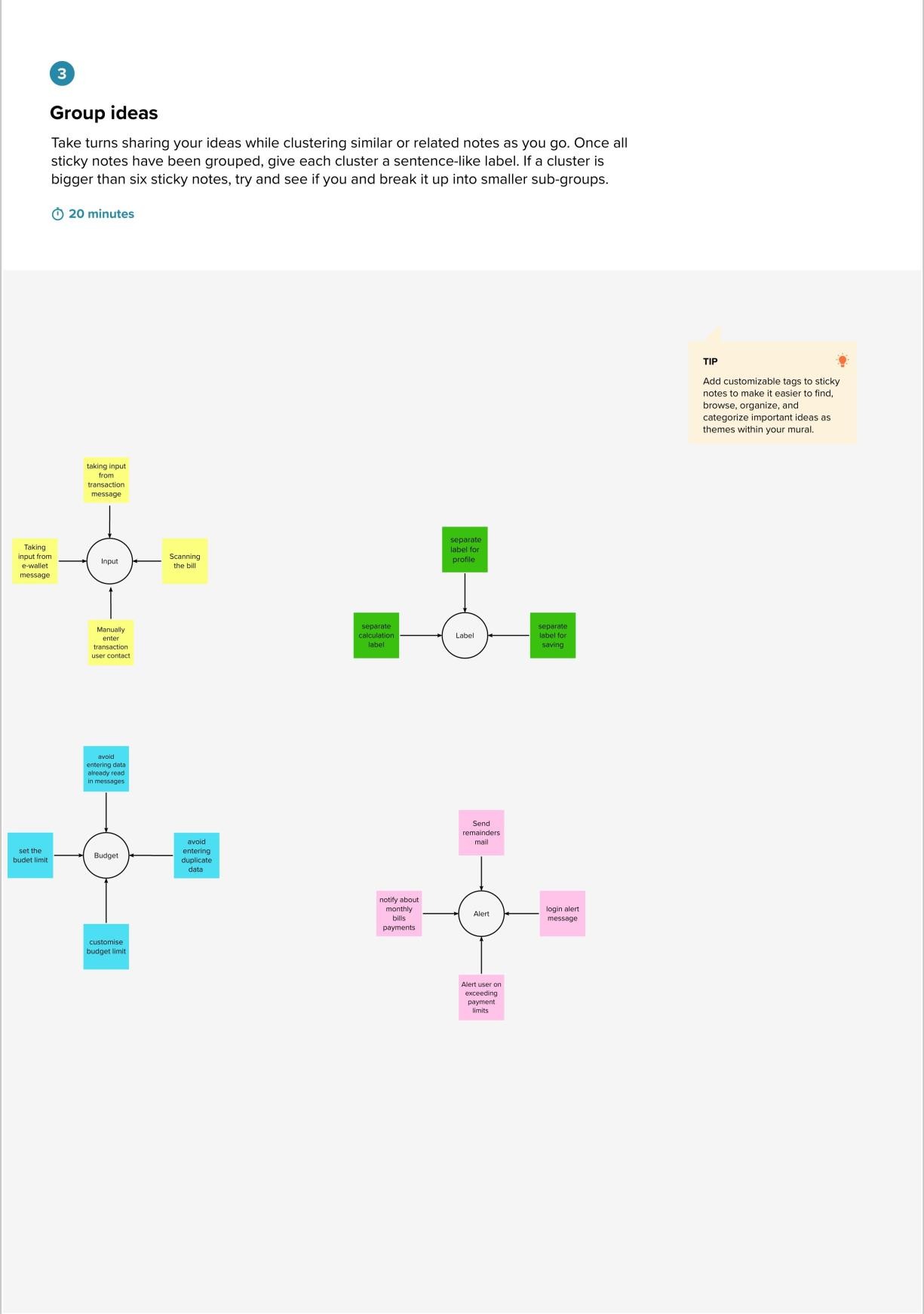
## Empathy Map:



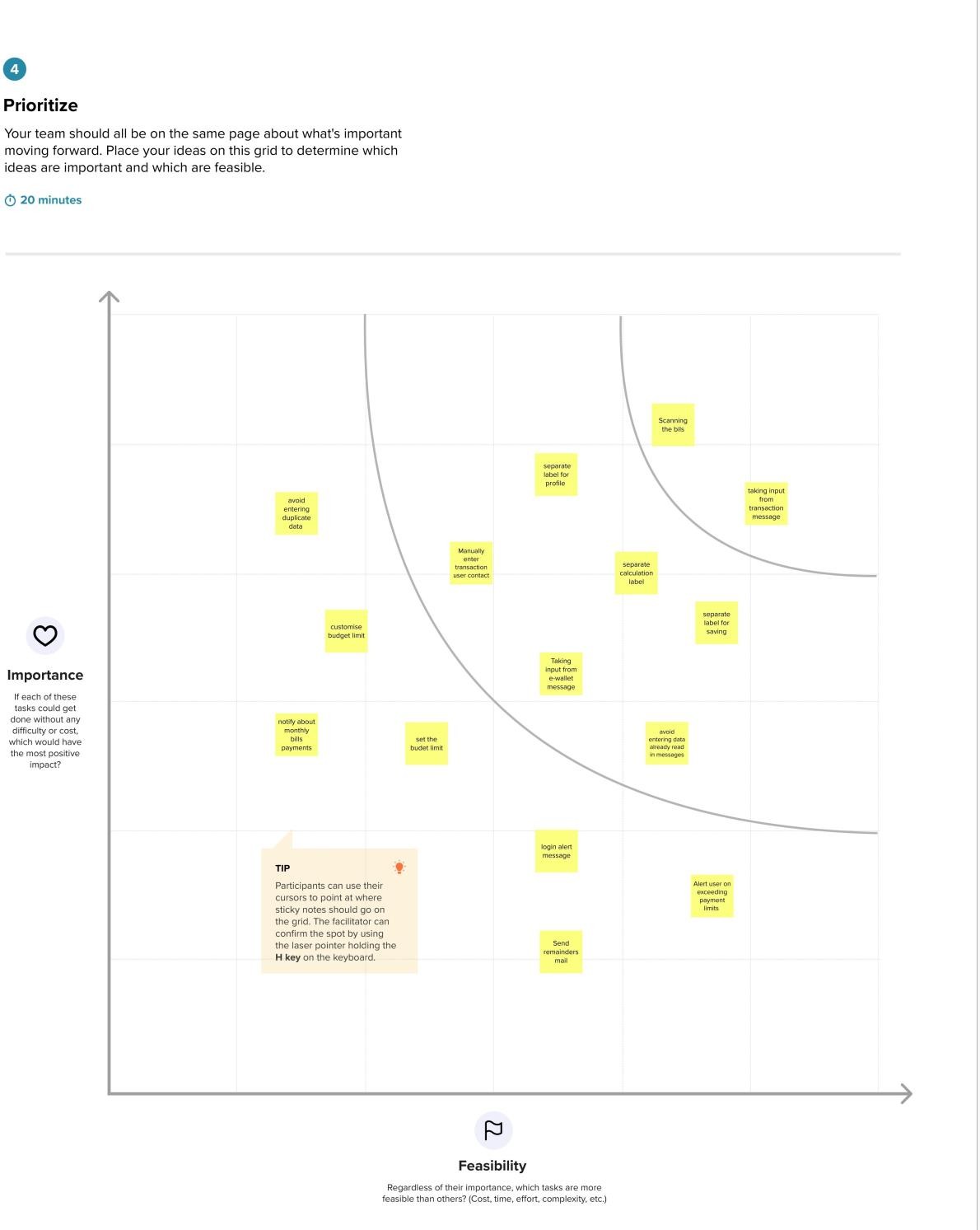
* 1. **Ideation and Brainstorming:**



### Step-2: Brainstorm, Idea Listing and Grouping



**Step-3: Idea Prioritization**



### Proposed Solution

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

|  |  |  |
| --- | --- | --- |
| **S.No**  **.** | **Parameter** | **Description** |
| 1. | Problem Statement (Problem to besolved) | 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. |

## Problem Solution Fit:

### PROBLEMS T

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Deﬁne CS, ﬁt into CC** | **1. CUSTO MER SEGME NT(S)**  Who is your customer?  i.e. working parents of 0-5 y.o. kids | **CS** | **6. CUSTOMER**  What constraints prevent your **CC**  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** 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 | **AS** | **Explore AS, differentiate** |
| **understand RC** | -person who make budget |  |  |  | Budget Bakers: it can be used only in android.QuickBooks: No chat supports  **E**  thihsaptriosbtlheemreeaxlisrtesa?soWnhtahtatis | **RC** |  |
| -person who plan for trip |  |  |  |
| -person who makes weekly or |  |  |  |
| monthly budget |  |  |  |
| -persons who makes more expensive |  |  |  |
| than salary |  |  |  |
|  | cu |  |  |
|  | st |  |  |
|  | o |  |  |
|  | m |  |  |
|  | er | **J&P** |  |
| **2. JOBS-TO-BE-DONE /** | s? |  | **9.** |

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

### PROB LEM ROOT CAUS

**F**

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

### o

**7. BEHAVIOUR**

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

cdoifufeldrebnet msidoeres.than one; explore

-People need to have their recorded on the penand paper

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

**c u**

### s

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 thedata everything they make a payment.

-People are not going open the app everythingthey make a payment While they enter the data, they must rememberall the payment they made for the day

i.e. directly related: ﬁnd the right solar panel installer, calculate usage and beneﬁts; indirectly

**B E**

**o** associated: customers spend free time on volunteering work (i.e. Greenpeace) **n**

-Half of the people make use of pen and paper to **J**

keep track. **&**

-people skip half of the payment they made for

### P

day **,**

### t a

**10. YOUR SOLUTION SL** If you are working on an existing business, write down your current solution ﬁrst,ﬁll in the canvas, and



**3. TRIGGERS**

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

**TR**

-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**

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

**EM**

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 ONLINE

What kind of actions do customers take onli

ne?

**Extract online & ofﬂine CH of BE**

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

# REQUIREMENT ANALYSIS:

## 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 userdaily 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 |

## Non-functional Requirements:

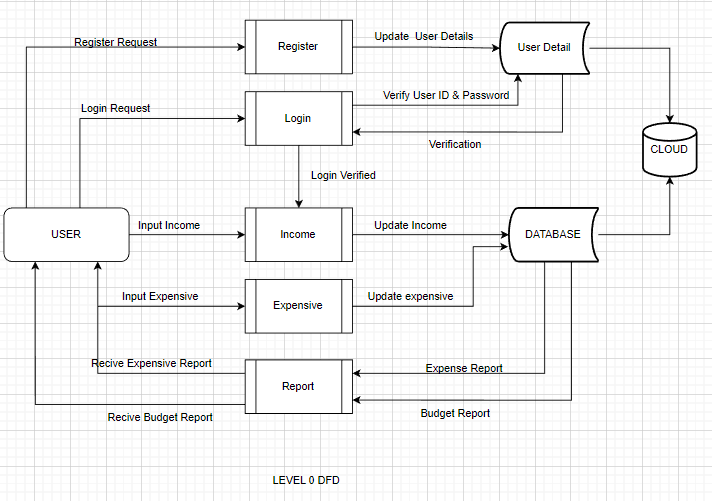
Following are the non-functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **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 |

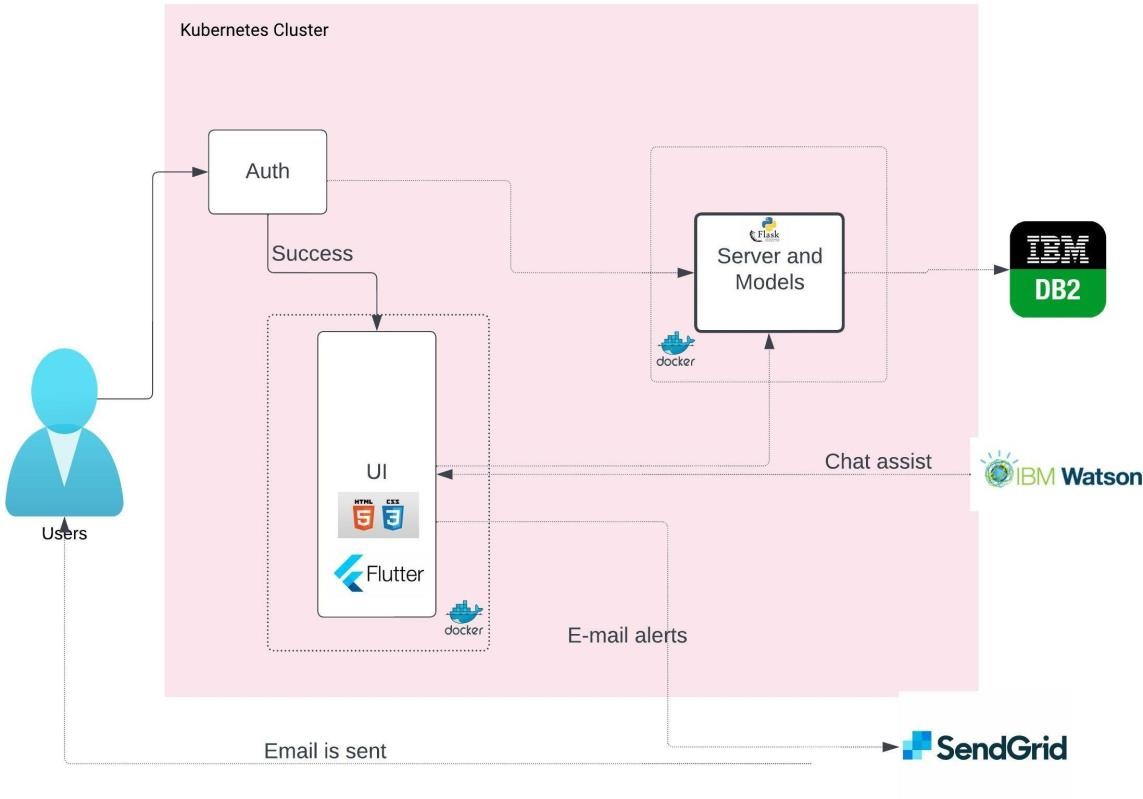
# PROJECT DESIGN:

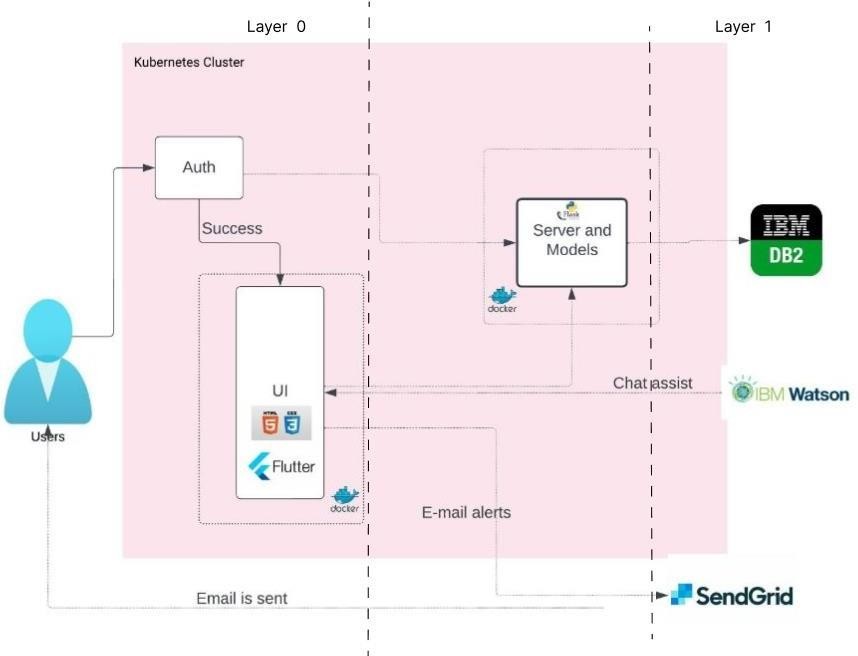
## Data Flow Diagram:

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.



## Solution and Technical Architecture Diagram:



**Technical Architecture:**

## 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 |

# PROJECT PLANNING &SCHEDULING:

## 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 | Madhan Raj S |
| Sprint-1 |  | USN-2 | As a user, I will receive confirmation email onceI have registered for the  application | 1 | High | Vethaprasanth M |
| Sprint-1 | Login | USN-3 | As a user, I used my Mail id and  password forlogin | 2 | High | Makeshvar S |
| Sprint-2 |  | USN-4 | As a user, I forget my password. Used forget  password , | 2 | High | Lalithraju A N |
| Sprint-2 | Dashboard | USN-5 | As a user, there is profile tab | 1 | High | Madhan Raj S |
| Sprint-2 |  | USN-6 | As a user, there is budget tab | 2 | High | Vethaprasath M |
| Sprint-3 | Budget | USN-7 | As a user, I create a budget, update the  budget | 1 | Low | Makeshvar |
| Sprint-3 |  | USN-9 | As a user, I can enter my expense into  category | 2 | High | Lalithraju A N |
| Sprint-4 | Report | USN-10 | As a user, I get a expense report anytime I need | 2 | High | Lalithraju A N,  Makeshvar S |
| Sprint-4 | Chat bot | USN-12 | As a admin, chat bot help to get familiar withapplication | 2 | High | Madhan Raj S,Vethaprasath M |

* 1. **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 |

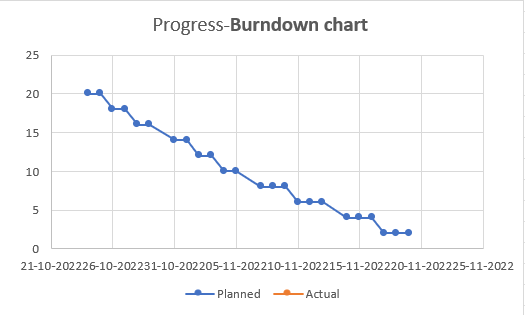
### Velocity:

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=20/6

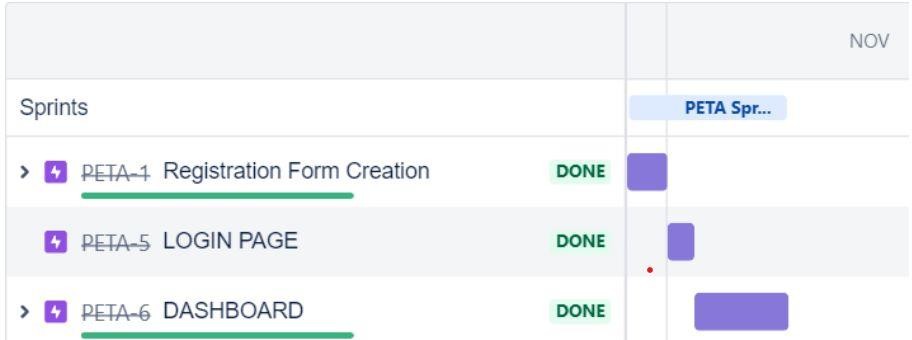
**AV=3.33**



### 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](https://www.visual-paradigm.com/scrum/what-is-agile-software-development/) methodologies suchas [Scrum](https://www.visual-paradigm.com/scrum/scrum-in-3-minutes/). However, burn down charts can be applied to any project containing measurable progress over time.

**6.3. Report from JIRA:**



# CODING &SOLUTIONING:

## 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](https://cdn.jsdelivr.net/npm/bulma%400.9.4/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>

## 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](https://cdn.jsdelivr.net/npm/bulma%400.9.4/css/bulma.min.css)"

/>

<link

href="~bulma-calendar/dist/css/bulma-calendar.min.css" rel="stylesheet"

/>

<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>

<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:

1. from django.db import models
2. from django.utils.timezone import now 10.from django.contrib.auth.models import User 11.from django.conf import settings

12.from django.db.models.signals import post\_save 13.from django.dispatch import receiver

14.from django.db.models import Sum 15.#Create your models here.

16.SELECT\_CATEGORY\_CHOICES = [

1. ("Food","Food"),
2. ("Travel","Travel"),
3. ("Shopping","Shopping"),
4. ("Necessities","Necessities"),
5. ("Entertainment","Entertainment"),
6. ("Other","Other")

23. ] 24.ADD\_EXPENSE\_CHOICES = [

1. ("Expense","Expense"),
2. ("Income","Income")

27. ]

28.PROFESSION\_CHOICES =[

1. ("Employee","Employee"),
2. ("Business","Business"),
3. ("Student","Student"),
4. ("Other","Other") 33.]
5. class Addmoney\_info(models.Model):
6. user = models.ForeignKey(User,default = 1, on\_delete=models.CASCADE)
7. add\_money = models.CharField(max\_length = 10 , choices = ADD\_EXPENSE\_CHOICES )
8. quantity = models.BigIntegerField()
9. Date = models.DateField(default = now)
10. Category = models.CharField( max\_length = 20, choices = SELECT\_CATEGORY\_CHOICES , default ='Food')
11. class Meta:
12. db\_table:'addmoney' 42.
13. class UserProfile(models.Model):
14. user = models.OneToOneField(User,on\_delete=models.CASCADE)
15. profession = models.CharField(max\_length = 10, choices=PROFESSION\_CHOICES)
16. Savings = models.IntegerField( null=True, blank=True)
17. income = models.BigIntegerField(null=True, blank=True)
18. # image = models.ImageField(upload\_to='profile\_image',blank=True)
19. def str (self):
20. return self.user.username

# Testing:

## 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.n et/ | 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   1. Reload and Test Simultaneously | https://pipeline- silly- tiger.mybluemix.n et/ | 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.n et/ | 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.n et/ | 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   1. Type or copy paste the URL to test 2. Check the website is legitimate or not 3. Continue if the website is   secure or be cautious if it is not secure | https://pi peline- silly- tiger.mybl uemix.net  / | User can able to identify the websites whether it is secure or not | Working as expected | Pass |  | N |  | Aakash b |

* 1. **User Acceptance Testing:**

1. ***Defect Analysis:***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Resolution** | **Severity 1** | **Severity 2** | **Severity 3** | **Severity 4** | **Subtota l** |
| 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 Reproduce d | 0 | 0 | 1 | 0 | 1 |
| Skipped | 0 | 0 | 1 | 1 | 2 |
| Won't Fix | 0 | 5 | 2 | 1 | 8 |
| Totals | 24 | 14 | 13 | 26 | 7  7 |

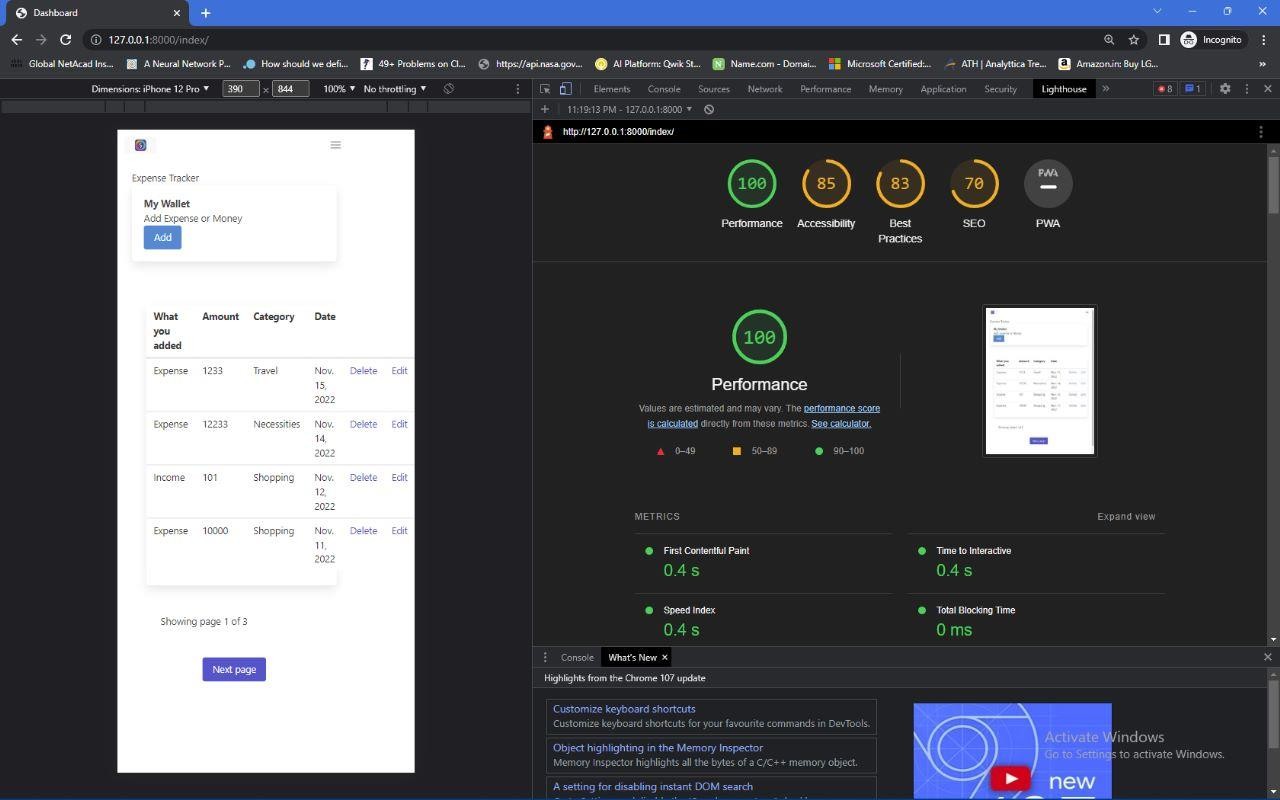
1. ***Test Case Analysis:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Section** | **Total Cases** | **Not Tested** | **F**  **ail** | **P**  **as s** |
| 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 |

# Result:

**9.1 Performance Metrics:**



# 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

# CONCLUSION:

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

# FUTURE SCOPE:

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

# 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

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()

else:

messages.success(request," Your account has been successfully created") return redirect("/")

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')

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':

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"),na me='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-26203-1660020960**](https://github.com/IBM-EPBL/IBM-Project-26203-1660020960)

**Demo Links:** [**https://drive.google.com/file/d/1eVY18Oss3Ja\_EgPsRSCfP8FS3TlfSNE6/view?usp=drivesdk**](https://drive.google.com/file/d/1eVY18Oss3Ja_EgPsRSCfP8FS3TlfSNE6/view?usp=drivesdk)