Personal Expense Tracker Application

NALAIYATHIRAN PROJECT BASED LEARNING

ON

PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP

A PROJECT REPORT

SEENIVASAN S 19106103

SRIRAM V M 19106115

SURESH KRISHNA M 19106121

THIRUKUMARAN T 19106128

VINITHRAJ K 19106134

BACHELOR OF ENGINEERING

IN

ELECTRONIC AND COMMUNICATION ENGINEERING

HINDUSTHAN COLLEGE OF ENGINEERING AND TECHOLOGY Approved by AICTE, New Delhi, Accredited with ‘A’ Grade by NAAC **(An Autonomous Institution, Affiliated to Anna University, Chennai)**

COIMBATORE-641 032

Hindusthan College of Engineering And Technology Approved by AICTE, New Delhi, Accredited with ‘A’ Grade by NAAC **(An Autonomous Institution, Affiliated to Anna University, Chennai)**
Valley Campus, Pollachi Highway, Coimbatore – 641 032

INTERNAL MENTOR

Dr. P. K. POONGUZHALI

Associate Professor

Department of Electronics and Communication Engineering

Hindusthan College of Engineering and Technology,

Coimbatore - 641 032

INDUSTRY MENTOR

KHUSBOO
IBM

Project Report Format

1.**INTRODUCTION**

1.1 Project Overview:

A web programme called "Expense Tracker" enables users to manage and keep track of both personal and professional costs. The users of this application can keep digital diaries. It will record a user's earnings and outgoings each day. With the aid of the internet, the user will be able to rapidly input his or her expenses and can check them whenever and wherever. Without putting his or her information at danger and effectively protecting his or her privacy, he or she can quickly import transactions from his or her mobile wallets. He is able to see the precise time that he has been using a particular product. The app will compare spending on a monthly and annual basis and inform the user of the area spent.

1.2 Purpose:

Expense tracker is an android based application. This application allows the user to maintain a computerized diary. Expense tracker application which will keep a track of Expenses of a user on a day-to-day basis. This application keeps a record of your expenses and also will give you a category wise distribution of your expenses. With the help of this application user can track their daily/weekly/monthly expenses. This application will also have a feature which will help you stay on budget because you know your expenses. Expense tracker application will generate report at the end of month to show Expense via a graphical representation. We also have added a special feature which will distributes your expenses in different categories suitable for the user. An expense history will also be provided in application.

2.LITERATURE SURVEY

2.1 Reference:

Loc Nguyen Hoang Vinh, Van Hoan Dinh, The Impact of Credit onEconomic Growth in Vietnam: A Comparison of TraditionalMethods and the Bayes Method, Prediction and Causality inEconometrics and Related Topics, 10.1007/978-3- 030-77094-5_23, (276-292), (2022).

Miguel Ángel Tinoco-Zermeño, Víctor Hugo Torres-Preciado,Francisco Venegas Martínez, Inflation and Bank Credit,Investigación Administrativa, 0.35426/IAv51n129.02, 51-1, (1-21), (2022).

Miguel Rodriguez Gonzalez, Christoph Wegener, Tobias Basse,Re-investigating the insurance-growth nexus using common factors,Finance Research Letters, 10.1016/j.frl.2021.102231, 46, (102231),(2022).

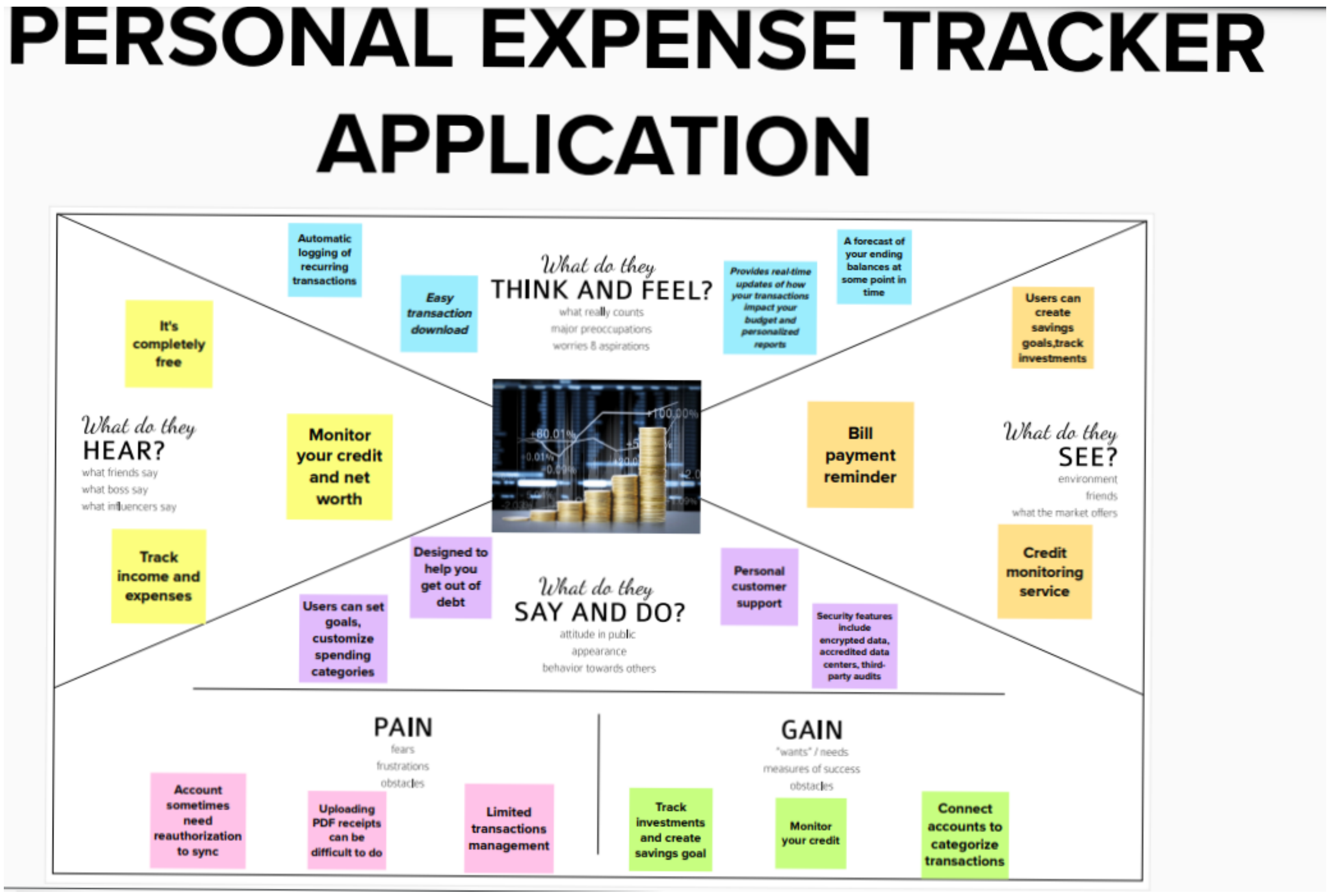
Cornelia Pop, Bucharest Stock Exchange development between1995 and 2020. From frontier to secondary emerging market, StudiaUniversitatis Babeș-Bolyai Negotia, 10.24193/subbnegotia.2022.1.04, 67, 1, (71-112), (2022).

2.3 **Problem Statement Definition:**

At the instant, there is no as such complete solution present easily or we should say free of cost which enables a person to keep a track of its daily expenditure easily. To do so a person has 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. 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. As the name itself suggests, this project is an attempt to manage our daily expenses in a more efficient and manageable way. The system attempts to free the user with as much as possible the burden of manual calculation and to keep the track of the expenditure.

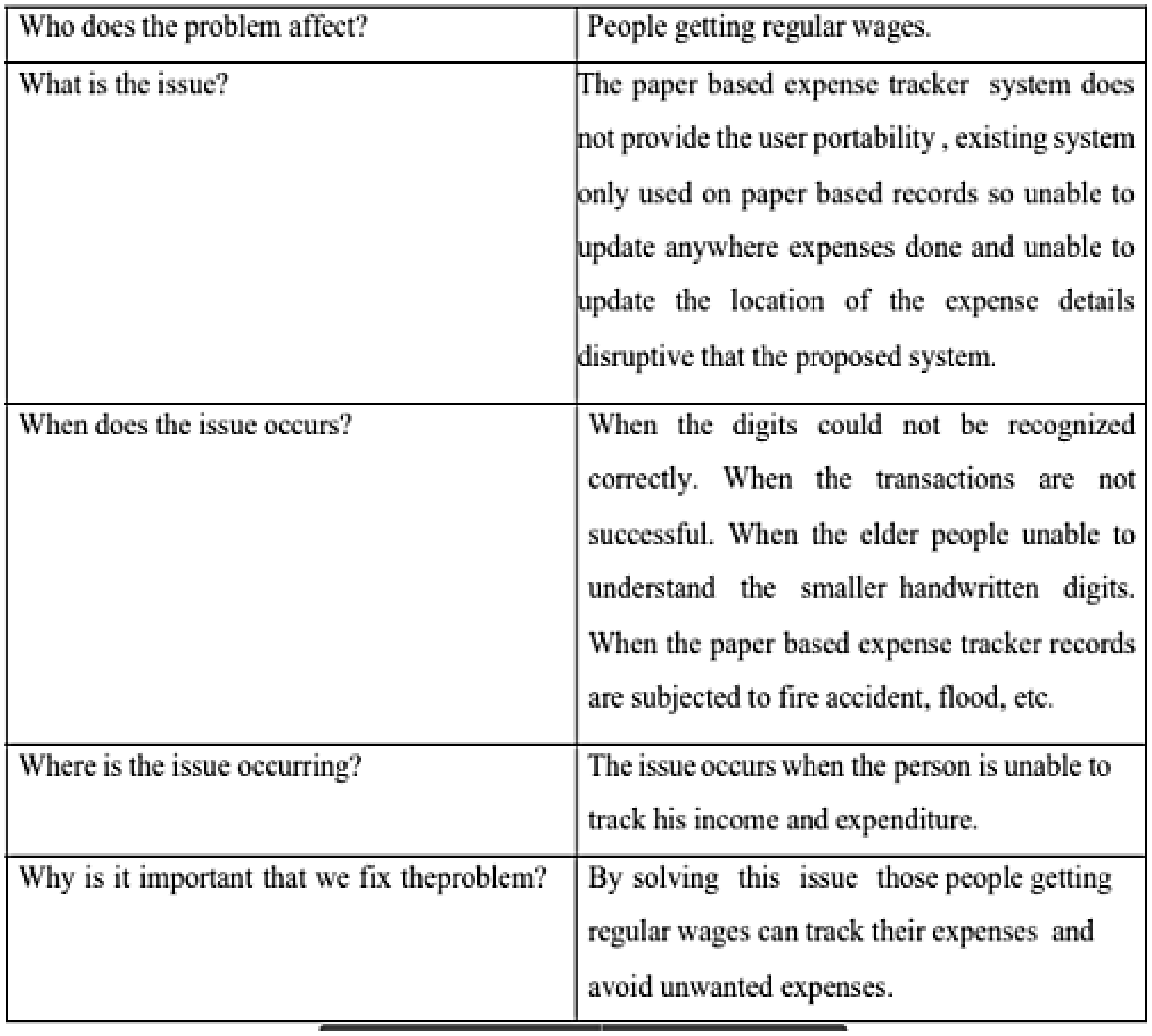
3.IDEATION & PROPOSED SOLUTION:

3.1 Empathy Map Canvas:


PERSONAL EXPENSE TRACKER APPLICATION
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Automatic logging of recurring transactions
Easy transaction download
Provides real-time updates of how your transactions impact your budget and personalized reports
A forecast of your ending balances at some point in time
What do they HEAR?
what friends say
what boss say
what influencers say
It's completely free
Monitor your credit and net worth
Track income and expenses
What do they SEE?
environment
friends
what the market offers
Users can create savings goals,track investments
Bill payment reminder
Credit monitoring service
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
Designed to help you get out of debt
Users can set goals, customize spending categories
Personal customer support
Security features include encrypted data, accredited data centers, third-party audits
PAIN
fears
frustrations
obstacles
Account sometimes need reauthorization to sync
Uploading PDF receipts can be difficult to do
Limited transactions management
GAIN
"wants" / needs
measures of success
obstacles
Track investments and create savings goal
Monitor your credit
Connect accounts to categorize transactions

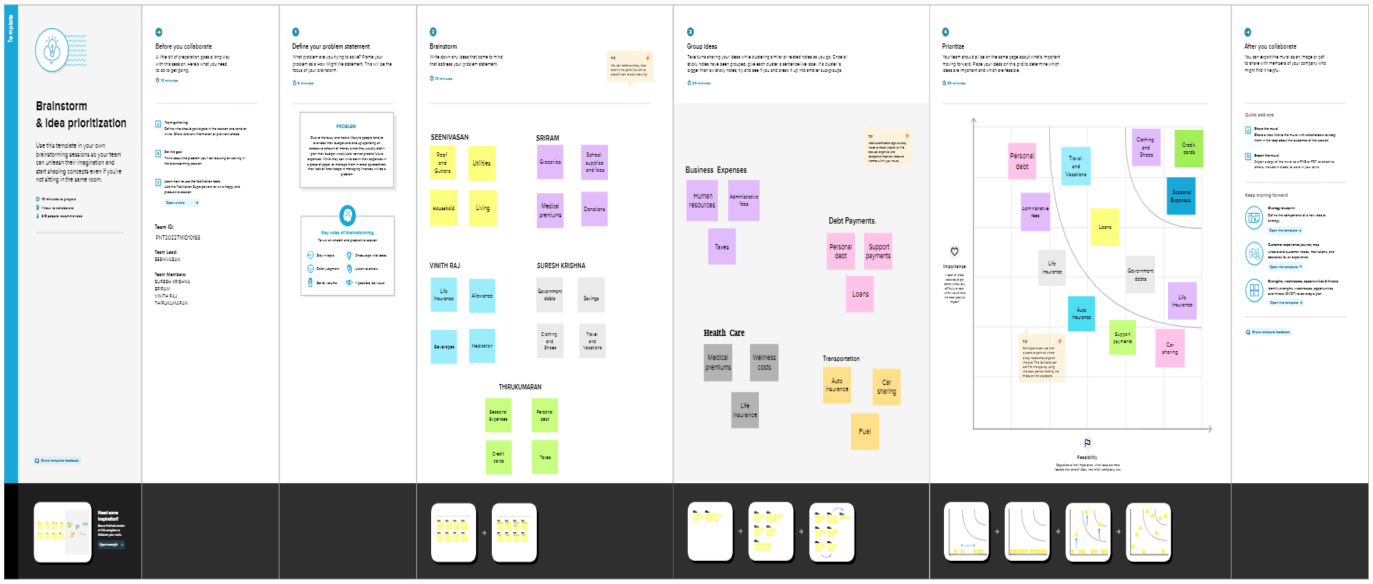
3.2 Ideation & Brainstorming:

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 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 2 constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.



Who does the problem affect?	People getting regular wages.
What is the issue?	The paper based expense tracker system does not provide the user portability , existing system only used on paper based records so unable to update anywhere expenses done and unable to update the location of the expense details disruptive that the proposed system.
When does the issue occurs?	When the digits could not be recognized correctly. When the transactions are not successful. When the elder people unable to understand the smaller handwritten digits. When the paper based expense tracker records are subjected to fire accident, flood, etc.
Where is the issue occurring?	The issue occurs when the person is unable to track his income and expenditure.
Why is it important that we fix theproblem?	By solving this issue those people getting regular wages can track their expenses and avoid unwanted expenses.

PERSONAL EXPENSE TRACKER APPLICATION

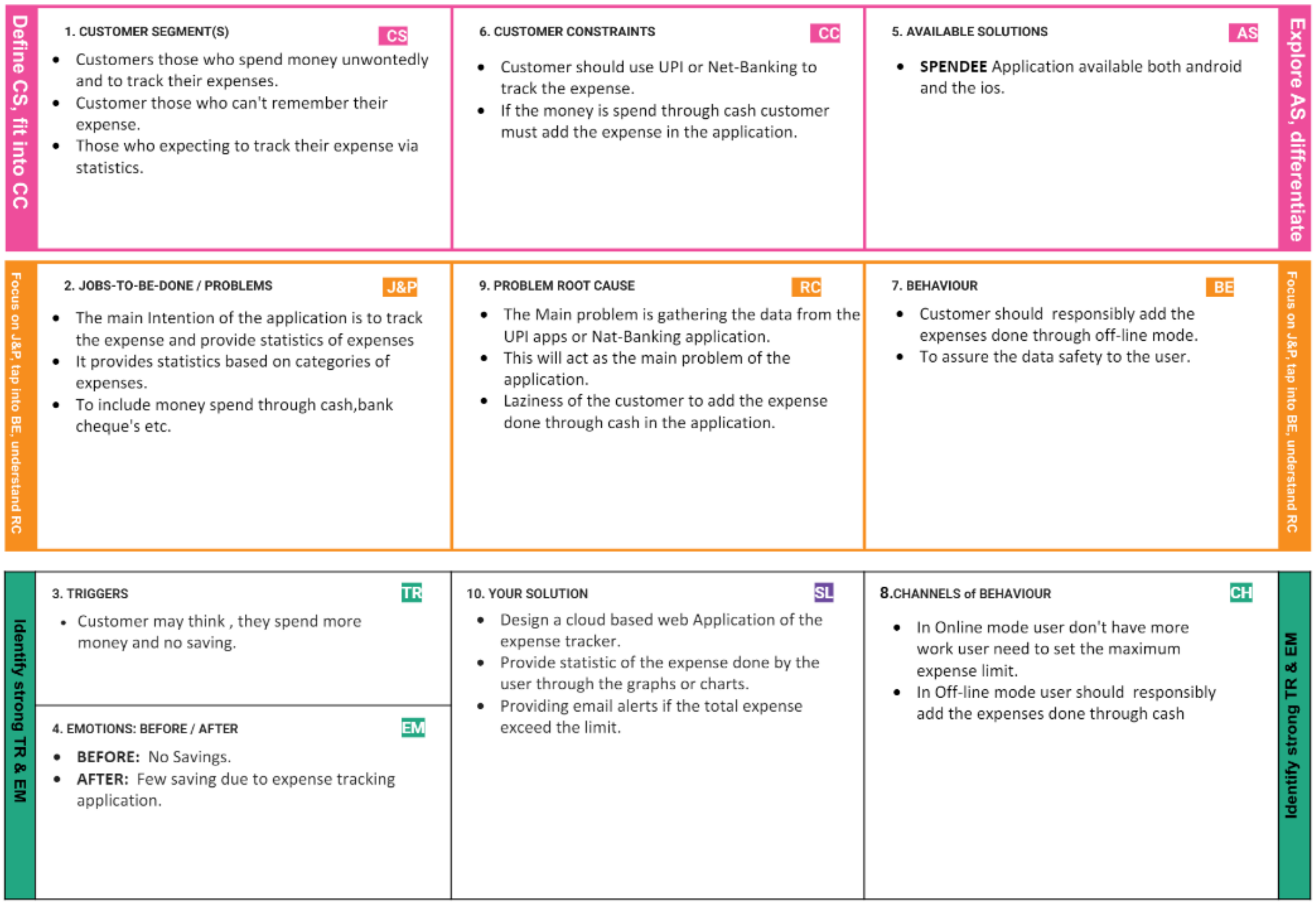


3.3 **Proposed Solution:**

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	There are many budgeting tools online, but not all of them are effective in assisting users in actually creating and adhering to a budget. The ongoing maintenance, the consolidation of all user financial accounts and activity into a single dashboard are some of the negatives. However, a lot of this current software includes convoluted features that are difficult to use. The major problem is to take count paper receipts and calculate the expense statistics.

2.	Idea / Solution description	People tend to neglect their budgets due to their busy and chaotic lifestyles, which results in them spending more than they planned. Future costs cannot be foreseen by the user. Their carelessness with money management will be a concern even though they can record their expense.
3.	Novelty / Uniqueness	Including all the expense including money spend using cash, Bank-cheques, etc. This application keeps track of all of your spending. To enter your expense, simply click. To decrease human error, prevent data loss, and expedite settlements. In this program to display the pie chart or graph lines.
4.	Social Impact / Customer Satisfaction	One can keep track of their own costs and create a monthly or annual budget with this tool. The application will display the statistics and sent alert message, if your spending exceeds the specified limit.
5.	Business Model (Revenue Model)	The subscription/premium to access extra features of this application can be used by business people, and also adding advertisement to generate revenue.
6.	Scalability of the Solution	Using IBM-cloud statistics can be shown to the user with high scalability, and with high accuracy.

3.4 **Proposed Solution Fit:**



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

- Customers those who spend money unwontedly and to track their expenses.
- Customer those who can't remember their expense.
- Those who expecting to track their expense via statistics.

6. CUSTOMER CONSTRAINTS CC

- Customer should use UPI or Net-Banking to track the expense.
- If the money is spend through cash customer must add the expense in the application.

5. AVAILABLE SOLUTIONS AS

- **SPENDEE** Application available both android and the ios.

Explore AS, differentiate

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

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

- The main Intention of the application is to track the expense and provide statistics of expenses
- It provides statistics based on categories of expenses.
- To include money spend through cash,bank cheque's etc.

9. PROBLEM ROOT CAUSE RC

- The Main problem is gathering the data from the UPI apps or Nat-Banking application.
- This will act as the main problem of the application.
- Laziness of the customer to add the expense done through cash in the application.

7. BEHAVIOUR BE

- Customer should responsibly add the expenses done through off-line mode.
- To assure the data safety to the user.

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

Identify strong TR & EM

3. TRIGGERS TR

- Customer may think , they spend more money and no saving.

4. EMOTIONS: BEFORE / AFTER EM

- **BEFORE:** No Savings.
- **AFTER:** Few saving due to expense tracking application.

10. YOUR SOLUTION SL

- Design a cloud based web Application of the expense tracker.
- Provide statistic of the expense done by the user through the graphs or charts.
- Providing email alerts if the total expense exceed the limit.

8.CHANNELS of BEHAVIOUR CH

- In Online mode user don't have more work user need to set the maximum expense limit.
- In Off-line mode user should responsibly add the expenses done through cash

Identify strong TR & EM

4. REQUIREMENT ANALYSIS:

4.1 Functional Requirement:

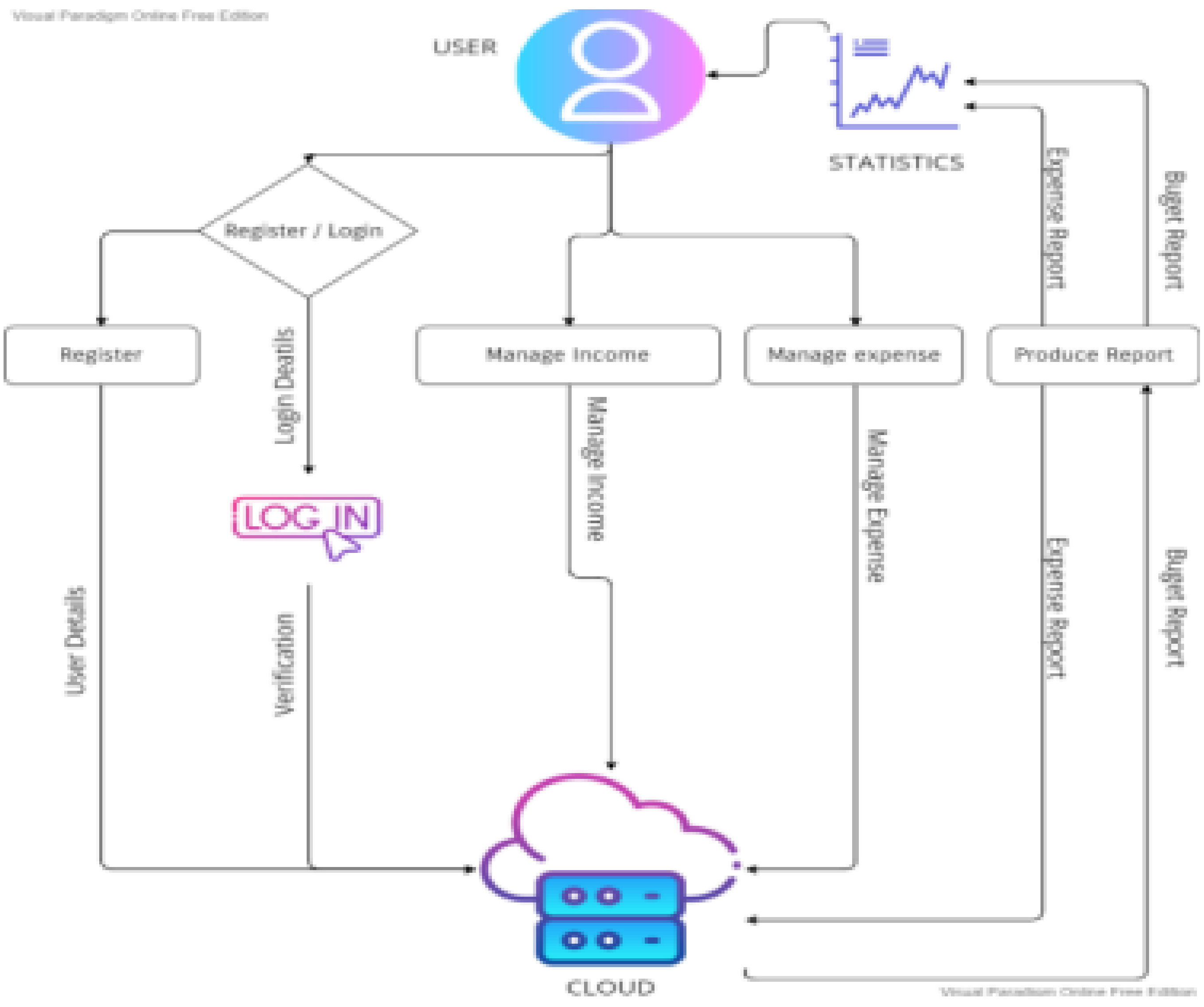
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Data	Data gathered in the application server is saved in the high security cloud server.
FR-4	Alert Notification	Alert messages through the Email or SMS.
FR-5	User Monthly Budget Plan	Setting Monthly budget to manage their expenses.
FR-6	Cloud Data Storage	To save the user valuable data high security cloud storage are used (AWS, IBM, GOOGLE, etc)

4.2 **Non-Functional Requirements:**

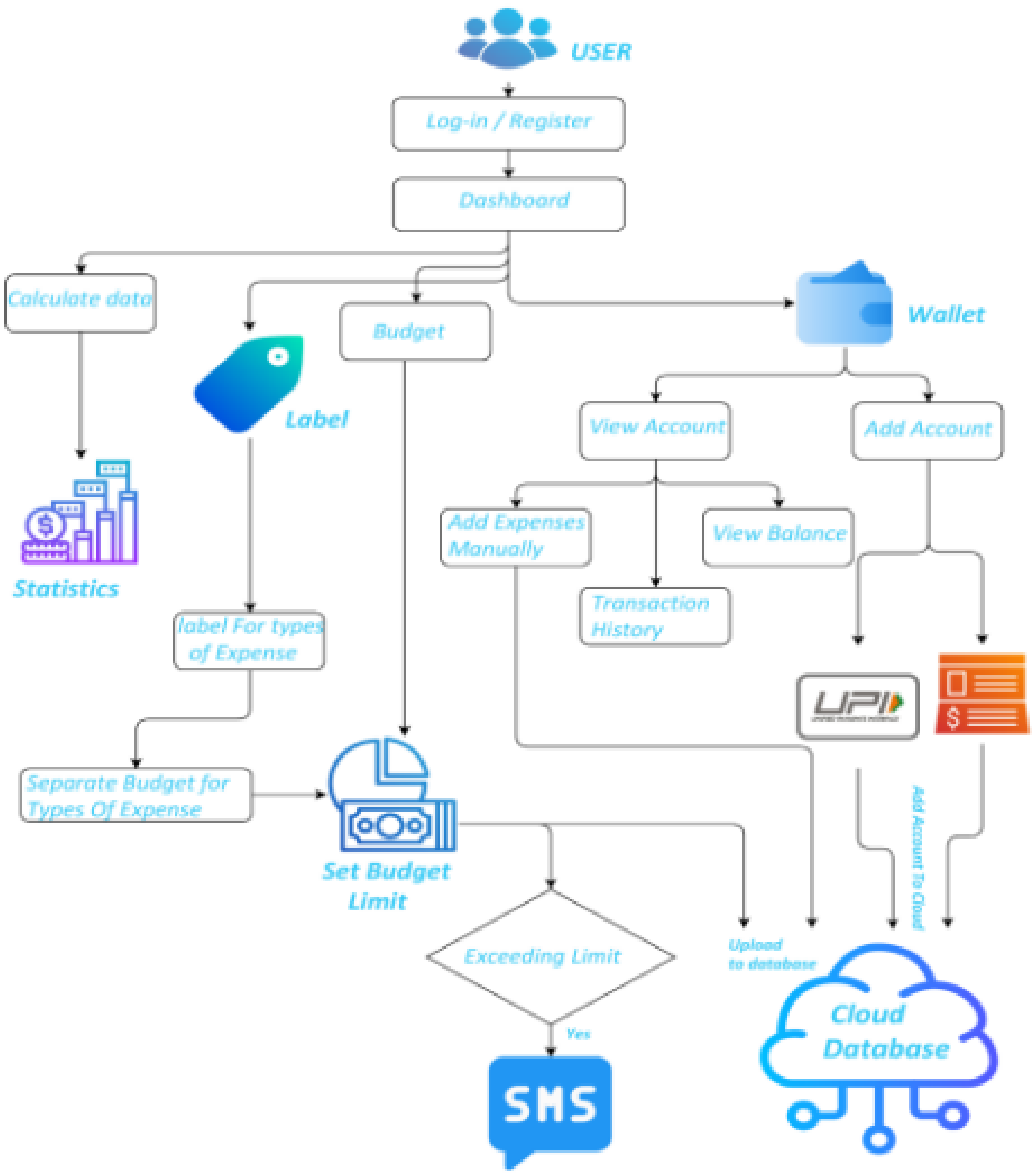
FR No.	**Non-Functional Requirement**	**Description**
NFR 1	**Usability**	Effectiveness, efficiency, and overall experience of the user interacting with our application should be maximised.
NFR 2	**Security**	Authentication, authorization, encryption of the data must be done in the application.
NFR 3	**Reliability**	Probability of error in the operations in a specified environment for a specified time should be minimised.
NFR 4	**Performance**	How the application is functioning accurately and effectively the application is to the end-users.
NFR 5	**Availability**	Using Cloud Storage and database, 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.

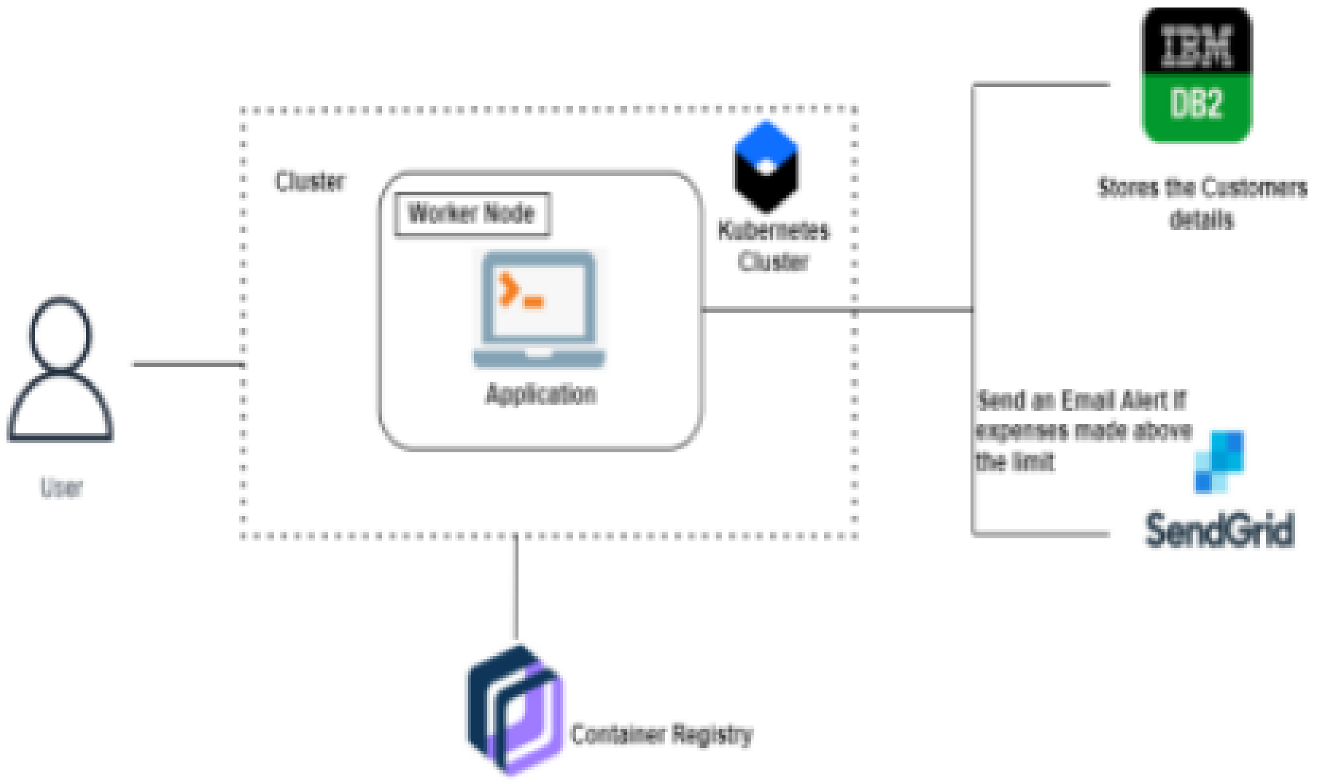
5.PROJECT DESIGN:

5.1 **Data Flow Diagrams:**



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)	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

		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	Sprint-1
		USN-2	As a user, I can register for the application through Facebook	I can register &access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard		As a user, I can access my detail, manage the expense, add		High	Sprint-1

			budget, expense report from the app etc..			
Customer (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 log into the applicatio n by entering email & password		High	Sprint-1
	Dashboard	USN-3	As a user, I can access my detail, manage the expense, add budget, expense report from the app etc..		High	Sprint-1
Customer Care Executive	Email or Customer Care no		As a user, I can contact the service administration for the support.	I can solve the Issue.	High	Sprint-3

6.PROJECT PLANNING & SCHEDULING:

6.1 Sprint Planning & Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Point	Priority	Team Members
Sprint 1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Seenivasan
		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Sriram
	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Suresh Krishna
	Dashboard	USN-4	Logging in takes to the dashboard for the logged user.	2	High	Thirukumaran
Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only						
Sprint 2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Vinithraj

	Charts	USN-2	Creating various graphs and statistics of customer’s data	1	Medium	Seenivasan
	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Sriram
		USN-4	Making dashboard interactive with JS	2	High	Suresh Krishna
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Thirukumaran
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user’s query	1	Medium	Vinithraj
	SendGrid	USN-3	Using SendGrid to send mail to the user about their expenses	1	Low	Seenivasan
		USN-4	Integrating both frontend and backend	2	High	Sriram
Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only						

Sprint-4	Docker	USN-1	Creating image of website using docker/	2	High	Suresh Krishna
	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Thirukumaran
	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Vinithraj
	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Seenivasan

6.2 Sprint Delivery Schedule:

Sprint	Total Story Points	Duration	Sprint Start Date Sprint End Date (Planned)	Story Points	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

7.CODING & SOLUTIONING:

7.1 Feature 1:

Step 1: Create style.css file with the following code.

```
  Body
{

 margin: 0;

 background-color: #EBEBEB;

 font-family: 'Roboto', sans-serif;

 }

.index-home {

 display: flex;

 flex-direction: column;

 align-items: center;

 }

.index-bg {

 background-color: #0a7206;

 height: 40vh;

 width: 100%;

 position: fixed;

 top: 0;

 left: 0;

 right: 0;

 z-index: -1;

 }

.index-content {
 width: 100%;

 display: flex;
flex-direction: column;

 align-items: center;

 margin-top: 4rem;

 }

.index-title {

 font-size: 3rem;

 color: white;

 }

.auth-box {

 display: flex;

 flex-direction: column;

 align-items: center;

 justify-content: center;

 background-color: white;

 padding: 2rem;

 border-radius: 1.5rem;

box-shadow: 0 2px 15px 1px rgba(0, 0, 0,

0.25); }

.auth-input {

 /*margin: 1rem 0;*/

 height: 4rem;

 width: 100%;

 display: flex;
 justify-content: center;

 align-items: center;

 }
input[type=text],

input[type=password], select {  box-

sizing: border-box;

 padding: 0.9rem 0.5rem 0.9rem

0.5rem;  display: inline-block;

 border: 2px solid #F5F5F5;

 border-radius: 0.5rem;

 background-color: #F5F5F5;

 font-size: 1rem;

 width: 17rem;

 }

.btn-submit {

 width: 100%;

 align-self: center;

 color: white;

 background-color: #0a7206;

 border: none;

 margin: 1.3rem 0 1rem 0;

 padding: 0.7rem;

 border-radius: 0.5rem;

 font-size: 1rem;

 cursor: pointer;
 }
.btn-submit:hover {
 background-color: #0bc106;
 }
.auth-shift-link {
color: #0a7206;
 font-weight: bolder;
 cursor: pointer;
 }
input:focus {
 outline: none;
 border: 2px solid #0a7206;
 }
input:hover {
 border: 2px solid #0a7206;
 }
.auth-title {
 font-size: 1.7rem;
 font-weight: bold;
 margin-bottom: 1rem;
 }
```

Step 2: Create home.css file with the following code.

7.2 Feature 2:

Step 1: Create a webpage using index.html

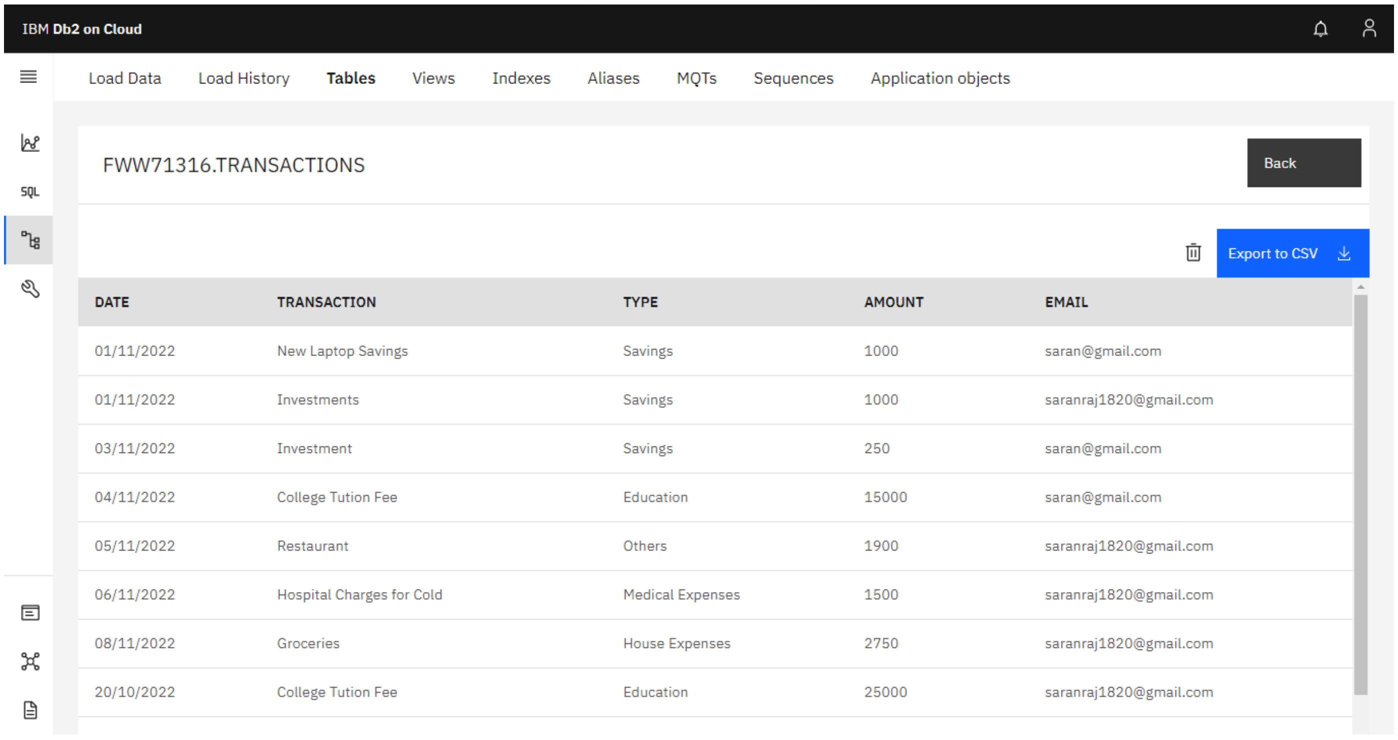
```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <link rel="stylesheet" href="../static/home.css">
<script
src="https://kit.fontawesome.com/514548070d.js"
crossorigin="anonymous"></script>
 {# <script src="../static/script.js"></script>#}
<title>Personal Expense Tracker</title>
</head>
<body>
<div class="track-home">
 <div class="track-left">
 <div class="track-avatar">
 <img class="avatar" src="../static/avatar.png" alt="avatar">
<span>Ivan Chris</span>
 </div>
 <hr class="solid-line">
 <div class="track-items">
 <div>
 <i class="fa-solid fa-folder"></i>
 <span>Dashboard</span>
 </div>
 <div>
 <i class="fa-solid fa-chart-simple"></i>
 <span>Statistics</span>
 </div>
 <div>
 <i class="fa-solid fa-gear"></i>
 <span>Settings</span>
 </div>
 <div>
 <i class="fa-solid fa-right-from-bracket"></i>
<span>Logout</span>
```

```
 </div>
 </div>
 </div>

<div class="track-right">
 <div class="track-right__top">
 <div class="track-right__top__types">
 <i class="fa-solid fa-bed fa-2x"></i>
 <span class="track-type__amount">₹10,300</span>
<span>Medical Expenses</span>
 </div>
 <div class="track-right__top__types">
 <i class="fa-solid fa-house fa-2x"></i>
 <span class="track-type__amount">₹25,700</span>
 <span>House Expenses</span>
</div>
 <div class="track-right__top__types">
 <i class="fa-solid fa-graduation-cap fa-2x"></i>  <span
class="track-type__amount">₹17,000</span>
<span>Educational Loan</span>
 </div>
 <div class="track-right__top__types">
 <i class="fa-solid fa-piggy-bank fa-2x"></i>
 <span class="track-type__amount">₹15,000</span>
<span>Savings</span>
 </div>
 <div class="track-right__top__types">
 <i class="fa-solid fa-money-check-dollar fa-2x"></i>
<span class="track-type__amount">₹9,732</span>
<span>Others</span>
 </div>
 </div>
 <div class="track-right__bottom">
 <div class="track-transactions">
 <h3>Recent Transactions</h3>
 <div class="transact-header">
 <span class="date-header"><b>Date</b></span>  <span
```

```
class="transaction-header"><b>Transaction</b></span>  <span
class="type-header"><b>Type</b></span>  <span class="amount-
header"><b>Amount</b></span>  </div>
 <div class="total-items">
 {% for i in range(5) %}
 <div class="transact-item">
 <span class="date-header">08/11/2022</span>  <span
class="transaction-header">Electricity Bill</span>  <div
class="type-header">House Expenses</div>  <span
class="amount-header">₹150</span>  </div>
 {% endfor %}
 </div>

 </div>
 <div class="track-add">
 <h3>Add Expenditure</h3>
 <form action="/add-expenditure" method="post">
<div class="expenditure-input">
 <input name="transaction" placeholder="Transaction" type="text">  </div>
 <div class="expenditure-input">
 <input name="type" placeholder="Transaction Type" type="text"  list="trans-
type">
 <datalist id="trans-type">
 <option value="Medical Expenses">
 <option value="House Expenses">
 <option value="Education" >
 <option value="Savings" >
 <option value="Others" >
 </datalist>
 </div>
 <div class="expenditure-input">
 <input name="amount" placeholder="Amount" type="number"  min="0">
 </div>
 <div class="expenditure-input">
 <input name="date" placeholder="Date" type="date">  </div>
 <button type="submit" class="btn-submit">Add</button>
</form>
```

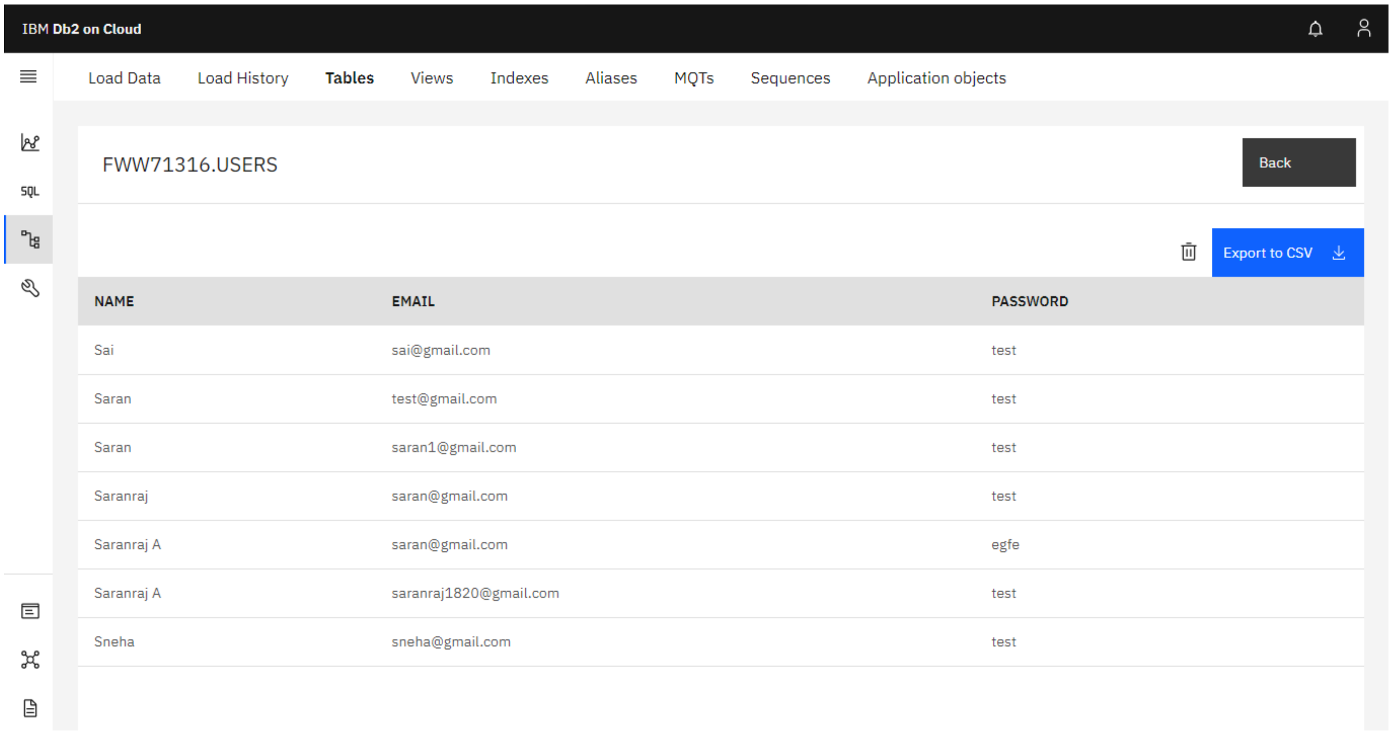
```
 </div>
 </div>
 </div>
</div>
</body>
</html>
```

7.3 Database Schema:



FWW71316.TRANSACTIONS

DATE	TRANSACTION	TYPE	AMOUNT	EMAIL
01/11/2022	New Laptop Savings	Savings	1000	saran@gmail.com
01/11/2022	Investments	Savings	1000	saranraj1820@gmail.com
03/11/2022	Investment	Savings	250	saran@gmail.com
04/11/2022	College Tution Fee	Education	15000	saran@gmail.com
05/11/2022	Restaurant	Others	1900	saranraj1820@gmail.com
06/11/2022	Hospital Charges for Cold	Medical Expenses	1500	saranraj1820@gmail.com
08/11/2022	Groceries	House Expenses	2750	saranraj1820@gmail.com
20/10/2022	College Tution Fee	Education	25000	saranraj1820@gmail.com



FWW71316.USERS

NAME	EMAIL	PASSWORD
Sai	sai@gmail.com	test
Saran	test@gmail.com	test
Saran	saran1@gmail.com	test
Saranraj	saran@gmail.com	test
Saranraj A	saran@gmail.com	egfe
Saranraj A	saranraj1820@gmail.com	test
Sneha	sneha@gmail.com	test

8.TESTING

8.1Test Cases

The main test case is to track and deliver the expenses made by the user and to save money for pre-defined expenses. Also, it plans on the user's future investments.

8.2 User Acceptance Testing

The main test case is to track and deliver the expenses made by the user and to save money for pre-defined expenses. Also, it plans on the user's future investments.

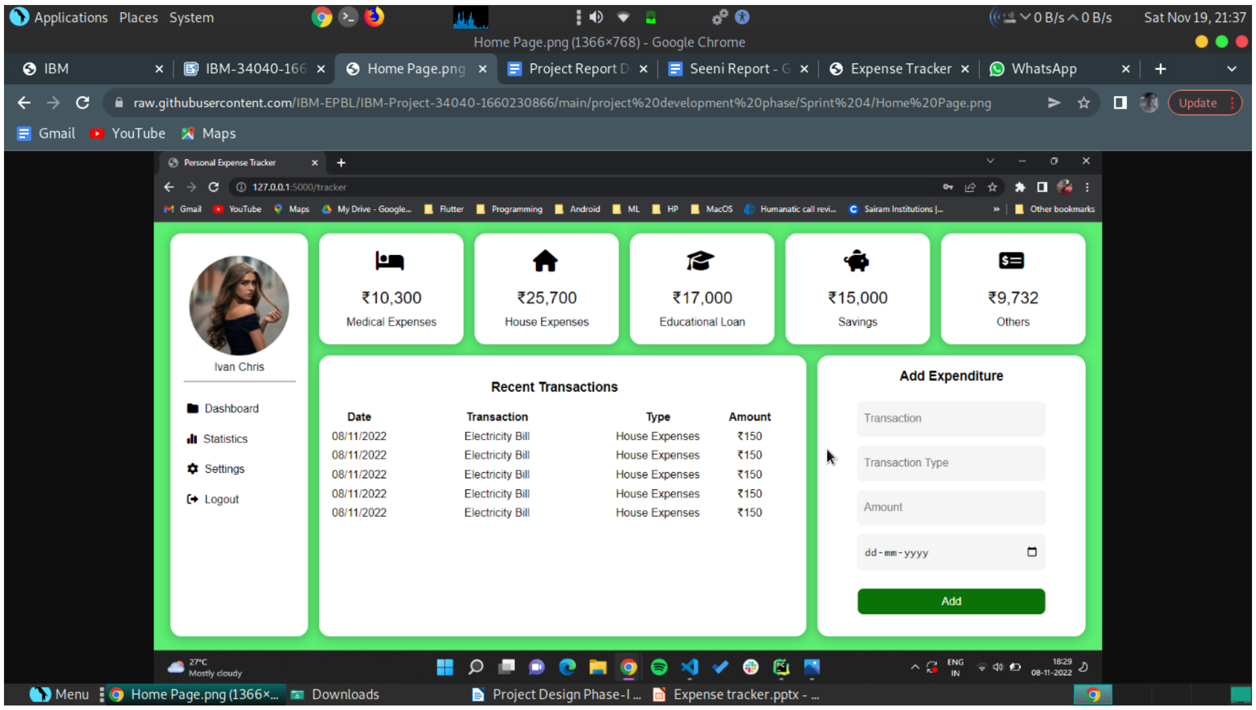
9.RESULTS

1. **Performance Metrics**

- Users have control over their money.
- User will be aware about what is happening with their money.

2. Save funds for emergencies.

- Aware about the financial condition.
- No more stress about the finance


Applications Places System
Home Page.png (1366×768) - Google Chrome
IBM
IBM-34040-166
Home Page.png
Project Report D
Seeni Report - G
Expense Tracker
WhatsApp
raw.githubusercontent.com/IBM-EPBL/IBM-Project-34040-1660230866/main/project%20development%20phase/Sprint%204/Home%20Page.png
Gmail YouTube Maps
Personal Expense Tracker
127.0.0.1:5000/tracker
Menu Home Page.png (1366×... Downloads Project Design Phase-I ... Expense tracker.pptx - ...

Ivan Chris

- Dashboard
- Statistics
- Settings
- Logout

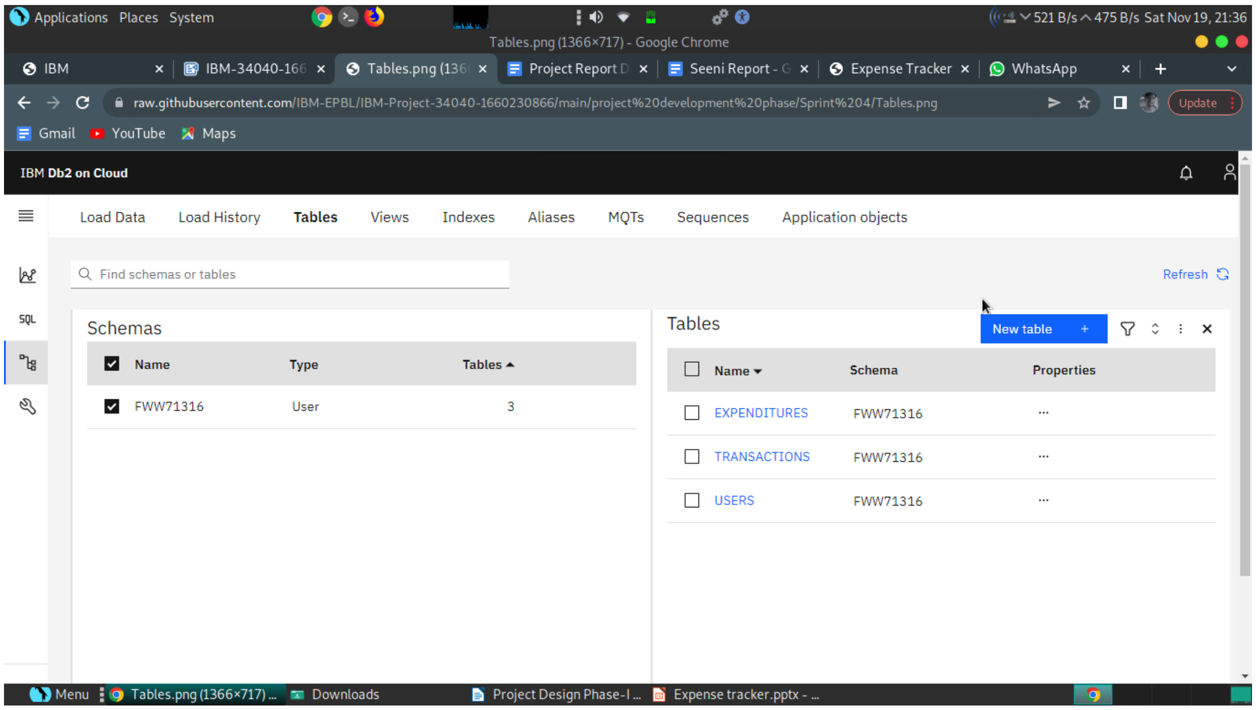
₹10,300	₹25,700	₹17,000	₹15,000	₹9,732
Medical Expenses	House Expenses	Educational Loan	Savings	Others

Recent Transactions

Date	Transaction	Type	Amount
08/11/2022	Electricity Bill	House Expenses	₹150
08/11/2022	Electricity Bill	House Expenses	₹150
08/11/2022	Electricity Bill	House Expenses	₹150
08/11/2022	Electricity Bill	House Expenses	₹150
08/11/2022	Electricity Bill	House Expenses	₹150

Add Expenditure

Transaction
Transaction Type
Amount
dd-mm-yyyy
Add

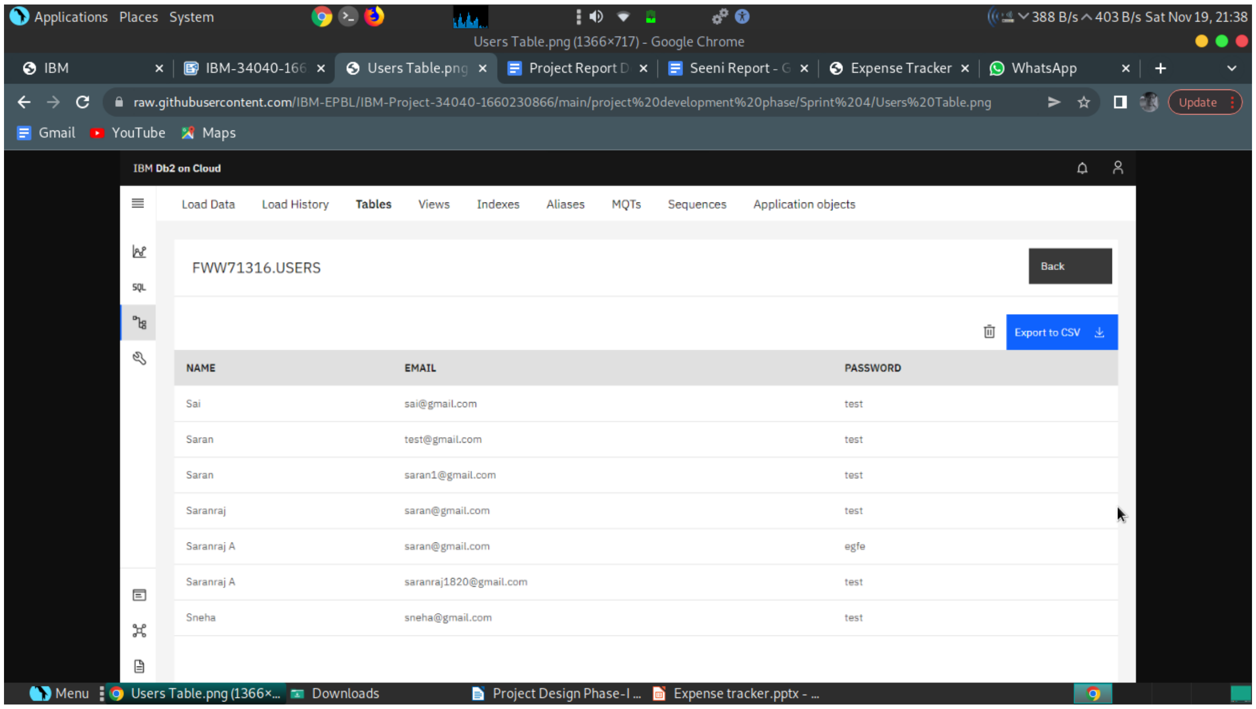

Applications Places System
Tables.png (1366×717) - Google Chrome
raw.githubusercontent.com/IBM-EPBL/IBM-Project-34040-1660230866/main/project%20development%20phase/Sprint%204/Tables.png
IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
Find schemas or tables
Refresh
New table
Menu Tables.png (1366×717) ... Downloads Project Design Phase-I ... Expense tracker.pptx - ...

Schemas

Name	Type	Tables
FWW71316	User	3

Tables

Name	Schema	Properties
EXPENDITURES	FWW71316	...
TRANSACTIONS	FWW71316	...
USERS	FWW71316	...



FWW71316.USERS

NAME	EMAIL	PASSWORD
Sai	sai@gmail.com	test
Saran	test@gmail.com	test
Saran	saran1@gmail.com	test
Saranraj	saran@gmail.com	test
Saranraj A	saran@gmail.com	egfe
Saranraj A	saranraj1820@gmail.com	test
Sneha	sneha@gmail.com	test

10.ADVANTAGE AND DISADVANTAGE

ADVANTAGE:

- Track your expenses anywhere, anytime.
- Seamlessly manage your money and budget without any financial paperwork.
- Access, submit, and approve invoices irrespective of time and location.
- Avoid data loss by scanning your tickets and bills and saving in the app.

DISADVANTAGE:

- 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 needs to be done by the user.

- Overload to rely on the daily entry of the expenditure

11.CONCLUSION

- We assure that this will help users to manage the cost of their daily expense.
- Help for the people who are frustrated with tha daily budget management.
- Wish to manage money.
- Preserve the record of their daily cost.
- Overcome wastage of money.

12.FUTURE SCOPE

- It will have various options to keep record (for example Food, Travelling Fuel, Salary etc.).

- Automatically it will keep on sending notifications for our daily expenditure.

- In today's busy and expensive life, we are in a great rush to make moneys, but at the end of the month we broke off. As we are unknowingly spending money on title and unwanted things. So, we have come over with the plan to follow our profit.

- Here user can define their own categories for expense type like food, clothing, rent and bills where they have to enter the money that has been spend and likewise can add some data in extra data to indicate the expense.

13.APPENDIX

Source Code

Back end

```
import pymysql

from flask import Flask

from flask import jsonify

from flask import request

import mysql.connector

from flask_cors import CORS, cross_origin



    app = Flask(__name__)

    CORS(app)


    # MySQL configurations

    app.config['MYSQL_DATABASE_USER'] = 'root'
    app.config['MYSQL_DATABASE_PASSWORD'] = 'root'

    app.config['MYSQL_DATABASE_DB'] = 'clgpractical'

    app.config['MYSQL_DATABASE_HOST'] = 'localhost'



    '''

    Expenses

    '''

    @app.route('/addexpense', methods=['POST'])

    def add_expense():

            conn = None

            cursor = None

            try:

                    _json = request.json

                    _user_id = _json['user_id']

                    _expense_id = _json['expense_id']

                    _expense_amount = _json['expense_amount']

                    _expense_description = _json['expense_description']

                    # validate the received values

                    if _user_id and _expense_id and _expense_description and
    _expense_amount and request.method == 'POST':
```

```
                # save edits
                sql = "INSERT INTO expenses(user_id, expense_id,
expense_amount, expense_description) VALUES(%s, %s, %s, %s)"
                data = (_user_id, _expense_id, _expense_amount,
_expense_description)
                conn = mysql.connector.connect(user = "root",
                                                                        password
= "root",
                                                                        database
= "clgpractical")
                cursor = conn.cursor()
                cursor.execute(sql, data)
                conn.commit()
                resp = jsonify('Expense added successfully!')
                resp.status_code = 200
                return resp
            else:
                return not_found()
        except Exception as e:
            print(e)
        finally:
            cursor.close()
            conn.close()



@app.route('/expenses')
def expenses():

        try:
            conn = mysql.connector.connect(user = "root",
            password = "root",
            database = "clgpractical")
            cursor = conn.cursor(pymysql.cursors.DictCursor)
            cursor.execute("SELECT * FROM expenses")
            rows = cursor.fetchall()
            resp = jsonify(rows)
            resp.status_code = 200
            return resp
        except Exception as e:
            print("1",e)
```

```
@app.route('/expense/<int:id>')

def expense(id):

        conn = None

        cursor = None

        try:

                conn = mysql.connector.connect(user = "root",

                password = "root",

                database = "clgpractical")

                cursor = conn.cursor(pymysql.cursors.DictCursor)

                cursor.execute("SELECT * FROM expenses WHERE expense_id
= %s", (id,))

                row = cursor.fetchone()

                resp = jsonify(row)

                resp.status_code = 200

                return resp

        except Exception as e:

                print("111111111",e)

        finally:

                cursor.close()
        conn.close()



@app.route('/updateexpense', methods=['PUT'])
def update_expense():
    conn = None
    cursor = None
    try:
        _json = request.json
        _id = _json['id']
        _expense_id = _json['expense_id']
        _expense_amt = _json['expense_amount']
        _expense_desc = _json['expense_description']
        # validate the received values
        if _expense_id and _expense_amt and _expense_desc and _id and
request.method == 'PUT':

            # save edits
            sql = "UPDATE expenses SET user_id=%s,
expense_description=%s , expense_amount=%s WHERE expense_id=%s"
            data = (_id, _expense_desc, _expense_amt,  _expense_id,)
            conn = mysql.connector.connect(user = "root",
        password = "root",
        database = "clgpractical")
            cursor = conn.cursor()
            cursor.execute(sql, data)
            conn.commit()
            resp = jsonify('User updated successfully!')
            resp.status_code = 200
            return resp
        else:
            return not_found()
    except Exception as e:
        print("1",e)
    finally:
        cursor.close()
        conn.close()



@app.route('/deleteexpense/<int:id>', methods=['DELETE'])
def delete_expense(id):
    conn = None
```

```
        cursor = None
        try:
                conn = mysql.connector.connect(user = "root",
                password = "root",
                database = "clgpractical")
                cursor = conn.cursor()
                cursor.execute("DELETE        FROM        expenses        WHERE
expense_id=%s", (id,))
                conn.commit()
                resp = jsonify('Expense deleted successfully!')
                resp.status_code = 200
                return resp
        except Exception as e:
                print(e)
        finally:
                cursor.close()
                conn.close()



@app.errorhandler(404)
def not_found(error=None):
        message = {
                'status': 404,
                'message': 'Not Found: ' + request.url,
        }
        resp = jsonify(message)
        resp.status_code = 404

        return resp


if __name__ == "__main__":
    app.run()
```

FRONT END

```
// add expense button reference
var btnAddExpense = document.querySelector("#addExpense");


// elements reference
var inputAmount = document.querySelector("#amount");
var inputDescription = document.querySelector("#description");
var totalAmount = document.querySelector("#total");
var errorAmountMessage = document.querySelector("#errorAmount");
var errorDescriptionMessage= document.querySelector("#errDesc");
```

```
//  expenseData reference where data will be displayed

var expenseDataEl = document.querySelector("#expenseData")


var counter =0;


//array for all expense objects

var allExpenses=[];

var totalExpense=0;


function retrieve(){

    console.log("called")

    allExpenses =[]

    var users;

    var url  = "http://127.0.0.1:5000/expenses";

    var xhr  = new XMLHttpRequest()

    xhr.open('GET', url, true)

    xhr.onload = function () {

        users = JSON.parse(xhr.responseText);

        updateDisplay(users)

        if (xhr.readyState == 4 && xhr.status == "200") {
            console.table(users);
            console.log(users[0][0]);
            console.log(users.length);
        } else {
            console.error(users);
        }
    }
    xhr.send(null);
}

function updateDisplay(users){
    for(var i = 0; i < users.length;i++ ){
        var expenseItem ={};
        expenseItem.amount = users[i][2];
        expenseItem.description = users[i][3];
        expenseItem.moment = new Date();
        expenseItem.id = users[i][1];
        totalExpense +=expenseItem.amount;
        allExpenses.push(expenseItem)
        counter = expenseItem.id;
    }
    totalAmount.innerHTML = totalExpense;
    //updating display
    renderList(allExpenses);
}

//Event listener to add expense
btnAddExpense.addEventListener("click",()=>{
    var isValidAmount = checkAmount();
    var isValidDescription = checkDescription();

    if(isValidAmount && isValidDescription){
        if(btnAddExpense.value=="Add Expense"){
            console.log("Adding")
            addExpense();
        }
        else
            updateExpense();
    }

});
```

```
// Check if amount is valid
function checkAmount(){

    if (!Number.isNaN(parseInt(inputAmount.value))) {
        return true;
    } else {
        errorAmountMessage.innerHTML="Enter a valid number";
        clearInputElements();
        return false;
    }
}

// Check if description is valid
function checkDescription(){
    if(inputDescription.value==""){
        console.log("hi")
        errorDescriptionMessage.innerHTML="Enter description";
        return false;
    }
    else{
        return true;
    }
}

// add expense function
function addExpense(){

    // object declaration
    var expenseItem ={};
    counter ++;
    //reading values from amount and description
    const amountValue = inputAmount.value;
    const descriptionValue = inputDescription.value;

    // adding the values to expenseItem object
    expenseItem.amount = amountValue;
    expenseItem.description = descriptionValue;
    expenseItem.moment = new Date();
    expenseItem.id = counter;

    //pushing it to allExpenses array
    allExpenses.push(expenseItem);
```

```
//updating display
renderList(allExpenses);

// totalcalculating total amount
totalExpense += parseInt(amountValue);
totalAmount.innerHTML = totalExpense;

//clearing input
clearInputElements();
clearErrorMessages();

// Db adding
var url = "http://localhost:5000/addexpense";

var data = {};
data.user_id = 11;
data.expense_id  = counter;
data.expense_amount = amountValue;
data.expense_description = descriptionValue;
var json = JSON.stringify(data);
```

```
    var xhr = new XMLHttpRequest();
    xhr.open("POST", url, true);
    xhr.setRequestHeader('Content-type','application/json; charset=utf-8');
    xhr.onload = function () {
        var users = xhr.response;
        if (xhr.readyState == 4 && xhr.status == "201") {
            console.table(users);
        } else {
            console.error(users);
        }
    }
    xhr.send(json);

}


function clearInputElements(){
    inputAmount.value="";
    inputDescription.value="";
```

```
}


function clearErrorMessages(){
    errorAmountMessage.innerHTML="";
    errorDescriptionMessage.innerHTML="";
}


// function to update the display of expenses list
function renderList(arr){
    console.log("1");
    var allExpenseHtml = arr.map(expense => createListItem(expense));
    var joinedAllExpenseHtml= allExpenseHtml.join(' ');
    expenseDataEl.innerHTML=joinedAllExpenseHtml;
}


// function to create individual tile of expense info
function createListItem({description,amount,moment,id}){
    return `
    <li class="list-group-item d-flex justify-content-between">
            <div class="d-flex flex-column">
               Rs. ${amount}
```

```
                <small class="text-muted">${description} </small>
            </div>
            <div>
                  <button
                  type="button"
                  class="btn btn-outline-secondary btn-sm"
                  onclick="updateItem(${id})"
                  >
                Update
                </button>
                <button
                  type="button"
                  class="btn btn-outline-danger btn-sm"
                  onclick="deleteItem(${id})"
                  >
                  Delete
                </button>
            </div>
        </li>
    `;
}
```

```
var update_id;
function updateItem(id){
    btnAddExpense.value= "Update Expense";
    update_id = id;

}

function updateExpense(){
    console.log("updating expense")

    var url = "http://localhost:5000/updateexpense";

    var data = {};
    data.id = 11;
    data.expense_id  = update_id;
    data.expense_amount = inputAmount.value;
    data.expense_description = inputDescription.value;
    var json = JSON.stringify(data);

    var xhr = new XMLHttpRequest();
```

```
xhr.open("PUT", url, true);
xhr.setRequestHeader('Content-type','application/json; charset=utf-8');
xhr.onload = function () {
 var users = JSON.parse(xhr.responseText);
 if (xhr.readyState == 4 && xhr.status == "200") {
        console.table(users);
 } else {
        console.error(users);
 }
}
xhr.send(json);


clearInputElements();
clearErrorMessages();
setTimeout(retrieve(),3000)
btnAddExpense.value = "Add Expense";
console.log("TEST",allExpenses)
console.log("TEST",allExpenses["0"])
for(var i = 0; i < allExpenses.length; i++){
   console.log("TESTing")
   if(allExpenses[i].id==update_id){
```

```
            allExpenses[i].amount = data.expense_amount;

            allExpenses[i].descriptionValue = data.expense_description;

            total += allExpenses[i].amount;

            console.log("in")

        }

    }

    console.log(allExpenses)

    renderList(allExpenses)

}



// function to delete item

function deleteItem(id){



    //deleting from db

    var url = "http://localhost:5000/deleteexpense";

    var xhr = new XMLHttpRequest();

    xhr.open("DELETE", url+'/'+id, true);

    xhr.onload = function () {

     var users = JSON.parse(xhr.responseText);
```

```
 if (xhr.readyState == 4 && xhr.status == "200") {
         console.table(users);
 } else {
         console.error(users);
 }
}
xhr.send(null);



//updating total
const delEl = allExpenses.filter(expense=>expense.id==id);
const el = delEl.pop();
totalExpense -= parseInt(el.amount);
totalAmount.innerHTML = totalExpense;

//deleting from array and displaying the new array
const newArr= allExpenses.filter(expense => expense.id!== id);

// updating allExpenses array
allExpenses = newArr.map(expense => expense);
```

```
    //updating display
    renderList(newArr);
}




function getDateString(moment) {
    return moment.toLocaleDateString('en-US', {
        year: 'numeric',
        month: 'long',
        day: 'numeric',
    });
}
```

GitHub Link

https://github.com/IBM-EPBL/IBM-Project-47967-1660803609

PROJECT DEMO LINK:

https://drive.google.com/file/d/179gE6AVpm_km_aPYZNA_K9qICh189veq/view?usp=sharing