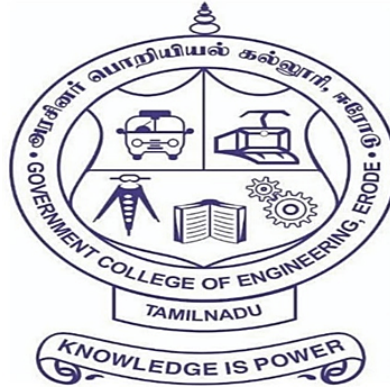


**GOVERNMENT COLLEGE OF ENGINEERING  
(Formerly IRTT) ERODE-638 316**



**BONAFIDE CERTIFICATE**

Certified that this project titled **“PERSONAL EXPENSE TRACKER APPLICATION”** is the bonafide work of **“NITHISHKUMAR P(731119104033), MOHAN RAJ P(731119104028), SHANMUGAM C K M(731119104044), SHREHARI GURUPRASAD B(731119104045)”** who carried out the project work under my supervision.

**SIGNATURE OF HOD**

Dr.A.SARADHA,M.E.,Ph.D.,  
HEAD OF THE DEPARTMENT  
DEPARTMENT OF CSE,  
GOVERNMENT COLLEGE OF  
ENGINEERING, ERODE – 638316

**SIGNATURE OF SPOC**

Dr.G.GOWRISON, M.E.,Ph.D.,  
ASSISTANT PROFESSOR(SR)  
DEPARTMENT OF ECE,  
GOVERNMENT COLLEGE OF  
ENGINEERING, ERODE - 638316

**SIGNATURE OF FACULTY  
EVALUATOR**

Dr.N.MAGESHNALLASAMY,M.E.,Ph.D.,  
ASSISTANT PROFESSOR(SR)  
DEPARTMENT OF CSE,  
GOVERNMENT COLLEGE OF  
ENGINEERING, ERODE – 638316

**SIGNATURE OF FACULTY  
MENTOR**

Dr.A.SARADHA  
ASSISTANT PROFESSOR(SR)  
DEPARTMENT OF CSE,  
GOVERNMENT COLLEGE OF  
ENGINEERING, ERODE - 638316

# PERSONAL EXPENSE TRACKER APPLICATION

<b>S.No.</b>	<b>TITLE</b>	<b>PAGE NO</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>4</b>
1.1.	Project Overview	4
1.2.	Purpose	4
<b>2.</b>	<b>LITERATURE SURVEY</b>	<b>4</b>
2.1.	Existing problem	4
2.2.	Reference	5
2.3.	Problem Statement Definition	6
<b>3.</b>	<b>IDEATION &amp; PROPOSED SOLUTION</b>	<b>7</b>
3.1.	Empathy Map Canvas	7
3.2.	Ideation & Brainstorming	7
3.3.	Proposed Solution	8
3.4.	Problem Solution Fit	9
<b>4.</b>	<b>REQUIREMENT ANALYSIS</b>	<b>10</b>
4.1.	Functional Requirement	10
4.2.	Non-Functional Requirement	10
<b>5.</b>	<b>PROJECT DESIGN</b>	<b>11</b>
5.1.	Data Flow Diagram	11
5.2.	Solution & Technical Architecture	11
5.3.	User Stories	12

<b>6.</b>	<b>PROJECT PLANNING &amp; SCHEDULING</b>	<b>13</b>
6.1.	Sprint Planning & Estimation	13
6.2.	Sprint Delivery Scheduling	15
6.3.	Report from JIRA	16
<b>7.</b>	<b>CODING &amp; SOLUTIONING</b>	<b>17</b>
7.1.	Feature 1	17
7.2.	Feature 2	17
<b>8.</b>	<b>TESTING</b>	<b>18</b>
8.1.	Test case	18
8.2.	User Acceptance Testing	18
<b>9.</b>	<b>RESULT</b>	<b>19</b>
9.1.	Performance Metrics	19
<b>10.</b>	<b>ADVANTAGES &amp; DISADVANTAGES</b>	<b>19</b>
<b>11.</b>	<b>CONCLUSION</b>	<b>20</b>
<b>12.</b>	<b>FUTURE SCOPE</b>	<b>20</b>
<b>13.</b>	<b>APPENDIX</b>	<b>20</b>
13.1.	Source Code	20
13.2.	GitHub & Project Demo Link	55

# **1. INTRODUCTION**

## **1.1. Project Overview**

The personal expense tracker is a web application which is used to maintain data of daily, weekly, monthly and yearly expenses in an eye-catching way. This project is aimed at developing a web application which will be helpful to users who run out of resources due to Mismanagement and also find it difficult to maintain records of their expenses. So personal Expense tracker will help them manage their needs and spending in a better way by accessing the web application directly from web browsers. It is designed and developed in a way that it is compatible with each and every device.

## **1.2. Purpose**

- A. No Need to install web application: - the problem of installing web application avoided on any device. So, reducing space and time related problems.
- B. Remotely Accessible: - a web-based application can be used remotely via a network connection that is platform independent.
- C. Movability and Repository: - to reduce the problem of movability and repository field by using to make the concept of web-based application.
- D. Cloud Storage: - Cloud Storage allows developers to store and retrieve data from the cloud database. The data stored in cloud will not expire. This means that data will persist even if the tab or the browser window is closed.
- E. Voice Features: - we have used Speechly for building real-time multimodal voice user interfaces. It enables developers and designers to enhance their current touch user interface with voice functionalities for better user experience.

# **2. LITERATURE SURVEY**

## **2.1. Existing problem**

- Lack of proper planning of out 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
- At the end of the month, we start to have money crisis.

## 2.2. Reference

S.NO	TITLE OF THE PAPER	AUTHOR	PUBLISHED YEAR	ABSTRACT
1	Expense Tracker	Aman Garg, Mukul Goel, Sagar Mittal, Mr. Shekhar Singh	April 2021	This Expense Tracker is a web application that facilitates the users to keep track and manage their personal as well as business expenses. This application helps the users to keep a digital diary. It will keep track of a user's income and expenses daily. Tracking your expenses daily can not only save your amount, but it can also assist you set financial goals for the longer term. If you know exactly where your amount goes every month, you will easily see where some cutbacks and compromises can be made.
2	Family Expense Manager Application	Rajaprabha M N	2017	The user can make use of this application in his/her daily life. After being used it can be a part of daily life to update and view daily expenses and family expenses. This helps to keep track of expenses & manage it for the user as they are busy in their daily routine, they are not able to keep track of their incomes & expenses
3	Daily Expense Tracker Mobile Application	Nuura Najati Binti Mustafa	2021	This application is an easier alternative to keeping track of users' use of money than the traditional way of writing their expenses in their diary. This application implements least squares method which helps to predict an outcome by finding the best fit line for a set of data. The use of the least squares method will help users in obtaining a successful budget planned with the prediction of the outcome of the budget based on expenses
4	Daily Expense	Shivam Mehra,	2021	This paper represents a Daily Expense Tracker is a tool that resides on a remote

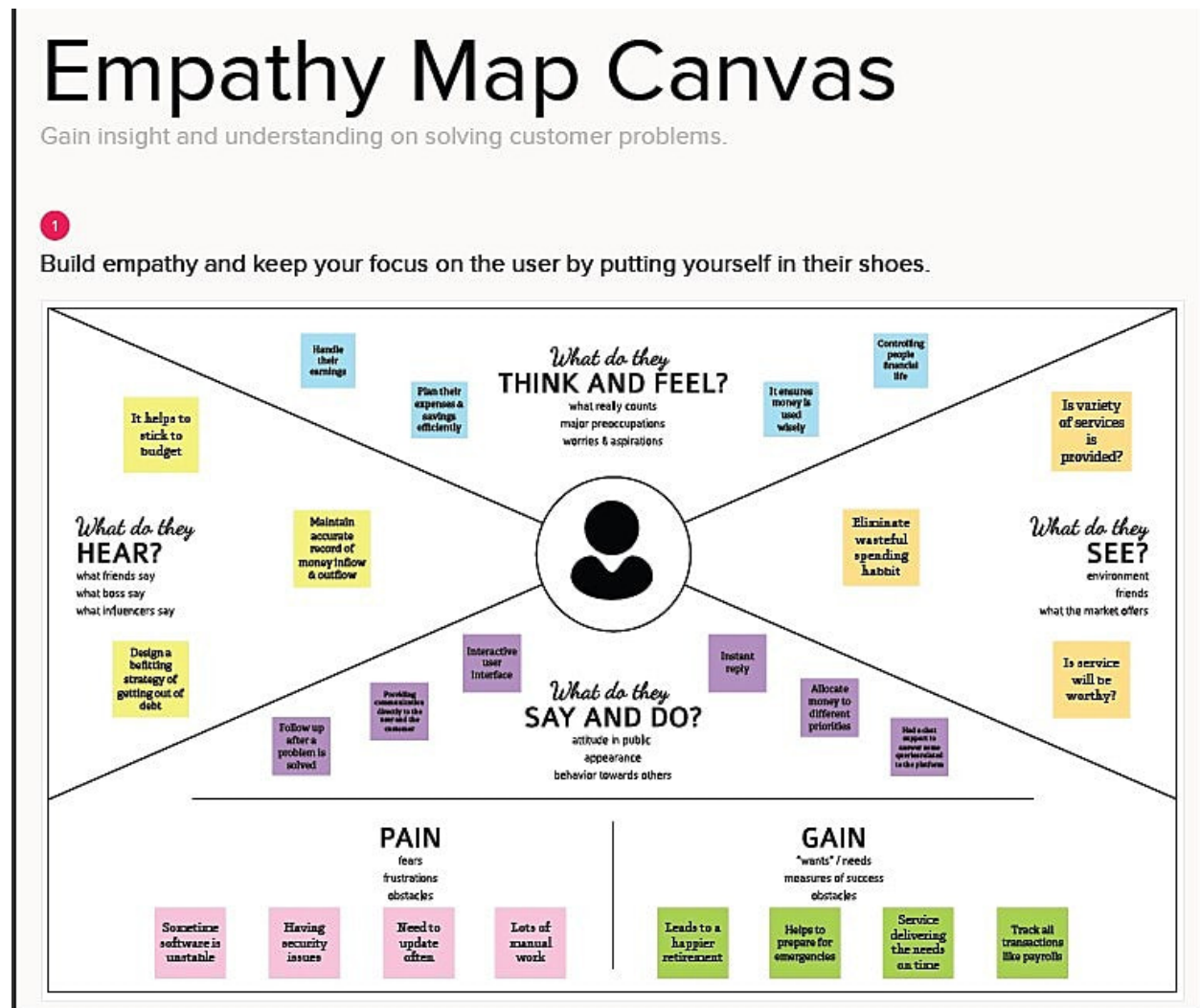
	Tracker	Prabhat Parashar		server and is accessible via browsers. It creates a digital record of the income and expense of the user. It input from the user a income, source of this income and the date of earning that income and creates a transaction entry under income category sums to the total amount of income and making real time changes. The web application will also be voice powered and all the functionalities can be used with voice commands.
5	Expenditure management system	Dr.V.Geetha,G. Nikhitha.	2022	In this method this device can be utilized by any individual to govern their income expenditure from each day to annual basis and to hold an eye on their spending, Including the person to whom the payments were made and the purpose for the payment. On a weekly, monthly, and yearly basis, details of expenses will be displayed in the form of a pie chart. It aids us in remembering and adding information about what money we receive from others and what costs or payments we must make on a given date or month.

### 2.3. Problem Statement Definition

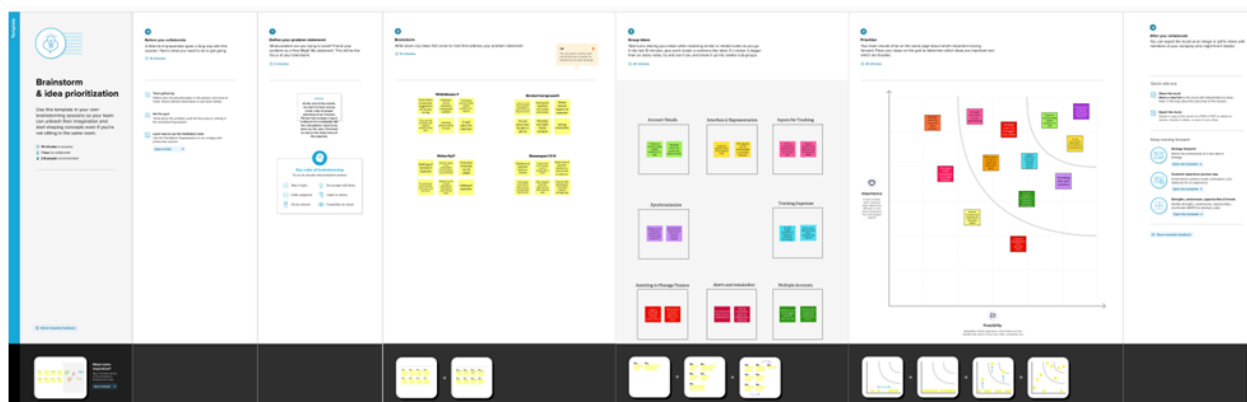
Many organizations have their own system to record their income and expenses, which they feel is the main key point of their business progress. It is a good habit for a person to record daily expenses and earning but due to unawareness and lack of proper applications to suit their privacy, lacking decision making capacity people are using traditional note keeping methods to do so. Due to lack of a complete tracking system, there is a constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.

### 3. IDEATION & PROPOSED SOLUTION

#### 3.1. Empathy Map Canvas



#### 3.2. Ideation & Brainstorming



### 3.3. Proposed Solution

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	<ul style="list-style-type: none"><li>a. It is challenging for the people to manage their cash flow day-to-day.</li><li>b. They always prepare a monthly budget for their expenses manually.</li></ul>
2.	Idea / Solution description	<ul style="list-style-type: none"><li>a. Due to the busy 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.</li><li>b. User cannot predict future expenses. While they can write down their expenses in a excel spreadsheet, their lack of knowledge in managing finances will be a problem.</li></ul>
3.	Novelty / Uniqueness	<ul style="list-style-type: none"><li>a. This application tracks your every expenses anywhere and anytime without using the paper work. Just click and enter your expenditure.</li><li>b. To avoid data loss, quick settlements and reduce human error.</li><li>c. To provide the charts or graph lines in this application.</li></ul>
4.	Social Impact / Customer Satisfaction	<ul style="list-style-type: none"><li>a. Using the application one can track their personal expenses and frame a monthly/annual budget.</li><li>b. If your expense exceeded than the specified limit, the application will show you an alert message in the form of an e-mail.</li></ul>
5.	Business Model (Revenue Model)	<ul style="list-style-type: none"><li>a. We can provide the application on a subscription basis.</li><li>b. We can provide pop-up ads, overlay ads, and other advertising services from third party advertisers.</li></ul>




6.	Scalability of the Solution	<p>a. This application will be available to all the user for life long time to maintain their own expenses in an effective way.</p> <p>b. IBM cloud will automatically allocate the storage for the users.</p>
----	-----------------------------	--

### 3.4. Proposed Solution fit

Problem-Solution fit canvas 2.0		Purpose / Vision	
Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <small>Who is your customer? I.e. working parents of 0-5 y.o. kids</small> 1) Customers who are not able keep track of their expenditure they do daily. 2) Customers who can't remember for what or when they have spent the money.	<b>6. CUSTOMER CONSTRAINTS</b> <small>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.</small> 1) This application will be submitted by most of the devices. 2) The solution we propose will have an alert via email feature, if expense exceed the given limit. 3) This solution also provides insights on their expenses on a graphical way.	<b>5. AVAILABLE SOLUTIONS</b> <small>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 &amp; cons do these solutions have? I.e. pen and paper is an alternative to digital notetaking</small> 1) Customers have used notes or papers to track their expenses. 2) Personal Expense Tracker developed in this project is an alternative.
	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <small>Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.</small> 1) The application allow the customers to keep track of their expenses. 2) They will be able to categories their expenses. 3) They will be also given option to set budget and will receive alert on mail when their expense exceeds the budget. 4) They can also have an insight of their expenses in a graphical representation either yearly or monthly.	<b>9. PROBLEM ROOT CAUSE</b> <small>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.</small> 1) Due to lot of payment options, customers tend to forget where or when they spent their money. 2) By tracking their expense they can save their money. 3) They can save lot of time and money.	<b>7. BEHAVIOUR</b> <small>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)</small> 1) Make sure he uses the app to track expenses. 2) Make sure they categorize the expenses correctly. 3) To set limit to their monthly expenses, to receive alerts via mail if expenses exceed the limit.
Focus on J&P, tap into BE, understand RC	<b>3. TRIGGERS</b> <small>What triggers customers to act? I.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.</small> 1) Customers can know how their money is being spent.	<b>10. YOUR SOLUTION</b> <small>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.</small> 1) To design a personal expense tracker using flask. 2) To provide insights on their spending in a graphical way based on categories. 3) To send an alert via email if their expense exceed the limit they set.	<b>8. CHANNELS of BEHAVIOUR</b> <b>8.1 ONLINE</b> <small>What kind of actions do customers take online? Extract online channels from #7</small> 1) All their data are being secured and updated to cloud storage.
Identify strong TR & EM	<b>4. EMOTIONS: BEFORE / AFTER</b> <small>How do customers feel when they face a problem or a job and afterwards? I.e. lost, insecure &gt; confident, in control - use it in your communication strategy &amp; design.</small> 1) They will be also to track their income and expense made by them.	<b>8.2 OFFLINE</b> <small>What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.</small> 1) Make sure their expenses is stored offline and updated to cloud once they are online.	<b>CH</b> <small>Extract online &amp; offline CH of BE</small>

Problem-Solution fit canvas is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 license  
Created by Darja Nepriakhina / Amaltama.com

 **AMALTAMA**

## 4. REQUIREMENT ANALYSIS

### 4.1. Functional Requirement

FR No.	Functional Requirements (Epic)	Sub Requirement (Story/ Sub-Task)
FR-1	User Registration	Registration through Application Registration through E-mail
FR-2	User Confirmation	Confirmation via E-mail Confirmation via OTP
FR-3	User monthly expense tentative data	Data to be registered in the app
FR-4	User monthly income data	Data to be registered in the app
FR-5	Alert/ Notification	Alert through E-mail Alert through SMS
FR-6	User Budget Plan	Planning and Tracking of user expense vs budget limit

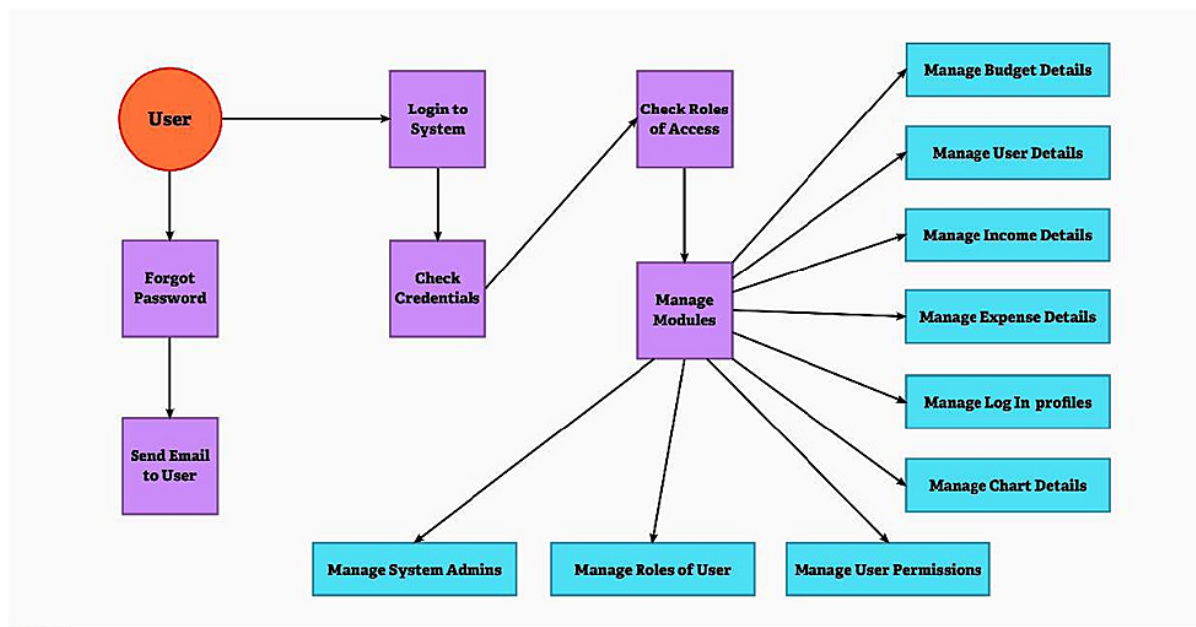
### 4.2. Non-Functional Requirements

NFR No.	Non-Functional Requirements	Description
NFR-1	Usability	Effectiveness, efficiency and overall satisfaction of the user while interacting with our application.
NFR-2	Security	Authentication, authorization, encryption of the application.
NFR-3	Reliability	Probability of failure-free operations in a specified environment for a specified time.
NFR-4	Performance	How the application is functioning and how responsive the application is to the end-users.
NFR-5	Availability	Without near 100% availability, application reliability and the user satisfaction will affect the solution.
NFR-6	Scalability	Capacity of the application to handle growth, especially in handling more users.

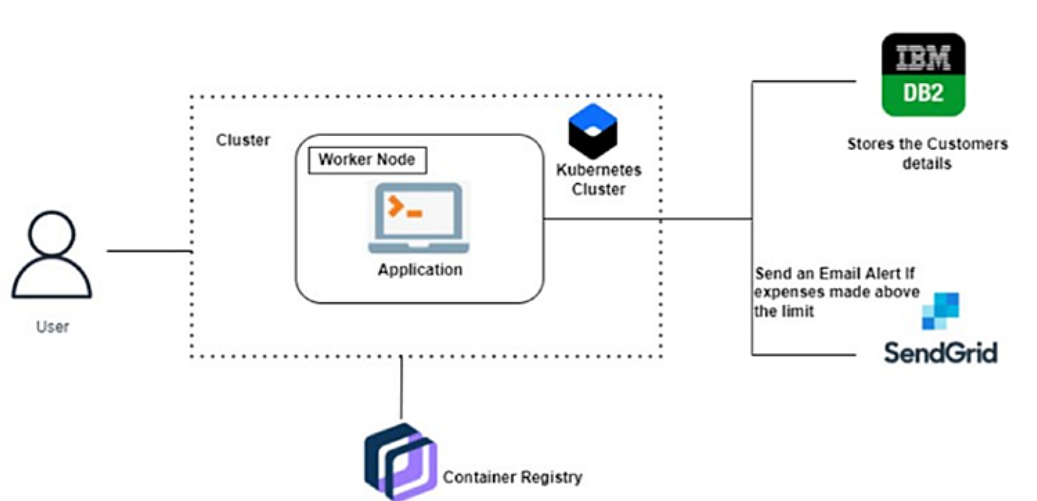
## 5. PROJECT DESIGN

### 5.1. Data Flow Diagrams

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



### 5.2. Solution & Technical Architecture



### 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	Sprint - 1
	Login	USN - 2	As a user, I can login to user dashboard and see the information about my incomes and expenses.	I can login to user dashboard and see the information.	High	Sprint - 1
	Dashboard	USN - 3	As a user, I can enter my income and expenditure details.	I can view my daily expenses.	High	Sprint - 2
	Expense Update	USN - 4	As a user, I can track my expenses and manage my monthly budget.	I can track my expenses and manage my monthly budget.	High	Sprint - 3

<b>User Type</b>	<b>Functional Requirement (Epic)</b>	<b>User Story Number</b>	<b>User Story / Task</b>	<b>Acceptance criteria</b>	<b>Priority</b>	<b>Release</b>
	Email Alert	USN - 5	As a user, I can see if there is an excessive expense and if there is such condition, I will be notified via e-mail.	I can receive e mail, if there is an excessive expense.	Medium	Sprint – 3
Customer Care Executive	Customer Care	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	Sprint – 4
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	Sprint – 4

## 6. PROJECT PLANNING & SCHEDULING

### 6.1. Sprint Planning & Estimation

<b>Sprint</b>	<b>Functional Requirements (Epic)</b>	<b>User Story Number</b>	<b>User Story / Task</b>	<b>Story Points</b>	<b>Priority</b>	<b>Team Members</b>
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	3	High	Nithishkumar P Shrehari Guruprasad B Mohan Raj P Shanmugam C K M
Sprint-1	Login	USN-2	As a user, I can login to user dashboard and see the information about my incomes and expenses.	3	High	Nithishkumar P Shrehari Guruprasad B Mohan Raj P Shanmugam C K M
Sprint-2	Dashboard	USN-3	As a user, I can enter my income and expenditure details.	4	High	Nithishkumar P Shrehari Guruprasad B Mohan Raj P Shanmugam C K M
Sprint-3	Expense Update	USN-4	As a user, I can track my expenses and manage my monthly budget.	2	High	Nithishkumar P Shrehari Guruprasad B Mohan Raj P Shanmugam C K M

Sprint-3	Email Alert	USN-5	As a user, I can see if there is an excessive expense and if there is such condition, I will be notified via email.	6	Medium	Nithishkumar P Shrehari Guruprasad B Mohan Raj P Shanmugam C K M
Sprint-4	Customer Care	USN-6	As a customer care executive, I can solve the login issues and other issues of the application.	3	Medium	Nithishkumar P Shrehari Guruprasad B Mohan Raj P Shanmugam C K M
Sprint-4	Application	USN-7	As an administrator, I can upgrade or update the application.	3	Medium	Nithishkumar P Shrehari Guruprasad B Mohan Raj P Shanmugam C K M

## 6.2. Sprint Delivery Schedule

<b>Sprint</b>	<b>Total Story Points</b>	<b>Duration</b>	<b>Sprint Start Date</b>	<b>Sprint End Date (Planned)</b>	<b>Story Points Completed (as on Planned End Date)</b>	<b>Sprint Release Date (Actual)</b>
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

The screenshot shows the Jira Software interface for the 'Personal Expense Tracker' project. The left sidebar contains navigation options: Roadmap, Backlog (selected), Board, Code, Project pages, Add shortcut, and Project settings. The main area displays the 'Backlog' view for the project. At the top, there's a banner to 'Set project and issue permissions. Try it in a 14-day trial of Jira Software Standard.' Below this, the 'Backlog' section shows two sprints: 'PET Sprint 1' (24 Oct – 29 Oct, 2 issues) and 'PET Sprint 2' (30 Oct – 5 Nov, 1 issue). Each sprint contains a list of issues with their status (e.g., DONE, IN PROGRESS) and a 'Create issue' button. A 'Quickstart' button is visible at the bottom right of the backlog view.

The screenshot shows the Jira Software interface for the 'Personal Expense Tracker' project, specifically the 'Roadmap' view. The left sidebar is the same as the previous screenshot. The main area displays a timeline view of the project's progress. The timeline is organized by months: OCT, NOV, DEC, JAN '23, and FEB. A vertical orange line indicates the current date. The roadmap shows several sprints and their associated issues, including 'PET-8 Registration', 'PET-9 Login', 'PET-10 Dashboard', 'PET-11 Expense Update', 'PET-12 Email Alert', 'PET-13 Customer Care', and 'PET-14 Application'. A 'Quickstart' button is visible at the bottom right of the roadmap view.



## **7. CODING & SOLUTIONING**

### **7.1. Feature 1**

In simple words, 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.

Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert.

#### **Software Required**

- Python
- Flask
- Docker
- Kubernetes
- IBM DB2

#### **System Required**

- 8GB RAM.
- Intel Core i3.
- OS-Windows/Linux/MAC .
- Laptop or Desktop

### **7.2. Feature 2**

Tracking your expenses can save your amount, but it can also help you set and work for financial goals for the future. If you know exactly where your amount is going every month, you can easily see where some cutbacks and compromises can be made and are possible. The project what we have developed words more efficient than the other available income and expense tracker. The project successfully avoids the manual calculation for avoiding calculation the income and expense per month and save time of user. The modules are developed with efficient, reliable and also in an attractive manner.

## 8. TESTING

### 8.1. Test case

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
1	Functional	Login Page	Verify user is able to Login into the Application		1) Open the Personal expense tracker application. 2) Login with user Credentials 3) Verify logged in to user account	Username: shrehari Password: Shre@2002	Login Successful	Working as expected	Pass		N		SHREHARI GURUPRASAD B
2	Functional	Signup Page	Verify user is able to Signup in the Application		1) Open the Personal expense tracker 2) Enter the Details and Create a new User 3) Verify if user is created and	Username: shanmugam Password: Shan@2002 Name: Shan DOB: 12/9/2001 Password: Shre@2002	Account Created Successfully	Working as expected	Pass		N		SHANMUGAM CK M
3	Functional	Dashboard page	Verify if all the user details are stored in Database		1) Open the Personal expense tracker application. 2) Enter the Details and Create a new User 3) Verify if user is created and inserted into DB Table	Username: mohan@gmail.com password: Testing@123	User should navigate to user account homepage	Working as expected	Pass				NITHSHKUMAR P
4	Functional	Login page	Verify user is able to log into application with Invalid credentials		1) Enter URL and click go 2) Click on Sign IN button 3) Enter Invalid username/email in Email test box 4) Enter valid password in password test box 5) Click on login button	Username: mohan@gmail.com password: Testing@123	Application should show 'Incorrect email or password' validation message.	Working as expected	Pass				MOHAN RAJ P
5	Functional	Login page	Verify user is able to log into application with Invalid credentials		1) Enter URL and click go 2) Click on Sign IN button 3) Enter Invalid username/email in Email test box 4) Enter valid password in password test box 5) Click on login button	Username: mohan@gmail.com password: Testing@123	Application should show 'Incorrect email or password' validation message.	Working as expected	Pass				MOHAN RAJ P

### 8.2. User Acceptance Testing

#### 1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

#### 2. 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	Severity 5
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

### 3. 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. RESULT

### 9.1. Performance Metrics

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	Records Page	1) Log in to Personal Expense Tracker Application 2) Enter all the pesonal details and expenses and mark it as expense or income	No Risks	N/A
4	Dashboard	1) Log in to Personal Expense Tracker Application 2) View the Analytics	No Risks	N/A
5	Bills Page	1) Log in to Personal Expense Tracker Application 2) Bills can be added.	No Risks	N/A
5	Email Acknowledgement	1) Mails are Sent to the Registered user if expenses>budget	No Risks	N/A

## 10. ADVANTAGES & DISADVANTAGES

### 10.1. Advantages

Knowing what you spend will help you to:

- Create a monthly budget
- Know where you're spending more than you actually think you are
- Figure out ways to cut back on your spending
- Know how much extra payments you can make towards your debt
- Plan for future large purchases
- Create a savings plan for putting money away every month
- Plan for retirement

- Create an investment strategy with extra money
- Prepare to file income taxes

## 10.2. Disadvantages

- Having security issues
- Need to update often
- Lots of manual work

## 11. CONCLUSION

Tracking your expenses can save your amount, but it can also help you set and work for financial goals for the future. If you know exactly where your amount is going every month, you can easily see where some cutbacks and compromises can be made and are possible. The project what we have developed words more efficient than the other available income and expense tracker. The project successfully avoids the manual calculation for avoiding calculation the income and expense per month and save time of user. The modules are developed with efficient, reliable and also in an attractive manner.

## 12. FUTURE SCOPE

Provision to add different currencies will be added so that this application is not just limited to India but also can be used worldwide and the currency converters will be designed and added in order to convert the different currency rates. In order to make it more user friendly and less user intensive, when the user tries to add the same category or vendor to an expense/income record, a duplicate alert will be presented showing the same category/vendor which the user entered previously for some expense/income and then he can tap on it and the entries will be automatically filled for the current record

## 13. APPENDIX

### 13.1. Source Code

#### Details.jsx

```
import React from 'react';
import { Card, CardContent } from '@mui/material';
import { Doughnut, Pie, PolarArea, Radar } from 'react-chartjs-2';
import './details.css';
import useTransactions from '../useTransactions';
import 'chart.js/auto';
```

```

const Details = ({ title }) => {
  /*const [doughnatC, setDoughnatC] = useState(true);
  const [polarAreaC, setPolarAreaC] = useState(false);
  const [pieC, setPieC] = useState(false);
  const [radarC, setRadarC] = useState(false);*/
  const { chartData } = useTransactions(title);
  console.log(chartData);
  return (
    <div>
      <div style={{ display: 'flex', flexDirection: 'row', gap: '5px', marginTop: '10px' }}>
        <Card style={{}} className={title === 'Income' ? "income" : "expense"} >
          {/*<div className="chartButtonContainer" >
            <button className={title === 'Income' ? (doughnatC ? "selectedIn" : "btnIn") :
(doughnatC ? "selectedEx" : "btnEx")} >
              onClick={() => {
                setDoughnatC(true);
                setPolarAreaC(false);
                setPieC(false);
                setRadarC(false);
              }}
            >Doughnat</button>
            <button className={title === 'Income' ? (polarAreaC ? "selectedIn" : "btnIn") :
(polarAreaC ? "selectedEx" : "btnEx")} >
              onClick={() => {
                setDoughnatC(false);
                setPolarAreaC(true);
                setPieC(false);
                setRadarC(false);
              }}
            >PolarArea</button>
            <button className={title === 'Income' ? (pieC ? "selectedIn" : "btnIn") : (pieC ?
"selectedEx" : "btnEx")} >
              onClick={() => {
                setDoughnatC(false);
                setPolarAreaC(false);
                setPieC(true);
                setRadarC(false);
              }}
            >Pie</button>
          </div>*/}
        </div>
      </div>
    </div>
  )
}

```

```

    }}
  >Pie</button>
  <button className={title === 'Income' ? (radarC ? "selectedIn" : "btnIn") : (radarC ?
"selectedEx" : "btnEx")}
    onClick={() => {
      setDoughnatC(false);
      setPolarAreaC(false);
      setPieC(false);
      setRadarC(true);
    }}
  >Radar</button>
</div>*/}
{ /*<CardHeader style={{ textAlign: "center", }} title={title+": Rs. "+total} />*/}
<CardContent>
  <Doughnut data={chartData} />
</CardContent>
</Card>
<Card className={title === 'Income' ? "income" : "expense"} >
  <CardContent>
    <PolarArea data={chartData} />
  </CardContent>
</Card>
<Card className={title === 'Income' ? "income" : "expense"} >
  <CardContent>
    <Pie data={chartData} />
  </CardContent>
</Card>
<Card className={title === 'Income' ? "income" : "expense"} >
  <CardContent>
    <Radar data={chartData} />
  </CardContent>
</Card>

</div>
</div>
);

```

```
};  
export default Details;
```

### **details.css**

```
.income {  
  border-top: 10px solid rgba(0, 255, 0, 0.7);  
  border-bottom: 10px solid rgba(0, 255, 0, 0.7);  
}  
.expense {  
  border-top: 10px solid rgba(255, 0, 0, 0.7);  
  border-bottom: 10px solid rgba(255, 0, 0, 0.7);  
}
```

### **DetailsTrack.jsx**

```
import React from 'react';  
import { Line } from 'react-chartjs-2';  
import useTransactions from '../useTransactions';  
import './detailsTrack.css';  
const DetailsTrack = ({ title }) => {  
  const { chartDataIncome, chartDataExpense } = useTransactions(title);  
  const chartData = (title === 'Income') ? chartDataIncome : chartDataExpense;  
  return (  
    <div className='chartContainer'>  
      <div className={title === 'Income' ? 'incomeChart' : 'expenseChart'} >  
        <Line data={chartData} />  
      </div>  
    </div>  
  );  
};  
export default DetailsTrack;
```

### **detailsTrack.css**

```
.incomeChart {  
  background-color: white;  
  border: 5px solid rgb(147, 147, 147);  
  margin: 20px 200px;
```

```

padding: 50px;
border-radius: 10px;
}
.expenseChart {
background-color: white;
border: 5px solid rgb(147, 147, 147);
margin: 20px 200px;
padding: 50px;
border-radius: 10px;
}

```

### main.jsx

```

import React from 'react';
import { Card, CardContent, Typography, Grid, Divider } from '@mui/material';
import './main.css';
import Form from './Form/Form';
import List from './List/List';
import { useContext } from 'react';
import { ExpenseTrackerContext } from '../context/context';
import InfoCard from '../InfoCard';
import useTransactions from '../useTransactions';
const Main = () => {
  const { balance } = useContext(ExpenseTrackerContext);
  const { total: incomeTotal } = useTransactions("Income");
  const { total: expenseTotal } = useTransactions("Expense");
  return (
    <Card style={{width: '370px', padding: '10px', backgroundColor: "transparent" }}>
      {/*<CardHeader className="cardHeader" title="Personal Expense Tracker"
subheader="speak to track" />*/}
      <CardContent className="cardContent1">
        <Typography style={{ color: 'rgb(144 122 0)' }} align="center" variant="h6">
          Total Balance: <span style={{ color: 'goldenrod' }}>Rs.{balance}</span>
        </Typography>
        <Typography style={{ color: 'green' }} align="center" variant="h6">
          Income: <span style={{ color: 'rgba(0, 255, 0, 0.9)' }}>Rs.{incomeTotal}</span>
        </Typography>
        <Typography style={{ color: 'red' }} align="center" variant="h6">
          Expense: <span style={{ color: 'rgba(255, 0, 0, 0.8)' }}>Rs.{expenseTotal}</span>
        </Typography>
        <Typography variant="subtitle1" style={{ lineHeight: '1.5em', marginTop: '20px' }} >

```



```

        <InfoCard />
      </Typography>
      <Divider style={{ borderTop: '1px dashed gray', backgroundColor: 'white',
paddingTop: '15px' }} />
      <Form />
    </CardContent>
    <CardContent className="cardContent2">
      <Grid container spacing={2}>
        <Grid item xs={12}>
          <List />
        </Grid>
      </Grid>
    </CardContent>
  </Card>
);
};
export default Main;

```

### **main.css**

```

.cardHeader {
  text-align: center;
  border: 2px solid rgb(144 122 0);
  color: rgb(144 122 0);
  border-radius: 5px;
  background-color: white;
}
.cardContent1 {
  border: 2px solid rgb(144 122 0);
  border-radius: 5px;
  background-color: white;
}
.cardContent2 {
  border: 2px solid rgb(144 122 0);
  border-radius: 5px;
  background-color: white;
}

```

### **List.jsx**

```

import React from 'react';

```

```

import { List as MUIList, ListItem, ListItemAvatar, ListItemText, Avatar,
ListItemSecondaryAction, IconButton, Slide } from '@mui/material';
import CurrencyRupeeIcon from '@mui/icons-material/CurrencyRupee';
import DeleteIcon from '@mui/icons-material/Delete';
import { useContext } from 'react';
import { ExpenseTrackerContext } from '../../context/context';
import { useState } from 'react';
import CustomizedSnackbar from '../Snackbar/Snackbar';
import './list.css';
const List = () => {
  const { deleteTransaction, transactions } = useContext(ExpenseTrackerContext);
  const [open, setOpen] = useState(false);
  return (
    <MUIList dense={false} className='listContainer'>
      <CustomizedSnackbar title="list" open={open} setOpen={setOpen} />

      {transactions.map((transaction) => (
        <Slide direction="down" in mountOnEnter unmountOnExit key={transaction.id}>
          <ListItem>
            <ListItemAvatar>
              <Avatar style={{ backgroundColor: transaction.type === 'Income' ? 'green' :
'red'}} >
                <CurrencyRupeeIcon />
              </Avatar>
            </ListItemAvatar>
            <ListItemText primary={transaction.category}
secondary={`Rs.${transaction.amount} - ${transaction.date}`} />
            <ListItemSecondaryAction>
              <IconButton edge="end" aria-label="delete" onClick={() => { setOpen(true);
deleteTransaction(transaction.id) }} >
                <DeleteIcon />
              </IconButton>
            </ListItemSecondaryAction>
          </ListItem>
        </Slide>
      )))
    </MUIList>
  );
}
export default List;

```

## ListStyles.js

```
import { makeStyles } from '@material-ui/core/styles';
import { red, green } from '@material-ui/core/colors';
export default makeStyles((theme) => ({
  avatarIncome: {
    color: '#fff',
    backgroundColor: green[500],
  },
  avatarExpense: {
    color: theme.palette.getContrastText(red[500]),
    backgroundColor: red[500],
  },
  list: {
    maxHeight: '240px',
    overflow: 'auto',
  },
})));
```

## list.css

```
.avatarIncome {
  color: #fff;
  background-color: green;
}
.avatarExpense {
  color: #fff;
  background-color: red;
}
.listContainer {
  max-height: 240px;
  overflow: auto;
}
```

## form.css

```
.createBtn {
  margin-top: 50px;
  margin-bottom: 20px;
  margin-left: 16px;
  width: 100%;
```

```

border-radius: 10px;
padding: 15px;
border: 2px solid rgb(111 95 0);
background-color: rgb(231 229 175);
cursor: pointer;
color: rgb(111 95 0);
font-weight: 600;
}

```

## Form.jsx

```

import React, { useContext, useEffect, useState } from 'react';
import { TextField, Typography, Grid, FormControl, InputLabel, Select, MenuItem,
OutlinedInput } from '@mui/material';
import { v4 as uuidv4 } from 'uuid';
import './form.css';
import { ExpenseTrackerContext } from '../../context/context';
import { expenseCategories, incomeCategories } from '../../constants/categories';
import formatDate from '../../utils/formatDate';
import { useSpeechContext } from '@speechly/react-client';
import CustomizedSnackbar from '../../Snackbar/Snackbar';
const initialState = {
  amount: "",
  category: "",
  type: 'Income',
  date: formatDate(new Date()),
}
const Form = () => {
  const [formData, setFormData] = useState(initialState);
  const { addTransaction } = useContext(ExpenseTrackerContext);
  const { segment } = useSpeechContext();
  const [open, setOpen] = useState(false);
  const [unfill, setUnfill] = useState(false);
  const createTransaction = () => {
    if (formData.category && formData.amount) {
      const transaction = { ...formData, amount: Number(formData.amount), id: uuidv4() }
      addTransaction(transaction);
      setFormData(initialState);
      setOpen(true);
    }
    else {

```

```

    setUnfill(true);
    /*swal("Unfulfilled Inputs", "Please fill all the fields", "warning");*/
  }
}
useEffect(() => {
  if (segment) {
    if (segment.intent.intent === 'add_expense') {
      setFormData({ ...formData, type: 'Expense' });
    }
    else if (segment.intent.intent === 'add_income') {
      setFormData({ ...formData, type: 'Income' });
    }
    else if (segment.isFinal && segment.intent.intent === 'create_transaction') {
      return createTransaction();
    }
    else if (segment.isFinal && segment.intent.intent === 'cancel_transaction') {
      return setFormData(initialState);
    }
    segment.entities.forEach((e) => {
      const category = `${e.value.charAt(0)}${e.value.slice(1).toLowerCase()}`;
      switch (e.type) {
        case 'amount':
          setFormData({ ...formData, amount: e.value });
          break;
        case 'category':
          if (incomeCategories.map((ic) => ic.type).includes(category)) {
            setFormData({ ...formData, type: 'Income', category });
          }
          else if (expenseCategories.map((ec) => ec.type).includes(category)) {
            setFormData({ ...formData, type: 'Expense', category });
          }
          break;
        case 'date':
          setFormData({ ...formData, date: e.value });
          break;
        default:
          break;
      }
    });
    if (segment.isFinal && formData.type && formData.category && formData.amount &&
    formData.date) {

```

```

        createTransaction();
    }
}
// eslint-disable-next-line
}, [segment]);
const selectedCategories = formData.type === 'Income' ? incomeCategories :
expenseCategories;
return (
  <Grid container spacing={2}>
    <CustomizedSnackbar title="form" open={open} setOpen={setOpen} />
    <CustomizedSnackbar title="unfilled" open={unfill} setOpen={setUnfill} />
    <Grid item xs={12}>
      <Typography align="center" variant="subtitle2" gutterBottom>
        {segment &&
          segment.words.map((w) => w.value).join(" ")
        }
      </Typography>
    </Grid>
    <Grid item xs={6}>
      <FormControl fullWidth>
        <InputLabel>Type</InputLabel>
        <Select
          value={formData.type}
          onChange={(e) => setFormData({ ...formData, type: e.target.value })}
          input={<OutlinedInput label="Type" />}
        >
          <MenuItem value="Income">Income</MenuItem>
          <MenuItem value="Expense">Expense</MenuItem>
        </Select>
      </FormControl>
    </Grid>
    <Grid item xs={6}>
      <FormControl fullWidth>
        <InputLabel>Category</InputLabel>
        <Select
          value={formData.category}
          onChange={(e) => setFormData({ ...formData, category: e.target.value })}
          input={<OutlinedInput label="Category" />}
        >
          {
            selectedCategories.map((c) =>

```

```

        <MenuItem key={c.type} value={c.type}>
          {c.type}
        </MenuItem>
      )
    </Select>
  </FormControl>
</Grid>
<Grid item xs={6}>
  <TextField type="number" onKeyDown={(e) => { e.key === 'e' &&
e.preventDefault(); e.key === '-' && e.preventDefault(); e.key === '+' && e.preventDefault();
}} label="Amount" fullWidth value={formData.amount} onChange={(e) => setFormData({
...formData, amount: e.target.value })} />
</Grid>
<Grid item xs={6}>
  <TextField type="date" label="Date" fullWidth value={formData.date}
onChange={(e) => setFormData({ ...formData, date: formatDate(e.target.value) })} />
</Grid>
  <button className="createBtn" onClick={createTransaction} >Create</button>
</Grid>
);
};
export default Form;

```

### navbar.css

```

.navbar {
  display: flex;
  flex-direction: row;
  height: 70px;
  background-color: white;
  border: 3px solid darkgoldenrod;
  border-radius: 5px;
  margin-top: 10px;
  align-items: center;
  justify-content: space-between;
}
.navTitle {
  font-size: 30px;
  margin-left: 15%;
  text-align: center;
  color: rgb(187, 134, 0);
}

```

```

    font-weight: bold;
    letter-spacing: 1px;
}
.navSubTitle {
    letter-spacing: normal;
    font-size: 20px;
    color: grey;
    margin-left: 75px;
}
.navBtn {
    height: 70%;
    margin-right: 220px;
    background-color: rgb(228, 227, 194);
    border-radius: 5px;
    border: 2px solid rgb(111, 95, 0);
    font-weight: 600;
    color: rgb(194, 128, 6);
    padding: 10px 20px;
    cursor: pointer;
}
/*
.navContainer {
    display: flex;
    flex-direction: row;
}
.navItems {
    display: flex;
    flex-direction: row;
}
.logoutBtn, .profileBtn {
    border-radius: 10px;
    padding: 0px 20px 0px 20px;
    border: 2px solid rgb(111 95 0);
    background-color: rgb(231 229 175);
    cursor: pointer;
    color: rgb(111 95 0);
    font-weight: 600;
}
.navHeader {
    text-align: left;
    border: 2px solid rgb(144 122 0);

```



```

    color: rgb(144 122 0);
    border-radius: 5px;
    background-color: white;
    width: 100%;
  }
  .navTitle {
  } */

```

### **Navbar.jsx**

```

import React from 'react';
import { useNavigate } from 'react-router';
import './navbar.css';
const Navbar = () => {
  const navigate = useNavigate();
  const handleLogout = () => {
    navigate('/');
  }
  return (
    <div className="navbar">
      <div className="navTitle">Personal Expense Tracker
        <span className='navSubTitle'>speak to track</span>
      </div>
      <div>
        <button className='navBtn' onClick={() => handleLogout()}>Logout</button>
      </div>
    </div>
  );
}
export default Navbar;

```

### **snackbar.css**

```

.root {
  width: 100%;
}

```

### **Snackbar.jsx**

```

import React from 'react'
import Snackbar from '@mui/material/Snackbar';

```

```

import MuiAlert from '@mui/material/Alert';
const CustomizedSnackbar = ({ title, open, setOpen }) => {
  const handleClose = (event, reason) => {
    if (reason === 'clickaway') return;

    setOpen(false);
  }
  return (
    <div>
      {title === 'list' &&
        <div className='root'>
          <Snackbar
            anchorOrigin={{ vertical: 'top', horizontal: 'right' }}
            open={open}
            autoHideDuration={3000}
            onClose={handleClose}
          >
            <MuiAlert onClose={handleClose} severity="success" elevation={6}
variant="filled" >
              Category successfully Removed
            </MuiAlert>
          </Snackbar>
        </div>
      }
      {title === 'form' &&
        <div className='root'>
          <Snackbar
            anchorOrigin={{ vertical: 'top', horizontal: 'right' }}
            open={open}
            autoHideDuration={3000}
            onClose={handleClose}
          >
            <MuiAlert onClose={handleClose} severity="success" elevation={6}
variant="filled" >
              Category successfully Added
            </MuiAlert>
          </Snackbar>
        </div>
      }
      {title === 'unfilled' &&
        <div className='root'>

```

```

        <Snackbar
          anchorOrigin={{ vertical: 'top', horizontal: 'right' }}
          open={open}
          autoHideDuration={3000}
          onClose={handleClose}
        >
          <MuiAlert onClose={handleClose} severity="warning" elevation={6}
variant="filled" >
            Unfulfilled fields..!!
          </MuiAlert>
        </Snackbar>
      </div>
    }
  </div>
);
};
export default CustomizedSnackbar;

```

### **SnackbarStyles.js**

```

import { makeStyles } from '@material-ui/core/styles';
export default makeStyles((theme) => ({
  root: {
    width: '100%',
    '& > * + *': {
      marginTop: theme.spacing(2),
    },
  },
})));

```

### **index.js**

```

export { default as Details } from './Details/Details';
export { default as Main } from './Main/Main';
export { default as Snackbar } from './Snackbar/Snackbar';
export { default as InfoCard } from './InfoCard';

```

### **InfoCard.jsx**

```

import React from 'react';
const isIncome = Math.round(Math.random());

```

```

const InfoCard = () => {
  return (
    <div style={{ textAlign: 'center', padding: '0 10%', paddingBottom: '5%' }}>
      Try saying: <br />
      Add {isIncome ? 'Income ' : 'Expense '}
      for {isIncome ? 'Rs.100 ' : 'Rs.50 '}
      in Category {isIncome ? 'Business ' : 'House '}
      for {isIncome ? 'Monday ' : 'Friday '}
    </div>
  );
};
export default InfoCard;

```

### categories.js

```

const incomeColors = ['#123123', '#154731', '#165f40', '#16784f', '#14915f', '#10ac6e', '#0bc77e',
'#04e38d', '#00ff9d'];
const expenseColors = ['#b50d12', '#bf2f1f', '#c9452c', '#d3583a', '#dc6a48', '#e57c58', '#ee8d68',
'#f79d79', '#ffae8a', '#cc474b', '#f55b5f'];
export const incomeCategories = [
  { type: 'Business', amount: 0, color: incomeColors[0] },
  { type: 'Investments', amount: 0, color: incomeColors[1] },
  { type: 'Extra income', amount: 0, color: incomeColors[2] },
  { type: 'Deposits', amount: 0, color: incomeColors[3] },
  { type: 'Lottery', amount: 0, color: incomeColors[4] },
  { type: 'Gifts', amount: 0, color: incomeColors[5] },
  { type: 'Salary', amount: 0, color: incomeColors[6] },
  { type: 'Savings', amount: 0, color: incomeColors[7] },
  { type: 'Rental income', amount: 0, color: incomeColors[8] },
];
export const expenseCategories = [
  { type: 'Bills', amount: 0, color: expenseColors[0] },
  { type: 'Car', amount: 0, color: expenseColors[1] },
  { type: 'Clothes', amount: 0, color: expenseColors[2] },
  { type: 'Travel', amount: 0, color: expenseColors[3] },
  { type: 'Food', amount: 0, color: expenseColors[4] },
  { type: 'Shopping', amount: 0, color: expenseColors[5] },
  { type: 'House', amount: 0, color: expenseColors[6] },
  { type: 'Entertainment', amount: 0, color: expenseColors[7] },
  { type: 'Phone', amount: 0, color: expenseColors[8] },
  { type: 'Pets', amount: 0, color: expenseColors[9] },
];

```

```

    { type: 'Other', amount: 0, color: expenseColors[10] },
  ];
  export const resetCategories = () => {
    incomeCategories.forEach((c) => c.amount = 0);
    expenseCategories.forEach((c) => c.amount = 0);
  };

```

### **context.js**

```

import React, { useReducer, createContext } from 'react';
import contextReducer from './contextReducer';
const initialState = JSON.parse(localStorage.getItem('transaction')) || [];
export const ExpenseTrackerContext = createContext(initialState);
export const Provider = ({ children }) => {
  const [transactions, dispatch] = useReducer(contextReducer, initialState);
  const deleteTransaction = (id) => dispatch({ type: 'DELETE_TRANSACTION', payload: id });
  const addTransaction = (transaction) => dispatch({ type: 'ADD_TRANSACTION', payload: transaction });
  const balance = transactions.reduce((acc, currVal) => (currVal.type === 'Expense' ? acc - currVal.amount : acc + currVal.amount), 0);
  return (
    <ExpenseTrackerContext.Provider value={{
      deleteTransaction,
      addTransaction,
      transactions,
      balance,
    }}>
      {children}
    </ExpenseTrackerContext.Provider>
  );
};

```

### **contextReducer.js**

```

const contextReducer = (state, action) => {
  let transactions;
  switch (action.type) {
    case 'DELETE_TRANSACTION':
      transactions = state.filter((t) => t.id !== action.payload);
      localStorage.setItem('transaction', JSON.stringify(transactions));

```

```

    return transactions;
  case 'ADD_TRANSACTION':
    transactions = [action.payload, ...state];
    localStorage.setItem('transaction', JSON.stringify(transactions));
    return transactions;
  default:
    return state;
}
}
export default contextReducer;

```

## Dashboard.jsx

```

import { PushToTalkButton, PushToTalkButtonContainer } from '@speechly/react-ui';
import React from 'react';
import { Details, Main } from '../components';
import DetailsTrack from '../components/DetailsTrack/DetailsTrack';
import Navbar from '../components/Navbar/Navbar';
const Dashboard = () => {
  return (
    <div>
      <div style={{ display: 'flex', flexDirection: 'row' }}>
        { /*<Grid className={classes.grid} container spaicng={0} alignItems="center"
justifyContent="center" style={{ height: '100vh' }} >
          <Grid item xs={12} sm={4} className={classes.main} >
            <Main />
          </Grid>
          <Grid item xs={12} sm={3} className={classes.last} >
            <Details title="Income" />
            <Details title="Expense" />
          </Grid>
          <Grid item xs={12} sm={3} className={classes.desktop} >
            <Details title="Income" />
          </Grid>
          <Grid item xs={12} sm={3} className={classes.last} >
            <Details title="Expense" />
          </Grid>
        </Grid> */ }
      <div>
        <Main />
      </div>
    </div>
  );
}

```

```

    <div>
      <div>
        <Navbar />
      </div>
      <span>
        <Details title="Income" />
      </span>
      <span>
        <Details title="Expense" />
      </span>
    </div>
    <PushToTalkButtonContainer>
      <PushToTalkButton />
    </PushToTalkButtonContainer>
  </div>
  <div>
    <DetailsTrack title="Income" />
    <DetailsTrack title="Expense" />
  </div>
</div>
);
};
export default Dashboard;

```

## DashboardStyles.js

```

import { makeStyles } from '@material-ui/core/styles';
export default makeStyles((theme) => ({
  desktop: {
    [theme.breakpoints.up('sm')]: {
      display: 'none',
    },
  },
  mobile: {
    [theme.breakpoints.down('sm')]: {
      display: 'none',
    },
  },
  last: {
    [theme.breakpoints.down('sm')]: {
      marginBottom: theme.spacing(3),
    },
  },
}));

```

```

        paddingBottom: '200px',
      },
    },
    grid: {
      '& > *': {
        margin: theme.spacing(2),
      },
    },
  },
));

```

### **forminput.css**

```

.inputField {
  padding: 15px;
  margin: 10px 0px;
  border-radius: 10px;
  border: 1px solid rgb(122 103 0);
}
.formInput {
  display: flex;
  flex-direction: column;
  width: 280px;
}
.labelField {
  font-size: 12px;
  color: rgb(50, 50, 0);
}
.errorMessage {
  font-size: 10px;
  padding: 3px;
  color: red;
  display: none;
}
.inputField:invalid:focus {
  border: 1px solid red;
}
.inputField:invalid:focus ~ span {
  display: block;
}

```



## FormInput.jsx

```
import React, { useState } from 'react';
import './formInput.css';
const FormInput = (props) => {
  const [focused, setFocused] = useState(false);

  const { label, errorMessage, onChange, id, ...inputProps } = props;
  const handleFocus = (e) => {
    setFocused(true);
  };
  return (
    <div className="formInput">
      <label className="labelField">{label}</label>
      <input
        {...inputProps}
        className="inputField"
        onChange={onChange}
        onBlur={handleFocus}
        onFocus={() => inputProps.name === "confirmPassword" && setFocused(true)}
        focused={focused.toString()} />
      <span className="errorMessage">{errorMessage}</span>
    </div>
  );
};
export default FormInput;
```

## login.css

```
.login {
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100vh;
}
.loginForm {
  background-color: rgb(244 228 140);
  padding: 0px 60px;
  border-radius: 10px;
}
.loginTitle {
```

```

    color: rgb(139 120 2);
    text-align: center;
    margin-top: 30px;
}
.loginBtn {
    width: 100%;
    height: 50px;
    padding: 10px;
    border: none;
    background-color: rgb(139 120 2);
    color: white;
    border-radius: 10px;
    font-weight: 600;
    font-size: 18px;
    cursor: pointer;
    margin-top: 15px;
    margin-bottom: 15px;
}
.loginBtn:hover {
    background-color: rgb(246 235 172);
    color: rgb(139 120 2);
    border: 2px solid rgb(139 120 2);
}
.registerLink {
    text-decoration: none;
    margin-left: 5px;
    color: rgb(72 62 1);
    font-weight: 500;
}

```

## Login.jsx

```

import React, { useState } from 'react';
import { Link, useNavigate, } from 'react-router-dom';
import swal from 'sweetalert';
import FormInput from '../formInput/FormInput';
import './login.css';
const Login = () => {

    const navigate = useNavigate();
    const [values, setValues] = useState({

```

```

    username: "",
    password: "",
  });
  const { username, password } = values;
  const inputs = [
    {
      id: 1,
      name: "username",
      type: "text",
      placeholder: "Username",
      label: "Username",
      errorMessage: "Username should be 3-16 characters and shouldn't include any special
characters",
      required: true,
      pattern: "^[A-Za-z0-9]{3,16}$",
    },
    {
      id: 3,
      name: "password",
      type: "password",
      placeholder: "Password",
      label: "Password",
      errorMessage: "Password should be 8-20 characters and include at least 1 letter, 1 number
and 1 special character",
      required: true,
      pattern: "^(?=.*[0-9])(?=.*[A-Za-z])(?=.*[!@#$%^&*])[a-zA-Z0-9!@#$%^&*]{8,20}$",
    },
  ]
  const handleSubmit = (e) => {
    e.preventDefault();
    if (username === "aaaa" && password === "aaaa@1111") {
      swal("Successfull Login", "Welcome to Dashboard", "success");
      navigate(`/dashboard/${username}`);
    }
    else {
      swal("Wrong Credentials", "Please try again!", "error");
    }
  };
  const onChange = (e) => {
    setValues({ ...values, [e.target.name]: e.target.value });
  };

```

```

    };
    return (
      <div className="login">
        <form className="loginForm" onSubmit={handleSubmit}>
          <h1 className="loginTitle">Login</h1>
          {inputs.map((input) => (
            <FormInput
              key={input.id}
              {...input}
              value={values[input.name]} onChange={onChange}
            />
          ))}
          <button className="loginBtn" >Login</button>
        <div>
          <p style={{ fontSize: "14px", marginBottom: "30px" }}>New user? <Link
to="/register" className="registerLink">Click to Register</Link></p>
        </div>
      </form>
    </div>
  );
};
export default Login;

```

### **register.css**

```

.register {
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100vh;
}
.registerForm {
  background-color: rgb(244 228 140);
  padding: 0px 60px;
  border-radius: 10px;
}
.registerTitle {
  color: rgb(139 120 2);
  text-align: center;
  margin-top: 30px;
}

```

```

.registerBtn {
  width: 100%;
  height: 50px;
  padding: 10px;
  border: none;
  background-color: rgb(139 120 2);
  color: white;
  border-radius: 10px;
  font-weight: bold;
  font-size: 18px;
  cursor: pointer;
  margin-top: 15px;
  margin-bottom: 15px;
}
.registerBtn:hover {
  background-color: rgb(246 235 172);
  color: rgb(139 120 2);
  border: 2px solid rgb(139 120 2);
}
.loginLink {
  text-decoration: none;
  margin-left: 5px;
  color: rgb(72 62 1);
  font-weight: 500;
}

```

### **Register.jsx**

```

import React, { useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import FormInput from '../formInput/FormInput';
import './register.css';
const Register = () => {
  const navigate = useNavigate();
  const [values, setValues] = useState({
    username: '',
    email: '',
    password: '',
    confirmPassword: '',
  });
  const { username } = values;

```

```

const inputs = [
  {
    id: 1,
    name: "username",
    type: "text",
    placeholder: "Username",
    label: "Username",
    errorMessage: "Username should be 3-16 characters and shouldn't include any special
characters",
    required: true,
    pattern: "^[A-Za-z0-9]{3,16}$",
  },
  {
    id: 2,
    name: "email",
    type: "email",
    placeholder: "Email",
    label: "Email",
    errorMessage: "It should be a valid email address",
    required: true,
  },
  {
    id: 3,
    name: "password",
    type: "password",
    placeholder: "Password",
    label: "Password",
    errorMessage: "Password should be 8-20 characters and include at least 1 letter, 1 number
and 1 special character",
    required: true,
    pattern: "^(?=.*[0-9])(?=.*[A-Za-z])(?=.*[!@#$%^&*])[a-zA-Z0-9!@#$%^&*]{8,20}$",
  },
  {
    id: 4,
    name: "confirmPassword",
    type: "password",
    placeholder: "Confirm Password",
    label: "Confirm Password",
    errorMessage: "Passwords don't match",
    required: true,
  },

```

```

        pattern: values.password,
      },
    ]
    const handleSubmit = (e) => {
      e.preventDefault();
      navigate(`/dashboard/${username}`);
    };
    const onChange = (e) => {
      setValues({ ...values, [e.target.name]: e.target.value });
    };
    return (
      <div className="register">
        <form className="registerForm" onSubmit={handleSubmit}>
          <h1 className="registerTitle">Register</h1>
          {inputs.map((input) => (
            <FormInput
              key={input.id}
              {...input}
              value={values[input.name]} onChange={onChange}
            />
          ))}
          <button className="registerBtn">Submit</button>
        </div>
        <p style={{ fontSize: "14px", marginBottom: "30px" }}>Already have an account?
        <Link to="/login" className="loginLink">Click to Login</Link></p>
      </div>
    </form>
  </div>
);
};
export default Register;

```

### home.css

```

.homeContainer {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  height: 100vh;
}

```

```
.homeBtn {
  width: 280px;
  background-color: deepskyblue;
  border-radius: 10px;
  padding: 20px;
  margin: 15px;
  cursor: pointer;
  color: white;
  border: ridge white;
}
```

### **Home.jsx**

```
import React from 'react';
import { Link } from 'react-router-dom';
import './home.css';
const Home = () => {
  return (
    <div className="homeContainer">
      <Link to="/login"><button className="homeBtn">Login</button></Link>
      <Link to="/register"><button className="homeBtn">Register</button></Link>
    </div>
  );
};
export default Home;
```

### **formatDate.js**

```
const formatDate = (date) => {
  const d = new Date(date);
  let month = `${d.getMonth() + 1}`;
  let day = `${d.getDate()}`;
  const year = d.getFullYear();
  if (month.length < 2) {
    month = `0${month}`;
  }
  if (day.length < 2) {
    day = `0${day}`;
  }
  return [year, month, day].join('-');
}
export default formatDate;
```



## App.css

```
.App {
  text-align: center;
}
.App-logo {
  height: 40vmin;
  pointer-events: none;
}
@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}
.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}
.App-link {
  color: #61dafb;
}
@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

## App.js

```
import React from 'react';
import { BrowserRouter, Routes, Route, } from "react-router-dom";
```

```

import Dashboard from './routes/dashboard/Dashboard';
import Home from './routes/home/Home';
import Login from './routes/forms/login/Login';
import Register from './routes/forms/register/Register';
const App = () => {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route exact path="/login" element={<Login />} />
        <Route exact path="/register" element={<Register />} />
        <Route exact path="/dashboard/:uname" element={<Dashboard />} />
      </Routes>
    </BrowserRouter>
  );
};
export default App;

```

### **useTransactions.js**

```

import { useContext } from "react"
import { expenseCategories, incomeCategories, resetCategories } from "./constants/categories";
import { ExpenseTrackerContext } from "./context/context"
const useTransactions = (title) => {
  resetCategories();
  const { transactions } = useContext(ExpenseTrackerContext);
  const transactionsPerType = transactions.filter((t) => t.type === title);
  const total = transactionsPerType.reduce((acc, currVal) => acc += currVal.amount, 0);
  const categories = title === 'Income' ? incomeCategories : expenseCategories;
  var month = "";
  var monthIncomeTotal = [
    {m:"January", amount:0},
    {m:"February", amount:0},
    {m:"March", amount:0},
    {m:"April", amount:0},
    {m:"May", amount:0},
    {m:"June", amount:0},
    {m:"July", amount:0},
    {m:"August", amount:0},
    {m:"September", amount:0},
    {m:"October", amount:0},
  ]

```

```

    {m:"November", amount:0},
    {m:"December", amount:0},
];
var monthExpenseTotal = [
    {m:"January", amount:0},
    {m:"February", amount:0},
    {m:"March", amount:0},
    {m:"April", amount:0},
    {m:"May", amount:0},
    {m:"June", amount:0},
    {m:"July", amount:0},
    {m:"August", amount:0},
    {m:"September", amount:0},
    {m:"October", amount:0},
    {m:"November", amount:0},
    {m:"December", amount:0},
];
// transactionsPerType.forEach((t) => {
//   console.log(t.amount, t.category, t.type);
// })
transactionsPerType.forEach((t) => {
  month = t.date.slice(5, 7);
  if (t.type === 'Income') {
    switch(month) {
      case '01': monthIncomeTotal[0].amount += t.amount;break;
      case '02': monthIncomeTotal[1].amount += t.amount;break;
      case '03': monthIncomeTotal[2].amount += t.amount;break;
      case '04': monthIncomeTotal[3].amount += t.amount;break;
      case '05': monthIncomeTotal[4].amount += t.amount;break;
      case '06': monthIncomeTotal[5].amount += t.amount;break;
      case '07': monthIncomeTotal[6].amount += t.amount;break;
      case '08': monthIncomeTotal[7].amount += t.amount;break;
      case '09': monthIncomeTotal[8].amount += t.amount;break;
      case '10': monthIncomeTotal[9].amount += t.amount;break;
      case '11': monthIncomeTotal[10].amount += t.amount;break;
      case '12': monthIncomeTotal[11].amount += t.amount;break;
      default: break;
    }
  }
  else if(t.type === 'Expense') {
    switch(month) {

```

```

        case '01': monthExpenseTotal[0].amount += t.amount;break;
        case '02': monthExpenseTotal[1].amount += t.amount;break;
        case '03': monthExpenseTotal[2].amount += t.amount;break;
        case '04': monthExpenseTotal[3].amount += t.amount;break;
        case '05': monthExpenseTotal[4].amount += t.amount;break;
        case '06': monthExpenseTotal[5].amount += t.amount;break;
        case '07': monthExpenseTotal[6].amount += t.amount;break;
        case '08': monthExpenseTotal[7].amount += t.amount;break;
        case '09': monthExpenseTotal[8].amount += t.amount;break;
        case '10': monthExpenseTotal[9].amount += t.amount;break;
        case '11': monthExpenseTotal[10].amount += t.amount;break;
        case '12': monthExpenseTotal[11].amount += t.amount;break;
        default: break;
    }
}
const category = categories.find((c) => c.type === t.category);
if (category) category.amount += t.amount;
});
const filteredCategories = categories.filter((c) => c.amount > 0);
const chartData = {
  datasets: [{
    data: filteredCategories.map((c) => c.amount),
    backgroundColor: filteredCategories.map((c) => c.color),
  }],
  labels: filteredCategories.map((c) => c.type)
};
const chartDataIncome = {
  datasets: [
    {
      label: 'Income',
      data: monthIncomeTotal.map((m) => m.amount),
      borderColor: '#165f40',
      backgroundColor: '#0bc77e',
      tension: 0.1,
    },
  ],
  labels: monthIncomeTotal.map((m) => m.m)
};
const chartDataExpense = {
  datasets: [
    {

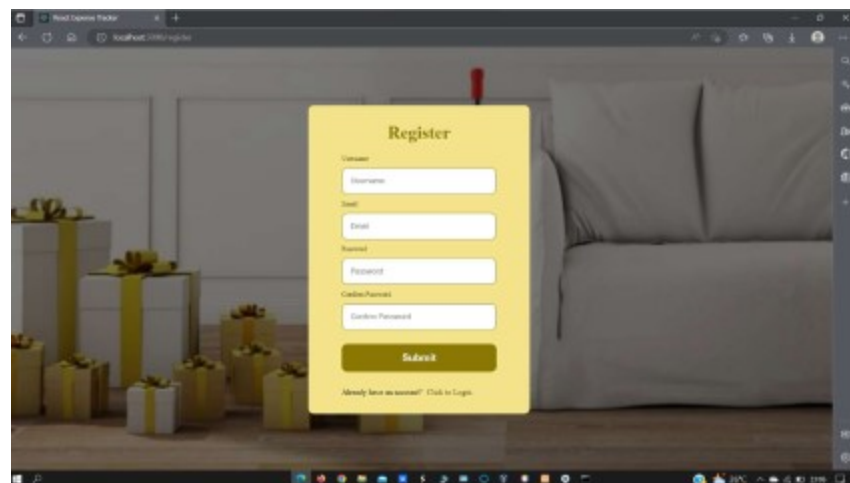
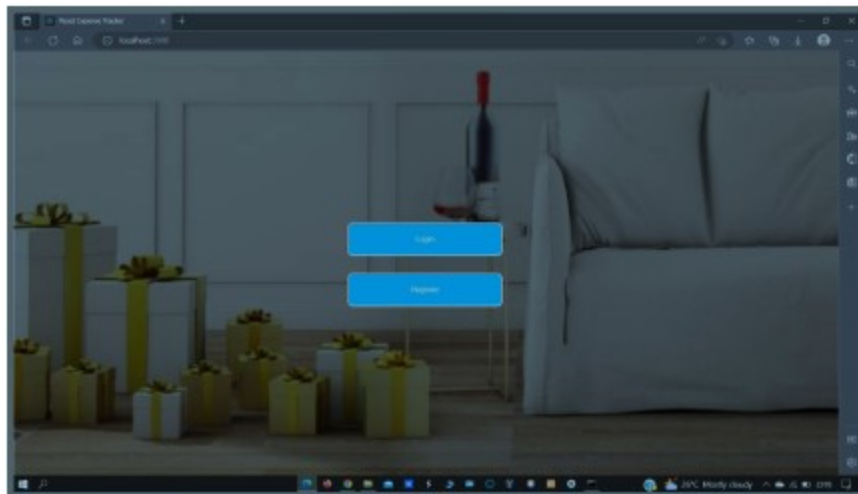
```

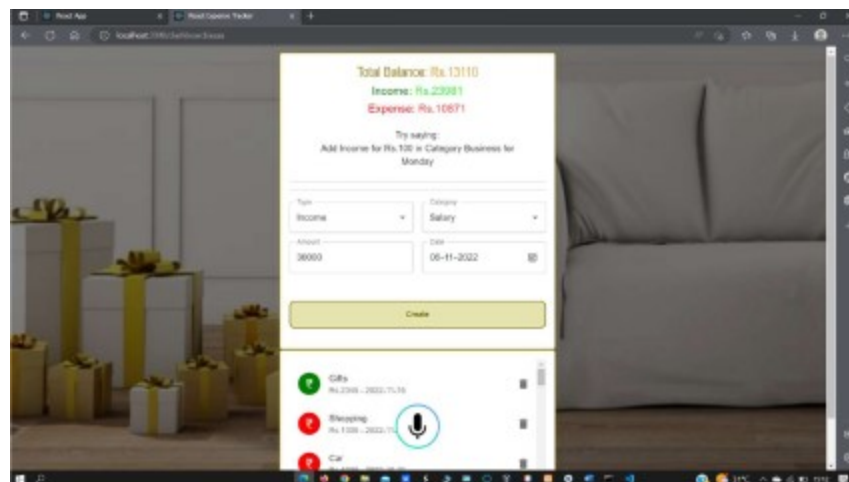
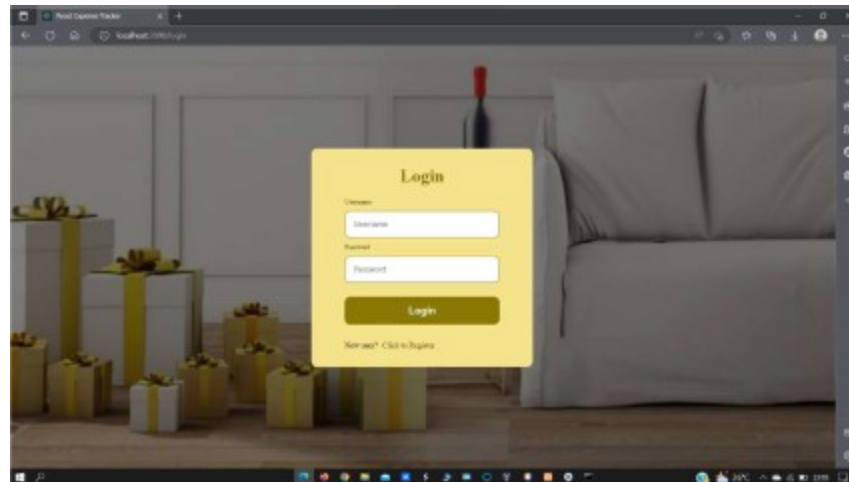
```

    label: 'Expense',
    data: monthExpenseTotal.map((m) => m.amount),
    borderColor: '#b50d12',
    backgroundColor: '#e57c58',
    tension: 0.1,
  },
],
labels: monthExpenseTotal.map((m) => m.m)
};
return { total, chartData, chartDataIncome, chartDataExpense };
};
export default useTransactions;

```

## OUTPUTS:







### 13.2. GitHub & Project Demo Link

**GitHub Link :** <https://github.com/IBM-EPBL/IBM-Project-1387-1658386550>

**Project Demo Link :** <https://youtu.be/kB8dApKmNhw>