NALAIYA THIRAN PROJECT BASED LEARNING ON PROFESSIONAL READLINESS FOR INNOVATION, EMPLOYNMENT AND ENTERPRENEURSHIP A PROJECT REPORT

PERSONAL EXPENSE TRACKER APPLICATION

IBM-Project-46011-1660734557

KALPANADEVI J - 721219104020

BINU ANGELA A - 721219104011

KAVYA K - 721219104022

DHARUN RAM RA - 721219104013

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KARPAGAM INSTITUTE OF TECHNOLOGY

COIMBATORE - 641105

LIST OF CONTENTS

1. INTRODUCTION

1.1 Project Overview

Category	:Cloud App Development
Team ID	:PNT2022TMID31805
Skills Required	:IBM Cloud ,HTML , Javascript , IBM Cloud Object Storage ,Python-Flask , Kubernetes , Docker , IBM DB2, IBM Container Registry.

In today's busy and expensive life we are in a great rush to make money. But at the end of the month we broke off. As we are unknowingly spending money on little and unwanted things. So, we have come over with the idea to track our earnings. Daily Expense Tracker (DET) aims to help everyone who are planning to know their expenses and save from it. DET is an android app which users can execute in their mobile phones and update their daily expenses so that they are well known to their expenses. User can also define expense categories. User will be able to see pie chart of expense. Also, DET app is capable of clustering. Personal and administration clustering is possible by the use of Apriori algorithm. Although this app is focused on new job holders, interns and teenagers, everyone who wants to track their expense can use this app.

1.2 Purpose

An expense tracking app is an exclusive suite of services for people who seek to handle their earnings and plan their expenses and savings efficiently. It helps you track all transactions like bills, refunds, payrolls, receipts, taxes, etc., on a daily, weekly, and monthly basis. A good app should allow you to capture all your receipts when receiving or making payments. Each receipt should be placed under an appropriate category. These receipts are stored in the cloud and can be retrieved anytime. This feature is especially beneficial for employees who travel for business. As such, many receipt tracking solutions are designed to function well on mobile apps, across various devices and with multiple linked accounts .People tend to overspend without realizing and this can prove to be disastrous .Using a daily expense manager can help you keep track of how much you spend every day and on what. At the end of the month, you will have a clear picture where your money is going .This is one of the best ways to get your expenses under control and bring some assemblance of order to your finances. Today, there are several expense manager applications in the market. Some are paid managers while others are free.Even banks like KVB,ICICI offer their customer expense tracker to help them out .Before You decide to go in for a money manager ,it is important to decide the type you want.

2. LITERATURE SURVEY

2.1 Existing Problem

In a study conducted by Forrester in 2016 surveying small and medium businesses (SMBs) across the world, 56% companies reported expense management as being the biggest challenge for their finance departments. In another survey conducted by Levvel Research in 2018 in North America, respondents reported the following pain points in expense management before adopting automation:

- Manual entry and routing of expense reports (62%)
- Lack of visibility into spend data (42%)
- Inability to enforce travel policies (29%)
- Lost expense reports (24%)
- Lengthy expense approval system and reimbursement cycles (23%).

2.2 References

- En.wikipedia.org. (2018). Systems design. [online] Available at: https://en.wikipedia.org/wiki/Systems_design [Accessed 2 May. 2018].

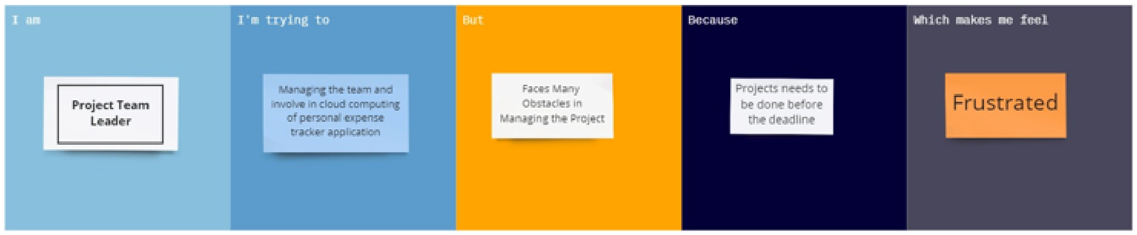
- Slideshare.net. (2018). Android ppt with example of budget manager. [online] Available at: https://www.slideshare.net/nalinimehta73/android-ppt-with example-of-budget-manager [Accessed 21 Apr. 2018].

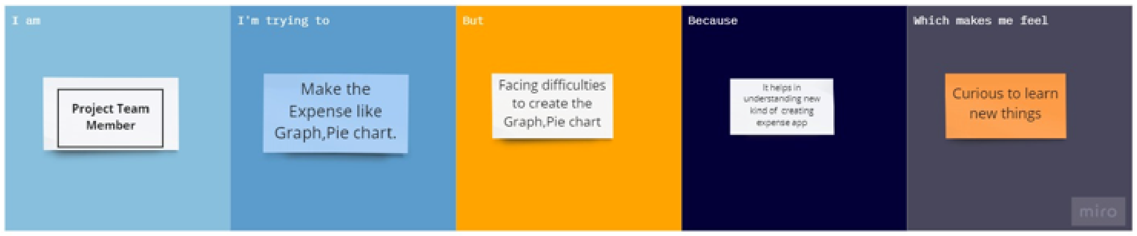
- Creately.com. (2018). Expense tracker. [online] Available at: https://creately.com/diagram/example/hv2esdzr2/expense%20tracker [Accessed 25 Apr. 2018].

- Slideshare.net. (2014). Apriori Algorithm by International School of Engineering. [online] Available at: https://www.slideshare.net/INSOFE/apriori-algorithm-36054672 [Accessed 14 Apr. 2018].

2.3 Problem Statement Definition

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.


I am
Project Team Leader
I'm trying to
Managing the team and involve in cloud computing of personal expense tracker application
But
Faces Many Obstacles in Managing the Project
Because
Projects needs to be done before the deadline
Which makes me feel
Frustrated

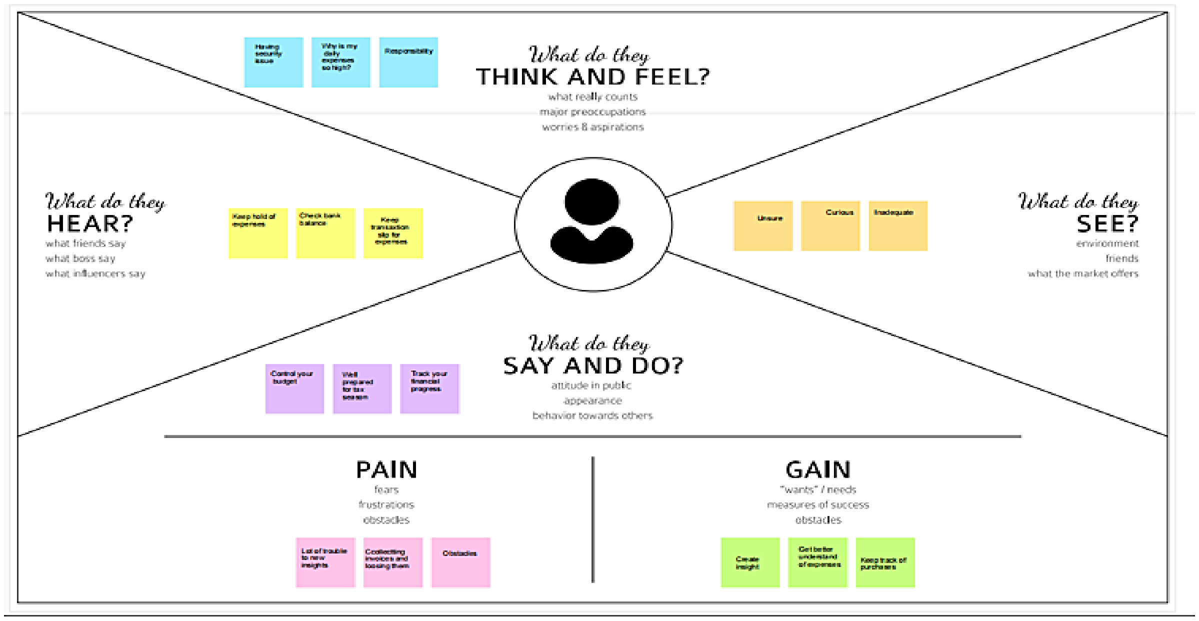

I am
Project Team Member
I'm trying to
Make the Expense like Graph,Pie chart.
But
Facing difficulties to create the Graph,Pie chart
Because
It helps in understanding new kind of creating expense app
Which makes me feel
Curious to learn new things
miro

Problem Statement (PS)	**I am (Customer)**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	**Project Team Leader**	Managing the team and involve in cloud computing of personal expense tracker application	Faces Many Obstacles in Managing the Project	Projects needs to be done before the deadline	Frustrated
PS-2	**Project Team Member**	Make the Expense like Graphs and Pie chart.	Facing difficulties to create the Graphs and Pie chart	It helps in understanding new kind of creating expense app	Curious to learn new things

3.Ideation and Proposed Solution

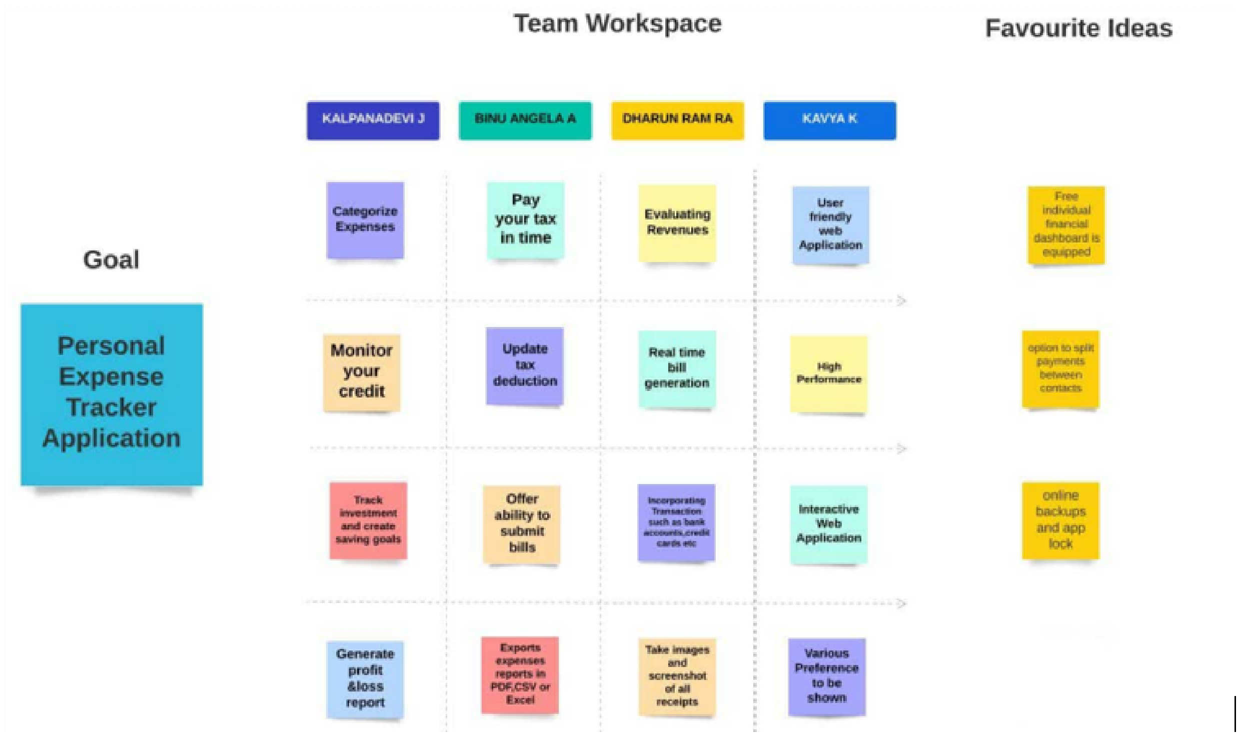
3.1 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to helps teams better understand their users. Empathy mapping is a simple workshop activity that can be done with stakeholders, marketing and sales, product development, or creative teams to build empathy for end users. For teams involved in the design and engineering of products, services, or experiences, an empathy mapping session is a great exercise for groups to "get inside the heads" of users. **Empathy maps are most useful at the beginning of the design process** after user research but before requirements and concepting. The mapping process can help synthesize research observations and reveal deeper insights about a user's needs.



3.2 Ideation & Brainstorming

Brainstorming is **a group problem-solving method that involve the spontaneous contribution of creative ideas and solutions.** This technique requires intensive, freewheeling discussion in which every member of the group is encouraged to think aloud and suggest as many ideas as possible based on their diverse knowledge.

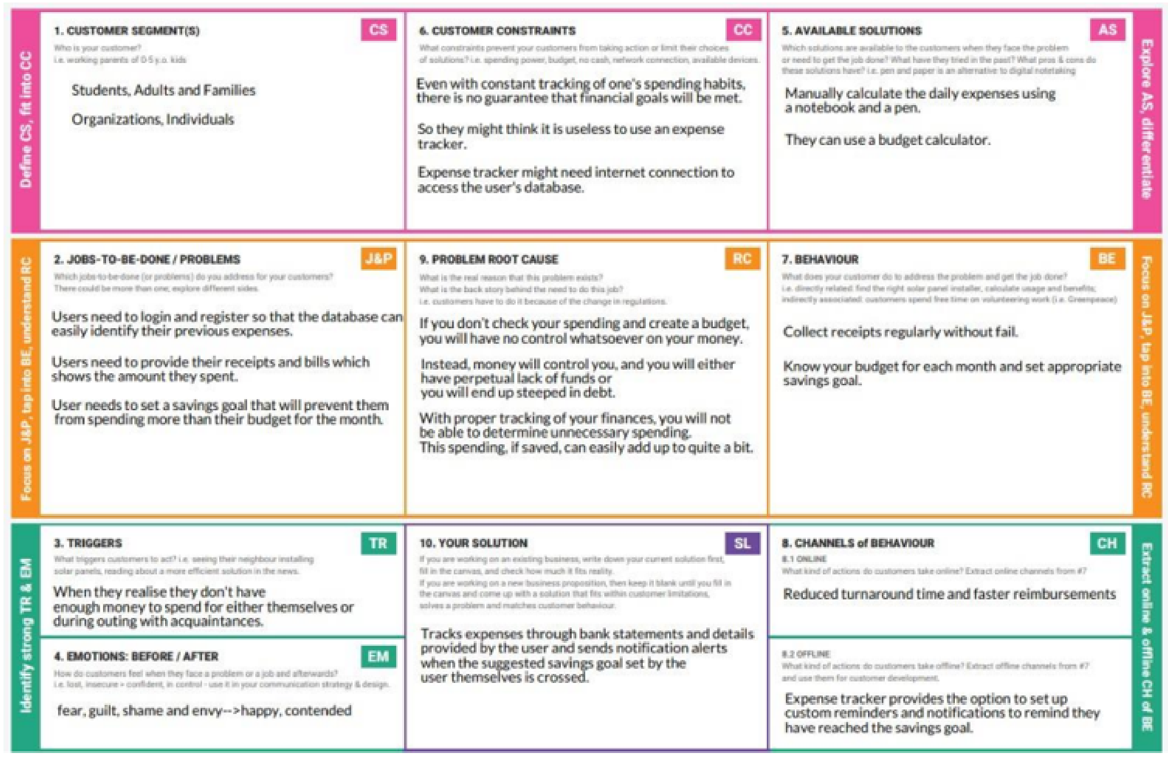

Team Workspace
Favourite Ideas
KALPANADEVI J
BINU ANGELA A
DHARUN RAM RA
KAVYA K
Goal
Personal Expense Tracker Application
Categorize Expenses
Pay your tax in time
Evaluating Revenues
User friendly web Application
Free individual financial dashboard is equipped
Monitor your credit
Update tax deduction
Real time bill generation
High Performance
option to split payments between contacts
Track investment and create saving goals
Offer ability to submit bills
Incorporating Transaction such as bank accounts,credit cards etc
Interactive Web Application
online backups and app lock
Generate profit &loss report
Exports expenses reports in PDF,CSV or Excel
Take images and screenshot of all receipts
Various Preference to be shown

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	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.
2.	Idea / Solution description	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.
3.	Novelty / Uniqueness	An Personal expense tracker helps you figure out what is happening to your money, and whether you can afford something you want.
4.	Social Impact / Customer Satisfaction	Personal Expense Tracker bundles together a collection of user-friendly tools that make it easy to track expenses, plan spending and save for future goals.
5.	Business Model (Revenue Model)	Effective financial management requires the proper tracking of income and expenses. There are many options to help you track all of your spending.
6.	Scalability of the Solution	The app will show you where your money is going. Keeping track of your finances frequently isn't the most pleasant thing in the world, and it requires specific skills and knowledge.

3.4 Problem Solution Fit

Problem-solution fit is a term used to describe the point validating that the base problem resulting in a business idea really exists and the proposed solution actually solves that problem.The least glamorous but most important part of starting a successful business is determining whether your idea actually solves a real problem for people. This process is known as finding a problem-solution fit.



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

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

Students, Adults and Families

Organizations, Individuals

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

Even with constant tracking of one's spending habits, there is no guarantee that financial goals will be met.

So they might think it is useless to use an expense tracker.

Expense tracker might need internet connection to access the user's database.

5. AVAILABLE SOLUTIONS AS

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

Manually calculate the daily expenses using a notebook and a pen.

They can use a budget calculator.

Explore AS, differentiate

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

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

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

Users need to login and register so that the database can easily identify their previous expenses.

Users need to provide their receipts and bills which shows the amount they spent.

User needs to set a savings goal that will prevent them from spending more than their budget for the month.

9. PROBLEM ROOT CAUSE RC

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

If you don't check your spending and create a budget, you will have no control whatsoever on your money.

Instead, money will control you, and you will either have perpetual lack of funds or you will end up steeped in debt.

With proper tracking of your finances, you will not be able to determine unnecessary spending.
This spending, if saved, can easily add up to quite a bit.

7. BEHAVIOUR BE

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

Collect receipts regularly without fail.

Know your budget for each month and set appropriate savings goal.

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

Identify strong TR & EM

3. TRIGGERS TR

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

When they realise they don't have enough money to spend for either themselves or during outing with acquaintances.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

fear, guilt, shame and envy-->happy, contended

10. YOUR SOLUTION SL

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

Tracks expenses through bank statements and details provided by the user and sends notification alerts when the suggested savings goal set by the user themselves is crossed.

8. CHANNELS of BEHAVIOUR CH

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

Reduced turnaround time and faster reimbursements

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

Expense tracker provides the option to set up custom reminders and notifications to remind they have reached the savings goal.

Extract online & offline CH of BE

4.Requirement Analysis

4.1 Functional Requirements

Following are the functional requirements of the proposed solution:

FR No.	**Functional Requirement(Epic)**	**Sub Requirement (Story/sub-Task)**
FR-1	User Registration	Registration through Application. Registration through Gmail.
FR-2	User Confirmation	Confirmation via Email. Confirmation via OTP.
FR-3	User Monthly Expensive 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

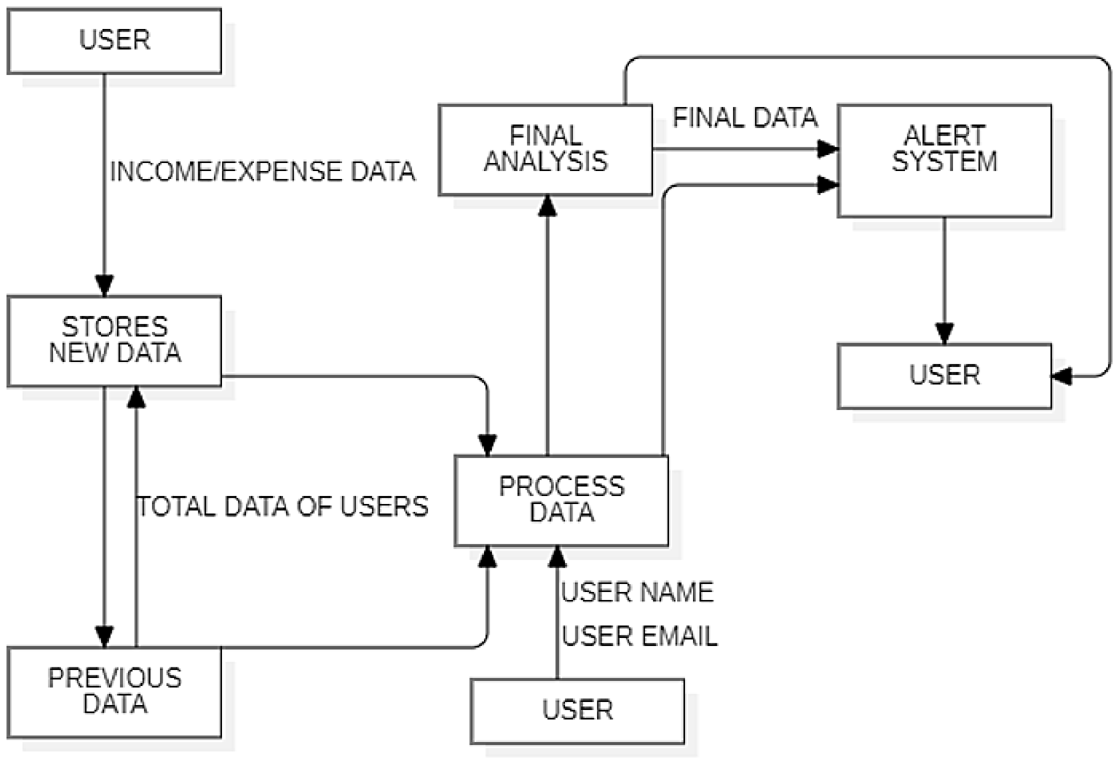
Following are the non-functional requirements of the proposed solution:

FR No	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	Effectiveness,Efficiency and overall satisfication of the user while interacting with our application.
NFR-2	**Security**	Authentication ,Authorization ,encryption of the application.
NFR-3	**Reliabilty**	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 satisfication will affect the solution.
NFR-6	**Scalability**	Capacity of the application to handle growth , especially in handling more users.

5.Project Design

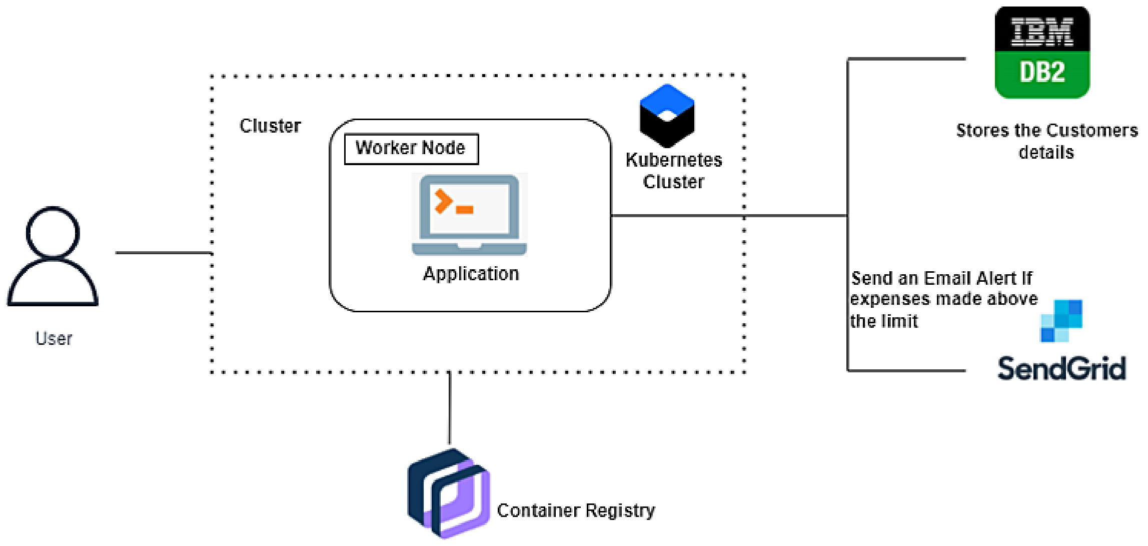
5.1.Data Flow Diagram

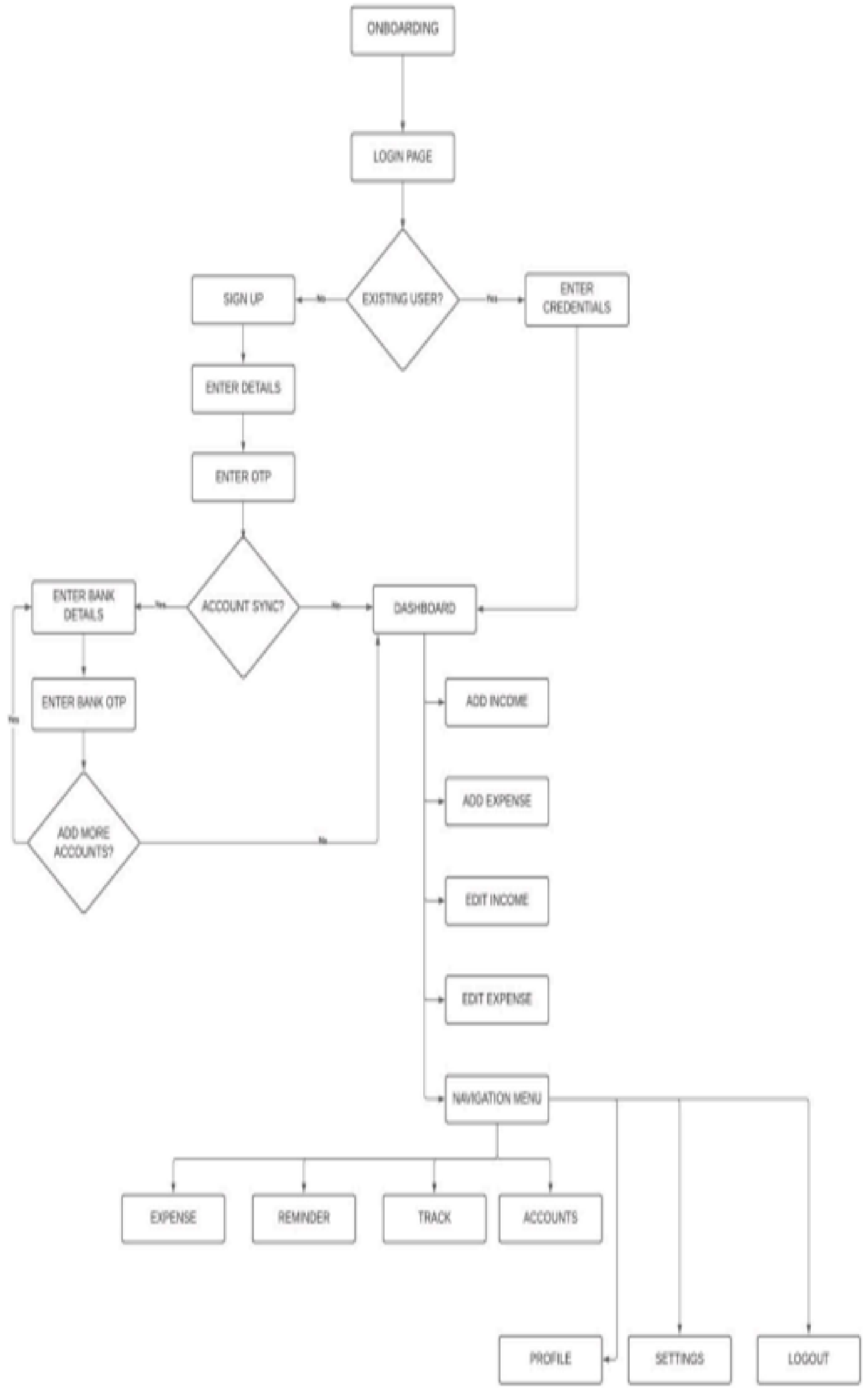
A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled.



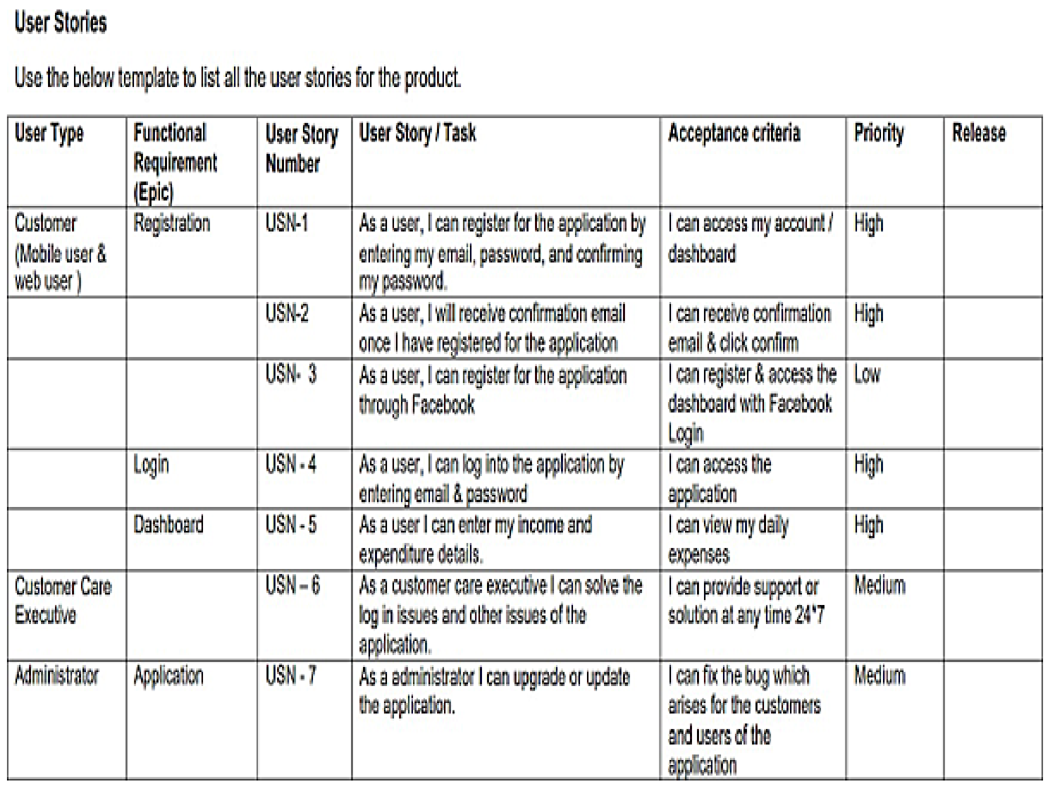
5.2.Solution & Technical Architecture

A solution architecture (SA) is an architectural description of a specific solution. SAs combine guidance from different enterprise architecture viewpoints (business, information and technical), as well as from the enterprise solution architecture (ESA).




ONBOARDING
LOGIN PAGE
EXISTING USER?
No
Yes
SIGN UP
ENTER CREDENTIALS
ENTER DETAILS
ENTER OTP
ACCOUNT SYNC?
Yes
No
ENTER BANK DETAILS
DASHBOARD
ENTER BANK OTP
Yes
ADD MORE ACCOUNTS?
No
ADD INCOME
ADD EXPENSE
EDIT INCOME
EDIT EXPENSE
NAVIGATION MENU
EXPENSE
REMINDER
TRACK
ACCOUNTS
PROFILE
SETTINGS
LOGOUT

5.3.User Stories



User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user & web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	
		USN- 3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	
	Login	USN - 4	As a user, I can log into the application by entering email & password	I can access the application	High	
	Dashboard	USN - 5	As a user I can enter my income and expenditure details.	I can view my daily expenses	High	
Customer Care Executive		USN – 6	As a customer care executive I can solve the log in issues and other issues of the application.	I can provide support or solution at any time 24*7	Medium	
Administrator	Application	USN - 7	As a administrator I can upgrade or update the application.	I can fix the bug which arises for the customers and users of the application	Medium	

6.Project Planning and Scheduling

6.1. Sprint Planning and Estimation

Sprint	**Functional Requirements**	**User Stories**	**User Story/Task**	**Story Points**	**Priority**	**Team Members**
Sprint 1	Registration	USN-1	As a user ,I can register for the application by entering my email ,password, and conforming my password	8	High	Kalpanadevi J Binu Angela A
Sprint 1	Login	USN-2	As a user I can Login to the application by entering email& password	8	High	Dharun Ram RA Kavya K
Sprint 2	Add Expenses	USN-3	As a user , I can add the day to day expense to the application	5	Medium	Kalpanadevi J Binu Angela A
Sprint 2	Edit and Delete Expenses	USN-4	As a user I can edit and delete the previously created expense	5	Medium	Dharun Ram RA Kavya K
Sprint 3	Creating time based filters in history	USN-5	As a user, I can see the time based history of expenses	8	High	Kalpanadevi J Binu Angela A
Sprint 3	Integrating with pie chart for analysis	USN-6	As a user , I can see view diagrammatic representation of expenses	5	Medium	Dharun Ram RA Kavya K
Sprint 4	Enabling limit feature	USN-7	As a user ,I can set monthly limit to expenses	5	Medium	Kalpanadevi J Binu Angela A
Sprint 4	Sending Email Alerts	USN-8	As a user, I will receive a mail if I cross a limit	8	High	Dharun Ram RA Kavya K

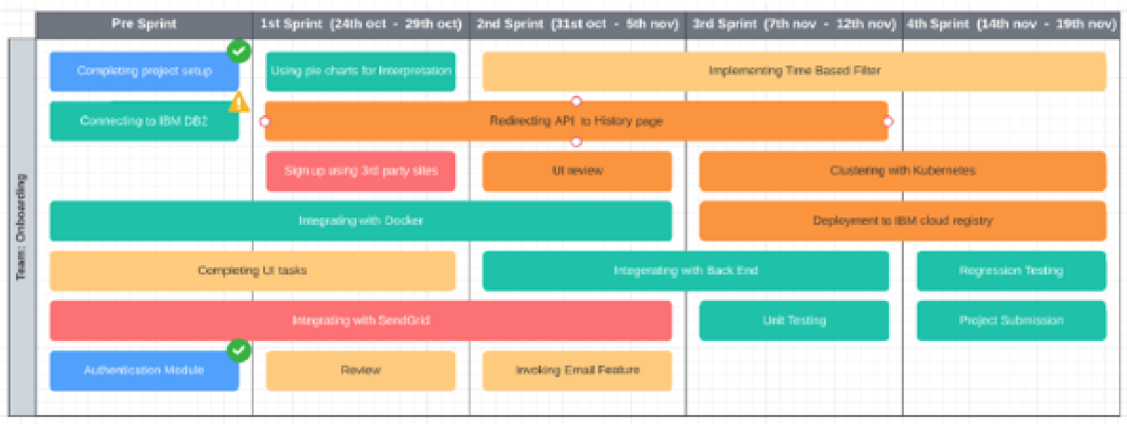
Project Tracker,Velocity,Burndown Chart

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	16	6 Days	24 Oct 2022	29 Oct 2022	16	29 Oct 2022
Sprint-2	12	6 Days	31 Oct 2022	05 Nov 2022	10	05 Nov 2022
Sprint-3	14	6 Days	07 Nov 2022	12 Nov 2022	13	12 Nov 2022
Sprint-4	14	6 Days	14 Nov 2022	19 Nov 2022	13	19 Nov 2022

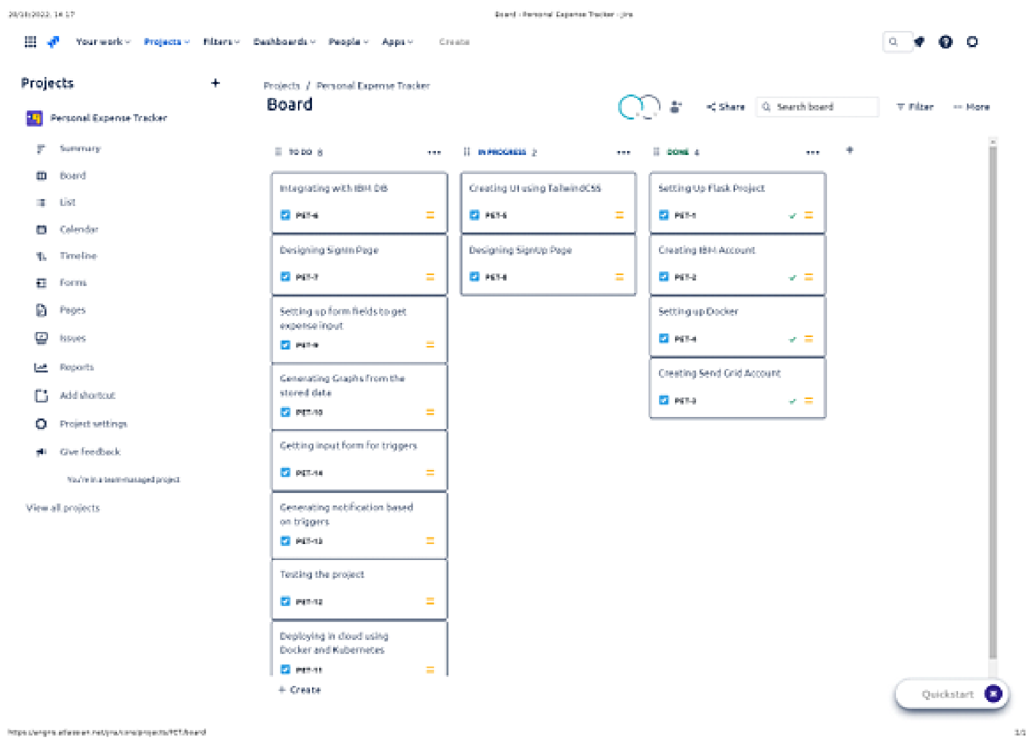
Velocity:
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

6.2.Sprint Delivery Schedule



6.3 Reports From JIRA



28/10/2022, 14:17 Board - Personal Expense Tracker - Jira

Your work · Projects · Filters · Dashboards · People · Apps · Create

Projects

Personal Expense Tracker

- Summary
- Board
- List
- Calendar
- Timeline
- Forms
- Pages
- Issues
- Reports
- Add shortcut
- Project settings
- Give feedback

You're in a team-managed project

View all projects

Projects / Personal Expense Tracker

Board

Share · Search board · Filter · More

TO DO 8	IN PROGRESS 2	DONE 4
Integrating with IBM DB — PET-6	Creating UI using TailwindCSS — PET-5	Setting Up Flask Project — PET-1 ✓
Designing SignIn Page — PET-7	Designing SignUp Page — PET-8	Creating IBM Account — PET-2 ✓
Setting up form fields to get expense input — PET-9		Setting up Docker — PET-4 ✓
Generating Graphs from the stored data — PET-10		Creating Send Grid Account — PET-3 ✓
Getting input form for triggers — PET-14		
Generating notification based on triggers — PET-13		
Testing the project — PET-12		
Deploying in cloud using Docker and Kubernetes — PET-11		

+ Create

Quickstart

https://engrs.atlassian.net/jira/software/projects/PET/boards 1/1

7. CODING & SOLUTIONING

7.1 Feature 1

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

● You only have positive values.

● You have less than seven categories since a larger number can

make it difficult to perceive each segment.

Code:

home.css

```
@import

'https://fonts.googleapis.com/css?family=Montserrat:300,400,700&display=swap';

* {

        padding: 0;

        margin: 0;

        box-sizing: border-box;

}

html {

        font-size: 10px;
        font-family: 'Montserrat', sans-serif;
        scroll-behavior: smooth;
}
a {
        text-decoration: none;
}
.container {
        min-height: 100vh;
        width: 100%;
        display: flex;
        align-items: center;
        justify-content: center;
}
img {
        height: 100%;
        width: 100%;
        object-fit: cover;
}
p {
        color: black;
        font-size: 1.4rem;
        margin-top: 5px;
        line-height: 2.5rem;
        font-weight: 300;
        letter-spacing: 0.05rem;
}
.section-title {
        font-size: 4rem;
        font-weight: 300;
        color: black;
        margin-bottom: 10px;
        text-transform: uppercase;
        letter-spacing: 0.2rem;
        text-align: center;
}
.section-title span {
        color: crimson;
}

.cta {
        display: inline-block;
        padding: 10px 30px;
        color: white;
        background-color: transparent;
        border: 2px solid crimson;
        font-size: 2rem;
        text-transform: uppercase;
        letter-spacing: 0.1rem;
        margin-top: 30px;
        transition: 0.3s ease;
        transition-property: background-color, color;
}
.cta:hover {
        color: white;
        background-color: crimson;
}
.brand h1 {
        font-size: 3rem;
        text-transform: uppercase;
        color: white;
}
.brand h1 span {
        color: crimson;
}

/* Header section */
#header {
        position: fixed;
        z-index: 1000;
```

```
    left: 0;
    top: 0;
    width: 100vw;
    height: auto;
}
#header .header {
    min-height: 8vh;
    background-color: rgba(31, 30, 30, 0.24);
    transition: 0.3s ease background-color;
}
#header .nav-bar {
    display: flex;
    align-items: center;
    justify-content: space-between;
    width: 100%;
    height: 100%;
    max-width: 1300px;
    padding: 0 10px;
}
#header .nav-list ul {
    list-style: none;
    position: absolute;
    background-color: rgb(31, 30, 30);
```

```
        width: 100vw;

        height: 100vh;

        left: 100%;

        top: 0;

        display: flex;

        flex-direction: column;

        justify-content: center;

        align-items: center;

        z-index: 1;

        overflow-x: hidden;

        transition: 0.5s ease left;

}

#header .nav-list ul.active {

        left: 0%;

}

#header .nav-list ul a {

        font-size: 2.5rem;

        font-weight: 500;

        letter-spacing: 0.2rem;

        text-decoration: none;

        color: white;

        text-transform: uppercase;

        padding: 20px;
```

```css
	display: block;
}
#header .nav-list ul a::after {
	content: attr(data-after);
	position: absolute;
	top: 50%;
	left: 50%;
	transform: translate(-50%, -50%) scale(0);
	color: rgba(240, 248, 255, 0.021);
	font-size: 13rem;
	letter-spacing: 50px;
	z-index: -1;
	transition: 0.3s ease letter-spacing;
}
#header .nav-list ul li:hover a::after {
	transform: translate(-50%, -50%) scale(1);
	letter-spacing: initial;
}
#header .nav-list ul li:hover a {
	color: crimson;
}
#header .hamburger {
	height: 60px;
```

```
        width: 60px;
        display: inline-block;
        border: 3px solid white;
        border-radius: 50%;
        position: relative;
        display: flex;
        align-items: center;
        justify-content: center;
        z-index: 100;
        cursor: pointer;
        transform: scale(0.8);
        margin-right: 20px;
}
#header .hamburger:after {
        position: absolute;
        content: '';
        height: 100%;
        width: 100%;
        border-radius: 50%;
        border: 3px solid white;
        animation: hamburger_puls 1s ease infinite;
}
#header .hamburger .bar {
	height: 2px;
	width: 30px;
	position: relative;
	background-color: white;
	z-index: -1;
}
#header .hamburger .bar::after,
#header .hamburger .bar::before {
	content: '';
	position: absolute;
	height: 100%;
	width: 100%;
	left: 0;
	background-color: white;
	transition: 0.3s ease;
	transition-property: top, bottom;
}
#header .hamburger .bar::after {
	top: 8px;
}
#header .hamburger .bar::before {
	bottom: 8px;
}
```

```
#header .hamburger.active .bar::before {
	bottom: 0;
}
#header .hamburger.active .bar::after {
	top: 0;
}
/* End Header section */

/* Hero Section */
#hero {
	background-image: url(../images/hero-bg.png);
	background-size: cover;
	background-position: top center;
	position: relative;
	z-index: 1;
}
#hero::after {
	content: '';
	position: absolute;
	left: 0;
	top: 0;
	height: 100%;
	width: 100%;
```

```
        background-color: black;

        opacity: 0.7;

        z-index: -1;

}

#hero .hero {

        max-width: 1200px;

        margin: 0 auto;

        padding: 0 50px;

        justify-content: flex-start;

}

#hero h1 {

        display: block;

        width: fit-content;

        font-size: 4rem;

        position: relative;

        color: transparent;

        animation: text_reveal 0.5s ease forwards;

        animation-delay: 1s;

}

#hero h1:nth-child(1) {

        animation-delay: 1s;

}

#hero h1:nth-child(2) {
	animation-delay: 2s;
}
#hero h1:nth-child(3) {
	animation: text_reveal_name 0.5s ease forwards;
	animation-delay: 3s;
}
#hero h1 span {
	position: absolute;
	top: 0;
	left: 0;
	height: 100%;
	width: 0;
	background-color: crimson;
	animation: text_reveal_box 1s ease;
	animation-delay: 0.5s;
}
#hero h1:nth-child(1) span {
	animation-delay: 0.5s;
}
#hero h1:nth-child(2) span {
	animation-delay: 1.5s;
}
#hero h1:nth-child(3) span {
        animation-delay: 2.5s;
}

/* End Hero Section */

/* Services Section */
#services .services {
        flex-direction: column;
        text-align: center;
        max-width: 1500px;
        margin: 0 auto;
        padding: 100px 0;
}
#services .service-top {
        max-width: 500px;
        margin: 0 auto;
}
#services .service-bottom {
        display: flex;
        align-items: center;
        justify-content: center;
        flex-wrap: wrap;
        margin-top: 50px;
```

```
}
#services .service-item {
	flex-basis: 80%;
	display: flex;
	align-items: flex-start;
	justify-content: center;
	flex-direction: column;
	padding: 30px;
	border-radius: 10px;
	background-image: url('../images/img-1.png');
	background-size: cover;
	margin: 10px 5%;
	position: relative;
	z-index: 1;
	overflow: hidden;
}
#services .service-item::after {
	content: '';
	position: absolute;
	left: 0;
	top: 0;
	height: 100%;
	width: 100%;
```

```
        background-image: linear-gradient(60deg, #29323c 0%, #485563 100%);
        opacity: 0.9;
        z-index: -1;
}
#services .service-bottom .icon {
        height: 80px;
        width: 80px;
        margin-bottom: 20px;
}
#services .service-item h2 {
        font-size: 2rem;
        color: white;
        margin-bottom: 10px;
        text-transform: uppercase;
}
#services .service-item p {
        color: white;
        text-align: left;
}
/* End Services Section */

/* Projects section */
#projects .projects {
        flex-direction: column;
        max-width: 1200px;
        margin: 0 auto;
        padding: 100px 0;
}
#projects .projects-header h1 {
        margin-bottom: 50px;
}
#projects .all-projects {
        display: flex;
        align-items: center;
        justify-content: center;
        flex-direction: column;
}
#projects .project-item {
        display: flex;
        align-items: center;
        justify-content: center;
        flex-direction: column;
        width: 80%;
        margin: 20px auto;
        overflow: hidden;
        border-radius: 10px;
```

```css
}
#projects .project-info {
	padding: 30px;
	flex-basis: 50%;
	height: 100%;
	display: flex;
	align-items: flex-start;
	justify-content: center;
	flex-direction: column;
	background-image: linear-gradient(60deg, #29323c 0%, #485563 100%);
	color: white;
}
#projects .project-info h1 {
	font-size: 4rem;
	font-weight: 500;
}
#projects .project-info h2 {
	font-size: 1.8rem;
	font-weight: 500;
	margin-top: 10px;
}
#projects .project-info p {
	color: white;
```

```
}

#projects .project-img {

	flex-basis: 50%;

	height: 300px;

	overflow: hidden;

	position: relative;

}

#projects .project-img:after {

	content: '';

	position: absolute;

	left: 0;

	top: 0;

	height: 100%;

	width: 100%;

	background-image: linear-gradient(60deg, #29323c 0%, #485563 100%);

	opacity: 0.7;

}

#projects .project-img img {

	transition: 0.3s ease transform;

}

#projects .project-item:hover .project-img img {

	transform: scale(1.1);

}
```

```
/* End Projects section */


/* About Section */
#about .about {
	flex-direction: column-reverse;
	text-align: center;
	max-width: 1200px;
	margin: 0 auto;
	padding: 100px 20px;
}
#about .col-left {
	width: 250px;
	height: 360px;
}
#about .col-right {
	width: 100%;
}
#about .col-right h2 {
	font-size: 1.8rem;
	font-weight: 500;
	letter-spacing: 0.2rem;
	margin-bottom: 10px;
}
```

```
#about .col-right p {

        margin-bottom: 20px;

}

#about .col-right .cta {

        color: black;

        margin-bottom: 50px;

        padding: 10px 20px;

        font-size: 2rem;

}

#about .col-left .about-img {

        height: 100%;

        width: 100%;

        position: relative;

        border: 10px solid white;

}

#about .col-left .about-img::after {

        content: '';

        position: absolute;

        left: -33px;

        top: 19px;

        height: 98%;

        width: 98%;

        border: 7px solid crimson;
        z-index: -1;
}
/* End About Section */


/* contact Section */
#contact .contact {
        flex-direction: column;
        max-width: 1200px;
        margin: 0 auto;
        width: 90%;
}
#contact .contact-items {
        /* max-width: 400px; */
        width: 100%;
}
#contact .contact-item {
        width: 80%;
        padding: 20px;
        text-align: center;
        border-radius: 10px;
        padding: 30px;
        margin: 30px;
        display: flex;
```

```
    justify-content: center;
    align-items: center;
    flex-direction: column;
    box-shadow: 0px 0px 18px 0 #0000002c;
    transition: 0.3s ease box-shadow;
}
#contact .contact-item:hover {
    box-shadow: 0px 0px 5px 0 #0000002c;
}
#contact .icon {
    width: 70px;
    margin: 0 auto;
    margin-bottom: 10px;
}
#contact .contact-info h1 {
    font-size: 2.5rem;
    font-weight: 500;
    margin-bottom: 5px;
}
#contact .contact-info h2 {
    font-size: 1.3rem;
    line-height: 2rem;
    font-weight: 500;
}
/*End contact Section */
```

```
/* Footer */
#footer {
        background-image: linear-gradient(60deg, #29323c 0%, #485563 100%);
}
#footer .footer {
        min-height: 200px;
        flex-direction: column;
        padding-top: 50px;
        padding-bottom: 10px;
}
#footer h2 {
        color: white;
        font-weight: 500;
        font-size: 1.8rem;
        letter-spacing: 0.1rem;
        margin-top: 10px;
        margin-bottom: 10px;
}
#footer .social-icon {
        display: flex;
        margin-bottom: 30px;
}
#footer .social-item {
        height: 50px;
```

```
	width: 50px;
	margin: 0 5px;
}
#footer .social-item img {
	filter: grayscale(1);
	transition: 0.3s ease filter;
}
#footer .social-item:hover img {
	filter: grayscale(0);
}
#footer p {
	color: white;
	font-size: 1.3rem;
}
/* End Footer */

/* Keyframes */
@keyframes hamburger_puls {
	0% {
		opacity: 1;
		transform: scale(1);
	}
	100% {
		opacity: 0;
		transform: scale(1.4);
```

```
    }
}
@keyframes text_reveal_box {
    50% {
        width: 100%;
        left: 0;
    }
    100% {
        width: 0;
        left: 100%;
    }
}
@keyframes text_reveal {
    100% {
        color: white;
    }
}
@keyframes text_reveal_name {
    100% {
        color: crimson;
        font-weight: 500;
    }
}
/* End Keyframes */
/* Media Query For Tablet */
@media only screen and (min-width: 768px) {
	.cta {
		font-size: 2.5rem;
		padding: 20px 60px;
	}
	h1.section-title {
		font-size: 6rem;
	}

	/* Hero */
	#hero h1 {
		font-size: 7rem;
	}
	/* End Hero */

	/* Services Section */
	#services .service-bottom .service-item {
		flex-basis: 45%;
		margin: 2.5%;
	}
	/* End Services Section */

	/* Project */
	#projects .project-item {
```

```
	flex-direction: row;
}
#projects .project-item:nth-child(even) {
	flex-direction: row-reverse;
}
#projects .project-item {
	height: 400px;
	margin: 0;
	width: 100%;
	border-radius: 0;
}
#projects .all-projects .project-info {
	height: 100%;
}
#projects .all-projects .project-img {
	height: 100%;
}
/* End Project */

/* About */
#about .about {
	flex-direction: row;
}
#about .col-left {
	width: 600px;
```

```
        height: 400px;
        padding-left: 60px;
}
#about .about .col-left .about-img::after {
        left: -45px;
        top: 34px;
        height: 98%;
        width: 98%;
        border: 10px solid crimson;
}
#about .col-right {
        text-align: left;
        padding: 30px;
}
#about .col-right h1 {
        text-align: left;
}
/* End About */


  /* contact  */
#contact .contact {
        flex-direction: column;
        padding: 100px 0;
        align-items: center;
```

```
	justify-content: center;
	min-width: 20vh;
}
#contact .contact-items {
	width: 100%;
	display: flex;
	flex-direction: row;
	justify-content: space-evenly;
	margin: 0;
}
#contact .contact-item {
	width: 30%;
	margin: 0;
	flex-direction: row;
}
#contact .contact-item .icon {
	height: 100px;
	width: 100px;
}
#contact .contact-item .icon img {
	object-fit: contain;
}
#contact .contact-item .contact-info {
	width: 100%;
	text-align: left;
```

```
        padding-left: 20px;
    }
    /* End contact  */
}
/* End Media Query For Tablet */

/* Media Query For Desktop */
@media only screen and (min-width: 1200px) {
    /* header */
    #header .hamburger {
        display: none;
    }
    #header .nav-list ul {
        position: initial;
        display: block;
        height: auto;
        width: fit-content;
        background-color: transparent;
    }
    #header .nav-list ul li {
        display: inline-block;
    }
    #header .nav-list ul li a {
        font-size: 1.8rem;
    }
```

```
        #header .nav-list ul a:after {
                display: none;
        }
        /* End header */

        #services .service-bottom .service-item {
                flex-basis: 22%;
                margin: 1.5%;
        }
}
/* End  Media Query For Desktop */
```

today expenses.html

```
{% extends 'base.html' %}


{% block body %}
<div class="container ">




<div class="row">
    <div class="col-md-5">
        <h3 class="mt-5">Today Expense Breakdown</h3>
        <div class="card shadow mb-2 bg-white rounded-pill">
            <div class="card-body ">
            <div class="row">
                <div class="col-md-6">TIME</div>
                <div class="col-md-6">  AMOUNT   </div>
             </div>
            </div>
        </div>
        {% for row in texpense %}

        <div class="card shadow mb-2 bg-white rounded-bottom">
            <div class="card-body ">
            <div class="row">
                <div id ="ttime" class="col-md-6">{{row [0]}}</div>
                <div id="tamount" class="col-md-6">   {{row[1] }}   </div>
             </div>
            </div>
        </div>



      {% endfor %}
    </div>
    </div>
    <section>
```

```
<div class="row">
    <div class="col-md-6">
        <h3 class="mt-5">Expense Breakdown BY Category</h3>

        <div class="card shadow mb-2 bg-white rounded-bottom">
            <div class="card-body ">
            <div class="row">
                <div class="col-md-6">Food</div>
                <div id="tfood" class="col-md-6">   {{ t_food}}   </div>
             </div>
            </div>
        </div>

        <div class="card shadow mb-2 bg-white rounded">
         <div class="card-body">
         <div class="row">
             <div class="col-md-6">Entertainment</div>
             <div id="tentertainment" class="col-md-6"> {{ t_entertainment}}   </div>
          </div>
         </div>
     </div>


     <div class="card shadow mb-2 bg-white rounded">
         <div class="card-body">
    <div class="row">
        <div class="col-md-6">Business</div>
        <div id="tbusiness" class="col-md-6"> {{t_business}}   </div>
     </div>
    </div>
</div>



<div class="card shadow mb-2 bg-white rounded">
    <div class="card-body">
    <div class="row">
        <div  class="col-md-6">Rent</div>
        <div id="trent" class="col-md-6"> {{  t_rent }}  </div>
     </div>
    </div>
</div>



<div class="card shadow mb-2 bg-white rounded">
    <div class="card-body">
    <div class="row">
        <div class="col-md-6">EMI</div>
        <div id="temi"  class="col-md-6">{{ t_EMI }}   </div>
     </div>
    </div>
```

```
 </div>



 <div class="card shadow mb-2 bg-white rounded">
     <div class="card-body">
     <div class="row">
         <div class="col-md-6">Other</div>
         <div id="tother" class="col-md-6"> {{ t_other}}</div>
      </div>
     </div>
 </div>

 <div class="card shadow mb-2 btn-outline-danger rounded-pill">
     <div class="card-body">
     <div class="row">
         <div class="col-md-6">Total</div>
         <div class="col-md-6">₹ {{total}}  </div>
      </div>
     </div>
 </div>

</div>
<div class="col-md-6">
 <canvas id="myChart" width="400" height="400"></canvas>
 <script>
```

```
    let food = document.getElementById('tfood').innerHTML
    let entertainment = document.getElementById('tentertainment').innerHTML
    let business = document.getElementById('tbusiness').innerHTML
    let rent = document.getElementById('trent').innerHTML
    let emi = document.getElementById('temi').innerHTML
    let other = document.getElementById('tother').innerHTML
var ctx = document.getElementById('myChart').getContext('2d');
var myChart = new Chart(ctx, {
    type: 'doughnut',
    data: {
        labels: ['Food', 'Entertainment', 'Business', 'Rent', 'EMI', 'Other'],
        datasets: [{
            label: 'Expenses Chart',
            data: [food, entertainment, business, rent, emi, other],
            backgroundColor: [
            'rgb(255, 99, 132)',
            'rgb(0, 0, 0)',
            'rgb(255, 205, 86)',
            'rgb(201, 203, 207)',
               'rgb(54, 162, 235)',
                'rgb(215, 159, 64)'
            ],


        }]
    },
            options: {
                responsive: true,
                plugins: {
      legend: {
        position: 'bottom',
      },
      title: {
        display: true,
        text: 'EXPENSE BREAKDOWN'
      }
    }
            }
        });




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

```
</section>



</div>


{% endblock %}
```

app.py

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""


from flask import Flask, render_template, request, redirect, session
# from flask_mysqldb import MySQL
# import MySQLdb.cursors
import re


from flask_db2 import DB2
import ibm_db
import ibm_db_dbi
from sendemail import sendgridmail,sendmail
```

```
# from gevent.pywsgi import WSGIServer
import os



app = Flask(__name__)


app.secret_key = 'a'


# app.config['MYSQL_HOST'] = 'remotemysql.com'
# app.config['MYSQL_USER'] = 'D2DxDUPBii'
# app.config['MYSQL_PASSWORD'] = 'r8XBO4GsMz'
# app.config['MYSQL_DB'] = 'D2DxDUPBii'
"""
dsn_hostname = "3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"
dsn_uid = "sbb93800"
dsn_pwd = "wobsVLm6ccFxcNLe"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "31498"
dsn_protocol = "tcpip"


dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"

    "PORT={3};"

    "PROTOCOL={4};"

    "UID={5};"

    "PWD={6};"

).format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol, dsn_uid,
dsn_pwd)

"""

# app.config['DB2_DRIVER'] = '{IBM DB2 ODBC DRIVER}'

app.config['database'] = 'bludb'

app.config['hostname']                    =                    '3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'

app.config['port'] = '31498'

app.config['protocol'] = 'tcpip'

app.config['uid'] = 'sbb93800'

app.config['pwd'] = 'wobsVLm6ccFxcNLe'

app.config['security'] = 'SSL'

try:

    mysql = DB2(app)

conn_str='database=bludb;hostname=3883e7e4-18f5-4afe-be8c
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=31498;protocol
=tcpip;\uid=sbb93800;pwd=wobsVLm6ccFxcNLe;security=SSL'

    ibm_db_conn = ibm_db.connect(conn_str,'','')



    print("Database connected without any error !!")

except:
    print("IBM DB Connection error   :      " + DB2.conn_errormsg())

# app.config['']

# mysql = MySQL(app)



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

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




#SIGN--UP--OR--REGISTER



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

```
@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    print("Break point1")
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

        print("Break point2" + "name: " + username + "------" + email + "------" + password)

        try:
            print("Break point3")
            connectionID = ibm_db_dbi.connect(conn_str, '', '')
            cursor = connectionID.cursor()
            print("Break point4")
        except:
            print("No connection Established")

        # cursor = mysql.connection.cursor()
        # with app.app_context():
        #     print("Break point3")
        #     cursor = ibm_db_conn.cursor()
        #     print("Break point4")


        print("Break point5")
        sql = "SELECT * FROM register WHERE username = ?"
        stmt = ibm_db.prepare(ibm_db_conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        result = ibm_db.execute(stmt)
        print(result)
        account = ibm_db.fetch_row(stmt)
        print(account)


        param = "SELECT * FROM register WHERE username = " + "\'" + username + "\'"
        res = ibm_db.exec_immediate(ibm_db_conn, param)
        print("---- ")
        dictionary = ibm_db.fetch_assoc(res)
        while dictionary != False:
            print("The ID is : ", dictionary["USERNAME"])
            dictionary = ibm_db.fetch_assoc(res)


        # dictionary = ibm_db.fetch_assoc(result)
        # cursor.execute(stmt)


        # account = cursor.fetchone()
        # print(account)
```

```
        # while ibm_db.fetch_row(result) != False:
        #     # account = ibm_db.result(stmt)
        #     print(ibm_db.result(result, "username"))

        # print(dictionary["username"])
        print("break point 6")
        if account:
            msg = 'Username already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            sql2 = "INSERT INTO register (username, email,password) VALUES (?, ?, ?)"
            stmt2 = ibm_db.prepare(ibm_db_conn, sql2)
            ibm_db.bind_param(stmt2, 1, username)
            ibm_db.bind_param(stmt2, 2, email)
            ibm_db.bind_param(stmt2, 3, password)
            ibm_db.execute(stmt2)
            # cursor.execute('INSERT INTO register VALUES (NULL, % s, % s, % s)', (username, email,password))
            # mysql.connection.commit()
            msg = 'You have successfully registered !'
        return render_template('signup.html', msg = msg)
```

```
 #LOGIN--PAGE


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


@app.route('/login',methods =['GET', 'POST'])
def login():
    global userid
    msg = ''



    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        # cursor = mysql.connection.cursor()
              # cursor.execute('SELECT * FROM register WHERE username = % s AND
password = % s', (username, password ),)
        # account = cursor.fetchone()
        # print (account)
        sql = "SELECT * FROM register WHERE username = ? and password = ?"

        stmt = ibm_db.prepare(ibm_db_conn, sql)

        ibm_db.bind_param(stmt, 1, username)

        ibm_db.bind_param(stmt, 2, password)

        result = ibm_db.execute(stmt)

        print(result)

        account = ibm_db.fetch_row(stmt)

        print(account)


         param = "SELECT * FROM register WHERE username = " + "\'" + username + "\'"
+ " and password = " + "\'" + password + "\'"

        res = ibm_db.exec_immediate(ibm_db_conn, param)

        dictionary = ibm_db.fetch_assoc(res)


        # sendmail("hello sakthi","sivasakthisairam@gmail.com")


        if account:

            session['loggedin'] = True

            session['id'] = dictionary["ID"]

            userid = dictionary["ID"]

            session['username'] = dictionary["USERNAME"]

            session['email'] = dictionary["EMAIL"]


            return redirect('/home')

        else:
            msg = 'Incorrect username / password !'

    return render_template('login.html', msg = msg)









#ADDING----DATA



@app.route("/add")
def adding():
    return render_template('add.html')



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

    date = request.form['date']
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category = request.form['category']


    print(date)
    p1 = date[0:10]
    p2 = date[11:13]
    p3 = date[14:]
    p4 = p1 + "-" + p2 + "." + p3 + ".00"
    print(p4)
    # cursor = mysql.connection.cursor()
    # cursor.execute('INSERT INTO expenses VALUES (NULL,  % s, % s, % s, % s, % s, % s)', (session['id'] ,date, expensename, amount, paymode, category))
    # mysql.connection.commit()
    # print(date + " " + expensename + " " + amount + " " + paymode + " " + category)


    sql = "INSERT INTO expenses (userid, date, expensename, amount, paymode, category) VALUES (?, ?, ?, ?, ?, ?)"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, session['id'])
    ibm_db.bind_param(stmt, 2, p4)
    ibm_db.bind_param(stmt, 3, expensename)
    ibm_db.bind_param(stmt, 4, amount)
    ibm_db.bind_param(stmt, 5, paymode)
    ibm_db.bind_param(stmt, 6, category)
    ibm_db.execute(stmt)

    print("Expenses added")

    # email part

    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)
    total=0

    for x in expense:

        total += x[4]


    param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER BY id DESC LIMIT 1"

    res = ibm_db.exec_immediate(ibm_db_conn, param)

    dictionary = ibm_db.fetch_assoc(res)

    row = []

    s = 0

    while dictionary != False:

        temp = []

        temp.append(dictionary["LIMITSS"])

        row.append(temp)

        dictionary = ibm_db.fetch_assoc(res)

        s = temp[0]


    if total > int(s):

        msg = "Hello " + session['username'] + " , " + "you have crossed the monthly limit of Rs. " + s + "/- !!!" + "\n" + "Thank you, " + "\n" + "Team Personal Expense Tracker."

        sendmail(msg,session['email'])


    return redirect("/display")
```

```
#DISPLAY---graph


@app.route("/display")
def display():
    print(session["username"],session['id'])


    # cursor = mysql.connection.cursor()
      # cursor.execute('SELECT * FROM expenses WHERE userid = % s AND date
ORDER BY `expenses`.`date` DESC',(str(session['id'])))
    # expense = cursor.fetchall()


      param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + "
ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    return render_template('display.html' ,expense = expense)





#delete---the--data

@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
def delete(id):
    #  cursor = mysql.connection.cursor()
    #  cursor.execute('DELETE FROM expenses WHERE  id = {0}'.format(id))
    #  mysql.connection.commit()

    param = "DELETE FROM expenses WHERE  id = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)

    print('deleted successfully')
    return redirect("/display")
```

```
#UPDATE---DATA


@app.route('/edit/<id>', methods = ['POST', 'GET' ])

def edit(id):

    # cursor = mysql.connection.cursor()

    # cursor.execute('SELECT * FROM expenses WHERE  id = %s', (id,))

    # row = cursor.fetchall()


    param = "SELECT * FROM expenses WHERE  id = " + id

    res = ibm_db.exec_immediate(ibm_db_conn, param)

    dictionary = ibm_db.fetch_assoc(res)

    row = []

    while dictionary != False:

        temp = []

        temp.append(dictionary["ID"])

        temp.append(dictionary["USERID"])

        temp.append(dictionary["DATE"])

        temp.append(dictionary["EXPENSENAME"])

        temp.append(dictionary["AMOUNT"])

        temp.append(dictionary["PAYMODE"])

        temp.append(dictionary["CATEGORY"])

        row.append(temp)

        print(temp)

        dictionary = ibm_db.fetch_assoc(res)
    print(row[0])

    return render_template('edit.html', expenses = row[0])




@app.route('/update/<id>', methods = ['POST'])

def update(id):

  if request.method == 'POST' :


      date = request.form['date']

      expensename = request.form['expensename']

      amount = request.form['amount']

      paymode = request.form['paymode']

      category = request.form['category']


    #   cursor = mysql.connection.cursor()

     #   cursor.execute("UPDATE `expenses` SET `date` = % s , `expensename` = % s ,
`amount` = % s, `paymode` = % s, `category` = % s WHERE `expenses`.`id` = % s
",(date, expensename, amount, str(paymode), str(category),id))

    #   mysql.connection.commit()


      p1 = date[0:10]

      p2 = date[11:13]

      p3 = date[14:]
      p4 = p1 + "-" + p2 + "." + p3 + ".00"


       sql = "UPDATE expenses SET date = ? , expensename = ? , amount = ?, paymode
= ?, category = ? WHERE id = ?"
      stmt = ibm_db.prepare(ibm_db_conn, sql)
      ibm_db.bind_param(stmt, 1, p4)
      ibm_db.bind_param(stmt, 2, expensename)
      ibm_db.bind_param(stmt, 3, amount)
      ibm_db.bind_param(stmt, 4, paymode)
      ibm_db.bind_param(stmt, 5, category)
      ibm_db.bind_param(stmt, 6, id)
      ibm_db.execute(stmt)


      print('successfully updated')
      return redirect("/display")









   #limit
@app.route("/limit" )
```

```
def limit():
       return redirect('/limitn')


@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
     if request.method == "POST":
         number= request.form['number']
        #  cursor = mysql.connection.cursor()
           #   cursor.execute('INSERT INTO limits VALUES (NULL, % s, % s) ',(session['id'],
number))
        #  mysql.connection.commit()


         sql = "INSERT INTO limits (userid, limitss) VALUES (?, ?)"
         stmt = ibm_db.prepare(ibm_db_conn, sql)
         ibm_db.bind_param(stmt, 1, session['id'])
         ibm_db.bind_param(stmt, 2, number)
         ibm_db.execute(stmt)


         return redirect('/limitn')




@app.route("/limitn")
def limitn():
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT limitss FROM `limits` ORDER BY `limits`.`id` DESC LIMIT
1')
    # x= cursor.fetchone()
    # s = x[0]

     param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER BY id DESC LIMIT 1"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = " /-"
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMITSS"])
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[0]

    return render_template("limit.html" , y= s)

#REPORT

@app.route("/today")
def today():
    #   cursor = mysql.connection.cursor()
    #   cursor.execute('SELECT TIME(date)   , amount FROM expenses  WHERE userid
= %s AND DATE(date) = DATE(NOW()) ',(str(session['id'])))

   #   texpense = cursor.fetchall()

   #   print(texpense)


       param1 = "SELECT TIME(date) as tn, amount FROM expenses WHERE userid = " + str(session['id']) + " AND DATE(date) = DATE(current timestamp) ORDER BY date DESC"

      res1 = ibm_db.exec_immediate(ibm_db_conn, param1)

      dictionary1 = ibm_db.fetch_assoc(res1)

      texpense = []


      while dictionary1 != False:

         temp = []

         temp.append(dictionary1["TN"])

         temp.append(dictionary1["AMOUNT"])

         texpense.append(temp)

         print(temp)

         dictionary1 = ibm_db.fetch_assoc(res1)


   #   cursor = mysql.connection.cursor()

      #    cursor.execute('SELECT * FROM expenses WHERE userid = % s AND DATE(date) = DATE(NOW()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))

   #   expense = cursor.fetchall()


      param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND DATE(date) = DATE(current timestamp) ORDER BY date DESC"
```

```
res = ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res)
expense = []
while dictionary != False:
    temp = []
    temp.append(dictionary["ID"])
    temp.append(dictionary["USERID"])
    temp.append(dictionary["DATE"])
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    expense.append(temp)
    print(temp)
    dictionary = ibm_db.fetch_assoc(res)


total=0
t_food=0
t_entertainment=0
t_business=0
t_rent=0
t_EMI=0
t_other=0
```

```
for x in expense:
    total += x[4]
    if x[6] == "food":
        t_food += x[4]

    elif x[6] == "entertainment":
        t_entertainment  += x[4]

    elif x[6] == "business":
        t_business  += x[4]
    elif x[6] == "rent":
        t_rent  += x[4]

    elif x[6] == "EMI":
        t_EMI  += x[4]

    elif x[6] == "other":
        t_other  += x[4]

print(total)

print(t_food)
print(t_entertainment)
print(t_business)
```

```
    print(t_rent)

    print(t_EMI)

    print(t_other)




      return render_template("today.html", texpense = texpense, expense = expense,
total = total ,

                         t_food = t_food,t_entertainment =  t_entertainment,

                         t_business = t_business,  t_rent =  t_rent,

                         t_EMI =  t_EMI,  t_other =  t_other )



@app.route("/month")

def month():

  #   cursor = mysql.connection.cursor()

     #    cursor.execute('SELECT DATE(date), SUM(amount) FROM expenses WHERE
userid= %s AND MONTH(DATE(date))= MONTH(now()) GROUP BY DATE(date)
ORDER BY DATE(date) ',(str(session['id'])))

  #   texpense = cursor.fetchall()

  #   print(texpense)


      param1 = "SELECT DATE(date) as dt, SUM(amount) as tot FROM expenses
WHERE userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current
timestamp) AND YEAR(date) = YEAR(current timestamp) GROUP BY DATE(date)
ORDER BY DATE(date)"

    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpense = []


    while dictionary1 != False:
        temp = []
        temp.append(dictionary1["DT"])
        temp.append(dictionary1["TOT"])
        texpense.append(temp)
        print(temp)
        dictionary1 = ibm_db.fetch_assoc(res1)



  #   cursor = mysql.connection.cursor()
      #     cursor.execute('SELECT * FROM expenses WHERE userid = % s AND MONTH(DATE(date))= MONTH(now()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))
  #   expense = cursor.fetchall()


     param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)


    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0


    for x in expense:
        total += x[4]
        if x[6] == "food":
            t_food += x[4]
        elif x[6] == "entertainment":
            t_entertainment  += x[4]

        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]

        elif x[6] == "EMI":
            t_EMI  += x[4]

        elif x[6] == "other":
            t_other  += x[4]

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)

        return render_template("today.html", texpense = texpense, expense = expense,
total = total ,
                        t_food = t_food,t_entertainment =  t_entertainment,
                        t_business = t_business,  t_rent =  t_rent,
                        t_EMI =  t_EMI,  t_other =  t_other )


@app.route("/year")
def year():
   #   cursor = mysql.connection.cursor()
    #   cursor.execute('SELECT MONTH(date), SUM(amount) FROM expenses WHERE userid= %s AND YEAR(DATE(date))= YEAR(now()) GROUP BY MONTH(date) ORDER BY MONTH(date) ',(str(session['id'])))
   #   texpense = cursor.fetchall()
   #   print(texpense)


        param1 = "SELECT MONTH(date) as mn, SUM(amount) as tot FROM expenses WHERE userid = " + str(session['id']) + " AND YEAR(date) = YEAR(current timestamp) GROUP BY MONTH(date) ORDER BY MONTH(date)"
     res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
     dictionary1 = ibm_db.fetch_assoc(res1)
     texpense = []


     while dictionary1 != False:
         temp = []
         temp.append(dictionary1["MN"])
         temp.append(dictionary1["TOT"])
         texpense.append(temp)
```

```
          print(temp)

          dictionary1 = ibm_db.fetch_assoc(res1)



    #   cursor = mysql.connection.cursor()

       #     cursor.execute('SELECT * FROM expenses WHERE userid = % s AND YEAR(DATE(date))= YEAR(now()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))

    #   expense = cursor.fetchall()


      param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " AND YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"

      res = ibm_db.exec_immediate(ibm_db_conn, param)

      dictionary = ibm_db.fetch_assoc(res)

      expense = []

      while dictionary != False:

          temp = []

          temp.append(dictionary["ID"])

          temp.append(dictionary["USERID"])

          temp.append(dictionary["DATE"])

          temp.append(dictionary["EXPENSENAME"])

          temp.append(dictionary["AMOUNT"])

          temp.append(dictionary["PAYMODE"])

          temp.append(dictionary["CATEGORY"])

          expense.append(temp)

          print(temp)
          dictionary = ibm_db.fetch_assoc(res)



      total=0

      t_food=0

      t_entertainment=0

      t_business=0

      t_rent=0

      t_EMI=0

      t_other=0



      for x in expense:

          total += x[4]

          if x[6] == "food":

              t_food += x[4]


          elif x[6] == "entertainment":

              t_entertainment  += x[4]


          elif x[6] == "business":

              t_business  += x[4]

          elif x[6] == "rent":

              t_rent  += x[4]
          elif x[6] == "EMI":
              t_EMI  += x[4]


          elif x[6] == "other":
              t_other  += x[4]


      print(total)


      print(t_food)
      print(t_entertainment)
      print(t_business)
      print(t_rent)
      print(t_EMI)
      print(t_other)


          return render_template("today.html", texpense = texpense, expense = expense,
  total = total ,
                      t_food = t_food,t_entertainment =  t_entertainment,
                      t_business = t_business,  t_rent =  t_rent,
                      t_EMI =  t_EMI,  t_other =  t_other )


  #log-out


  @app.route('/logout')
```

```
def logout():
   session.pop('loggedin', None)
   session.pop('id', None)
   session.pop('username', None)
   session.pop('email', None)
   return render_template('home.html')



port = os.getenv('VCAP_APP_PORT', '8080')
if __name__ == "__main__":
    app.secret_key = os.urandom(12)
    app.run(debug=True, host='0.0.0.0', port=port)
```

7.2 Feature 2

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

.

SendMail.py

```
import smtplib

import sendgrid as sg

import os

from sendgrid.helpers.mail import Mail, Email, To, Content

SUBJECT = "expense tracker"

s = smtplib.SMTP('smtp.gmail.com', 587)

def sendmail(TEXT,email):

    print("sorry we cant process your candidature")

    s = smtplib.SMTP('smtp.gmail.com', 587)

    s.starttls()

    # s.login("il.tproduct8080@gmail.com", "oms@1Ram")

    s.login("tproduct8080@gmail.com", "lxixbmpnexbkiemh")

    message  = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)

    # s.sendmail("il.tproduct8080@gmail.com", email, message)

    s.sendmail("il.tproduct8080@gmail.com", email, message)

    s.quit()

def sendgridmail(user,TEXT):
      # from_email = Email("shridhartp24@gmail.com")
    from_email = Email("tproduct8080@gmail.com")
    to_email = To(user)
    subject = "Sending with SendGrid is Fun"
    content = Content("text/plain",TEXT)
    mail = Mail(from_email, to_email, subject, content)
    # Get a JSON-ready representation of the Mail object
    mail_json = mail.get()
    # Send an HTTP POST request to /mail/send
    response = sg.client.mail.send.post(request_body=mail_json)
    print(response.status_code)
    print(response.headers)
```

7.3 Database Schema

Tables :

REGISTER

```
 id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
username VARCHAR(255) NOT NULL,
email VARCHAR(255) NOT NULL,
password VARCHAR(255) NOT NULL
```

EXPENSES

```
id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
 userid  INT NOT NULL,
date TIMESTAMP NOT NULL,
expensename  VARCHAR(255) NOT NULL,
amount  INT NOT NULL,
paymode VARCHAR(255) NOT NULL,
category VARCHAR(255) NOT NULL
```

LIMITS

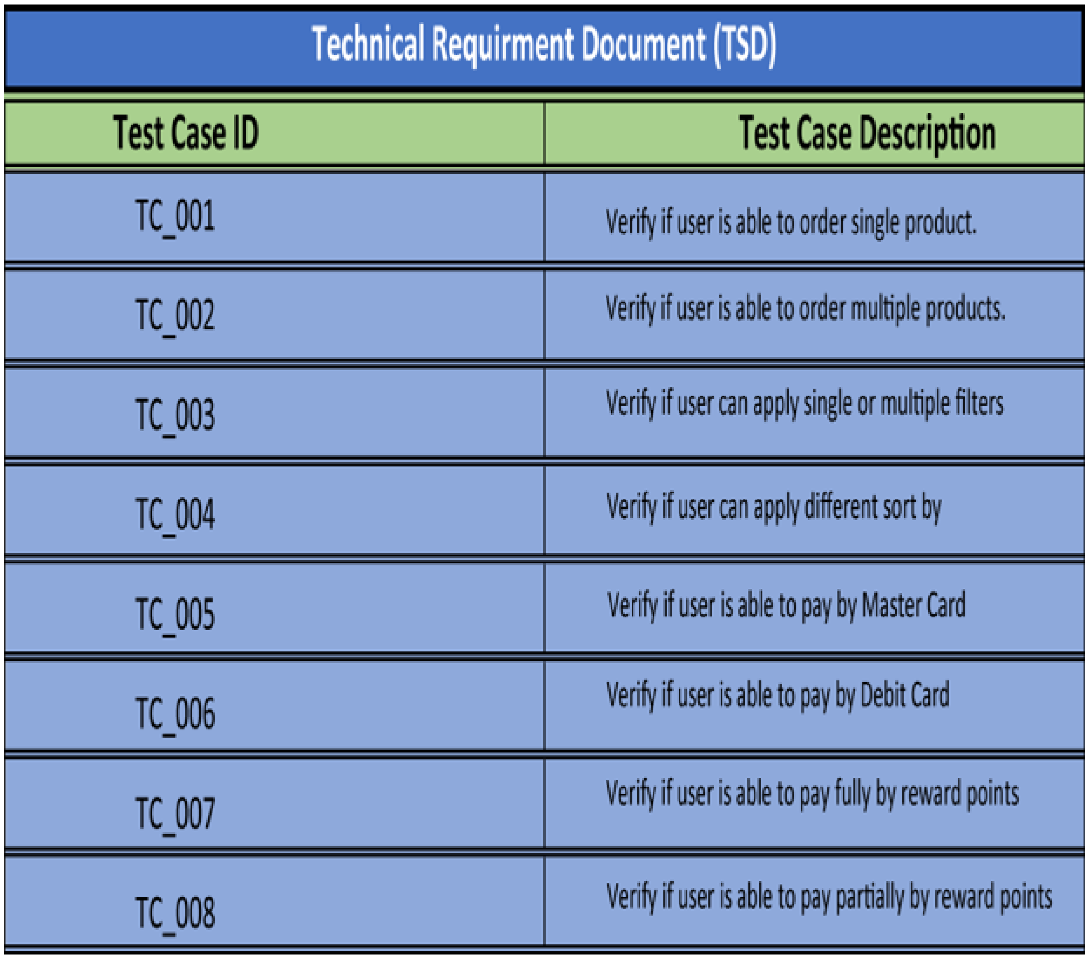
```
id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
userid VARCHAR(255) NOT NULL,
limitss VARCHAR(255) NOT NULL
```

8.TESTING

8.1 Test Cases

Test Case ID	Purpose	Test Cases	Result
TC1	Authentication	Password with length less than 4 characters	Password cannot be lessthan 4 characters
TC2	Authentication	User name with lengthless than 2 characters	User name cannot be less than 2 Characters
TC3	Authentication	Valid user name with minimum 2 characters	User name Accepted
TC4	Authentication	User name left blank	User name cannot be less than 2 characters
TC5	Authentication	Password field left blank	Password cannot be empty
TC6	Authentication	Minimum 4 characters valid password	Password Accepted
TC7	Authentication	Password and Confirm password did not match	Please enter same password
TC8	Authentication	Confirm Password field left blank	Please enter same password

8.2.User Acceptance Testing



Technical Requirment Document (TSD)	
Test Case ID	**Test Case Description**
TC_001	Verify if user is able to order single product.
TC_002	Verify if user is able to order multiple products.
TC_003	Verify if user can apply single or multiple filters
TC_004	Verify if user can apply different sort by
TC_005	Verify if user is able to pay by Master Card
TC_006	Verify if user is able to pay by Debit Card
TC_007	Verify if user is able to pay fully by reward points
TC_008	Verify if user is able to pay partially by reward points

9.RESULTS

9.1.Performance Metrics

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

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

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

iv. Payments & Invoices: Accept and pay from credit cards, debit cards, net banking, mobilewallets, and bank transfers, and track the status of your invoicesand bills in the mobileapp itself. Also,the tracking app sends reminders for payments and automatically matchesthe payments with invoices.

v. Reports: The expense tracking app generates and sends reports to give a detailed insight about profits, losses, budgets, income, balance sheets, etc.,

vi. E-commerce integration: Integrate your expense tracking app with your eCommerce store and track your sales through paymentsreceived via multiplepayments.

vii. Vendors and Contractors: Manage and track all the payments to the vendors and contractors added to the mobile app.

viii. Access control: Increase your team productivity by providing access control to particular usersthrough custom permissions.

ix. Track Projects: Determine project profitability by tracking labor costs, payroll, expenses, etc., of your ongoingproject.

x. Inventory tracking: An expense tracking app can do it all. Right from tracking products or the cost of goods, sending alert notifications when the product is running out of stock or the product is not selling,to purchase orders.

xi. In-depth insights and analytics: Provides in-built tools to generate.

xii. Recurrent Expenses: Rely on your budgeting app to track, streamline, and automate all the recurrent expenses and remind you on a timely basis.

10.ADVANTAGES & DISADVANTAGES

1. **Achieve your business goals** with a tailored mobile app that perfectlyfits your business.
2. **Scale-up** at the pace your businessis growing.
3. Deliver an **outstanding** customerexperience through additionalcontrolover the app.
4. Control the **security**of your businessand customer data.
5. Open **direct marketingchannels** with no extra costs with methodssuch as push notifications.
6. **Boost the productivity** of all the processes within the organization.
7. Increase **efficiency** and **customer satisfaction** with an app aligned totheir needs.
8. **Seamlesslyintegrate** with existing infrastructure.
9. Ability to provide **valuable insights**.

11 .CONCLUSION

From this project,we are able to manageand keep tracking the dailyexpenses as well as income. While making this project, we gained a lot ofexperience of working as a team.We discovered various predictedandunpredicted problems and we enjoyeda lot solving them as a team. We adoptedthings like video tutorials, text tutorials, internetand learningmaterials to makeour project complete.

12.FUTURE SCOPE

The project assists well to record the income and expenses in general.However,this project has some limitations:

1. The application is unable to maintain the backup of data once it is uninstalled.
2. This application does not provide higher decision capability further enhance the capability of this application.
3. Multiple language interface.
4. Provide backup and recoveryof data.
5. Provide better user interface for user.
6. Mobile apps advantage.

13.APPENDIX

Source Code Github Link

https://github.com/IBM-EPBL/IBM-Project-46011-1660734557

Demo Link

https://drive.google.com/file/d/1h1Fo7T2GkPEQNjaRVVx8DWuJozP_4Bv2/view?usp=drivesdk