PERSONAL EXPENSE TRACKER APPLICATION

1. INTRODUCTION

1.1 Project Overview

Modern life offers several options of services and goods for consumers. As a result, people's expenses have gone up dramatically, e.g., compared to a decade ago, and the cost of living has been increasing day by day. Thus it becomes essential to keep a check on expenses in order to live a good life with a proper budget set up. This application is a full detailed expense tracker tool that will not only help users keep a check on their expenses, but also cut down the unrequired expenses, and thus will help provide a responsible lifestyle.

1.2 Purpose

This project offers some opportunities that will help the user to sustain all financial activities like digital automated diary. Most of the people cannot track their expenses and income one way they face a money crisis, in this case daily expense tracker can help the people to track income-expense day to day and making life tension free. Money is the most valuable portion of our daily life and without money we will not last one day on the earth. So this application is important to lead a happy family. It helps the user to avoid unexpected expenses and bad financial situations. This Project will save time and provide a responsible lifestyle.

2. LITERATURE SURVEY

2.1 Existing problem

In existing, we need to maintain the Excel sheets, CSV etc. files for the user daily and monthly expenses. In existing, there is no as such complete solution to keep a track of its daily expenditure easily. To do so a person to keep a log in a diary or in a computer, also all the calculations needs to be done by the user which may sometimes results in errors leading to losses.

2.2 References

1. Expense Tracker

Authors: Atiya Kazi, Praphulla S. Kherade, Raj S. Vilankar, Parag M.Sawant

Year: 2021

This project is an android app which maintain a digital record to track the daily expenses of the user. It is developed using Angular 8 for front end and SQL lite for back end. This application takes the income of a user and manage its daily expenses so that the user can save money. If you exceed daily expense allowed amount it will give you a warning, so that you don't spend much, at that specific day. If you spend less money than the daily expense allowed amount, the money left after spending is added into user's savings. The application generates report of the expenses of each end of the month. The Expense Tracker app tracks all the expenses and helps the user to manage his/her expenses so that the user is the path of financial stability. The Tracking of expenses is categorised by week, month and year, it helps to see the more expenses made. To use the Expense Tracker the user has to sign up into such as name, phone no., address, email address, username, password and confirm password of the user.

MERITS:

➢ Expense Tracker helps to maintain the record of daily expenses and monthly income of a user from anywhere.

➢ It helps the user to be financially stable.

DEMERITS:

➢ Continuous internet connection.

➢ Information is less secure.

2. DAILY EXPENSE TRACKER

AUTHORS: Shivam Mehra, Prabhat Parashar

YEARS: 2021

This project is a tool that resides on a remote server and is accessible via browsers which allows to track the daily expense of the user and help them to better manage their resources. It creates a digital record of the income and expense of the user. It input from the user a income, source of this income and the date of earning that income and creates a transaction entry under income category sums to the total amount of income and making real time changes. The various sources of income can be added and thus the distribution of your income is also illustrated by real time functioning charts that will keep updating as per your transactions. Similarly, it will also have an expense category where you can make similar transaction about the source of your expense, amount and date. On creating such transaction a different chart for distribution of expense will also be made in real time. The web application will also be voice powered and all the functionalities can be used with voice commands.

MERITS:

➢ The application will be accessible and compatible to all the devices.

➢ It has voice functionalities for better user experience.

DEMERITS:

➢ Some human errors may occur.

➢ Information is less secure.

3. A NOVEL EXPENSE TRACKER USING STATISTICAL ANALYSIS

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

YEARS: 2021

Expense Tracker is used to maintain and manage data of daily expenditure in a more precise way it can give profound knowledge of their expenses. User can choose the kind of spending they wanted to do, even the amount etc. and all these details is going to be saved by the internal database expenses so that the user is the path of financial stability. The Tracking of expenses is categorised by week, month and year, it helps to see the more expenses made. To use the Expense Tracker the user has to sign up into such as name, phone no., address, email address, username, password and confirm password of the user.

MERITS:

➢ Expense Tracker helps to maintain the record of daily expenses and monthly income of a user from anywhere.

➢ It helps the user to be financially stable.

DEMERITS:

➢ Continuous internet connection.

➢ Information is less secure.

2. DAILY EXPENSE TRACKER

AUTHORS: Shivam Mehra, Prabhat Parashar

YEARS: 2021

This project is a tool that resides on a remote server and is accessible via browsers which allows to track the daily expense of the user and help them to better manage their resources. It creates a digital record of the income and expense of the user. It input from the user a income, source of this income and the date of earning that income and creates a transaction entry under income category sums to the total amount of income and making real time changes. The various sources of income can be added and thus the distribution of your income is also illustrated by real time functioning charts that will keep updating as per your transactions. Similarly, it will also have an expense category where you can make similar transaction about the source of your expense, amount and date. On creating such transaction a different chart for distribution of expense will also be made in real time. The web application will also be voice powered and all the functionalities can be used with voice commands.

MERITS:

➢ The application will be accessible and compatible to all the devices.

➢ It has voice functionalities for better user experience.

DEMERITS:

➢ Some human errors may occur.

➢ Information is less secure.

3. A NOVEL EXPENSE TRACKER USING STATISTICAL ANALYSIS

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

YEARS: 2021

Expense Tracker is used to maintain and manage data of daily expenditure in a more precise way it can give profound knowledge of their expenses. User can choose the kind of spending they wanted to do, even the amount etc. and all these details is going to be saved by the internal database storage. In this system user can have a knowledge about their expenditure on their daily basis, weekly as well as monthly basis. This systematic way of storing your information related to your expenses would help you to keep a track of your expenditure and further you do not have to do the manual stuff. Some statistical analysis must be done to be able to give users correct information on their expenses and help them spend better. This helps the society to prevent the issues like bankruptcy and save time from manual calculations. User can provide his/her income to calculate the total expense per day and the results will be stored for each individual user. People when usually go for trips with friends, can use this tracker to maintain their expense.

MERITS:

➢ The application includes a wish list, savings, and settlement modules.

➢ It helps to organise expenses for pre-planned trips, festival, parties and so on.

DEMERITS:

➢ It is chaotic while organising the plan and do not get notification for the settlement. ➢ Information is less secure and maybe lost.

4. A REVIEW ON BUDGET ESTIMATOR ANDROID APPLICATION

AUTHORS: Namita Jagtap, Priyanka Joshi, Aditya Kamble

YEARS: 2021

This project is about mobile application Expenses system. In existing, we need to maintain the excel sheets, csv etc. files for the user daily and monthly expenses. In existing, there is no as such complete solution to keep a track of its daily expenditure easily. to do so a person as to keep a log in a diary or in a computer, also all the calculations need to be done by the user which may sometimes results in errors leading to losses. So, this project is introduced to overcome the disadvantage with the proposed features like geo-location tracking, based on the location of the user, it using Google Places, to check, the available store in the area, provides a notification for offers purpose, in term of security design, this system may implement a login authentication such as OTP message to your mobile device, this function may bring more security confidence to user.

MERITS:

➢ The application includes a wish list and savings.

➢ In additional gives offers of recently visited hotels, shops and even e-shopping.

DEMERITS:

➢ The information about the offers is not properly maintained and updated.

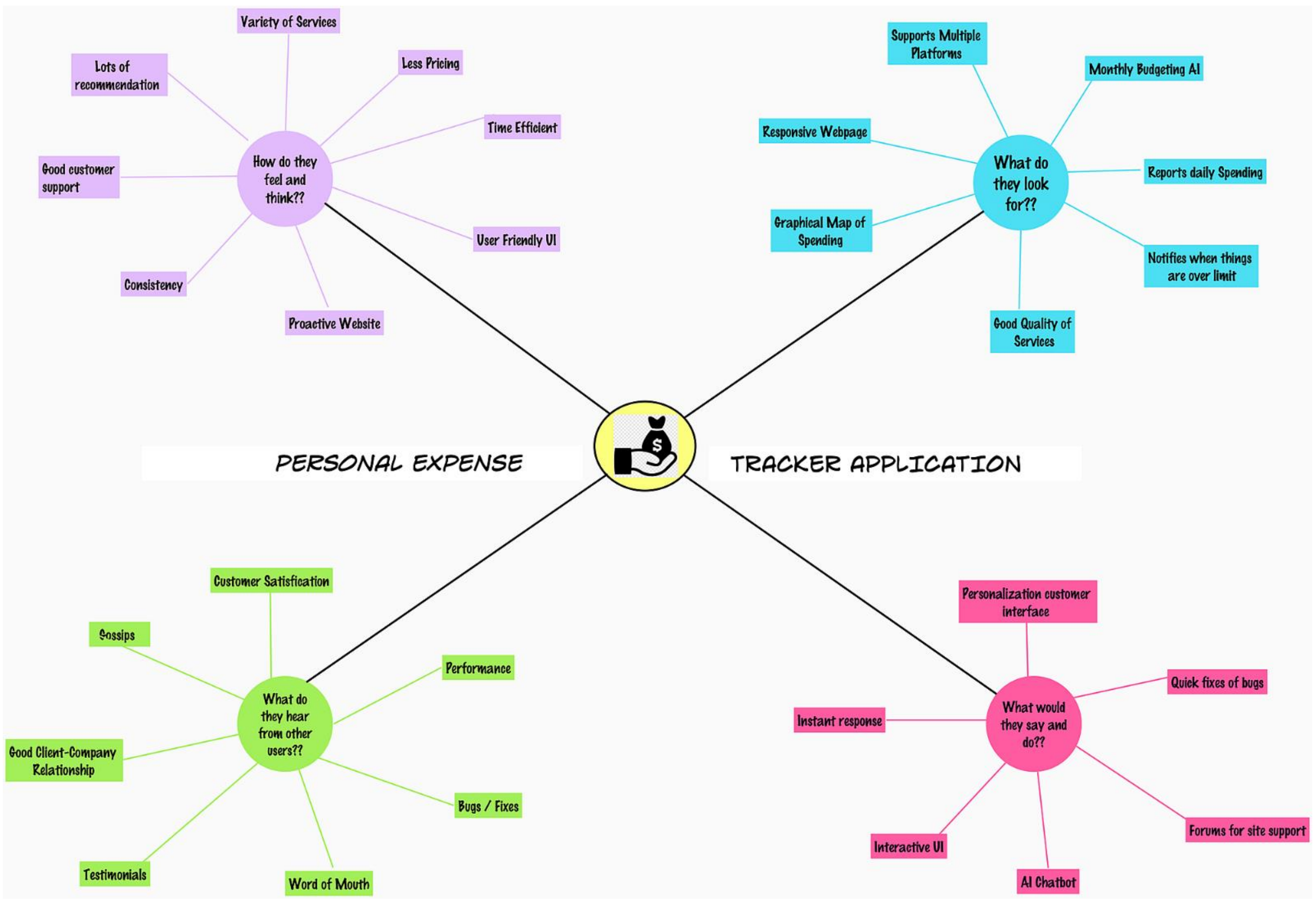
➢ Due to improper information about the offers may bring miscalculation in the budget.

2.3 Problem Statement Definition

As for the time being, there a lot of budget planner software that are available online but some of this software fall short in helping users to actually create and stick to a budget. One of the drawbacks is the on-going maintenance, a lot of budget software offer the simplicity of integrating with all user’s financial accounts and consolidating their activity into one dashboard. However though, some of this existing software mostly have complicated features that are not user friendly. Also, due to the busy and hectic lifestyle people tend to overlook their budget and end up spending an excessive amount of money since they usually didn’t plan their budget wisely. Last but not least, user cannot predict future expenses. While they can write down their expenses in a piece of paper or manage them in excel spreadsheet, their lack of knowledge in managing finances will be a problem.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Share template feedback

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

How might we allow the user a simple way to track expense and how might we define a remainder system for the user

Key rules of brainstorming
To run an smooth and productive session

Stay in topic.	Encourage wild ideas.
Defer judgment.	Listen to others.
Go for volume.	If possible, be visual.

Need some inspiration?
See a finished version of this template to kickstart your work.

Open example →

2

Brainstorm

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

10 minutes

TIP

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

Abitha

Navigate to the dashboard	Edit user profile	Visualize the expenses
Add income and expenses	Add remainder and get notify	Set budget

Harshini

Filter the expenses graphically	Edit income and expenses	Keep accurate records
Create a additional steam of income	Show case flow	Generate monthly report

Hemavarshini

Set smart budget to help you not over spend money in a choosen category	No need for complicated excel sheets	Categorize you expenses
Feedback system	Get monthly report as pdf or excel sheet	Overspending Underspending of money

Phooja shree

To remind user to enter the spendings	Categorize the expenses	Limitations for budget
Filter the expenses periodically	Add multiple stream of income	Helps you to stick on your budget and cut out impulse spending

3

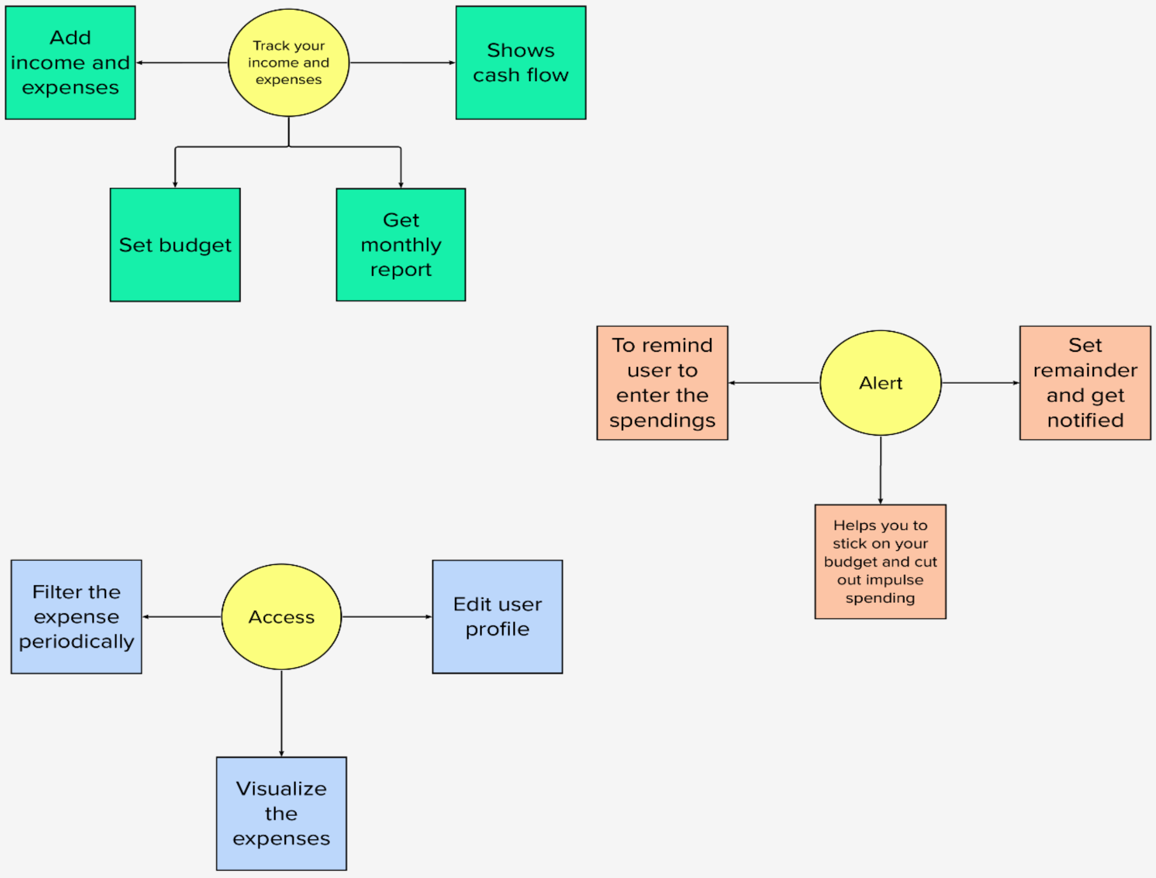
Group ideas

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

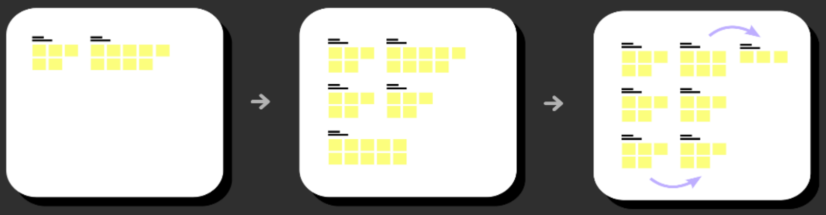
20 minutes

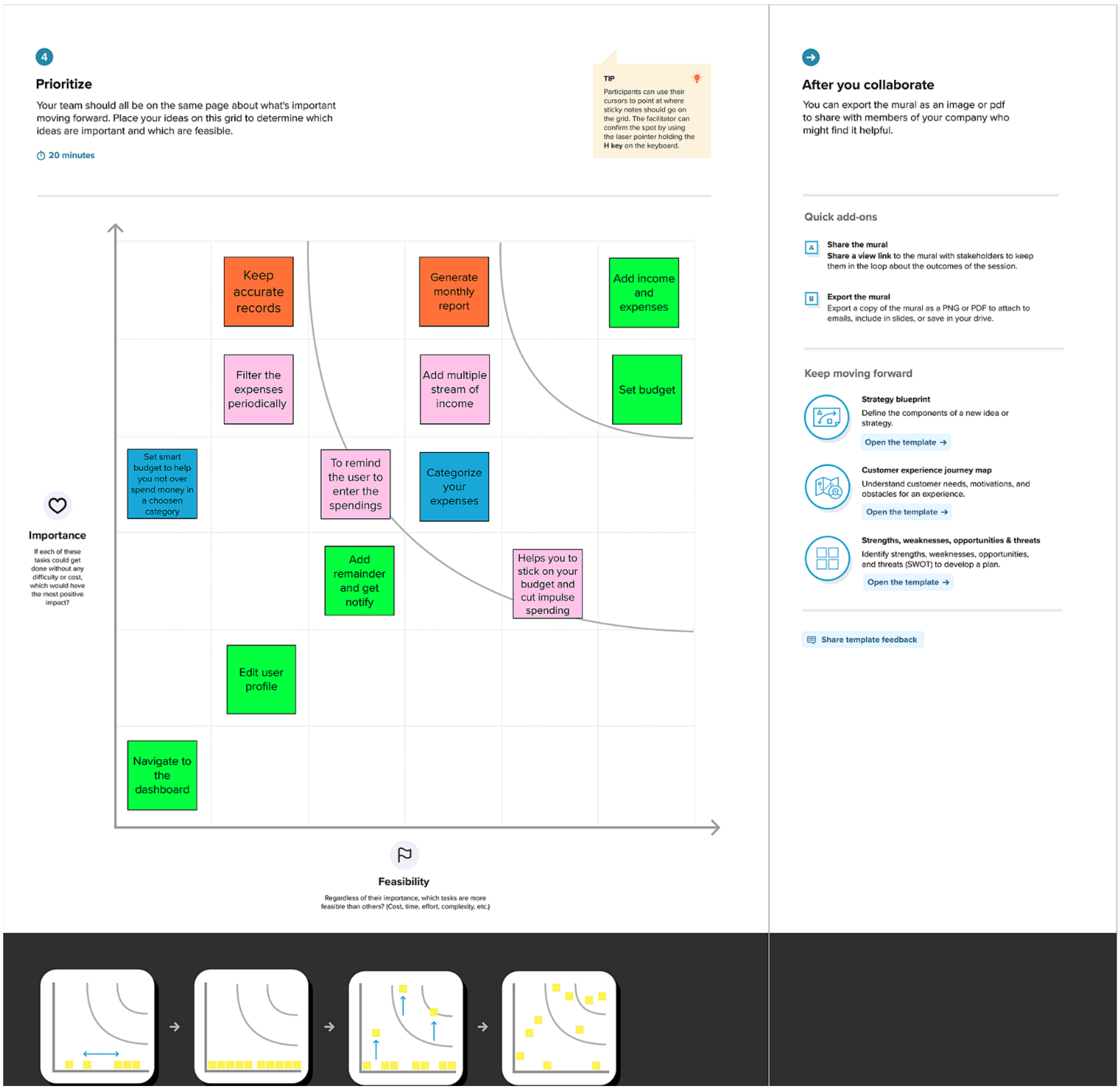
TIP

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



Track your income and expenses
Add income and expenses
Shows cash flow
Set budget
Get monthly report
Alert
To remind user to enter the spendings
Set remainder and get notified
Helps you to stick on your budget and cut out impulse spending
Access
Filter the expense periodically
Edit user profile
Visualize the expenses





Keep accurate records
Generate monthly report
Add income and expenses
Filter the expenses periodically
Add multiple stream of income
Set budget
Set smart budget to help you not over spend money in a choosen category
To remind the user to enter the spendings
Categorize your expenses
Add remainder and get notify
Helps you to stick on your budget and cut impulse spending
Edit user profile
Navigate to the dashboard

4

Prioritize

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

20 minutes

TIP

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

Importance

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

Feasibility

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

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A **Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B **Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template →

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template →

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template →

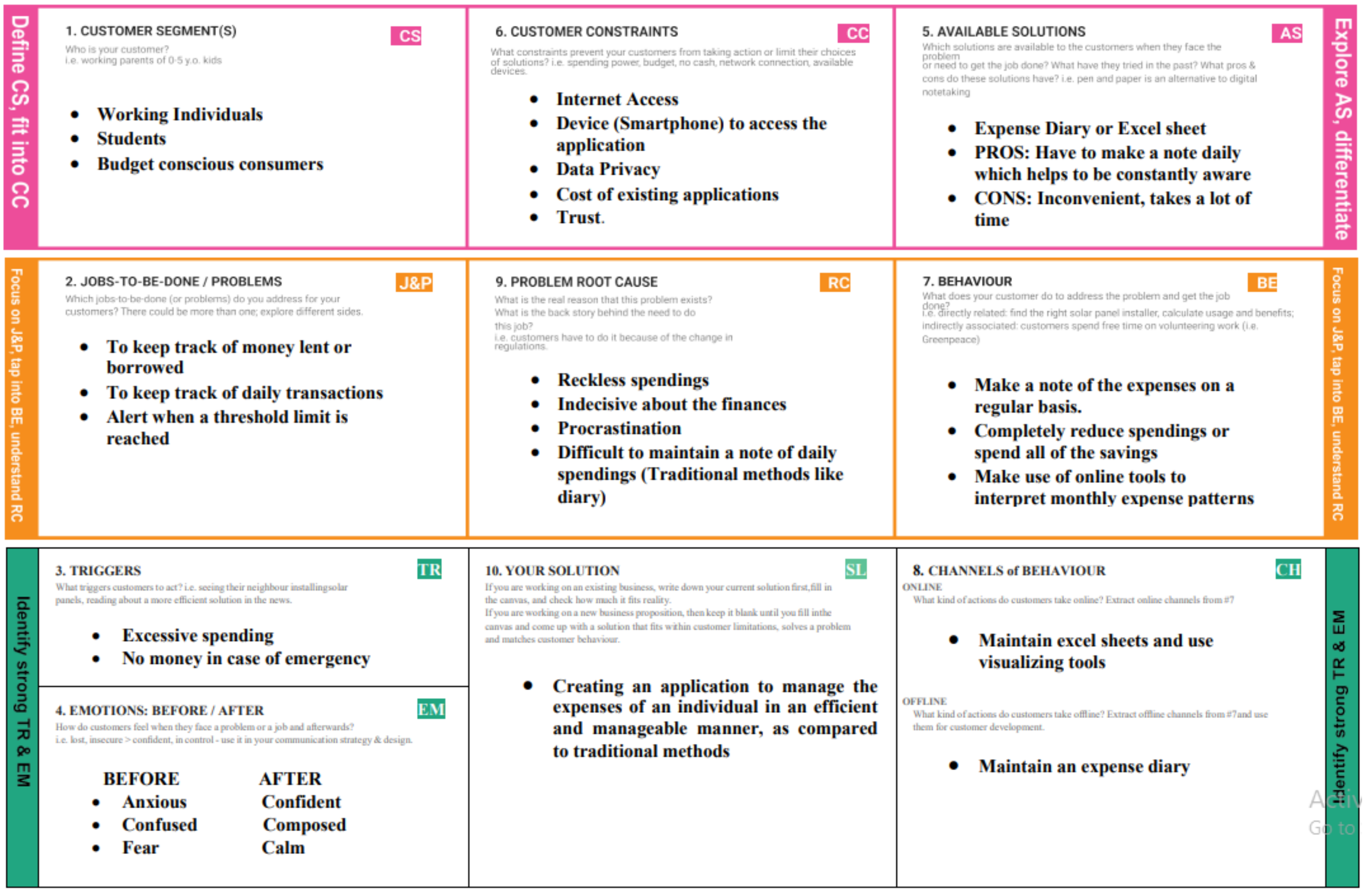
Share template feedback

3.3 <u>Proposed Solution</u>

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	At the end of the month we start to have money crisis. Lack of proper planning of our income. Person has to keep a log in a diary or in a computer. All the calculations need to be done by the user. Overload to rely on the daily entry of the expenditure.
2.	Idea / Solution description	An expense tracker app allows you to monitor and categorize your expenses. This application will be helpful for us in not just managing their expenses, but also in enabling them to improve their investments.

3.	Novelty / Uniqueness	Expense tracker apps help you collect and classify your purchases so that you can identify areas that might be trimmed. Tracking your expenditures also allows you to understand why you're in debt and how you got there. The user gets a notification when the budget limit exceeds.
4.	Social Impact / Customer Satisfaction	A good financial plan can spot positive and negative trends. This will help you better allocate funds to the areas that are making your business money, and avoid expenditures that didn't yield enough results.
5.	Business Model (Revenue Model)	In the point of business, this can make paying employees and vendors easier. Cost Effective one.
6.	Scalability of the Solution	Improves financial management. Secured and safe to use. Insights about money management.

3.4 Problem Solution fit



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

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

- **Working Individuals**
- **Students**
- **Budget conscious consumers**

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.

- **Internet Access**
- **Device (Smartphone) to access the application**
- **Data Privacy**
- **Cost of existing applications**
- **Trust.**

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

- **Expense Diary or Excel sheet**
- **PROS: Have to make a note daily which helps to be constantly aware**
- **CONS: Inconvenient, takes a lot of time**

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.

- **To keep track of money lent or borrowed**
- **To keep track of daily transactions**
- **Alert when a threshold limit is reached**

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.

- **Reckless spendings**
- **Indecisive about the finances**
- **Procrastination**
- **Difficult to maintain a note of daily spendings (Traditional methods like diary)**

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)

- **Make a note of the expenses on a regular basis.**
- **Completely reduce spendings or spend all of the savings**
- **Make use of online tools to interpret monthly expense patterns**

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

Identify strong TR & EM

3. TRIGGERS TR

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

- **Excessive spending**
- **No money in case of emergency**

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
• Anxious	Confident
• Confused	Composed
• Fear	Calm

10. YOUR SOLUTION SL

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

- **Creating an application to manage the expenses of an individual in an efficient and manageable manner, as compared to traditional methods**

8. CHANNELS of BEHAVIOUR CH

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

- **Maintain excel sheets and use visualizing tools**

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

- **Maintain an expense diary**

Identify strong TR & EM

4. REQUIREMENT ANALYSIS

4.1 Functional 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 for collecting details
FR-2	User Confirmation	Confirmation via Email and Confirmation via OTP
FR-3	Login	Enter Username and Password
FR-4	Calender	Personal expense tracker application must allow the user to add the data to their expenses
FR-5	Alert /Notification	Alert through E-mail or through SMS
FR-6	Category	This application shall allow user to add categories of their expenses

4.2 Non-Functional requirements

Following are the non-functional requirements of the proposed solution

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	Helps to keep an accurate record of user's income and expenses
NFR-2	Security	Budget tracking apps are considered very safe for from those who commit cyber crimes
NFR-3	Reliability	Each data records is stored on a well built efficient database schema. There is no risk of data loss.
NFR-4	Performance	The types of expenses are categorised along with an option . Throughput of the system is increased due to light weight database support.
NFR-5	Availability	The application must have 100% up-time
NFR-6	Scalability	Capacity of the application to handle growth, especially in handling more users.

5. PROJECT DESIGN

5.1 Data Flow Diagrams

LEVEL 1

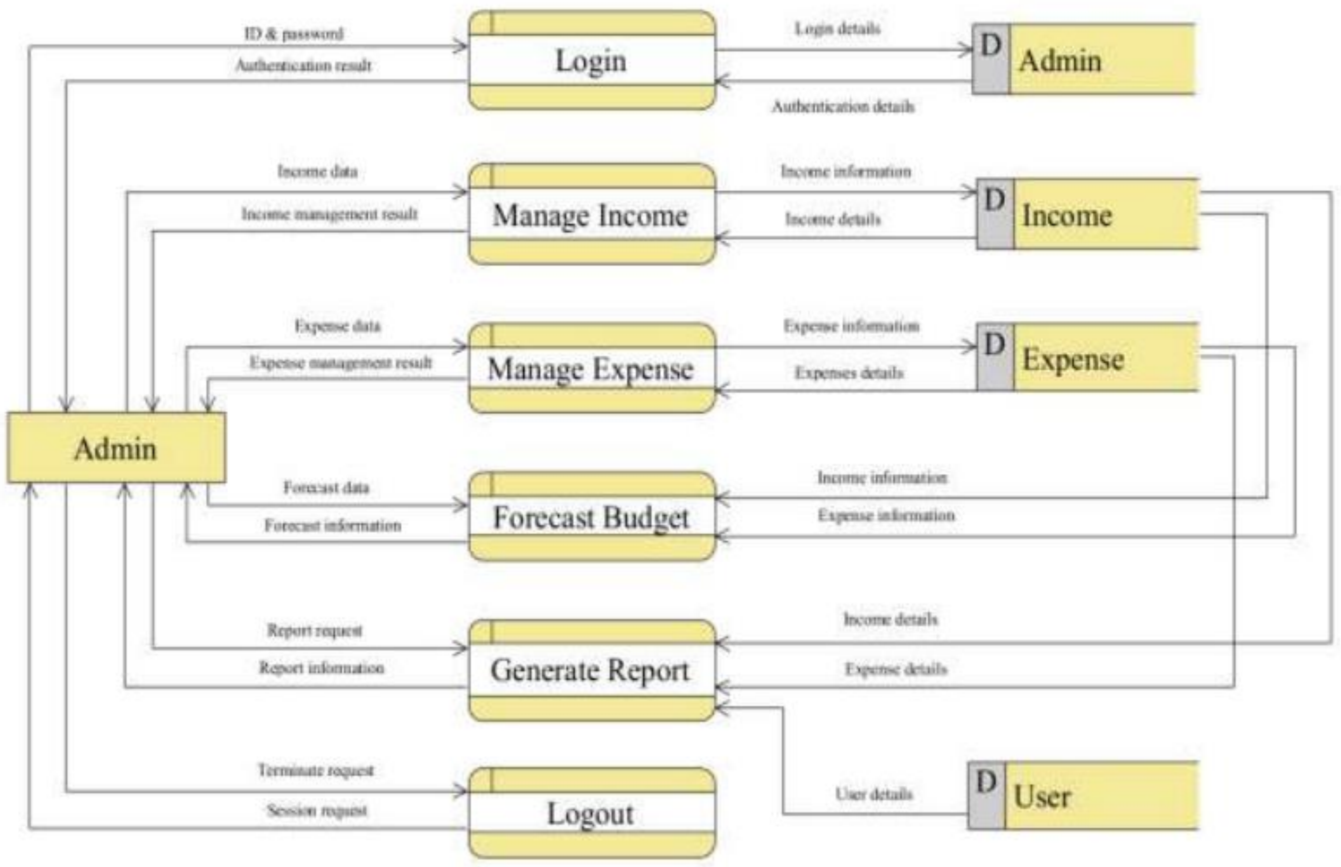


Figure 1.0 DFD Level 1 for Admin

In figure 1.0 , there are six processes involve in Admin module. Admin can be login to the system as a first step to get into the system. After login, process that involve admin is Manage Income, Manage Expenses, Forecast Budget, and Generate Report from the system. At the end on the process, admin can logout from the system.

LEVEL 1

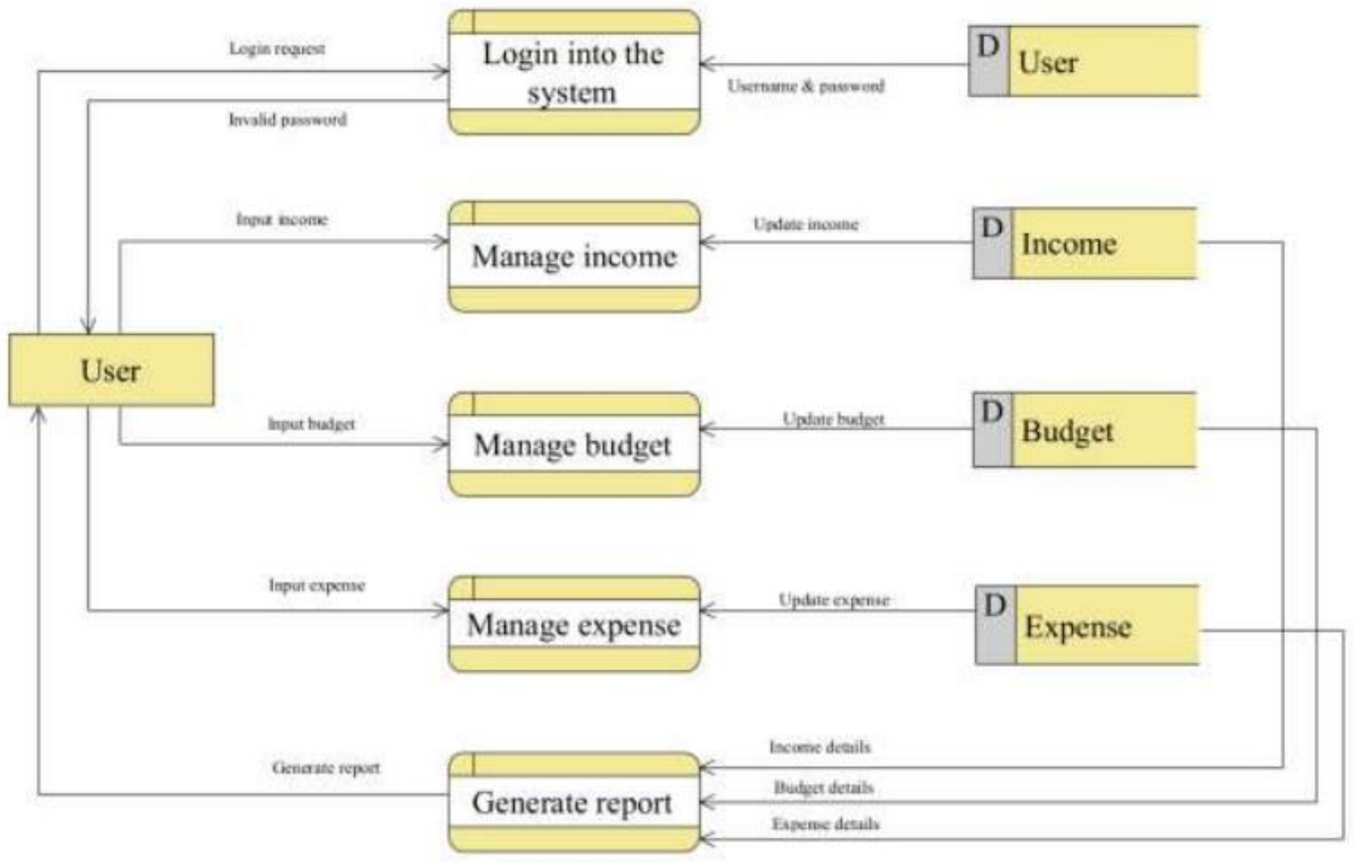


Figure 1.1 DFD Level 1 for User

Figure 1.1 above shows that there are four processes involve in User module. User can be login to the system as a first step to get into the system. Then, other processes carried in this module are Manage Income, Manage Expenses and Generate Report from the system

LEVEL 2

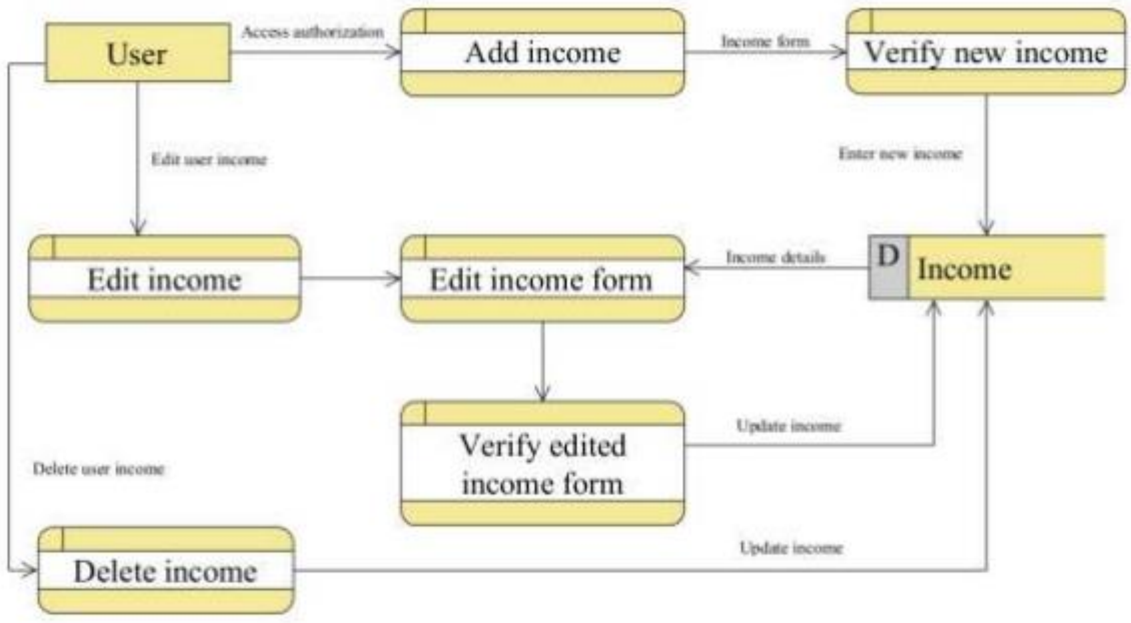


Figure 1.2 DFD Level 2 Manage Income

Figure 1.2, the process above suggest that the user can enter their income, and can edit and delete if needed

LEVEL 2

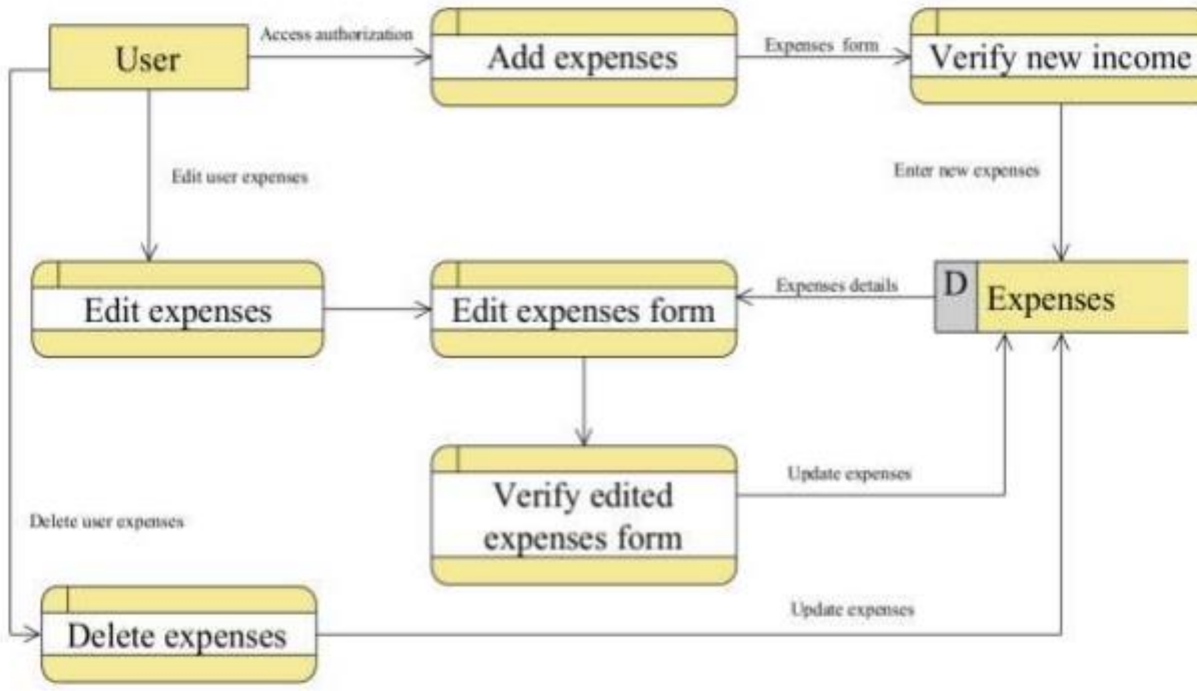


Figure 1.3 DFD Level 2 Manage Expenses

Figure 1.3, the process above shows that the user can enter their expenses, and can edit and delete if needed.

LEVEL 2

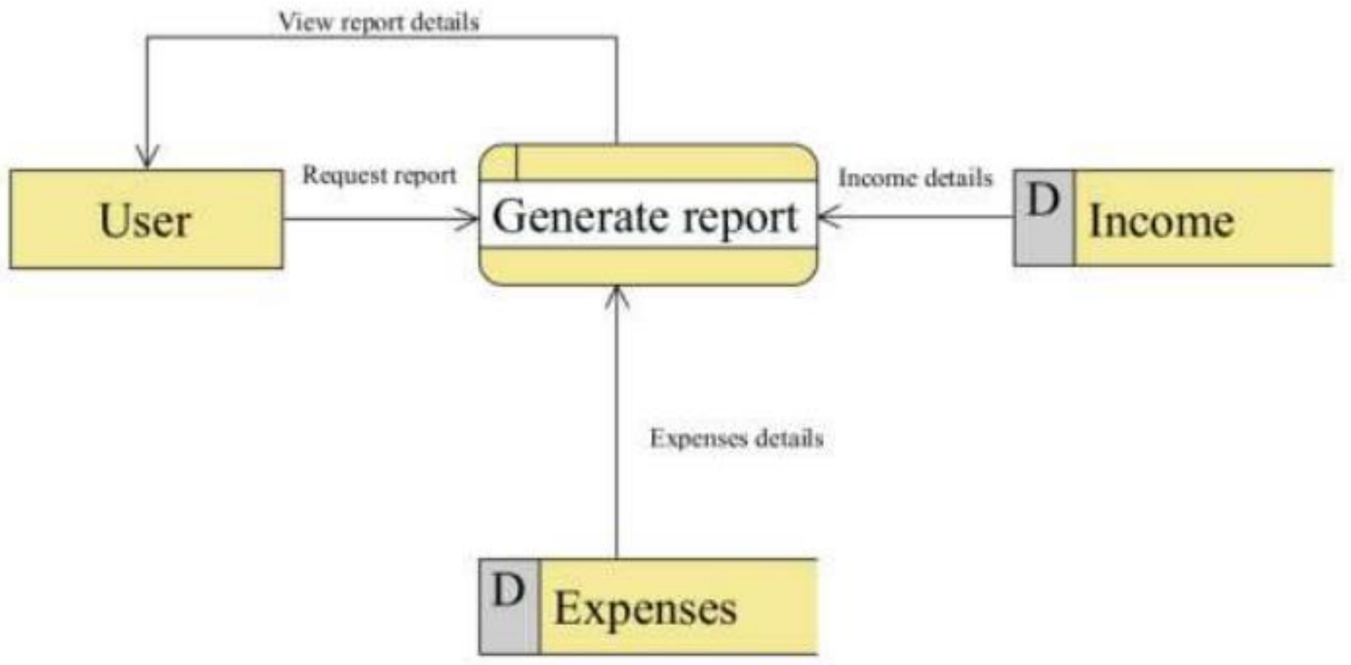


Figure 1.4 DFD Level 2 Report Generation

Figure 1.4, it shows the process to generate a report based on data has been entered in income and expenses. The data from these data store will be retrieve to display a report and forecast next month budget

5.2 Solution & Technical Architecture

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

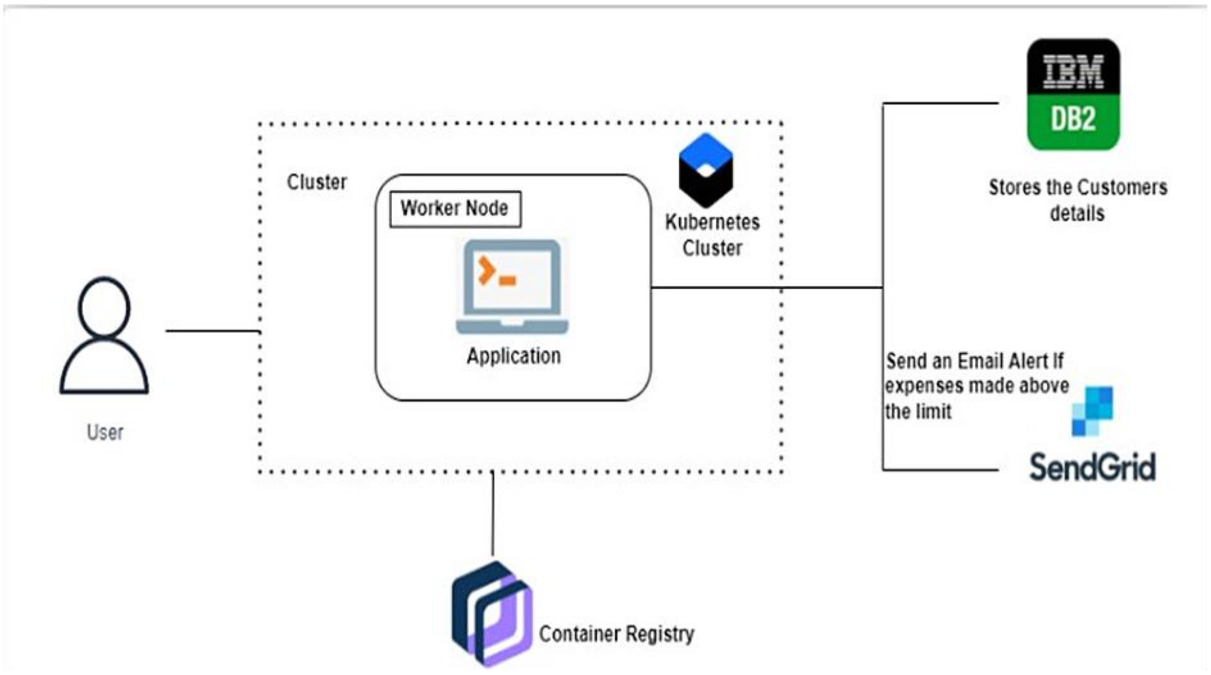


Table – 1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	The user can interact with the application with use of Chatbot.	HTML,CSS,JavaScript/ ReactJS, etc.
2.	Application Logic-1	The application contains the sign in/sign up where the user will login into the main dashboard.	Python
3.	Application Logic-2	Dashboard contains the fields like Add income, Add Expenses.	IBM Watson STT service
4.	Application Logic-3	The user will get the expense report in the graph form and also get alerts if the expense limit exceed.	IBM Watson Assistant, Send Grid
5.	Database	The Income and Expense data are stored in the MySQL database.	MySQL, NoSQL, etc.
6.	Cloud Database	With use of Database Service on Cloud, the User data are stored in a well secured Manner.	IBM DB2, IBM Cloudant etc.
7.	File Storage	IBM Block Storage used to store the financial data of the user.	IBM Block Storage or Other Storage Service or Local Filesystem

Table – 2 : Application Characteristics:

S.No	Component	Description	Technology
1.	Open-Source Frameworks	Flask Framework in Python is used to implement this Application	Python-Flask.
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, Kubernetes Cluster
3.	Scalable Architecture	Expense Tracker is a life time access supplication. Its demand will increase when the user's incomes are high.	Container Registry, Kubernetes Cluster.
4.	Availability	This application will be available to the user at any part of time.	Container Registry, Kubernetes Cluster
5.	Performance	The performance will be high because there will be no network traffics in the application	Kubernetes Cluster.

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story/Task	Acceptance Criteria	Priority	Release
Customer (Mobile user & web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password and confirming my password	I can access my account/dashboard	High	
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	
		USN-3	As a user, I can register for the	I can register & access the	Low	

			application through Facebook	dashboard with Facebook login		
	Login	USN-4	As a user, I can log into the application by entering email & password	I can access the application	High	
	Dashboard	USN-5	As a user, I can enter my income and expenditure details	I can view my daily expenses	High	
Customer Care Executive		USN-6	As a customer care executive, I can solve the log in issues and other issues of the application	I can provide support or solution at any time 24/7	Medium	
Administrator	Application	USN-7	As an administrator, I can upgrade or update the application	I can fix the bug which arises for the customers and users of the application	Medium	

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information by referring the, technical papers, research publications etc.	24 SEPTEMBER 2022
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	24 SEPTEMBER 2022
Ideation	List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	24 SEPTEMBER 2022

Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	19 OCTOBER 2022
Problem Solution Fit	Prepare problem - solution fit document.	19 OCTOBER 2022
Solution Architecture	Prepare solution architecture document.	17 OCTOBER 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.	3 OCTOBER 2022
Data Flow Diagrams	Draw the data flow diagrams and submit for review.	3 OCTOBER 2022
Technology Architecture	Prepare the technology architecture diagram	20 OCTOBER 2022
Prepare Milestone & Activity List	Prepare the milestones & activity list of the project.	26 OCTOBER 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	19 NOVEMBER 2022

6.2 <u>Sprint Delivery Schedule</u>

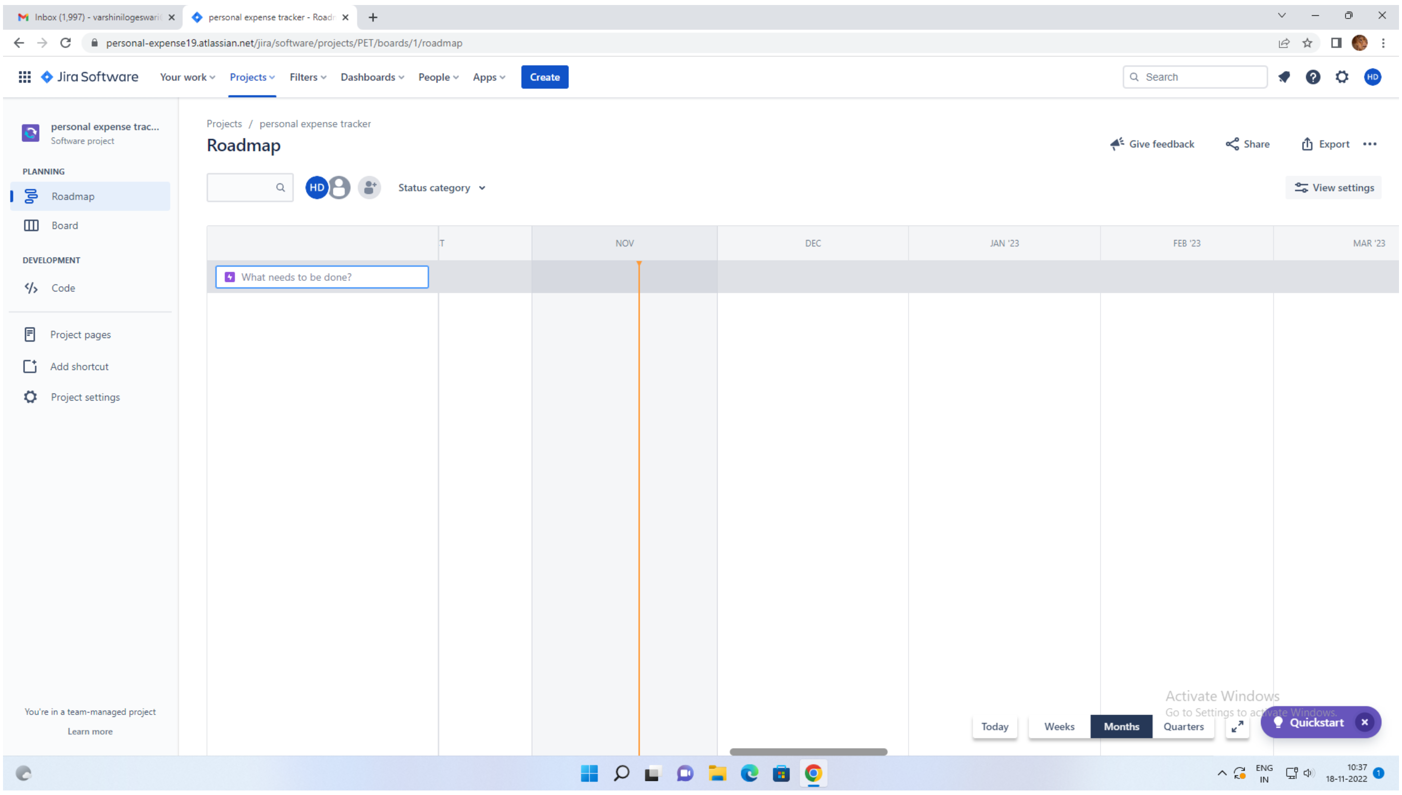
Sprint	Total Story Points	Durati on	Sprint Start Date	Sprint End Date (Planned)	Sprint End Date (Planned)	Sprint Release Date (Actual)
Sprint -1	20	6 Days	23 Oct 2022	28 Oct 2022	20	29 Oct 2022
Sprint -2	20	6 Days	30 Oct 2022	04 Nov 2022	20	05 Nov 2022
Sprint -3	20	6 Days	06 Nov 2022	11 Nov 2022	20	12 Nov 2022
Sprint -4	20	6 Days	13 Nov 2022	18 Nov 2022	20	19 Nov 2022

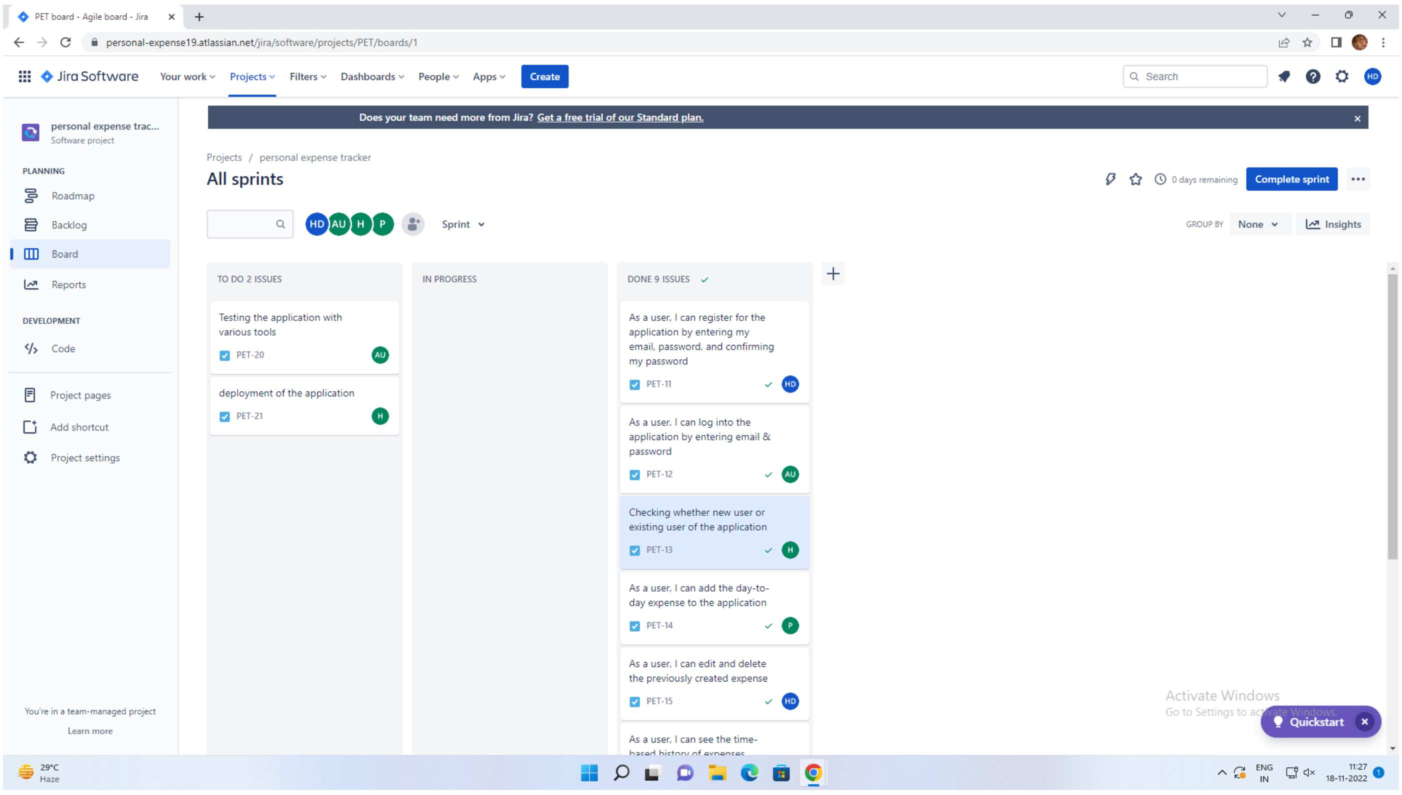
Velocity

We have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). Calculating the team’s average velocity (AV) per iteration unit (story points per day)

$$AV = \text{sprint duration} / \text{velocity} = 20/6 = 3.33$$

6.3 <u>Reports from JIRA</u>





TO DO 2 ISSUES

Testing the application with various tools
PET-20 AU

deployment of the application
PET-21 H

IN PROGRESS

DONE 9 ISSUES

As a user, I can register for the application by entering my email, password, and confirming my password
PET-11 HD

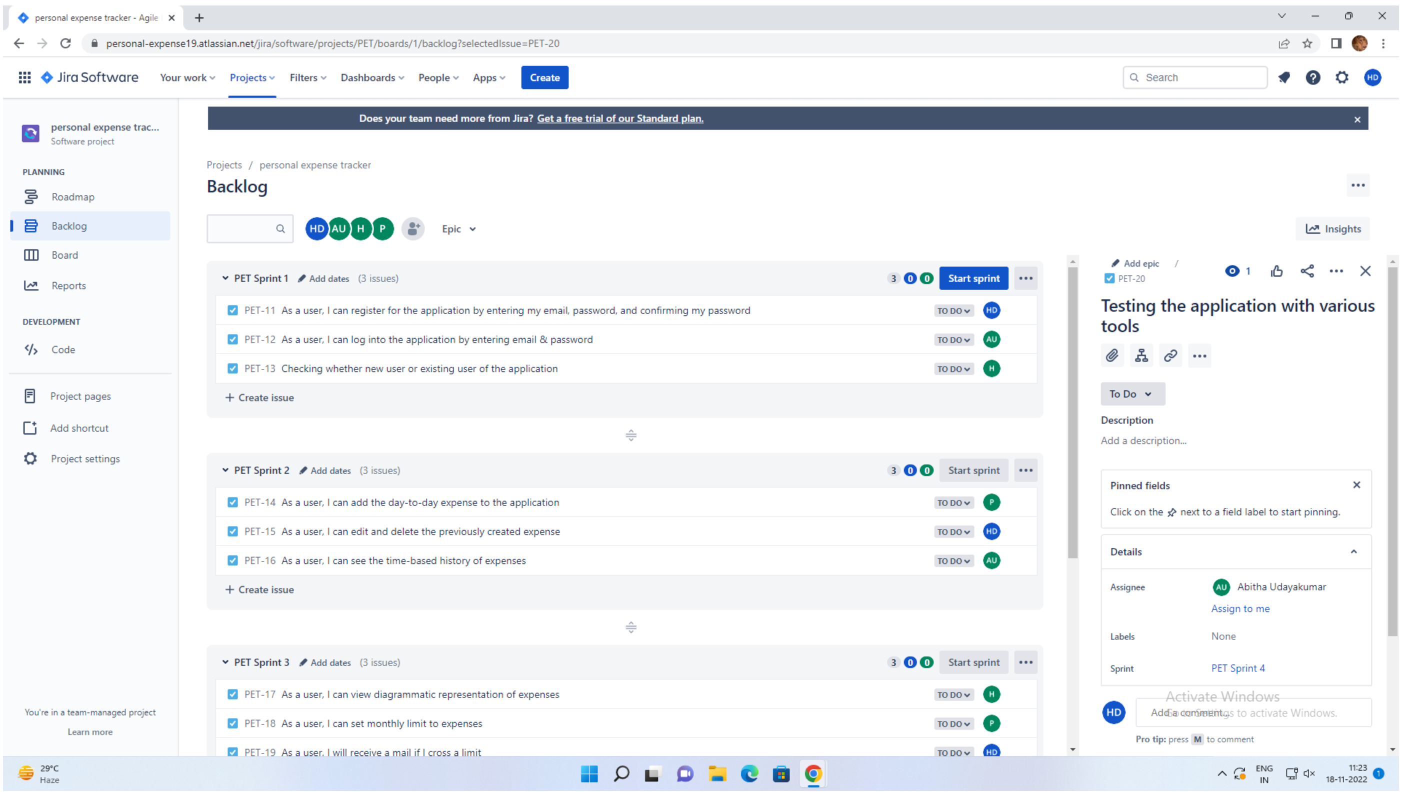
As a user, I can log into the application by entering email & password
PET-12 AU

Checking whether new user or existing user of the application
PET-13 H

As a user, I can add the day-to-day expense to the application
PET-14 P

As a user, I can edit and delete the previously created expense
PET-15 HD

As a user, I can see the time-based history of expenses


personal expense tracker - Agile

personal-expense19.atlassian.net/jira/software/projects/PET/boards/1/backlog?selectedIssue=PET-20

Jira Software Your work Projects Filters Dashboards People Apps Create Search

Does your team need more from Jira? Get a free trial of our Standard plan.

personal expense trac... Software project

PLANNING
Roadmap
Backlog
Board
Reports

DEVELOPMENT
Code

Project pages
Add shortcut
Project settings

You're in a team-managed project
Learn more

Projects / personal expense tracker

Backlog

HD AU H P Epic Insights

PET Sprint 1 Add dates (3 issues) 3 0 0 Start sprint
PET-11 As a user, I can register for the application by entering my email, password, and confirming my password TO DO HD
PET-12 As a user, I can log into the application by entering email & password TO DO AU
PET-13 Checking whether new user or existing user of the application TO DO H
+ Create issue

PET Sprint 2 Add dates (3 issues) 3 0 0 Start sprint
PET-14 As a user, I can add the day-to-day expense to the application TO DO P
PET-15 As a user, I can edit and delete the previously created expense TO DO HD
PET-16 As a user, I can see the time-based history of expenses TO DO AU
+ Create issue

PET Sprint 3 Add dates (3 issues) 3 0 0 Start sprint
PET-17 As a user, I can view diagrammatic representation of expenses TO DO H
PET-18 As a user, I can set monthly limit to expenses TO DO P
PET-19 As a user, I will receive a mail if I cross a limit TO DO HD

Add epic / PET-20

Testing the application with various tools

To Do

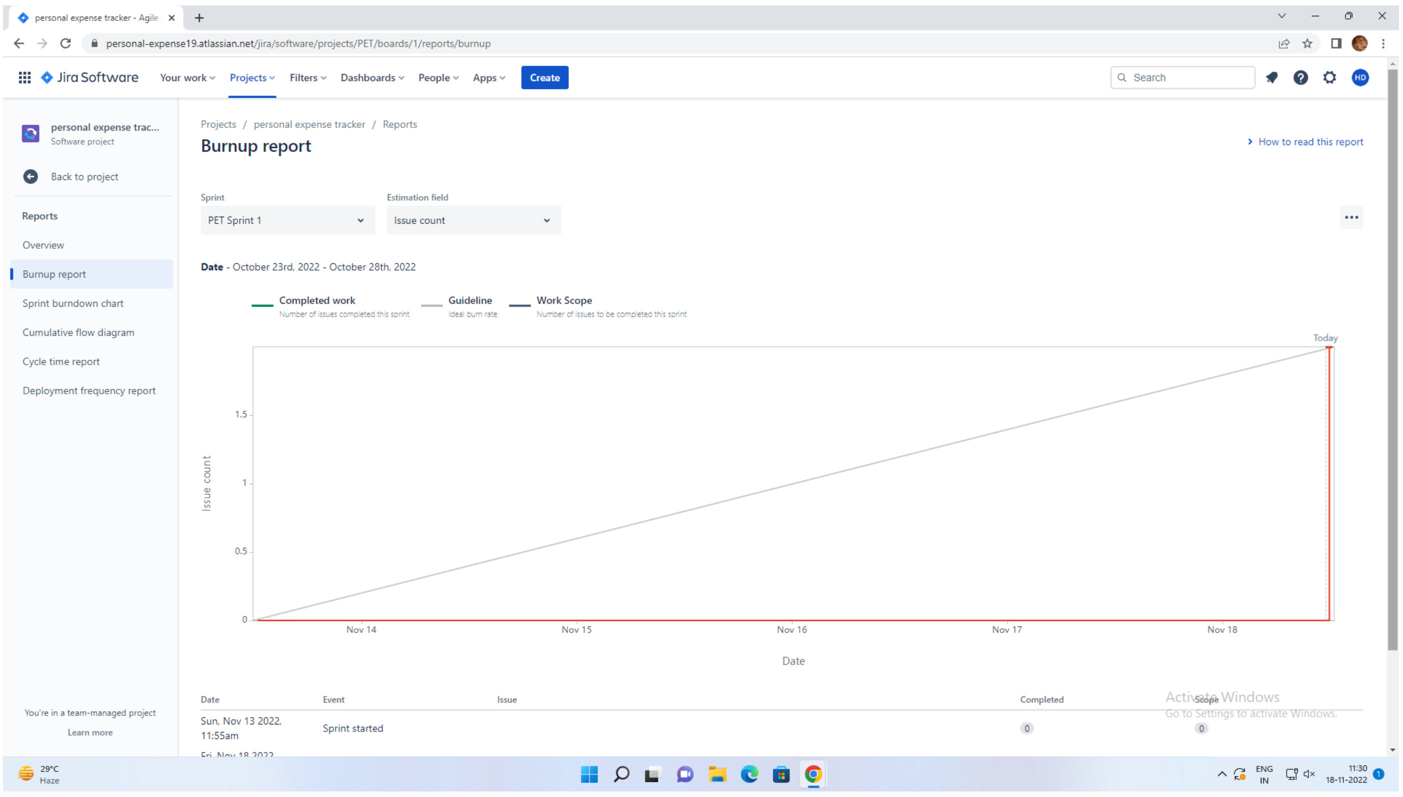
Description
Add a description...

Pinned fields
Click on the next to a field label to start pinning.

Details
Assignee AU Abitha Udayakumar
Assign to me
Labels None
Sprint PET Sprint 4

Add a comment...
Pro tip: press M to comment

29°C Haze ENG IN 11:23 18-11-2022


personal expense tracker - Agile

personal-expense19.atlassian.net/jira/software/projects/PET/boards/1/reports/burnup

Jira Software Your work Projects Filters Dashboards People Apps Create Search

personal expense trac... Software project

Back to project

Reports
Overview
Burnup report
Sprint burndown chart
Cumulative flow diagram
Cycle time report
Deployment frequency report

You're in a team-managed project
Learn more

Projects / personal expense tracker / Reports

Burnup report

How to read this report

Sprint: PET Sprint 1
Estimation field: Issue count

Date - October 23rd, 2022 - October 28th, 2022

Date	Event	Issue	Completed	Scope
Sun, Nov 13 2022, 11:55am	Sprint started		0	0

29°C Haze ENG IN 11:30 18-11-2022

7. CODING & SOLUTIONING(Explain the features added in the project along with code)

7.1 Feature 1

We have added the data visualize on methods for expenditure. The pie chart have been used to represent the monthly expenses. The pie chart is a pictorial representation of data that makes it possible to visualize the relationships between the parts and the whole of a variable. For example, it is possible to understand the industry count or percentage of a variable level from the division by areas or sectors. The recommended use for pie charts is two-dimensional, as three-dimensional use can be confusing.

The dimensions form sectors of the measurement values; they can have one or two sizes and up to two measures. The first dimension is used to define the angle of each sector that makes up the chart and the second dimension optionally determines the radius of each sector. Additionally, these plots are useful for comparing data over a fixed period since they do not show changes over time.

Therefore, their use should be considered if:

● You are looking to categorize and compare a set of data.

1. You only have positive values.
2. You have less than seven categories since a larger number can make it difficult to perceive each segment.

7.2 Feature 2

Email notifications will be sent to the users once they cross the expenditure limit through send grid mail system. Most notifications are transactional, meaning a recipient's action or account activity triggers them. But some notifications are marketing related, encouraging the recipient to take a specific action. Ecommerce product notifications inform recipients about new products or discounts. Plus, unlike general marketing emails, these are highly personalized and focus on a single product. For example, if a customer views an item on your website and that item goes on sale, you can send the customer a notification to let them know this is the best time to buy. Users can also opt into receiving notifications when an out-of-stock item is back in stock. Notification emails tend to perform well because the content is highly relevant to the recipient. But the only way for the recipient to know this is if you state the content clearly in the subject line. For example, the subject line "New Sign-in to Your Account" gets straight to the point, letting the user know why you sent this notification.

8. TESTING

8.1 Test Cases

Test Case Id	Test Description	Input Test Data	Expected Result	Actual Result	Remarks
TC-1	Install PET app in android phone	Transfer PET app	Open Application with it home page	Application executed with home page	Pass
TC-2	Enter valid data in username and password field	hema	Show home page for user hema	Displayed home page for user hema	Pass
TC-3	Enter a valid data in username and leave password field empty	hema	Show error	Didn't show any error	Fail

8.2 <u>User Acceptance Testing</u>
Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	1	0	0	0	1
Duplicate	1	0	0	0	1
External	3	1	0	0	4
Fixed	4	1	0	0	5
Not Reproduced	0	0	0	0	1
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	9	2	0	0	11

Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	0	0	0	0
Client Application	5	0	0	5
Security	0	0	0	0
Outsource Shipping	0	0	0	0
Exception Reporting	5	0	0	5
Final Report Output	0	0	0	0
Version Control	0	0	0	0

9. <u>RESULTS</u>

9.1 <u>Performance Metrics</u>

1. Tracking income and expenses: Monitoring the income and tracking all expenditures (through bank accounts, mobile wallets, and credit & debit cards).

2. Transaction Receipts: Capture and organize your payment receipts to keep track of your expenditure.

3. Organizing Taxes: Import your documents to the expense tracking app, and it will streamline your income and expenses under the appropriate tax categories.

4. Payments & Invoices: Accept and pay from credit cards, debit cards, net banking, mobile wallets, and bank transfers, and track the status of your invoices and bills in the mobile app itself. Also, the tracking app sends reminders for payments and automatically matches the payments with invoices.

5. Reports: The expense tracking app generates and sends reports to give a detailed insight about profits, losses, budgets, income, balance sheets, etc.
6. E-commerce integration: Integrate your expense tracking app with your eCommerce store and track your sales through payments received via multiple payment methods.
7. Access control: Increase your team productivity by providing access control to particular users through custom permissions.
8. Track Projects: Determine project profitability by tracking labor costs, payroll, expenses, etc., of your ongoing project.
9. Inventory tracking: An expense tracking app can do it all. Right from tracking products or the cost of goods, sending alert notifications when the product is running out of stock or the product is not selling, to purchase orders.
10. In-depth insights and analytics: Provides in-built tools to generate reports with easy-to-understand visuals and graphics to gain insights about the performance of your business.
11. Recurrent Expenses: Rely on your budgeting app to track, streamline, and automate all the recurrent expenses and remind you on a timely basis.

10. ADVANTAGES & DISADVANTAGES

1. Scale-up at the pace your business is growing.
2. Deliver an outstanding customer experience through additional control over the app.
3. Control the security of your business and customer data
4. Open direct marketing channels with no extra costs with methods such as push notifications.
5. Boost the productivity of all the processes within the organization.
6. Increase efficiency and customer satisfaction with an app aligned to their needs.
7. Seamlessly integrate with existing infrastructure.
8. Ability to provide valuable insights.
9. Optimize sales processes to generate more revenue through enhanced data collection.

11. CONCLUSION

After making this application we assure that this application will help its users to manage the cost of their daily expenditure. It will guide them and aware them about there daily expenses. It will prove to be helpful for the people who are frustrated with their daily budget management, irritated because of amount of expenses and wishes to manage money and to preserve the record of their daily cost which may be useful to change their way of spending money. In short, this application will help its users to overcome the wastage of money.

12. FUTURE SCOPE

The project assists well to record the income and expenses in general. However, this project has some limitations:
1. The application is unable to maintain the backup of data once it is uninstalled.
2. This application does not provide higher decision capability.

To further enhance the capability of this application, we recommend the following features to be incorporated into the system:

3. Multiple language interface.
4. Provide backup and recovery of data.
5. Provide better user interface for user.
6. Mobile apps advantage.

13. <u>APPENDIX</u>

<u>Source Code</u>

```
from flask import Flask, render_template, request, redirect, session ,url_for
import ibm_db
import re
import sendemail
app = Flask(__name__)
hostname = '8e359033-a1c9-4643-82ef-
8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;'
uid = 'jsl93327'
pwd = 'HTm8BvLQSFBDJLYE'
driver = "{IBM DB2 ODBC DRIVER}"
db_name = 'Bludb'
port = '30120'
protocol = 'TCPIP'
cert = "certi.crt"
dsn = (
    "DATABASE ={0};"
    "HOSTNAME ={1};"
    "PORT ={2};"
    "UID ={3};"
    "SECURITY=SSL;"
    "PROTOCOL={4};"
    "PWD ={6};"
).format(db_name, hostname, port, uid, protocol, cert, pwd)
connection = ibm_db.connect(dsn, "", "")
app.secret_key = 'a'

#HOME--PAGE
@app.route("/home")
def home():
    return render_template("homepage.html")
@app.route("/")
def add():
    return render_template("home.html")

#SIGN--UP--OR--REGISTER
@app.route("/signup")
def signup():
    return render_template("signup.html")
@app.route('/register', methods =['GET', 'POST'])
def register():
    global user_email
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        query = "SELECT * FROM register WHERE email=?;"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            query = "INSERT INTO register values(?,?,?);"
            stmt = ibm_db.prepare(connection, query)
            ibm_db.bind_param(stmt, 1, username)
            ibm_db.bind_param(stmt, 2, email)
            ibm_db.bind_param(stmt, 3, password)
            ibm_db.execute(stmt)
            session['loggedin'] = True
            session['id'] = email
            user_email = email
            session['email'] = email
            session['username'] = username
            msg = 'You have successfully registered ! Proceed Login Process'
            return render_template('login.html', msg = msg)
    else:
        msg = 'PLEASE FILL OUT OF THE FORM'
        return render_template('register.html', msg=msg)
```

#LOGIN--PAGE

```
@app.route("/signin")
def signin():
    return render_template('login.html')
@app.route('/login',methods =['GET', 'POST'])
def login():
    global user_email
    msg = ''
    if request.method == 'POST' :
        email = request.form['email']
        password = request.form['password']
        sql = "SELECT * FROM register WHERE email =? AND password=?;"
        stmt = ibm_db.prepare(connection, sql)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print (account)
        if account:
            session['loggedin'] = True
            session['id'] = account['EMAIL']
            user_email=  account['EMAIL']
            session['email']=account['EMAIL']
            session['username'] = account['USERNAME']
            return redirect('/home')
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)
```

#CHANGE FORGOT PASSWORD

```
@app.route("/forgot")
def forgot():
    return render_template('forgot.html')
@app.route("/forgotpw", methods =['GET', 'POST'])
def forgotpw():
    msg = ''
    if request.method == 'POST' :
        email = request.form['email']
        password = request.form['password']
        query = "SELECT * FROM register WHERE email=?;"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            query = "UPDATE register SET password = ? WHERE email = ?;"
            stmt = ibm_db.prepare(connection, query)
            ibm_db.bind_param(stmt, 1, password)
            ibm_db.bind_param(stmt, 2, email)
            ibm_db.execute(stmt)
            msg = 'Successfully changed your password ! Proceed Login Process'
            return render_template('login.html', msg = msg)
    else:
        msg = 'PLEASE FILL OUT THE CORRECT DETAILS'
        return render_template('forgot.html', msg=msg)
```

#ADDING----DATA

```
@app.route("/add")
def adding():
    return render_template('add.html')
@app.route('/addexpenses',methods=['GET', 'POST'])
def addexpense():
    global user_email
    que = "SELECT * FROM expenses where id = ? ORDER BY 'dates' DESC"
    stm = ibm_db.prepare(connection, que)
    ibm_db.bind_param(stm, 1, session['email'])
    ibm_db.execute(stm)
    dictionary=ibm_db.fetch_assoc(stm)
    expense=[]
    while dictionary !=
False:        exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary[
"AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"])
        expense.append(exp)
        dictionary = ibm_db.fetch_assoc(stm)
    i=len(expense)+1
    id=str(i)
    dates = request.form['date']
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category = request.form['category']
    query = "INSERT INTO expenses VALUES (?,?,?,?,?,?,?);"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, session['email'])
    ibm_db.bind_param(stmt, 2, dates)
    ibm_db.bind_param(stmt, 3, expensename)
    ibm_db.bind_param(stmt, 4, amount)
    ibm_db.bind_param(stmt, 5, paymode)
    ibm_db.bind_param(stmt, 6, category)
    ibm_db.bind_param(stmt, 7, id)
    ibm_db.execute(stmt)
    print(id + " "+dates + " " + expensename + " " + amount + " " + paymode + " " + category)
    return redirect("/display")
```

#DISPLAY---graph

```
@app.route("/display")
def display():
    query = "SELECT * FROM expenses where id = ? ;"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, session['email'])
    ibm_db.execute(stmt)
    dictionary=ibm_db.fetch_assoc(stmt)
    rexpense=[]
    while dictionary !=
False:        exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary[
"AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"])
        rexpense.append(exp)
        dictionary = ibm_db.fetch_assoc(stmt)
    que = "SELECT MONTH(dates) as DATES, SUM(amount) as AMOUNT FROM expenses
WHERE id=? AND YEAR(dates)= YEAR(now()) GROUP BY MONTH(dates);"
    stm = ibm_db.prepare(connection, que)
    ibm_db.bind_param(stm, 1,session['email'])
    ibm_db.execute(stm)
    dictionary=ibm_db.fetch_assoc(stm)
    texpense=[]
    while dictionary != False:
        exp=(dictionary["DATES"],dictionary["AMOUNT"])
        texpense.append(exp)
        dictionary = ibm_db.fetch_assoc(stm)
    print(texpense)
    quer = "SELECT * FROM expenses WHERE id = ? AND YEAR(dates)= YEAR(now());"
    st = ibm_db.prepare(connection, quer)
    ibm_db.bind_param(st, 1,session['email'])
    ibm_db.execute(st)
    dictionary=ibm_db.fetch_assoc(st)
    expense=[]
    while dictionary !=
False:        exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary[
"AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"])
        expense.append(exp)
        dictionary = ibm_db.fetch_assoc(st)
    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0
    for x in expense:
        total += x[3]
        if x[5] == "food":
            t_food += x[3]
        elif x[5] == "entertainment":
            t_entertainment  += x[3]
        elif x[5] == "business":
            t_business  += x[3]
        elif x[5] == "rent":
            t_rent  += x[3]
        elif x[5] == "EMI":
            t_EMI  += x[3]
        elif x[5] == "other":
            t_other  += x[3]
    print(total)
    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)
    qur = "SELECT * FROM expenses WHERE id = ? AND MONTH(dates)= MONTH(now());"
    stt = ibm_db.prepare(connection, qur)
    ibm_db.bind_param(stt, 1, session['email'])
    ibm_db.execute(stt)
    dictionary=ibm_db.fetch_assoc(stt)
    lexpense=[]
    while dictionary !=
False:        exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary[
"AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"])
        lexpense.append(exp)
        dictionary = ibm_db.fetch_assoc(stt)
    ttotal=0
    to_food=0
    to_entertainment=0
    to_business=0
    to_rent=0
    to_EMI=0
    to_other=0
    for x in lexpense:
        ttotal += x[3]
        if x[5] == "food":
            to_food += x[3]
        elif x[5] == "entertainment":
            to_entertainment  += x[3]
        elif x[5] == "business":
            to_business  += x[3]
        elif x[5] == "rent":
            to_rent  += x[3]
        elif x[5] == "EMI":
            to_EMI  += x[3]
        elif x[5] == "other":
            to_other  += x[3]
    print(ttotal)
    qy = "SELECT max(IDX) as IDX FROM limits where id=?;"
    smt = ibm_db.prepare(connection, qy)
    ibm_db.bind_param(smt, 1, session['email'])
    ibm_db.execute(smt)
    dictionary = ibm_db.fetch_assoc(smt)
    uexpense=[]
    while dictionary != False:
        exp=(dictionary["IDX"])
        uexpense.append(exp)
        dictionary = ibm_db.fetch_assoc(smt)
    k=uexpense[0]
    qu = "SELECT NUMBER FROM limits where id=? and idx=?"
    sm = ibm_db.prepare(connection, qu)
    ibm_db.bind_param(sm, 1, session['email'])
    ibm_db.bind_param(sm, 2, k)
    ibm_db.execute(sm)
    dictionary = ibm_db.fetch_assoc(sm)
    fexpense=[]
    while dictionary != False:
        exp=(dictionary["NUMBER"])
        fexpense.append(exp)
        dictionary = ibm_db.fetch_assoc(stmt)
    if len(fexpense) <= 0:
        print("Enter the limit First")
    else:
        if ttotal > fexpense[0]:
            m=sendemail.sendgridmail(session["email"])
            print(m)
        else: print("Error")
    return render_template("display.html",rexpense=rexpense, texpense = texpense, expense = 
expense,  total = total ,
                           t_food = t_food,t_entertainment =  t_entertainment,
                           t_business = t_business,  t_rent =  t_rent,
                           t_EMI =  t_EMI,  t_other =  t_other )
```

#delete---the--data

```
@app.route('/delete/<idx>', methods = ['POST', 'GET' ])
def delete(idx):
    query = "DELETE FROM expenses WHERE id=? and idx=?;"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, session["email"])
    ibm_db.bind_param(stmt, 2, idx)
    ibm_db.execute(stmt)
    print('deleted successfully')
    return render_template("display.html")
```

#UPDATE---DATA

```
@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    query = "SELECT * FROM expenses WHERE id=? and idx=?;"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, session['email'])
    ibm_db.bind_param(stmt, 2, id)
    ibm_db.execute(stmt)
    dictionary=ibm_db.fetch_assoc(stmt)
    expense=[]
    while dictionary != 
False:        exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary[
"AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"])
        expense.append(exp)
        dictionary = ibm_db.fetch_assoc(stmt)
    print(expense)
    return render_template('edit.html', expenses = expense[0])
@app.route('/update/<id>', methods = ['POST'])
def update(id):
  if request.method == 'POST' :
      dates = request.form['date']
      expensename = request.form['expensename']
      amount = request.form['amount']
      paymode = request.form['paymode']
      category = request.form['category']
      query = "UPDATE expenses SET dates = ? , expensename = ? , amount = ?, paymode = ?, 
category = ? WHERE id = ? and idx=?;"
      stmt = ibm_db.prepare(connection, query)
      ibm_db.bind_param(stmt, 1, dates)
```

```
            ibm_db.bind_param(stmt, 2, expensename)
            ibm_db.bind_param(stmt, 3, amount)
            ibm_db.bind_param(stmt, 4, paymode)
            ibm_db.bind_param(stmt, 5, category)
            ibm_db.bind_param(stmt, 6, session['email'])
            ibm_db.bind_param(stmt, 7, id)
            ibm_db.execute(stmt)
            print('successfully updated')
            return redirect("/display")
```

#limit

```
@app.route("/limit" )
def limit():
       return render_template('limit.html')
@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
    que = "SELECT * FROM limits where id = ? ;"
    stm = ibm_db.prepare(connection, que)
    ibm_db.bind_param(stm, 1, session['email'])
    ibm_db.execute(stm)
    if request.method == "POST":
        dictionary=ibm_db.fetch_assoc(stm)
        expense=[]
        while dictionary != False:
            exp=(dictionary['ID'],dictionary['NUMBER'],dictionary['IDX'])
            expense.append(exp)
            dictionary = ibm_db.fetch_assoc(stm)
        i=len(expense)+1
        idx=str(i)
        number= request.form['number']
        query = "INSERT INTO limits VALUES(?,?,?)"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, session['email'])
        ibm_db.bind_param(stmt, 2, number)
        ibm_db.bind_param(stmt, 3, idx)
        ibm_db.execute(stmt)
        return redirect('/limitn')
@app.route("/limitn")
def limitn():
    query = "SELECT max(IDX) as IDX FROM limits where id=?;"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, session['email'])
    ibm_db.execute(stmt)
    dictionary = ibm_db.fetch_assoc(stmt)
    expense=[]
    while dictionary != False:
        exp=(dictionary["IDX"])
        expense.append(exp)
        dictionary = ibm_db.fetch_assoc(stmt)
    k=expense[0]
    que = "SELECT NUMBER FROM limits where id=? and idx=?"
    stmt = ibm_db.prepare(connection, que)
    ibm_db.bind_param(stmt, 1, session['email'])
    ibm_db.bind_param(stmt, 2, k)
    ibm_db.execute(stmt)
    dictionary = ibm_db.fetch_assoc(stmt)
    texpense=[]
    while dictionary != False:
        exp=(dictionary["NUMBER"])
        texpense.append(exp)
        dictionary = ibm_db.fetch_assoc(stmt)
    s=texpense[0]
    return render_template("limit.html" , y= s)
```

#REPORT

```
@app.route("/today")
def today():
      query = "SELECT dates, amount FROM expenses  WHERE id = ? AND DATE(dates) = 
DATE(NOW()); "
      stmt = ibm_db.prepare(connection, query)
      ibm_db.bind_param(stmt, 1, str(session['email']))
      ibm_db.execute(stmt)
      dictionary=ibm_db.fetch_assoc(stmt)
      texpense=[]
      while dictionary != False:
        exp=(dictionary["DATES"],dictionary["AMOUNT"])
        texpense.append(exp)
        dictionary = ibm_db.fetch_assoc(stmt)
      print(texpense)
      query = "SELECT * FROM expenses WHERE id = ? AND DATE(dates) = DATE(NOW())"
      stmt = ibm_db.prepare(connection, query)
      ibm_db.bind_param(stmt, 1, session['email'])
      ibm_db.execute(stmt)
      dictionary=ibm_db.fetch_assoc(stmt)
      expense=[]
      while dictionary != False:
          exp=(dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"])
          expense.append(exp)
          dictionary = ibm_db.fetch_assoc(stmt)
      total=0
      t_food=0
      t_entertainment=0
      t_business=0
      t_rent=0
      t_EMI=0
      t_other=0
      for x in expense:
          total += x[0]
          if x[2] == "food":
              t_food += x[0]
          elif x[2] == "entertainment":
            t_entertainment  += x[0]
         elif x[2] == "business":
            t_business  += x[0]
         elif x[2] == "rent":
            t_rent  += x[0]
         elif x[2] == "EMI":
            t_EMI  += x[0]
         elif x[2] == "other":
            t_other  += x[0]
      print(total)
      print(t_food)
      print(t_entertainment)
      print(t_business)
      print(t_rent)
      print(t_EMI)
      print(t_other)
      return render_template("today.html", texpense = texpense, expense = expense,  total = total ,
                        t_food = t_food,t_entertainment =  t_entertainment,
                        t_business = t_business,  t_rent =  t_rent,
                        t_EMI =  t_EMI,  t_other =  t_other )
@app.route("/month")
def month():
      query = "SELECT dates, SUM(amount) as AMOUNT FROM expenses WHERE id= ? AND
MONTH(dates)= MONTH(now()) GROUP BY dates ORDER BY dates;"
      stmt = ibm_db.prepare(connection, query)
      ibm_db.bind_param(stmt, 1, str(session['email']))
      ibm_db.execute(stmt)
      dictionary=ibm_db.fetch_assoc(stmt)
      texpense=[]
      while dictionary != False:
        exp=(dictionary["DATES"],dictionary["AMOUNT"])
        texpense.append(exp)
        dictionary = ibm_db.fetch_assoc(stmt)
      print(texpense)
      query = "SELECT * FROM expenses WHERE id = ? AND MONTH(dates)= MONTH(now());"
      stmt = ibm_db.prepare(connection, query)
      ibm_db.bind_param(stmt, 1, session['email'])
      ibm_db.execute(stmt)
      dictionary=ibm_db.fetch_assoc(stmt)
      expense=[]
      while dictionary !=
False:        exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary[
"AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"])
        expense.append(exp)
        dictionary = ibm_db.fetch_assoc(stmt)
      total=0
      t_food=0
      t_entertainment=0
      t_business=0
      t_rent=0
      t_EMI=0
```

```
    t_other=0
    for x in expense:
      total += x[3]
      if x[5] == "food":
        t_food += x[3]
      elif x[5] == "entertainment":
        t_entertainment  += x[3]
      elif x[5] == "business":
        t_business  += x[3]
      elif x[5] == "rent":
        t_rent  += x[3]
      elif x[5] == "EMI":
        t_EMI  += x[3]
      elif x[5] == "other":
        t_other  += x[3]
    print(total)
    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)
    return render_template("month.html", texpense = texpense, expense = expense,  total = total ,
                           t_food = t_food,t_entertainment =  t_entertainment,
                           t_business = t_business,  t_rent =  t_rent,
                           t_EMI =  t_EMI,  t_other =  t_other )
@app.route("/year")
def year():
    query = "SELECT MONTH(dates) as DATES, SUM(amount) as AMOUNT FROM expenses
WHERE id=? AND YEAR(dates)= YEAR(now()) GROUP BY MONTH(dates);"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1,session['email'])
    ibm_db.execute(stmt)
    dictionary = ibm_db.fetch_assoc(stmt)
    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0
    for x in expense:
      total += x[3]
      if x[5] == "food":
        t_food += x[3]
      elif x[5] == "entertainment":
        t_entertainment  += x[3]
      elif x[5] == "business":
        t_business  += x[3]
      elif x[5] == "rent":
        t_rent  += x[3]
        elif x[5] == "EMI":
            t_EMI  += x[3]
        elif x[5] == "other":
            t_other  += x[3]
    print(total)
    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)
    return render_template("year.html", texpense = texpense, expense = expense,  total = total ,
                           t_food = t_food,t_entertainment =  t_entertainment,
                           t_business = t_business,  t_rent =  t_rent,
                           t_EMI =  t_EMI,  t_other =  t_other )
```

#log-out

```
@app.route('/logout')
def logout():
  session.pop('loggedin', None)
  session.pop('id', None)
  session.pop('username', None)
  return render_template('home.html')
```

GitHub & Project Demo Link

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-1923-1658420812

Demo Link:

https://github.com/IBM-EPBL/IBM-Project-1923-1658420812/tree/main/Final%20Deliverables