Personal Expense Tracker Application

Cloud Application Development

Submitted by

Narayan Kumar M

Shai Prashanth K

Sathish Kumar G

Preethi S

Project ID PNT2022MID07222

Department of Information Technology,
Jerusalem College of Engineering, Chennai - 600100.

Institution Mentor: Sathea Sree.S
Industrial Mentor: Kusboo

1.Introduction:

A Personal Expense Tracker Application is a particular form of digital diary that aids in keeping track of all of our cash transitions and moreover offers daily, weekly, monthly, and yearly reports on all financial activities. User receives alerts to keep track of income and expenses that can system for tracking the application. All data is kept in offline mode for easy access at any time and from any location. The Daily Expense Tracker's user interface is incredibly straightforward and appealing, making it simple to grasp and the finest approach to record our financial data.

1.1 Project Overview:

Simply put, personal finance includes all of the financial decisions and actions that a finance software facilitates by assisting you in effectively managing your finances. A personal finance software will not only assist you with accounting and budgeting, but it will also provide you with valuable advice on money management. Users of personal finance applications will be prompted to enter their costs, after which their wallet balance will be updated and displayed to them. Users can also receive a graphical analysis of their expenses. They can choose to establish a cap on how much can be used in that month, and if the cap is surpassed, the user will receive an email alert

1.2 Purpose:

When you keep track of your spending, you can make sure your money is being utilized wisely and you will know where it goes. You can learn why you're in debt and how you got there by keeping track of your spending. You can then use this information to create a debt relief plan that works for you.
You may plan for both short-term and long-term expenses by using a budget to make sure you're not spending more than you're earning. It's a simple, practical solution for folks with all types of income and expenses to maintain order in their finances.

2.Literature Survey

2.1 Existing Problem

The current problem is that users are not motivated to monitor their expenses and not willing to alter their spending trends. Our system introduces a goal and reward system to beat the problem. Secondly, the feature of sending reminders for recurring bills is a key feature in this app. Most often, users do not have the right guidance with spending and it is hard for them to analyze their mistakes. To make this process simpler, our app displays the spending trends as a graph to help users analyse and find the real reason behind his inability to meet their goals

2.2 References

https://www.ijraset.com/fileserve.php?FID=3379

https://www.irjmets.com/uploadedfiles/paper/issue_4_april_2022/21604/fina%20l/fin_irjmets1651132467.pdf

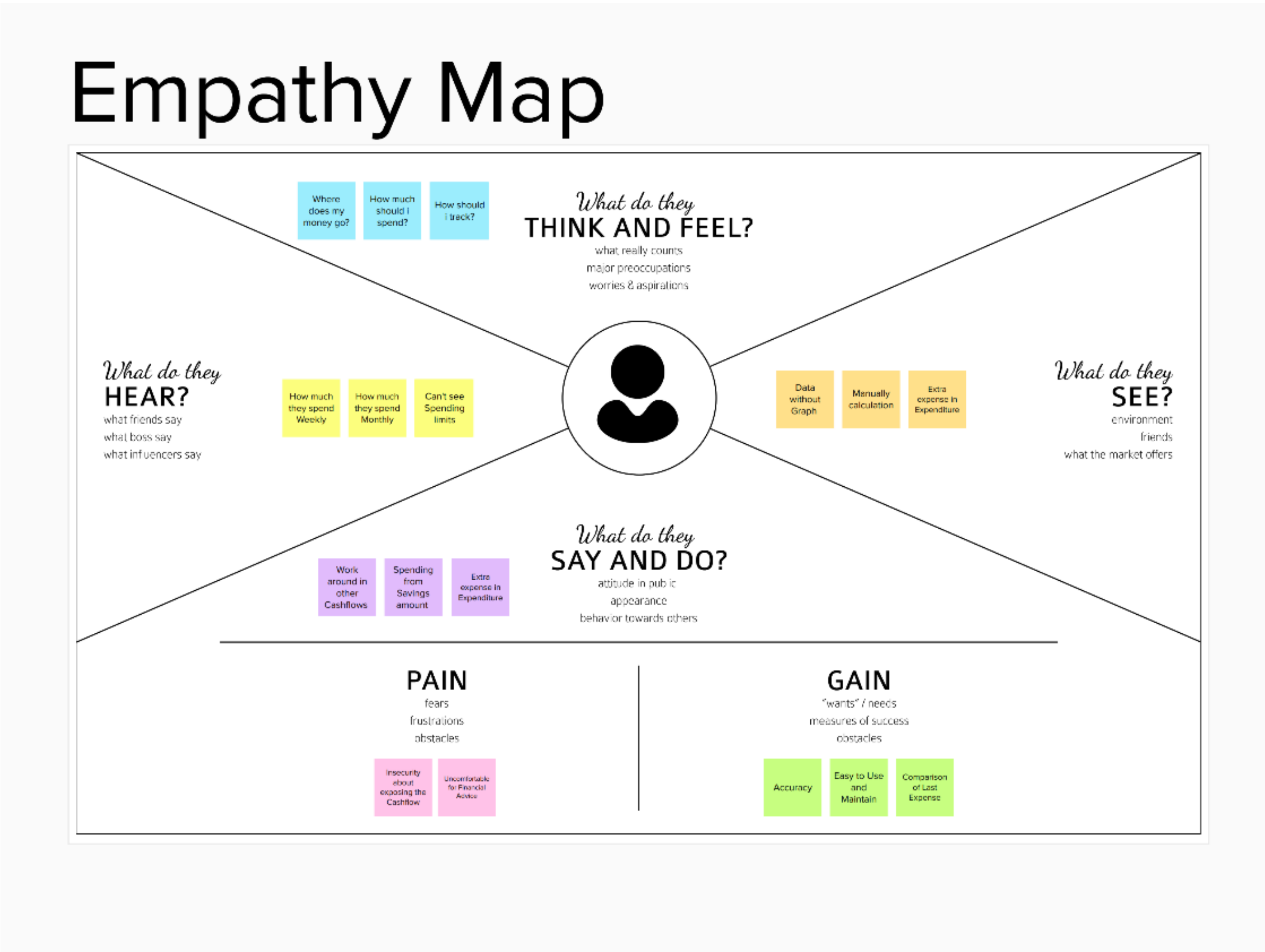
https://www.ijres.org/papers/Volume-9/Issue-12/Ser-4/L09127073.pdf

2.3 Problem Statement Definition

Users spend money on various transactions that may be a part of their daily routine or could be a one-time transaction. Every user has different priorities and thus different expenses. Our team aims to develop a customizable Personal Expense Tracker that allows users to tailor-make the application to suit their needs. We aim to do so through the provision of user-defined expense categories, rewards, goals, and limits to name a few. The application will also provide users with the feature to view a graphical analysis of their expenditure to understand their spending patterns and reach conclusions accordingly.

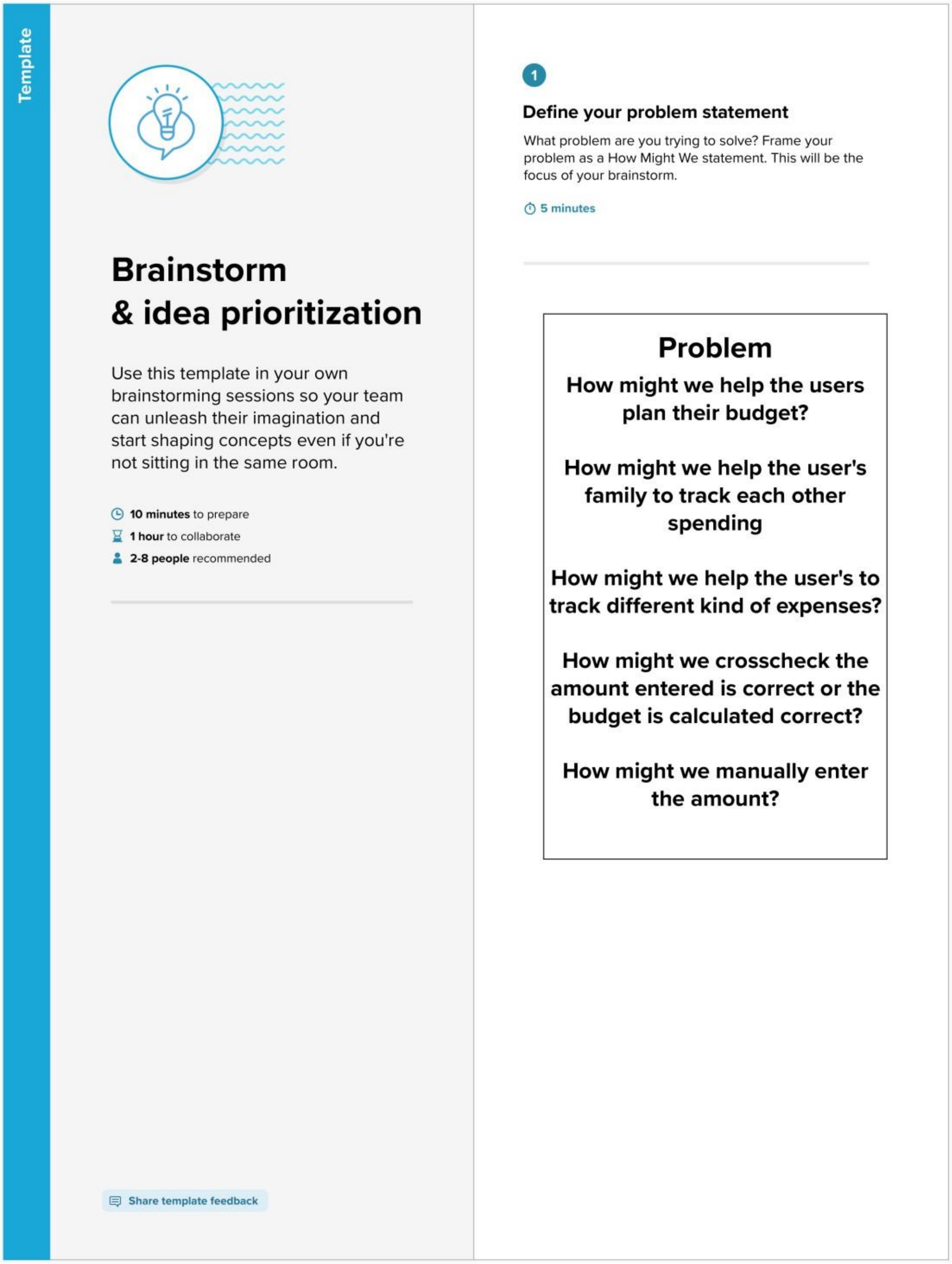
3.Ideation and Proposed Solution:

3.1 Empathy Map Canvas

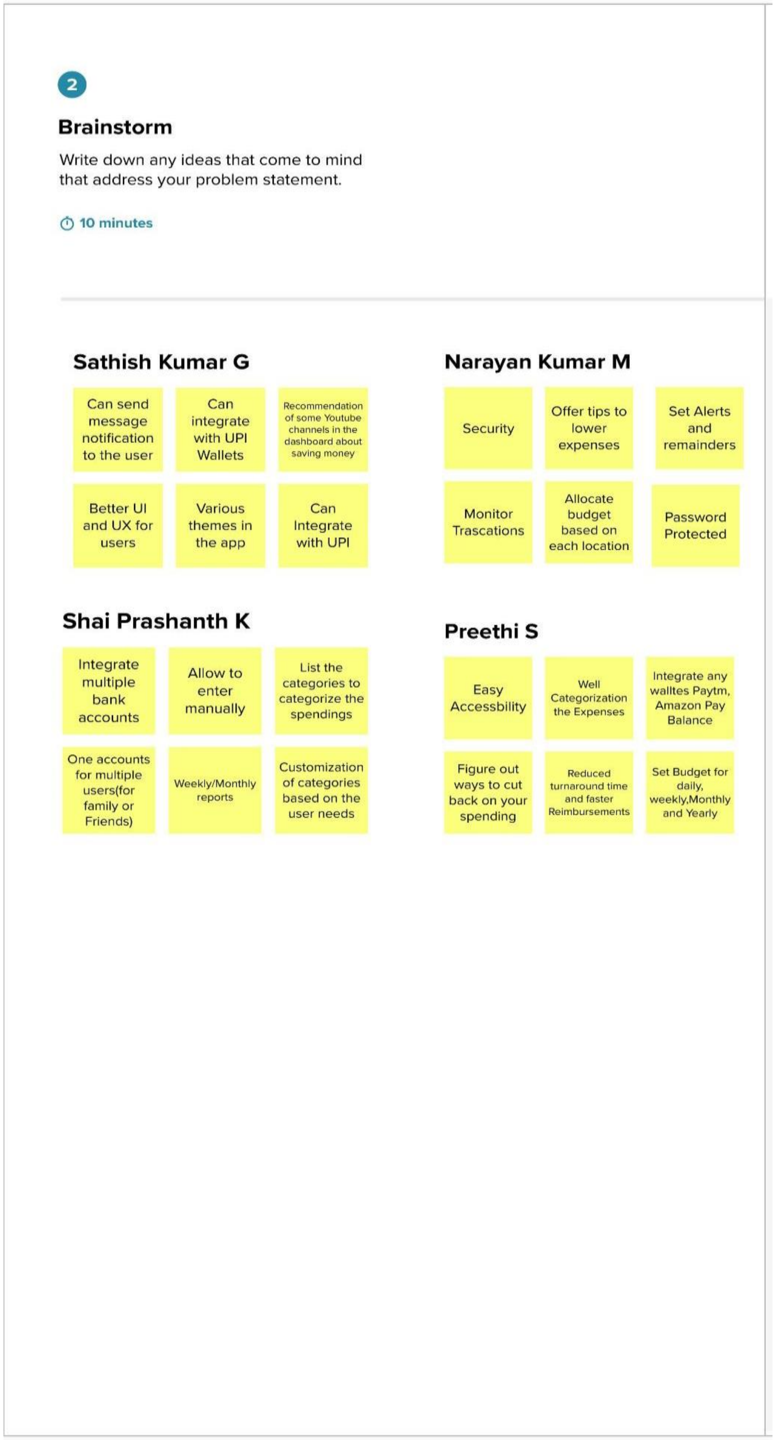


3.2 Ideation and Brainstorming

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping



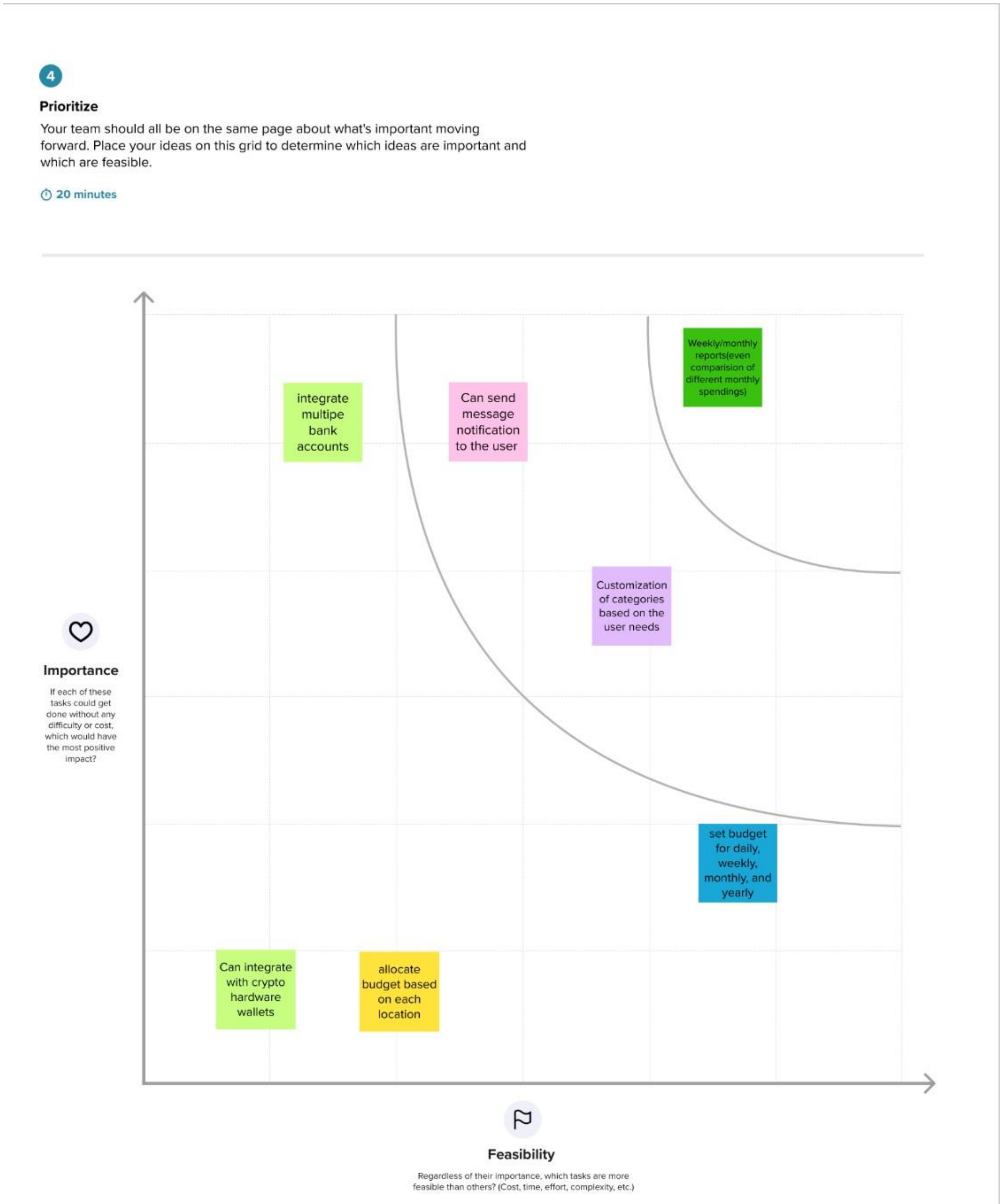
3

Group ideas

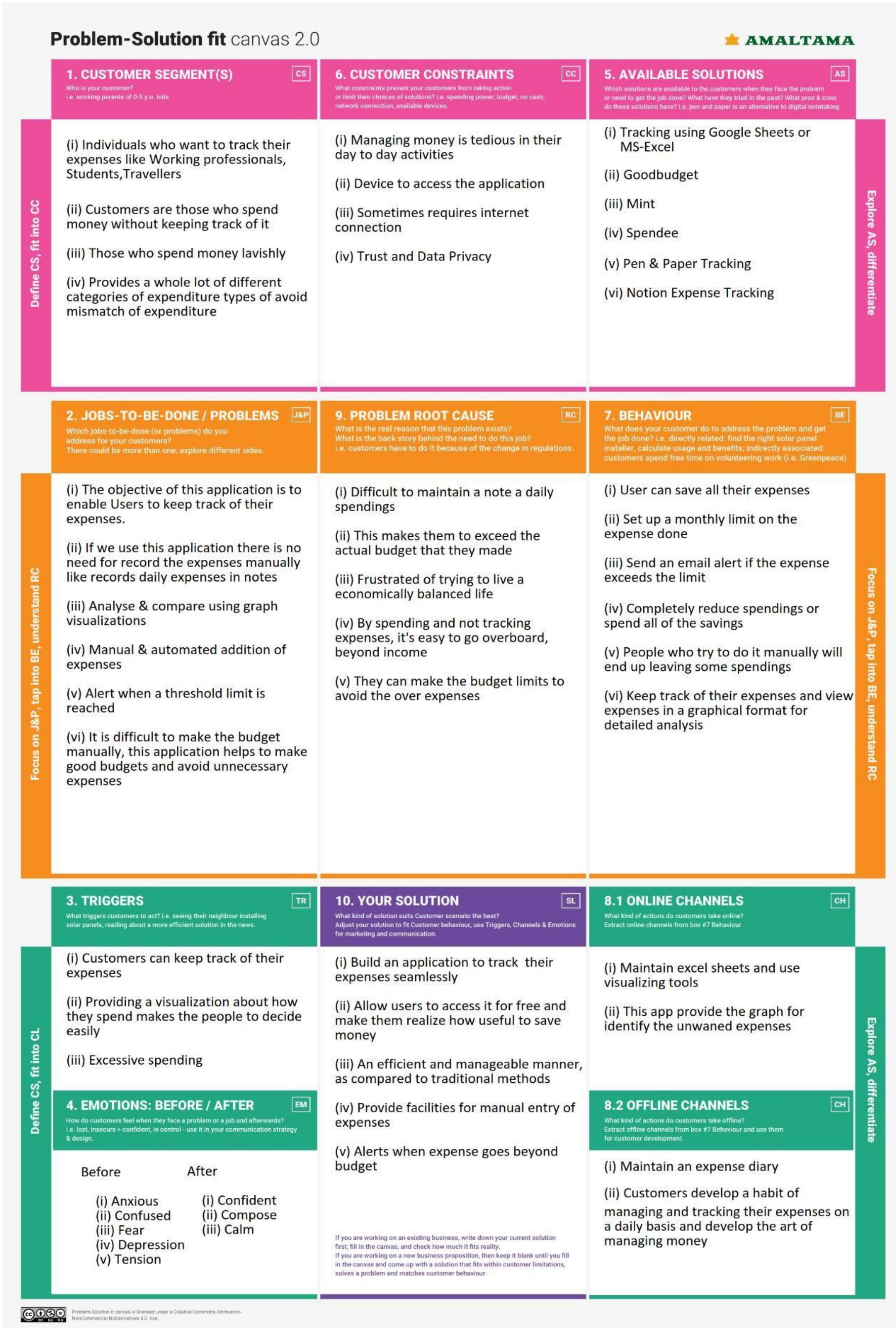
Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

Step-3: Idea Prioritization



3.3 Proposed Solution Fit



Problem-Solution fit canvas 2.0

AMALTAMA

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

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

(i) Individuals who want to track their expenses like Working professionals, Students,Travellers

(ii) Customers are those who spend money without keeping track of it

(iii) Those who spend money lavishly

(iv) Provides a whole lot of different categories of expenditure types of avoid mismatch of expenditure

6. CUSTOMER CONSTRAINTS 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.

(i) Managing money is tedious in their day to day activities

(ii) Device to access the application

(iii) Sometimes requires internet connection

(iv) Trust and Data Privacy

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

(i) Tracking using Google Sheets or MS-Excel

(ii) Goodbudget

(iii) Mint

(iv) Spendee

(v) Pen & Paper Tracking

(vi) Notion Expense Tracking

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS J&P

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

(i) The objective of this application is to enable Users to keep track of their expenses.

(ii) If we use this application there is no need for record the expenses manually like records daily expenses in notes

(iii) Analyse & compare using graph visualizations

(iv) Manual & automated addition of expenses

(v) Alert when a threshold limit is reached

(vi) It is difficult to make the budget manually, this application helps to make good budgets and avoid unnecessary expenses

9. PROBLEM ROOT CAUSE RC

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

(i) Difficult to maintain a note a daily spendings

(ii) This makes them to exceed the actual budget that they made

(iii) Frustrated of trying to live a economically balanced life

(iv) By spending and not tracking expenses, it's easy to go overboard, beyond income

(v) They can make the budget limits to avoid the over expenses

7. BEHAVIOUR BE

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

(i) User can save all their expenses

(ii) Set up a monthly limit on the expense done

(iii) Send an email alert if the expense exceeds the limit

(iv) Completely reduce spendings or spend all of the savings

(v) People who try to do it manually will end up leaving some spendings

(vi) Keep track of their expenses and view expenses in a graphical format for detailed analysis

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

Define CS, fit into CL

3. TRIGGERS TR

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

(i) Customers can keep track of their expenses

(ii) Providing a visualization about how they spend makes the people to decide easily

(iii) Excessive spending

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.

Before	After
(i) Anxious	(i) Confident
(ii) Confused	(ii) Compose
(iii) Fear	(iii) Calm
(iv) Depression	
(v) Tension	

10. YOUR SOLUTION SL

What kind of solution suits Customer scenario the best?
Adjust your solution to fit Customer behaviour, use Triggers, Channels & Emotions for marketing and communication.

(i) Build an application to track their expenses seamlessly

(ii) Allow users to access it for free and make them realize how useful to save money

(iii) An efficient and manageable manner, as compared to traditional methods

(iv) Provide facilities for manual entry of expenses

(v) Alerts when expense goes beyond budget

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

8.1 ONLINE CHANNELS CH

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

(i) Maintain excel sheets and use visualizing tools

(ii) This app provide the graph for identify the unwaned expenses

8.2 OFFLINE CHANNELS CH

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

(i) Maintain an expense diary

(ii) Customers develop a habit of managing and tracking their expenses on a daily basis and develop the art of managing money

Explore AS, differentiate

3.4 Problem Solution

S.No.	**Parameter**	**Description**
1.	Problem Statement (Problem to be solved)	All the financial decisions and activities that you make are unable to keep a track of it.This app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about financial management
2.	Idea / Solution description	personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management
3.	Novelty / Uniqueness	We will show the expenses in the Pie Chart In monthly and weekly basis
4.	Social Impact / Customer Satisfaction	It will help the people to track their expenses and also alerts when you exceed the limit of your budget.

5.	Business Model (Revenue Model)	We can provide the application in a subscription based
6.	Scalability of the Solution	IBM Cloud will automatically allocate storage for upcoming users.

4. Requirement Analysis

4.1 Function Requirement

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 LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Profile	View User's Personal details Add a car to their favorites list

FR-4	Car Registration	User can input information like car’s date of purchase, price, damages incurred etc
FR - 5	Viewing Past Predictions	Users are able to view past predictions for the price of the car. (Graph displaying the price of the car for a month)

4.2 Non-function Requirement

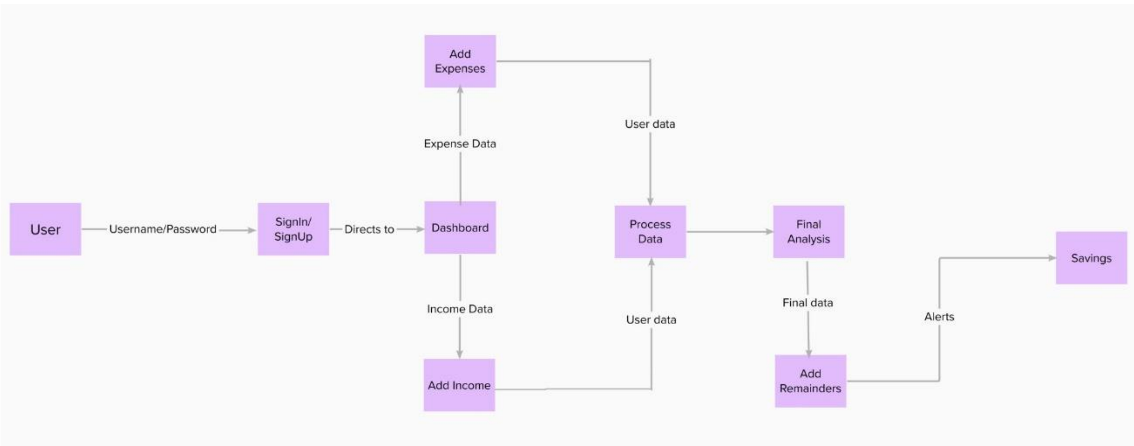
Following are the non-functional requirements of the proposed solution.

FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	It is used to track and manage user’s expenses.Easy navigation is provided through an integrated side menu.
NFR-2	**Security**	The security and the integrity of data is done by providing a password login system for each authorized user. Application is highly secure as all data is encrypted using a secure encryption algorithm.
NFR-3	**Reliability**	Application is highly reliable as it is deployed with IBM cloud assistance. The user information and complaints are stored carefully. There is no risk and loss of data.
NFR-4	**Performance**	Performance is stable and smooth as it is very light weight application built with flask framework.

NFR-5	**Availability**	Available all the time as it is deployed in IBM Cloud Servers
NFR-6	**Scalability**	Application is scalable as it uses IBM cloud resources and microservices architecture. Kubernetes cluster will manage the scalability parts by creating new pods in the cluster whenever required.

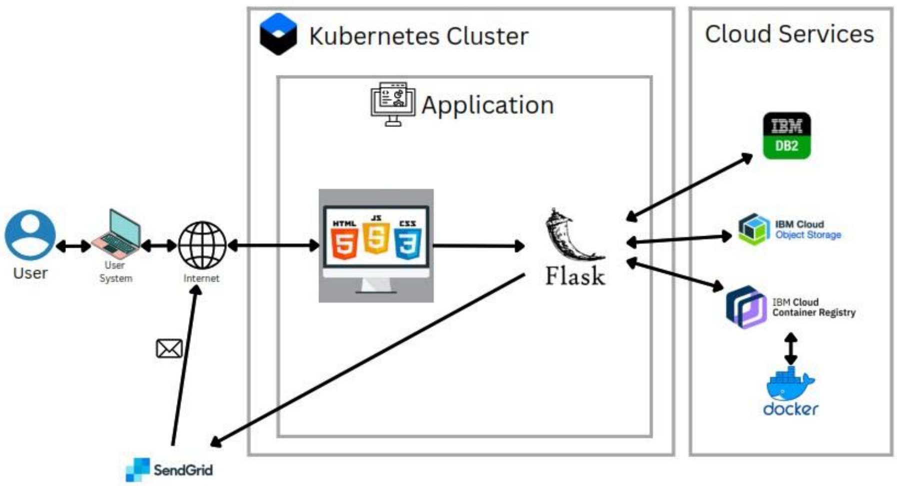
5. Project Design

5.1 Data Flow diagrams



5.2 Solution and Technical Architecture

Solution Architecture



Technical Interface

Table-1: Components & Technologies:

S.No	**Component**	**Description**	**Technology**
1.	User Interface	The user can interact with the application with the use of a Chatbot Web UI	HTML, CSS, JavaScript etc.
2.	Application Logic-1	The application contains the sign in/sign up where the user will login into the main dashboard	Flask(Python)
3.	Application Logic-2	The user will get the expense report in the graph form and also get alerts if the expense limit exceed	IBM Watson Assistant, SendGrid
4.	Cloud Database	The Income and Expense data are stored in the IBM DB2 database. With use of Database Service on Cloud, the User data are stored in a well secured Manner	IBM DB2.
5.	File Storage	IBM Block storage used to store the financial data of the user	IBM Block Storage.
6.	External API-1	To alert users when the limit of expense reaches of the budget	Send Grid

7.	Infrastructure (Server / Cloud)	Application will be deployed in cloud.	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Using flask to implement backend and connect with external services and database	Flask, Python, Kubernetes
2.	Security Implementations	This application provides high security to the user financial data. It can be done by using the container registry in IBM cloud	Container Registry, Docker, Kubernetes Cluster
3.	Scalable Architecture	Meet up the high demands of the user, as the number of users increases the application should be able to meet the requirements. Using Microservices Architecture to provide scalable application	Container Registry, Docker, Kubernetes Cluster
4.	Availability	This application will be available to the user at any part of time. Deploying the application with Kubernetes cluster to make application available across the globe on the internet	Container Registry, Docker, Kubernetes Cluster
5.	Performance	Can handle a large number of requests per second. The performance will be high because there will be no network traffics in the application	Docker, Kubernetes Cluste

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web/Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	Login	USN-2	As a user, I can log into the application by entering email & password	I can access the application	High	Sprint-1
	Dashboard	USN-3	As a user can enter the daily expense and view Graphs and see how money is spent	I can view my daily expenses	High	Sprint-2
	Add Expense	USN-4	User can add the daily expense by entering in this page	I can add the daily expense	High	Sprint-2
	Report Generation	USN-5	User can use this to generate the report in PDF or Excel	I can download the report	High	Sprint-3
	Add the Payments Remainders	USN-6	In this user can add their monthly Payments remainder like EB Bill,Broadband Bill,Insurance.	I can add the Payments Remainder	Medium	Sprint-4
Customer Care Executive		USN-7	As a customer care executive ,it is easy to solve the problem that faced by the customers	I can provide support to customers at any time 24*7.	Medium	Sprint-3
Administrator	Application	USN-8	As an administrator I can upgrade or add new features through update	I can fix the bug which arises for the customers and users of the application	Medium	

6.Project Planning and Scheduling

6.1 Sprint Planning and Estimation

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the selected project and collecting other information	28 SEPTEMBER 2022
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	19 SEPTEMBER2022

Ideation	List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	19 SEPTEMBER2022
Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	23 SEPTEMBER 2022
Problem Solution Fit	Prepare problem - solution fit document.	30 SEPTEMBER 2022
Solution Architecture	Prepare a solution architecture document.	28 SEPTEMBER 2022
Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	20 OCTOBER 2022
Functional Requirement	Prepare the functional requirement document.	8 OCTOBER 2022
Data Flow Diagrams	Draw the data flow diagrams and submit for review.	9 OCTOBER2022

Technology Architecture	Prepare the technology architecture diagram.	10 OCTOBER 2022
Prepare Milestone & Activity List	Prepare the milestones & activity list of the project.	22 OCTOBER 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	IN PROGRESS

6.2 Sprint Delivery Schedule

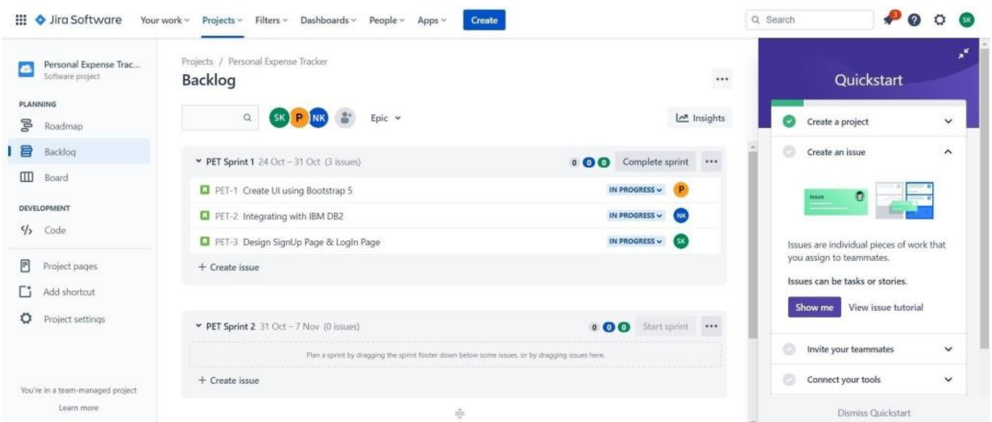
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration Page	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	8	High	Narayan Kumar, Shai Prashanth
Sprint-1	Login Page	USN-2	As a user, I can log into the application by entering email & password	8	High	Narayan Kumar, Shai Prashanth
Sprint-1	Add Transactions	USN-3	As a user, I can add the day to day expense to the application	8	High	Narayan Kumar, Shai Prashanth
Sprint-2	Delete the expesnse	USN-4	As a user, I can delete the previously created expenses	8	High	Narayan Kumar, Shai Prashanth
Sprint-2	Passbook	USN-5	As a user, I can see the time-based history of expenses.	4	High	Narayan Kumar
Sprint-3	Adding the Profile Page	USN-6	As a user, I can see the see and edit the name, email, Income and Edit the Limit	4	Medium	Narayan Kumar, Shai Prashanth

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3	Creating Pie Charts based on the expense	USN-7	As a user, I can view diagrammatic representation of expenses	6	High	Narayan Kumar, Shai Prashanth
Sprint-3	Dashboard	USN-8	As a user,I can see the Pie Chart, Add transactions Button and Last Few Transactions	10	High	Narayan Kumar, Shai Prashanth
Sprint-4	Sending E-mail	USN-9	Sending a Email to user when the expense limit reaches it	4	High	Narayan Kumar
Sprint-4	Testing	USN-10	Testing the application with various tools	8	High	Narayan Kumar, Shai Prashanth
Sprint-4	Deployment	USN-11	Deployment of the Application	8	High	Narayan Kumar, Shai Prashanth

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

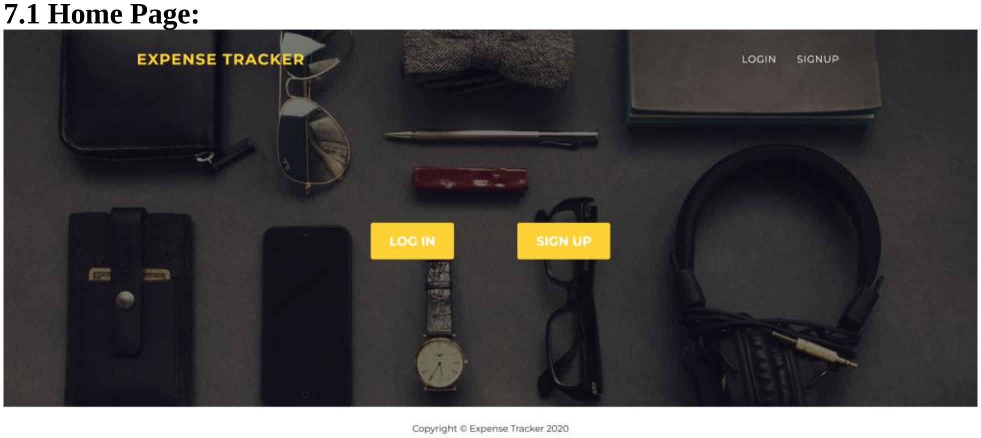
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Reports from JIRA

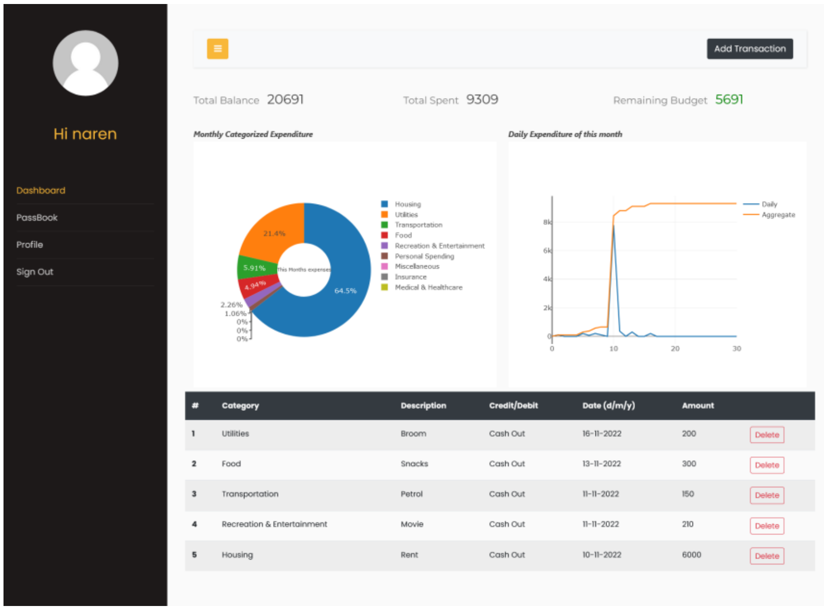


7.Coding and Solutioning

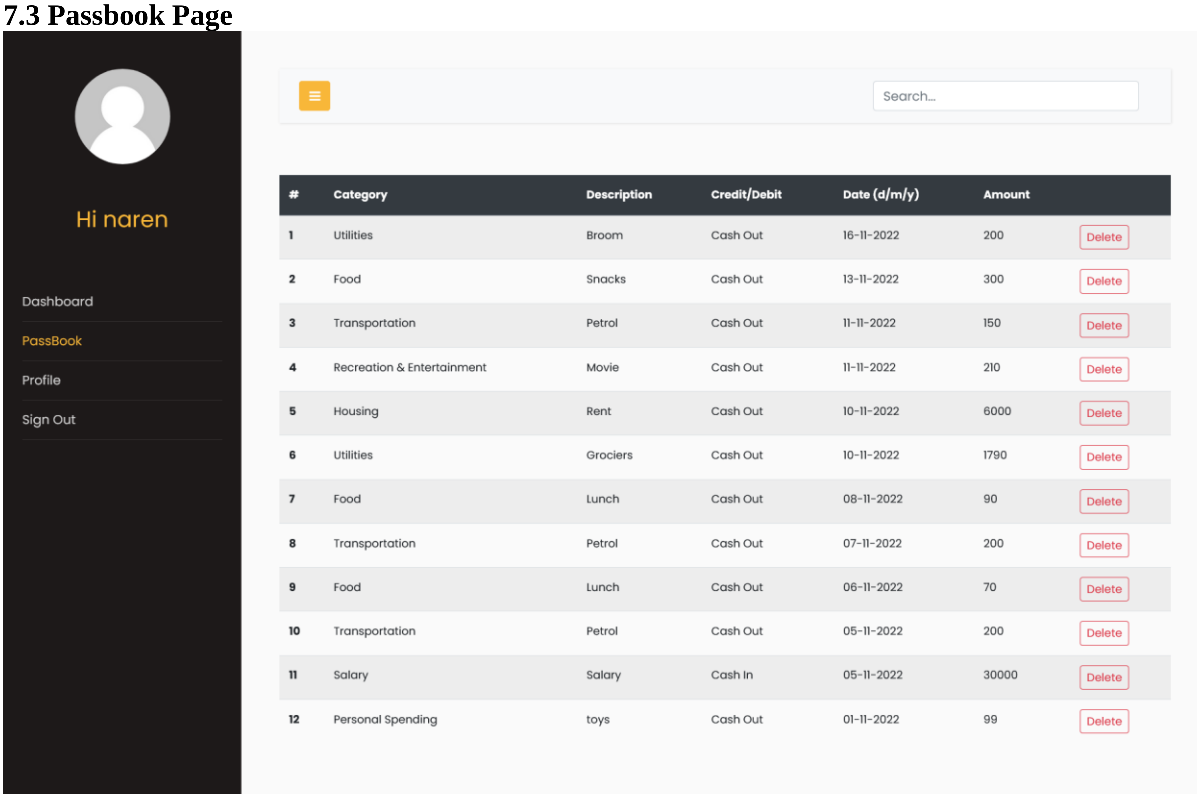
7.1 Home Page:



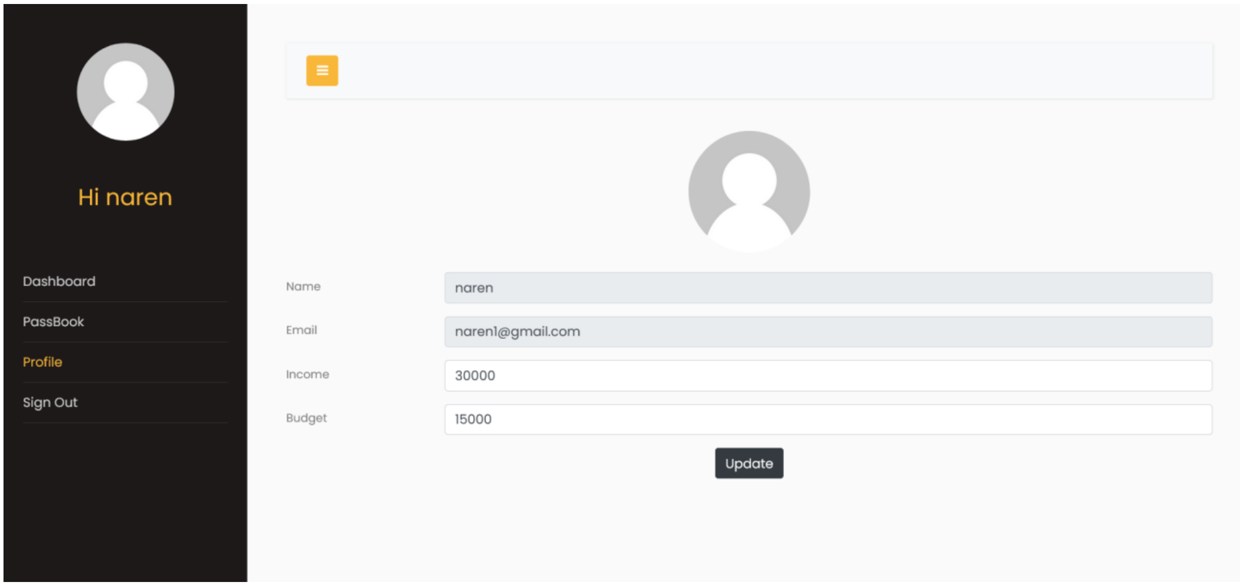
7.2 Dashboard Page



7.3 Passbook Page



7.4 Profile Page



8.Testing

8.1 Test Cases

1	Verify that the user is able to login successfully on entering appropriate credentials.
2	The UI elements, such as the card for the login, the button to submit etc are functional and are rendered properly in all devices.
3	Users who enter invalid credentials will not be redirected to the dashboard
4	Verify user is able to register themselves to the application.
5	The UI elements, such as the card for the registration, the button to submit etc are functional and are rendered properly in all devices.
6	The user should not be able to register successfully if any of the fields are left empy.
7	The user should receive the email from nunnaaarthi@gmail.com, stating a successfully registration.
8	The UI part of the dashboard, the side navbar, the logout option, the cards showing various expenses, and the wallet edit icon must be functioning and rendered properly
9	The UI part of the add expense page must be rendered and functioning.
10	The user must be able to add an expense.
11	The user should not be able to add the same expense again.
12	The user should be able to update the balance and it must be reflected in the dashboard where the wallet balance is displayed
13	The user views all the UI components rendered properly and functioning accordingly.
14	The user should be able to set a monthly limit for his expenditures.
15	The page must render properly and function appropriately.
16	The user must be able to view the analysis of their expenses.
17	The UI which contains of 2 graphs must be rendered properly.
18	The user must be able to view the analysis of their expenses.

8.2 User Acceptance Testing

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_001	Functional	Login Page	Verify that the user is able to login successfully on entering appropriate credentials.	User must have already been registered.	1. Enter the URL for the Application. 2. Click on the existing user option. 3. Enter the credentials for logging in.	Username: naren1@gmail.com Password: 123	User should be able to submit the login request and be redirected to the dashboard page.	Working as expected	Pass				Narayan Kumar
LoginPage_TC_002	UI	Login Page	The UI elements, such as the card for the login, the button to submit etc are functional and are rendered properly in all devices.	User must have already been registered.	1. Enter the URL for the Application. 2. Click on the existing user option. 3. The following UI elements must be found: a. Card for Login b. Textbox for username c. Textbox for password	Nil	Application should show below UI elements: a.Username textbox b.Password textbox c.Login button in green d.Link for registration	Working as expected	Pass				Narayan Kumar
LoginPage_TC_003	Functional	Login Page	Users who enter invalid credentials will not be redirected to the dashboard	User must have already been registered.	1. Enter the URL for the Application. 2. Click on the existing user option. 3. Enter the credentials for logging in.	Username: naren1@gmail.com Password: 123456	User should not be redirected to the dashboard page.	Working as expected	Pass				Narayan Kumar
RegistrationPage_TC_003	Functional	Registration page	Verify user is able to register themselves to the application.		1. Enter the URL for the applcation. 2. Enter the following details: a. Email ID / Username b. Password c. Confirm Balance d. Initial Wallet Amount	Username: test1@gmail.com Password: Testing@123 Confirm password: Testing@123 Intitial Wallet Amount: 15,000	User should be directed to the dashboard page, and wallet balance must 15,000.	Working as expected	Pass				Narayan Kumar
RegistrationPage_TC_004	UI	Registration page	The UI elements, such as the card for the registration, the button to submit etc are functional and are rendered properly in all devices.		1. Enter the URL for the Application. 2. Click on the existing user option. 3. The following UI elements must be found: a. Card for Registration b. Textbox for username c. Textbox for password d. Textbox for confirm password e. Textbox for Wallet amount	Nil	Application should show below UI elements: a. Card for Registration b. Textbox for username c. Textbox for password d. Textbox for confirm password e. Textbox for Wallet amount f. Submit button	Working as expected	Pass				Shai Prashanth
RegistrationPage_TC_005	Functional	Registration page	The user should not be able to register successfully if any of the fields are left empy.		1. Enter the URL for the Application. 2. Click on the existing user option. 3. The following UI elements must be found: a. Card for Registration b. Textbox for username c. Textbox for password d. Textbox for confirm password e. Textbox for Wallet amount	Username: test1@gmail.com Password: Testing@123 Confirm password: Testing@123 Intitial Wallet Amount: 15,000	User is not able to register.	Working as expected	Pass				Shai Prashanth
SendGridEmailService_TC_006	Functional	Registration page	The user should receive the email from nunnaaarthi@gmail.com, stating a successfully		1. Enter the URL for the Application. 2. Click on the existing user option. 3. The following UI elements must be found: a. Card for Registration b. Textbox for username	Username: test@gmail.com Password: Testing123! Confirm password: Testing123! Intitial Wallet Amount:	User receives the mail from nunnaaarthi@gmail.com stating a successful reigstration.	Working as expected	Pass				Shai Prashanth

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
lashboard_TC_00	UI	Dashboard Page	The UI part of the dashboard, the side navbar, the logout option, the cards showing various expenses, and the wallet edit icon must be		1. Enter the URL for the application. 2. Login to with user credentials. 3. View the dashboard.	username: naren1@gmail.com password: 123	Application should show below UI elements: a.Side navbar b.Log Out button c. Wallet Edit Icon	Working as expected	Pass				Shai Prashanth
ddExpense_TC_0(	UI	Add Expense Page	The UI part of the add expense page must be rendered and functioning.		1. Enter the URL for the application. 2. Login to with user credentials. 3. View the dashboard. 4. Click on Add Expense. 5. View the card for the expense creation.	username: naren1@gmail.com password: 123	Application should show the below UI elements: a. Card for the expense addition b. Textbox for amount spent c. Selecting a category d.Date of expense e. Description of expense f.Group	Working as expected	Pass				Sathish Kumar
ddExpense_TC_0(	Functional	Add Expense Page	The user must be able to add an expense.	The user must have already logged in and created a group called Invente.	1. The user creates the expense by filling in the fields. 2. On filling the fields, the user submits the expense.	Amount Spent: 1000 Category: Food Date of Expense: 12/11/2022 Description of Expense: Swiggy Group: Invente	The expense gets created and the same must be updated in the dashboard and rendered.	king as expec	Pass				Sathish Kumar
AddExpense_TC_010	Functional	Add Expense Page	The user should not be able to add the same expense again.	The user must have already logged in and must have already registered an identical expense.	1. The user creates the expense by filling in the fields. 2. On filling the fields, the user submits the expense.	Amount Spent: 1000 Category: Food Date of Expense: 12/11/2022 Description of Expense: Swiggy	The expense must not be added to the database again.	Expense is added again.	Failed	Add implementation to ensure that the same expense cannot be made twice.		BUG_01	Sathish Kumar
UpdateBalance_TC_011	Functional	Update Balance Page	The user should be able to update the balance and it must be reflected in the dashboard where the wallet balance is displayed.	The user must have already logged in.	1. The user logs in and views the dashboard. 2. The user clicks on the edit icon to edit the balance. 3. The user views the update balance page. 4. The user enters new balance	New Balance: 30000	The dashboard shows Rs 30000 as the wallet balance.	Working as expected.	Pass				Sathish Kumar
UpdateBalance_TC_012	UI	Update Balance Page	The user views all the UI components rendered properly and functioning accordingly.	The user must have already logged in.	1. The user logs in and views the dashboard. 2. The user clicks on the edit icon to edit the balance. 3. The user views the following UI components: a. The current wallet balance (readonly) b. Textbox for updated balance	Nil	Application should show the below UI elements: a. Card for the wallet balance updation b. Readonly textbox showing current balance c. Textbox for updated balance d. Update balance button	Working as expected	Pass				Sathish Kumar
SetMonthlyLimit_TC_013	Functional	Set Monthly Limit Page	The user should be able to set a monthly limit for his expenditures.	The user must have already logged in.	1. The user logs in. 2. The user clicks on the set monthly limit and views the set limit card.	Monthly Limit: 5000	The monthly limit must be set, and if thee user spends more than 5000 within the month, a mail must be sent to the user.	Working as expected	Pass				Preethi

9.Results

9.1 Performance Metrics

S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes	Risk Score
1	Personal Expense Tracker Application	New	Low	No Changes	Moderate	Yes, 2hrs	>10 to 30%	GREEN

NFT - Detailed Test Plan

S.No	Project Overview	NFT Test approach	Assumptions/Dependencies/Risks	Approvals/SignOff
1	Login Page	1) Open the Personal Expense Tracker Application 2) Login with user Credentials	No Risks	N/A
2	Signup Page	1) Open the Personal Expense Tracker Application 2) Enter the Details and Create a new User	No Risks	N/A
3	Passbook Page	1) Log in to Personal Expense Tracker Application 2) Where history of transaction can be seen	No Risks	N/A
4	Dashboard	1) Log in to Personal Expense Tracker Application 2) View the Pie-Chart 3) View the Last 5 Transactions	No Risks	N/A
5	Profile Page	1) Log in to Personal Expense Tracker Application 2) Where user can change the income and Budget	No Risks	N/A
6	Email Acknowledgement	1) Mails are Sent to the Registered if user exceeds budget	No Risks	N/A

End Of Test Report

S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff
1	Personal Expense Tracker Application	1) Log in to Personal Expense Tracker Application 2) Test for all Testcases 3) Log out to Personal Expense Tracker Application	YES	Test Passed	GO/NO-GO decision	N/A	None	N/A

10.Advantages and Disadvantages

Advantages

- The users will be able to track their expenses easily.
- Avoid papers and calculations
- Better understanding of their spending behavior
- Avoid overspending

Disadvantages

- Good internet connection is need
- Manual way of adding transactions will break the user experience
- Complex analysis based on categories is not supported
- Recurring expenses have a period of exactly one month, which is not customizable

11.Conclusion

A spending plan (also called a budget) is simply a plan you create to help you meet expenses and spend money the way you want to spend it. A good spending plan can help you stop “spending leaks”; in other words, it can keep you from spending money without thinking. It can help you make sure you have money to pay bills on time, even when your bills and income change each month

12. Future Scope

- The users can be allowed to integrate their bank accounts, crypto wallets, etc to avoid wasting time on manual way of adding transactions.

- Summary can be provided to the users based on the spending behavior in the application
- Having the application itself reward the user for accomplishment of various goals, such as coupons, vouchers, etc

13. Appendix

Source Code

App.py

```
from connect import app, db
from flask import Flask, render_template, session, redirect, url_for, request,
flash
from flask_login import login_user, logout_user, login_required, current_user
from forms import loginForm, registrationForm, transactionForm, profiles
from models import Users, Transactions
from werkzeug.security import generate_password_hash, check_password_hash
from flask_msearch import Search
from picture_handler import add_profile_pic
from graphs import ghaint_chart, baseGraph, baseGraph2, savingGraph, savingGraph2
from datetime import datetime
from values import totalBal, leftBal, totalSpent

today = datetime.today()
now = datetime.now()


@app.route("/", methods=["GET", "POST"])
def index():

    RegistrationForm = registrationForm()
    LoginForm = loginForm()

    if RegistrationForm.validate_on_submit():
        user = Users.query.filter_by(email=RegistrationForm.email2.data).first()

        if user is not None:
            flash("Email Id already registered!")

        else:

            passw = generate_password_hash(RegistrationForm.password2.data)

            user = Users(
                email=RegistrationForm.email2.data,
```

```
                name=RegistrationForm.username2.data,
                pasword_hash=passw,
            )

            db.session.add(user)
            db.session.commit()
            flash("Thanks for registeration! Login to continue")
            return redirect(url_for("index"))

    if LoginForm.validate_on_submit():
        user = Users.query.filter_by(email=LoginForm.email1.data).first()

        if user is not None and user.check_password(LoginForm.password1.data):

            login_user(user)
            # flash('Log in Success')

            next = request.args.get("next")

            if next == None or not next[0] == "/":
                next = url_for("dashboard")

            return redirect(next)

        elif user is None:
            flash("Email Id not registered!")

        else:
            flash("Wrong Password!")

    return render_template("index.html", logForm=LoginForm,
signForm=RegistrationForm)


@app.route("/instructions")
def instructions():
    return render_template("instructions.html")


# @app.route("/test")
# def test():
#     bar = testchart()
#     return render_template("test.html", plot=bar)
```

```
@app.route("/logout")
def logout():
    logout_user()
    return redirect(url_for("index"))


@app.route("/dashboard", methods=["GET", "POST"])
@login_required
def dashboard():
    transForm = transactionForm()
    bar = ghaint_chart()
    line = baseGraph()
    line2 = baseGraph2()
    line3 = savingGraph()
    line4 = savingGraph2()
    TotalBal = totalBal()
    LeftBal = leftBal()
    TotalSpent = totalSpent()

    if transForm.validate_on_submit():
        print("In form")
        data = Transactions(
            cashFlow=transForm.flow.data,
            amount=transForm.amount.data,
            description=transForm.description.data,
            cat=transForm.category.data,
            date=transForm.date.data,
            userId=current_user.id,
        )

        db.session.add(data)
        db.session.commit()
        flash("Transaction added successfully!")
        print("data send")
        return redirect(url_for("dashboard"))
    # elif not transForm.validate_on_submit():
    #     flash("Some error occured! Make sure you have filled all feilds
correctly")
    uid = current_user.id
    trans_data = Transactions.query.filter_by(userId=uid).order_by(
        Transactions.date.desc()
    ).limit(5)
    return render_template(
        "dashboard.html",
        transForm=transForm,
```

```
        plot=bar,
        plot2=line,
        plot3=line2,
        plot4=line3,
        plot5=line4,
        totalBal=TotalBal,
        leftBal=LeftBal,
        trans_data=trans_data,
        totalSpent=TotalSpent,
    )


search = Search()
search.init_app(app)


@app.route("/search")
def search():
    data = Transactions.query.msearch(request.args.get("query")).all()
    return render_template("passbook.html", trans_data=data)


@app.route("/<int:row_id>/delete", methods=["POST"])
@login_required
def delete(row_id):
    delete__row = Transactions.query.get_or_404(row_id)
    db.session.delete(delete__row)
    db.session.commit()
    flash("Transaction deleted!")

    return redirect(url_for("passbook"))


@app.route("/passbook", methods=["GET", "POST"])
@login_required
def passbook():
    uid = current_user.id
    trans_data = Transactions.query.filter_by(userId=uid).order_by(
        Transactions.date.desc()
    )
    return render_template("passbook.html", trans_data=trans_data)


@app.route("/profile", methods=["GET", "POST"])
@login_required
def profile():
```

```
    form = profiles()

    if form.validate_on_submit():
        print("in if")
        if form.image.data:
            # print("image added")
            username = current_user.id
            image_data = current_user.profile_image
            # print(image_data)
            current_user.profile_image = "default_profile.jpeg"
            db.session.commit()
            pic = add_profile_pic(form.image.data, username, image_data)
            current_user.profile_image = pic

        current_user.budget = form.budget.data
        current_user.income = form.income.data
        db.session.commit()
        flash("User Account Updated")
        return redirect(url_for("profile"))

    elif request.method == "GET":
        form.name.data = current_user.name
        form.email.data = current_user.email
        form.budget.data = current_user.budget
        form.income.data = current_user.income
        form.image.data = current_user.profile_image

    profile_image = url_for(
        "static", filename="profile_pics/" + current_user.profile_image
    )


    return render_template("profile.html", proForm=form,
profile_image=profile_image)


if __name__ == "__main__":
    app.run(host="0.0.0.0",port=int("3000"), debug=True)
```

forms.py

```
from flask_wtf import FlaskForm
```

```
from wtforms import (StringField, BooleanField, DateTimeField, RadioField,
SelectField, PasswordField,
                     TextField, TextAreaField, SubmitField, IntegerField,
FileField, ValidationError, DateField, DecimalField)
from wtforms.validators import DataRequired, Email, EqualTo, Required
from flask_wtf.file import FileField, FileAllowed
from categories import cats


class profiles(FlaskForm):
    name = StringField('Name')
    email = StringField('Email')
    image = FileField('Update Image', validators=[
                      FileAllowed(['jpg', 'png', 'jpeg','jfif'])])
    budget = IntegerField('Monthly Budget', validators=[DataRequired()])
    income = IntegerField('Monthly Income', validators=[DataRequired()])
    submit = SubmitField('Update')


class loginForm(FlaskForm):
    email1 = StringField('Email', validators=[DataRequired(), Email(message=('Not
a valid email address!'))])
    password1 = PasswordField('Password', validators=[DataRequired()])
    submit1 = SubmitField('Log In')


class registrationForm(FlaskForm):
    username2 = StringField('UserName', validators=[DataRequired()])
    email2 = StringField('Email', validators=[DataRequired(), Email(message=('Not
a valid email address!'))])
    password2 = PasswordField('Password', validators=[DataRequired(), EqualTo(
        'pass_confirm', message='Passwords must match')])
    pass_confirm = PasswordField(
        'Confirm Password', validators=[DataRequired()])
    submit2 = SubmitField('Register')

    def check_email(self, field):
        if Users.query.filter_by(email=field.data).first():
            raise ValidationError('Your email has been registered already!')


class transactionForm(FlaskForm):
    amount = IntegerField('Amount', validators=[DataRequired()])
    date = DateField('Date of Transaction', validators=[
                     DataRequired()], format='%d/%m/%Y',
render_kw={'placeholder': '20/6/2015 for June 20, 2015'})
```

```
    description = TextField('Description', validators=[DataRequired()])
    flow = RadioField('flow',coerce=int, choices=[
                      (1, 'Cash In'), (2, 'Cash Out')],
validators=[DataRequired()])
    submit = SubmitField('Add Transaction')
    category = SelectField(
        u'Category', choices=cats, validators=[DataRequired()])
```

Dockerfile

```
FROM python:3.9
WORKDIR /app
ADD . /app
COPY requirements.txt /app
RUN python3 -m pip install -r requirements.txt
EXPOSE 3000
CMD ["python","app.py"]
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no" />
        <meta name="description" content="" />
        <meta name="author" content="" />
        <title>Expense Tracker</title>
        <link rel="icon" type="image/x-icon" href="../static/img/favicon.ico" />
        <!-- Font Awesome icons (free version)-->
        <script src="https://use.fontawesome.com/releases/v5.13.0/js/all.js"
crossorigin="anonymous"></script>
        <!-- Google fonts-->
        <link href="https://fonts.googleapis.com/css?family=Montserrat:400,700"
rel="stylesheet" type="text/css" />
        <link
href="https://fonts.googleapis.com/css?family=Droid+Serif:400,700,400italic,700it
alic" rel="stylesheet" type="text/css" />
        <link
href="https://fonts.googleapis.com/css?family=Roboto+Slab:400,100,300,700"
rel="stylesheet" type="text/css" />
        <!-- Core theme CSS (includes Bootstrap)-->
```

```
        <link href="{{ url_for('static', filename='css/styles.css') }}"
rel="stylesheet" />
    </head>
    <body id="page-top">

        <!-- Navigation-->
        <nav class="navbar navbar-expand-lg navbar-dark fixed-top" id="mainNav">
            <div class="container">
                <a class="navbar-brand js-scroll-trigger" href="#page-
top">Expense Tracker  </a>

                <button class="navbar-toggler navbar-toggler-right" type="button"
data-toggle="collapse" data-target="#navbarResponsive" aria-
controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">
                    Menu
                    <i class="fas fa-bars ml-1"></i>
                </button>
                <div class="collapse navbar-collapse" id="navbarResponsive">
                    <ul class="navbar-nav text-uppercase ml-auto">
                        <li class="nav-item"><a class="nav-link js-scroll-
trigger" data-toggle="modal" data-target="#loginModal">Login</a></li>
                        <li class="nav-item"><a class="nav-link js-scroll-
trigger" href="#contact">SignUp</a></li>
                    </ul>
                </div>
            </div>
        </nav>
        <!-- Masthead-->

        <header class="masthead">
            {% for mess in get_flashed_messages() %}
            <center>
                <div class="alert alert-warning alert-dismissible fade show"
role="alert" style="width: 50%;">
                    <button
                        type="button"
                        class="fad close"
                        data-dismiss="alert"
                        aria-label="Close"
                    >
                        <span aria-hidden="true">&times;</span>
                    </button>
                    {{mess}}
                </div>
            </center>
```

```
            {% endfor %}
            <div class="container">
                <a class="btn btn-primary btn-xl text-uppercase js-scroll-
trigger"  data-toggle="modal" data-target="#loginModal" style="margin:2rem 3rem
2rem">Log In</a>
                <a class="btn btn-primary btn-xl text-uppercase js-scroll-
trigger"  data-toggle="modal" data-target="#signupModal" style="margin:2rem 3rem
2rem">Sign Up</a>
            </div>
        </header>

        <!-- Footer-->
        <footer class="footer py-4">
            <div class="container">
                <div class="row align-items-center">
                    <div class="col-lg-4 text-lg-left"></div>
                    <div class="col-lg-4 my-3 my-lg-0">
                        Copyright © Expense Tracker 2020
                    </div>
                    <div class="col-lg-4 text-lg-right">

                    </div>
                </div>
            </div>
        </footer>


        <!-- user model (login) -->

        <div class="modal fade" style="font-family: Montserrat;" id="loginModal"
data-backdrop="static" data-keyboard="false" tabindex="-1"
            aria-labelledby="staticBackdropLabel" aria-hidden="true">
            <div class="modal-dialog modal-dialog-centered">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title" id="staticBackdropLabel">Log
In</h5>
                        <button type="button" class="close" data-dismiss="modal"
aria-label="Close">
                            <span aria-hidden="true">&times;</span>
                        </button>
                    </div>
                    <div class="modal-body">
                        <form method="POST" id="loginform">
                            {{logForm.hidden_tag()}}
```

```
                            <div class="form-group row">
                                <label class="col-sm-2 col-form-
label">Email</label>
                                <div class="col-sm-10">
                                    {% if logForm.email1.errors %}
                                    <span class="text-danger">
                                        {% for error in logForm.email1.errors %}
                                          {% if error != "This field is
required." %}
                                            {{ error }}
                                        {%endif%}
                                        {% endfor %}
                                    </span>
                                    {% endif %}
                                    {{logForm.email1(class='form-control')}}
                                </div>
                            </div>
                            <div class="form-group row">
                                <label class="col-sm-2 col-form-
label">Password</label>
                                <div class="col-sm-10">
                                    {{logForm.password1(class='form-control')}}
                                </div>
                            </div>
                        </form>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>
                        {{logForm.submit1(class="btn btn-
primary",form="loginform")}}
                    </div>
                </div>
            </div>
        </div>

        <!-- user model (signup) -->

        <div class="modal fade" style="font-family: Montserrat;" id="signupModal"
data-backdrop="static" data-keyboard="false" tabindex="-1"
            aria-labelledby="staticBackdropLabel" aria-hidden="true">
            <div class="modal-dialog modal-dialog-centered">
                <div class="modal-content">
                    <div class="modal-header">
```

```
                        <h5 class="modal-title" id="staticBackdropLabel">Sign
Up</h5>
                        <button type="button" class="close" data-dismiss="modal"
aria-label="Close">
                            <span aria-hidden="true">&times;</span>
                        </button>
                    </div>
                    <div class="modal-body">
                        <form method="POST" id="signupform">
                            {{signForm.hidden_tag()}}
                              <div class="form-group row">
                                <label class="col-sm-2 col-form-
label">Name</label>
                                <div class="col-sm-10">
                                    {{signForm.username2(class='form-
control',required="required")}}
                                </div>
                            </div>
                            <div class="form-group row">

                                <label class="col-sm-2 col-form-
label">Email</label>
                                <div class="col-sm-10">
                                    {% if signForm.email2.errors %}
                                    <span class="text-danger">
                                        {% for error in signForm.email2.errors %}
                                        {% if error != "This field is required."
%}

                                            {{ error }}
                                            {%endif%}
                                        {% endfor %}
                                    </span>
                                    {% endif %}
                                    {{signForm.email2(class="form-control")}}

                                </div>
                            </div>
                            <div class="form-group row">
                                <label class="col-sm-2 col-form-
label">Password</label>
                                <div class="col-sm-10">
                                    <!-- <input type="password" class="form-
control" id="inputPassword"> -->
                                    {% if signForm.password2.errors %}
                                    <span class="text-danger">
```

```
                                        {% for error in signForm.password2.errors
%}
                                        {% if error != "This field is required."
%}
                                            {{ error }}
                                            {%endif%}
                                        {% endfor %}
                                    </span>
                                    {% endif %}
                                    {{signForm.password2(class='form-control')}}
                                </div>
                            </div>
                            <div class="form-group row">
                                <label class="col-sm-2 col-form-label">Confirm
Password</label>
                                <div class="col-sm-10">
                                    <!-- <input type="password" class="form-
control" id="inputPassword"> -->
                                    {{signForm.pass_confirm(class='form-
control')}}
                                </div>
                            </div>
                        </form>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>
                        <!-- <button type="button" class="btn btn-primary">Sign
Up</button> -->
                        {{signForm.submit2(class='btn btn-
primary',form="signupform")}}
                    </div>
                </div>
            </div>
        </div>


        <!-- Bootstrap core JS-->
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
        <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.bundle.min.j
s"></script>
        <!-- Third party plugin JS-->
```

```
        <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-
easing/1.4.1/jquery.easing.min.js"></script>
        <!-- Contact form JS-->
        <script src="{{ url_for('static',
filename='mail/jqBootstrapValidation.js') }}"></script>
        <script src="{{ url_for('static', filename='mail/contact_me.js')
}}"></script>
        <!-- Core theme JS-->
        <script src="{{ url_for('static', filename='js/scripts.js') }}"></script>
    </body>
</html>
```

Dashboard.html

```
<!doctype html>
<html lang="en">
  <head>
    <title>Expense Tracker</title>
    <meta charset="utf-8">
    <link rel="icon" type="image/x-icon" href="../static/img/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-
to-fit=no">

    <link
href="https://fonts.googleapis.com/css?family=Poppins:300,400,500,600,700,800,900
" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css?family=Montserrat:400,700"
rel="stylesheet" type="text/css" />
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='css1/style.css')
}}">
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/d3/3.5.6/d3.min.js"></script>
  </head>
  <body style="overflow-x: hidden;">

    <div class="wrapper d-flex align-items-stretch">
      <nav id="sidebar">
        <div class="p-4 pt-5">
          <a href="#" class="img logo rounded-circle mb-5" style="background-
image:
```

```
url({{url_for('static',filename='profile_pics/'+current_user.profile_image)}});">
</a>
          <h3 style="color: #f8b739; font-family: Poppins, Arial, sans-serif;
text-align: center;">Hi {{current_user.name}}</h3>
          <br>
          <br>
          <ul class="list-unstyled components mb-5">

            <li class="active">
              <a href="{{url_for('dashboard')}}" >Dashboard</a>
            </li>

            <li>
              <a href="{{url_for('passbook')}}">PassBook</a>
            </li>

            <li>
              <a href="{{url_for('profile')}}">Profile</a>
            </li>
            <li>
              <a href="{{url_for('logout')}}">Sign Out</a>
            </li>

          </ul>

        </div>
      </nav>

        <!-- Page Content  -->
      <div id="content" class="p-4 p-md-5">

        <nav class="navbar navbar-expand-lg navbar-light bg-light">
          <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn btn-primary">
              <i class="fa fa-bars"></i>
              <span class="sr-only">Toggle Menu</span>
            </button>

            <button class="btn btn-dark ml-auto d-inline-block" data-
toggle="modal" data-target="#transaction">Add Transaction</button>

          </div>
        </nav>
          {% for mess in get_flashed_messages() %}
```

```
          <div class="alert alert-warning alert-dismissible fade show"
role="alert">
            <button
              type="button"
              class="fad close"
              data-dismiss="alert"
              aria-label="Close"
            >
              <span aria-hidden="true">&times;</span>
            </button>
            {{mess}}
          </div>
          {% endfor %}
        <div class="row">
          <div class="col-md-4">
            <h4 style="font-family: Montserrat; color: #444;"><span style="color:
grey; font-size: large;">Total Balance   </span>{{totalBal}}</h4>
          </div>
          <div class="col-md-4">
            <h4 style="font-family: Montserrat; color: #444;"><span style="color:
grey; font-size: large;">Total Spent   </span>{{totalSpent}}</h4>
          </div>
          <div class="col-md-4">
            {%if leftBal>=0%}
            <h4 style="font-family: Montserrat; color: #444;"><span style="color:
grey; font-size: large;">Remaining Budget   </span><span style="color:
green;">{{leftBal}}</span></h4>
            {%else%}
            <h4 style="font-family: Montserrat; color: #444;"><span style="color:
grey; font-size: large;">Overspent   </span><span style="color: red;">{{(-
leftBal)}}</span></h4>
            {%endif%}
          </div>
        </div>
        <br>

        <!-- <div class="container"> -->
          <div class="row">
            <div class="col-md-6">
              <em style="font-family:-apple-system, BlinkMacSystemFont, 'Segoe
UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-
serif; color: #444; font-weight: bold;">
                Monthly Categorized Expenditure</em>
```

```
              <div class="chart" id="catChart" style="display: inline-
block;width: 100%">
                <script>
                  var graphs = {{ plot | safe}};
                  Plotly.plot('catChart', graphs, {});
                </script>
              </div>
            </div>
            <div class="col-md-6">
              <em
                  style="font-family:-apple-system, BlinkMacSystemFont, 'Segoe
UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-
serif; color: #444; font-weight: bold;">
              Daily Expenditure of this month</em>
              <div class="chart" id="baseGraph">
                <script>
                  var graphs = {{ plot2 | safe}};
                  Plotly.plot('baseGraph', graphs, {});
                </script>
              </div>
              <div class="chart" id="baseGraph">
                <script>
                  var graphs = {{ plot3 | safe}};
                  Plotly.plot('baseGraph', graphs, {});
                </script>
              </div>
            </div>

            </div>
            <div class="row">
              <table class="table table-striped">
                <thead class="thead-dark">
                    <tr>
                        <th scope="col">#</th>
                        <th scope="col">Category</th>
                        <th scope="col">Description</th>
                        <th scope="col">Credit/Debit</th>
                        <th scope="col">Date (d/m/y)</th>
                        <th scope="col">Amount</th>
                        <th scope="col"></th>
                    </tr>
                </thead>
                <tbody>
                    {%for r in trans_data%}
                    <tr>
```

```
                        <th scope="row">{{loop.index0 + 1}}</th>
                        <!-- <th scope="row">{{r.id}}</th> -->

                        <td>{{r.cat}}</td>
                        <td>{{r.description}}</td>
                        {%if r.cashFlow == 1 %}
                        <td>Cash In</td>
                        {%else%}
                        <td>Cash Out</td>
                        {%endif%}
                        <td>{{r.date.strftime('%d-%m-%Y')}}</td>
                        <td>{{r.amount}}</td>
                        <td><button type="button" class="btn btn-outline-danger
btn-sm" data-toggle="modal" data-target="#del{{r.id}}">
                            Delete
                        </button></td>
                        <div class="modal fade" tabindex="-1" role="dialog"
id="del{{r.id}}">
                            <div class="modal-dialog" role="document">
                                <div class="modal-content">
                                    <div class="modal-header">
                                        <h5 class="modal-title">Delete Post Pop
up Modal</h5>

                                        <button type="button" class="close" data-
dismiss="modal" aria-label="Close">
                                            <span aria-
hidden="true">&times;</span>
                                        </button>
                                    </div>
                                    <div class="modal-body">
                                        <p>Are you sure you want to delete this
Transaction ?</p>

                                    </div>
                                    <div class="modal-footer">
                                        <button type="button" class="btn btn-
secondary" data-dismiss="modal">
                                            Cancel
                                        </button>

                                        <form action="{{url_for('delete',
row_id=r.id) }}" method="POST">
                                            <input class="btn btn-danger"
type="submit" value="Delete" />
                                        </form>
                                    </div>
```

```
                                </div>
                            </div>
                        </div>
                    </tr>
                    {%endfor%}
                </tbody>
            </table>
            </div>
        </div>
      <!-- </div> -->
    </div>



<!-- Modal -->
<div class="modal fade" id="transaction" data-backdrop="static" data-
keyboard="false" tabindex="-1"
  aria-labelledby="staticBackdropLabel" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="staticBackdropLabel">Add Transaction</h5>
        <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <form method="POST" id="transactionform">
          {{transForm.hidden_tag()}}
          <div class="form-group row">
            <div class="col-sm-10">
              {% for subfield in transForm.flow %}
              <tr >
                           
                           
                           

                    <td>{{ subfield }}</td>
                    <td>{{ subfield.label }}</td>
              </tr>
              {% endfor %}
            </div>
          </div>
          <div class="form-group row">
```

```
            <label for="staticEmail" class="col-sm-2 col-form-
label">Amount</label>
            <div class="col-sm-10">
              {{transForm.amount(class='form-control',maxlength=18)}}
            </div>
          </div>
          <div class="form-group row">
            <label for="staticEmail" class="col-sm-2 col-form-
label">Description</label>
            <div class="col-sm-10">
              {{transForm.description(class='form-control',maxlength=20)}}

            </div>
          </div>
          <div class="form-group row">
            <label for="inputPassword" class="col-sm-2 col-form-
label">Date</label>
            <div class="col-sm-10">
              {{transForm.date(class='datepicker form-control')}}
            </div>
          </div>
          <div class="form-group row">
            <label for="inputPassword" class="col-sm-2 col-form-
label">Category</label>
            <div class="col-sm-10">
              {{transForm.category(class='dropdown form-control')}}
            </div>
          </div>
        </form>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>
        {{transForm.submit(class="btn btn-
primary",form="transactionform",id="trans",onsubmit="alert('thank you')")}}
      </div>
    </div>
  </div>
</div>

    <script src="{{ url_for('static', filename='js1/jquery.min.js') }}"></script>
    <script src="{{ url_for('static', filename='js1/popperjs') }}"></script>
    <script src="{{ url_for('static', filename='js1/bootstrap.min.js')
}}"></script>
    <script src="{{ url_for('static', filename='js1/main.js') }}"></script>
```

```
  </body>
</html>
```

GitHub & Project Demo Link

Github Link	https://github.com/IBM-EPBL/IBM-Project-17344-1659634885
Project Demo Link	https://drive.google.com/file/d/1PtWvLut-1SeyYq3oVdk_u-bzhl--zGqr/view http://169.51.207.18:30233/