NALAIYA THIRAN PROJECT – EXPERIENTIAL PROJECT BASED LEARNING

PERSONAL EXPENSE TRACKER APPLICATION

TEAM ID: PNT2022TMID03042

BATCH: B4-4M6E

TEAM MEMBERS:

SHAKTHI DHARUN R - 727720EUIT510
RAJA GURU J - 727719EUIT123
RANGANATHAN R - 727719EUIT125
VISHNU R - 727719EUIT177

INDUSTRY MENTOR NAME: Ms. KUSHBOO

FACULTY MENTOR NAME: Mr. M MOHAMMED MUSTAFA

INTRODUCTION

1.1 Project Overview

This project is based on an expense and income tracking system. This project aims to create an easy, faster and smooth tracking system between the expense and the income. This project also offers some opportunities that will help the user to sustain all financial activities like digital automated diary. So, for the better expense tracking system, we developed our project that will help the users a lot. Most of the people cannot track their expenses and income one way they face a money crisis, in this case daily expense tracker can help the people to track income-expense day to day and making life tension free. Money is the most valuable portion of our daily life and without money we will not last one day on the earth. So using the daily expense tracker application is important to load a happy family. Daily expense tracker helps the user to avoid unexpected expenses and bad financial situations. This Project will save time and provide a responsible lifestyle.

1.2 Purpose

The main reason you should track your expenses is **to identify and eliminate wasteful spending habits in your financial life**. Moreover, consistently tracking your expenses will help you maintain control of your finances and promote better financial habits like saving and investing. Recording your expenses daily can ensure that you are financially aware all year long and not just during tax season. Knowing where your money is going and how much you're spending can improve your spending habits. Plus, you'll have a better idea of where you can allocate money to positively impact your bottom line.

LITERATURE SURVEY

2.1 Existing problem

As a user We face many difficulties in our daily file. In our daily life money is the most important portion and without it we cannot last one day on earth but if we keep on track all financial data then we can overcome this problem. Most of the people cannot track their expenses and income one way they face the money crisis and depression. This situation motivates us to make an android app to track all financial activities. Using the Daily Expense Tracker user can be tracking expenses day to day and making life tension free.

2.2 References

Paper 1

Title: Application for Predictive Recommendation and Visualization of Personal Expenses

Authors: *Darsh Shah, Sanay Shah, Ritik Savani, Dr. Bhavesh Patel, Ashwini Deshmukh*

Paper 2
Title: Utilization of QR Code for Tracking Digital Expenses
Authors: Farhan Putra Salsabil1, Gerardo Axel Lwiantoro2, Gregorius A. N. Aditantyo3

Paper 3
Title: A Novel Expense Tracker using Statistical Analysis
Authors: *Muskaan Sharma, Ayush Bansal, Dr. Raju Ranjan, Shivam Sethi*

Paper 4
Title: Daily Expense Tracker
Authors: Shivam Mehra, Prabhat Parashar

Paper 5
Title: A Review on Budget Estimator Android Application Authors: *Namita Jagtap, Priyanka Joshi, Aditya Kamble*

Paper 6
Title: STUDENT EXPENSE TRACKING APPLICATION
Authors: Saumya Dubey1, Pragya Dubey2, Rigved Rishabh Kumar 3, Aaisha Khatoon4

Paper 7
Title: Time Scheduling and Finance Management: University Student Survival Kit
Authors:*J.L. Yeo, P.S. JosephNg, K.A. Alezabi, H.C. Eaw, K.Y. Phan (2020, IEEE)*

Paper 8
Title: Development of an Application for Expense Accounting
Authors: *Denis E. Yurochkin; Anton A. Horoshiy; Saveliy A. Karpukhin(2021, IEEE)*

2.3 Problem Statement Definition
Personal finance entails all the financial decisions and activities that a Finance app makes life easier by helping the user to manage their finances efficiently.
✓ A personal finance app will not only help them with budgeting and accounting but also give helpful insights about money management.
✓ The price tracker is an web app which runs on all web platforms. It allows person to control all their expenses in an effective manner and

it helps to budget and save money.

✓ This might avoid price range managing problems and offers us green effects on our savings. In everyone's life, cash performs an essential role.

✓ A person who can't control his costs can't efficiently lead a household and satisfy his goals. In the present day global where mobile telephones and laptops have come to be part of living, such an app might be available to address all our costs.

✓ A individual commonly can't preserve music of all his expenses through the conventional pen and paper technique and might miss some of his small expenses and might even miss some bills.

✓ Such a scenario will in no way rise up while we use an app. We could make smooth comparisons through seeing the graphs, that's not possible with inside the rigorous methods.

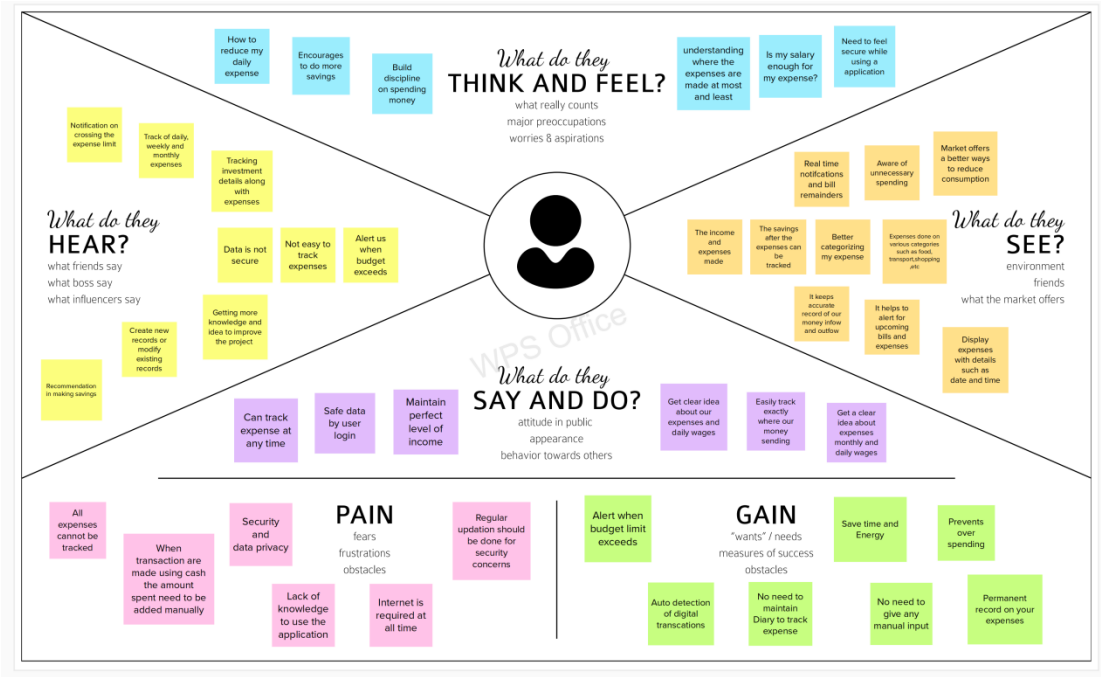
✓ In this scope of this project, we focus on some algorithms for expense Tracking one using pie-chart & bar graphs and another using notifications.

✓ To build a web application which uses a cloud object storage, trained on calculating the expenses, and get the prediction of amount saved when an income is given, by using IBM container registry to efficiently curb out the following constraints:

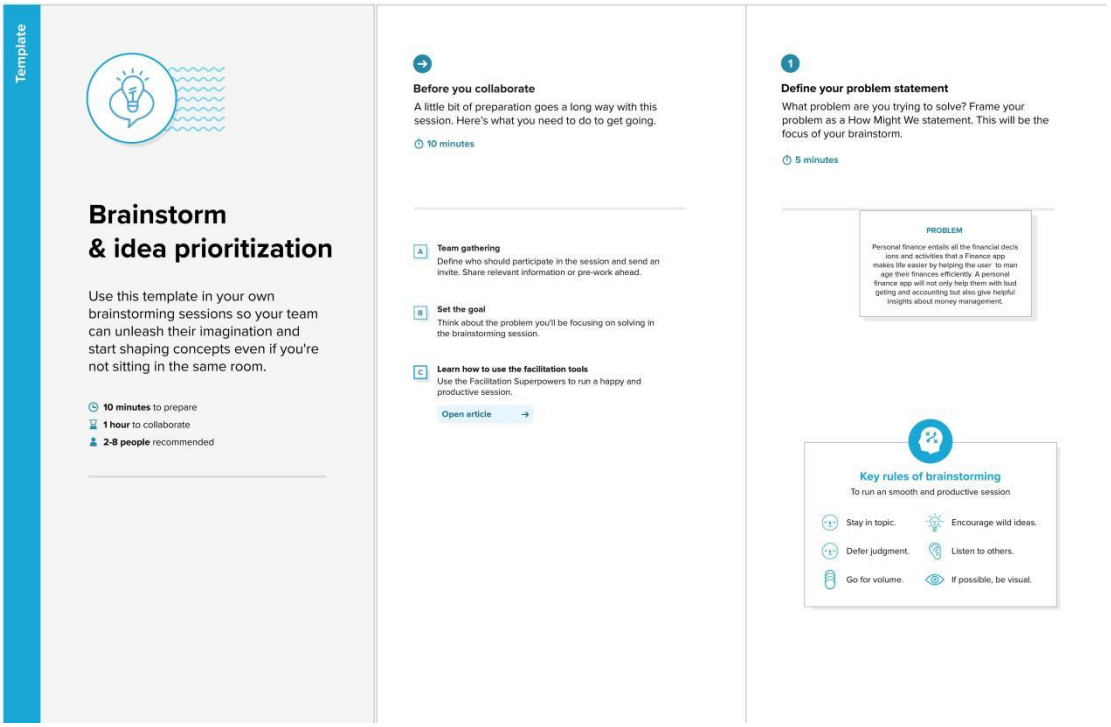
- Tracking expenses in daily basis, which shows how the income is spend.
- By monitoring the spending which make sure all monthly expenses are
- covered.
- By tracking we can see and avoid unnecessary spendings.
- It will help to save money for future use.
- It provides an alert message when it exceeds the limit, and this system
- will help us from suffering at the end of the month.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



2

Brainstorm

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

10 minutes

TIP

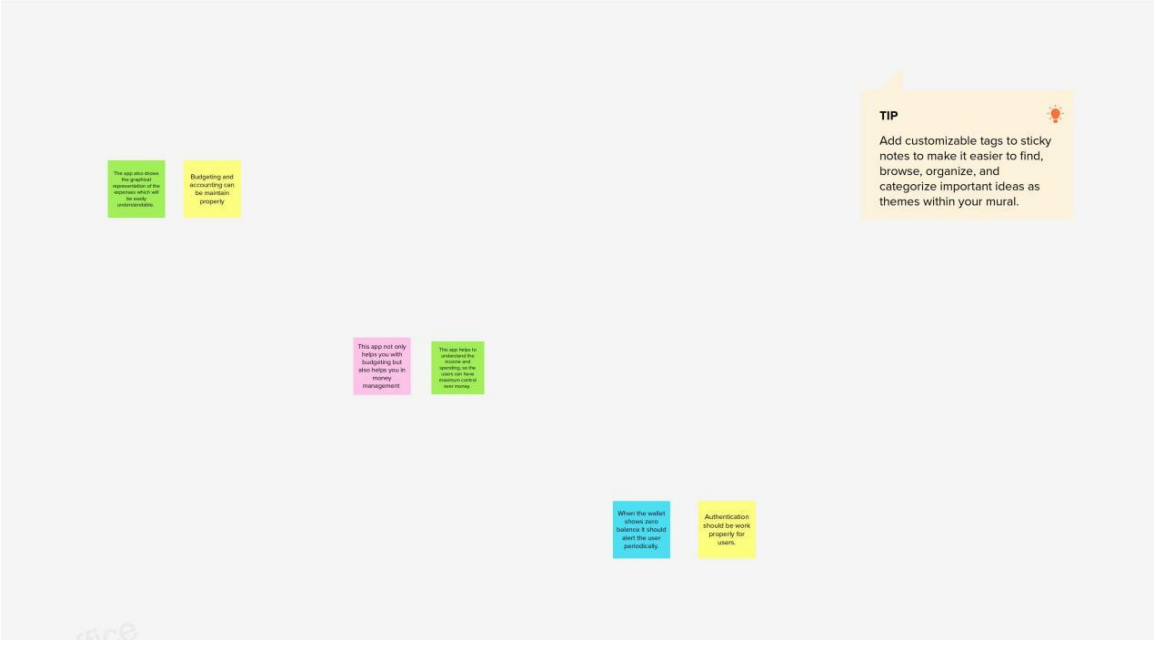
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

3

Group ideas

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

20 minutes

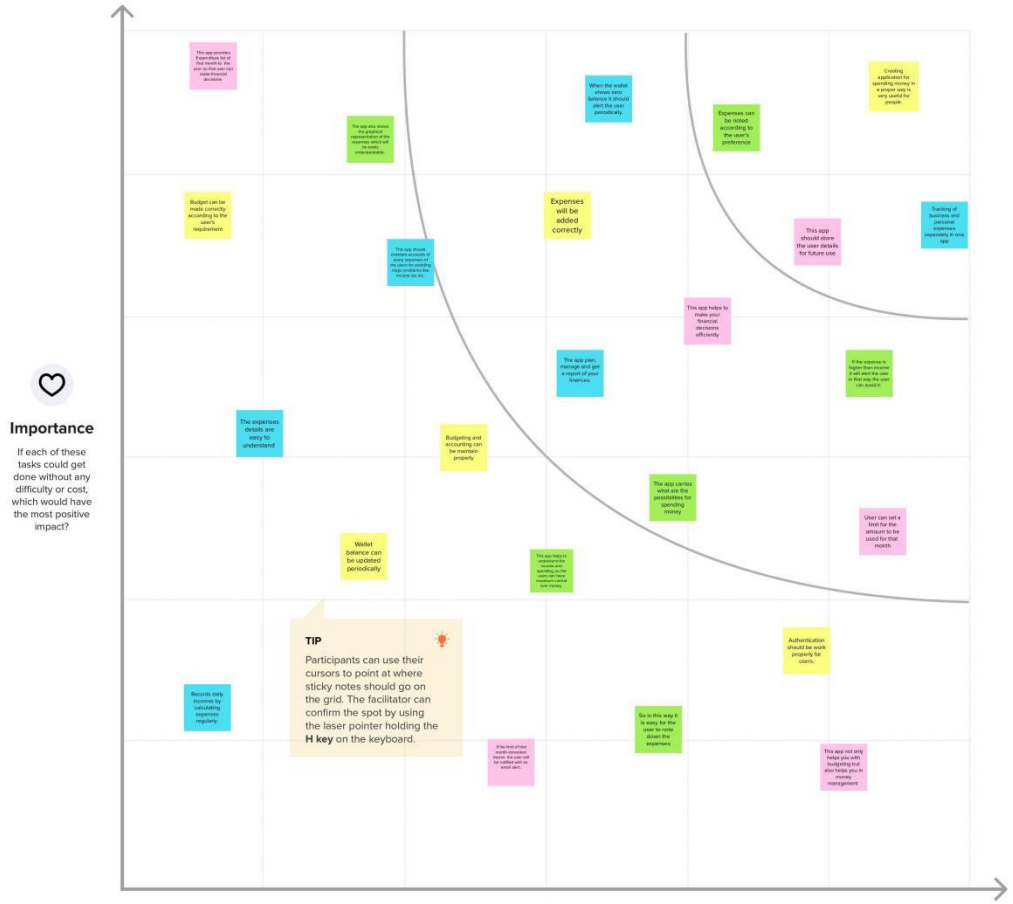


4

Prioritize

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

⏱ 20 minutes

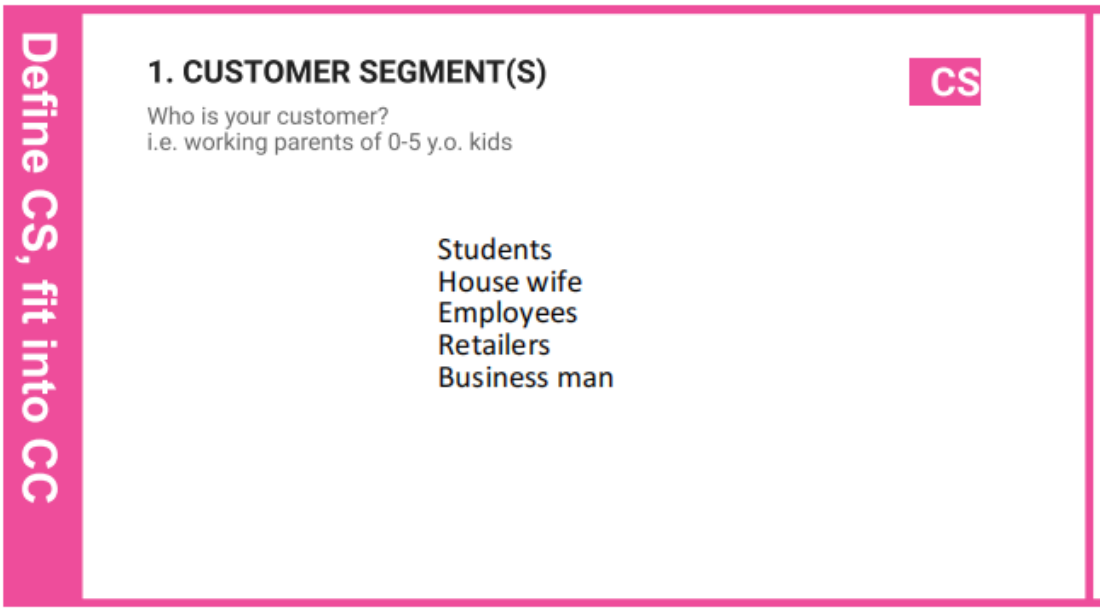


3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Modern life offers a excessive options of services and goods for consumers. As a result, people's expenses have gone up dramatically, e.g., compared to a decade ago, and the cost of living has been increasing day by day. Thus it becomes essential to keep a check on expenses in order to live a good life with a proper budget set up.
2.	Idea / Solution description	Our goal is to create an personal expense tracking application where user can be tracking all financial activities and view previous income and expense report. ➢ User can easily review the reports daily, weekly, monthly or yearly. ➢ User can update or delete records. ➢ User can get notification daily. ➢ Create Category and Change currency. ➢ User can also change Notification time and modify some features. ➢ Add Expense and Income
3.	Novelty / Uniqueness	Personal Expense Tracker is a simple application. Writing in a user's pocket handbook every time during income and expense is not convenient and it is easy because of this application, the user will be able to manage the money through his/her smart devices without any hassle under any circumstances. Users just need to enter income and expense and the app calculates it for users. This application is very easy, fast and secure with money calculation and online mode service
4.	Social Impact / Customer Satisfaction	➢ Eliminate paper, automatically route expense report to the user and reducing time & cost of processing which help customer to save their time. ➢ Software reduces like-hood of data entry errors and can find duplicate entries so

		that customer don't do any extra payments ➢ Reporting and analytics provide real time insight into expense by user as category which enable customer to spend their time on higher value task
5.	Business Model (Revenue Model)	Freemium Model
6.	Scalability of the Solution	➢ Payment mode embedded with the app ➢ The application can be used to collect samples of data related to user's expenses with permissions and use those sample data as parameters to evaluate patterns of spending. ➢ Using some data mining technique expenses can be classified and can be used in market analysis and planning. ➢ This application will not only helps users to manage their expenses but also help marketing executives to plan marketing according to the needs of users.

3.4 Problem Solution fit



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.

Customers need to manage their expense
Need to save money
Know their daily/ weekly/ monthly/ yearly expense
Tracking and visualization of income and expense
Alert when epense exceeds the limit

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.

Customers are triggered when they see their expense is higher than their income.
When they came to know about more productive and efficient way to manage their expense.
When they want to save the money

4. EMOTIONS: BEFORE / AFTER

EM

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

Before:
Feel like spendthrift
Spending money on unwanted things
Incapable to manage money
Not saving for future

After:
Feel like thrifty
Capable to manage money
Saving for future plans
Using income on important works

6. CUSTOMER CONSTRAINTS

CC

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

Internet Dependence
Reduced Speed
Internet reliance
Security
Restricted Functionality
Availability
Web Issues
Browser Support

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.

People dont check their spending and create a budget, theyl have no control whatsoever on money. Instead, money control them, and theyl either have perpetual lack of funds or will end up steeped in debt.

So many people don’ t have great financial management skills, they willl not know how to categorize your expenses.

When they don’ t keep a watch on your spending, there will be short of money, always. This will stress them out.

People are spending money frivolously, they not have money to set financial goals.

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.

This application help the users to add their expenses so they can get an analysis of their expenditure in graphical form. They have option to set the limit of amount to be used for a particular month and if the limit exceed the user will be notified with alert message.

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

Customer in past tried a lot of things to manage their expense like sticky notes, spreadsheet and ledger that cause confusion, data inconsistency problems while recording and splitting of expenses
All they need is a person to maintain their expense and show their statistics. But all people can’ t afford a separate person to manage their own expenses. So in this modern world they need a app which make their management easier.

Explore AS, differentiate

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)

Customers is given instant access to online chat features with knowledgeable staff members
Customers is given option to email the application support team with questions or concerns and make sure the response time is fast
Give customers the option of using the phone and talking to a live operator because some consumers are averse to or unfamiliar with virtual communication forms.

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

8. CHANNELS of BEHAVIOUR

CH

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

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

Online:
Social Media Marketing
Google Advertisement
Youtube Advertisement

Offline:
Newspaper
Radio
Existing Customer Recommendation

Identify strong TR & EM

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 Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Calendar	Personal expense tracker application shall allow user to add the data to their expenses.
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

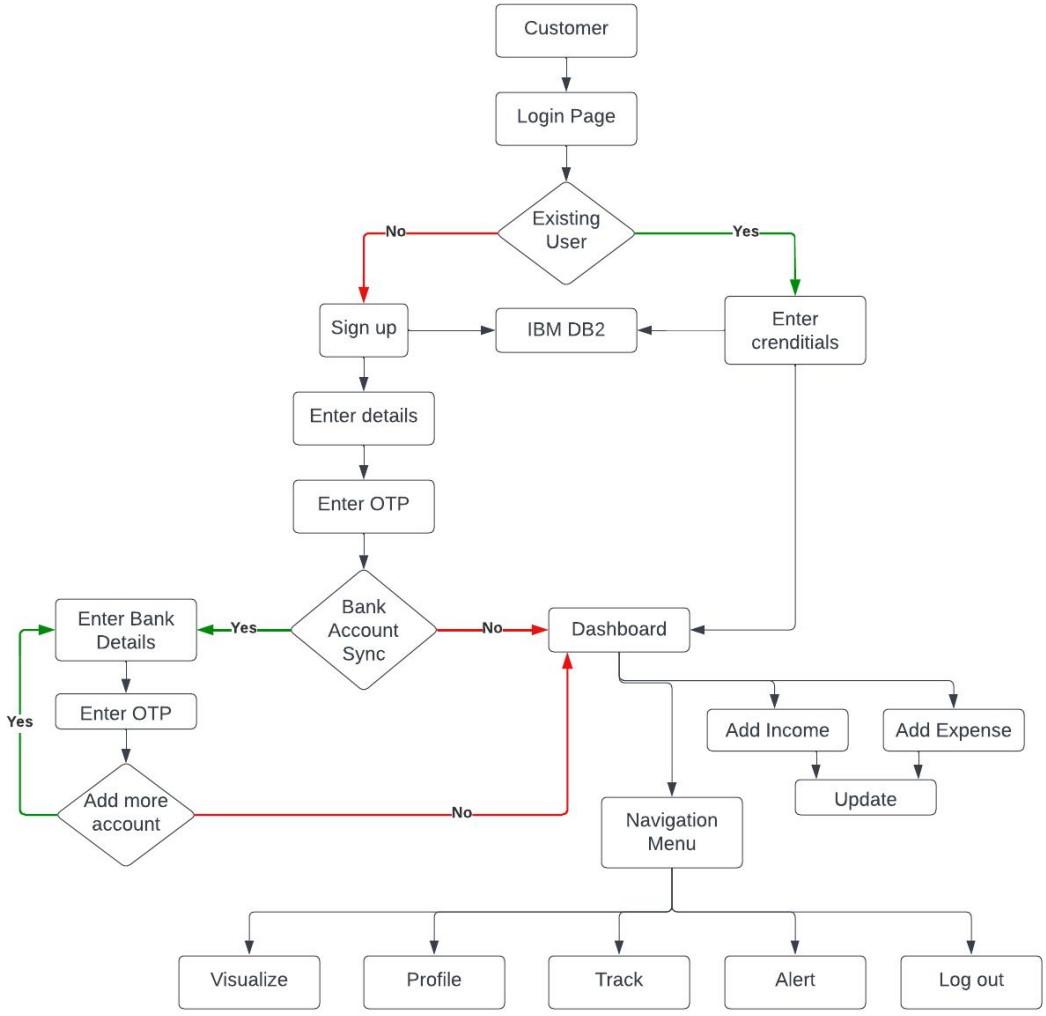
4.2 Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Expense Tracker is highly reusable. The application can be modified for just about anything!
NFR-2	**Security**	Authentication, Authorization, Encryption, Application security testing. More security of the customer data and bank account details.
NFR-3	**Reliability**	Each data record is stored on a well built efficient database schema. There is no risk of data loss.
NFR-4	**Performance**	Application performance monitoring (APM)
NFR-5	**Availability**	It is available all the time, no time constraint
NFR-6	**Scalability**	Capacity of the application to handle growth, especially in handling more users.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



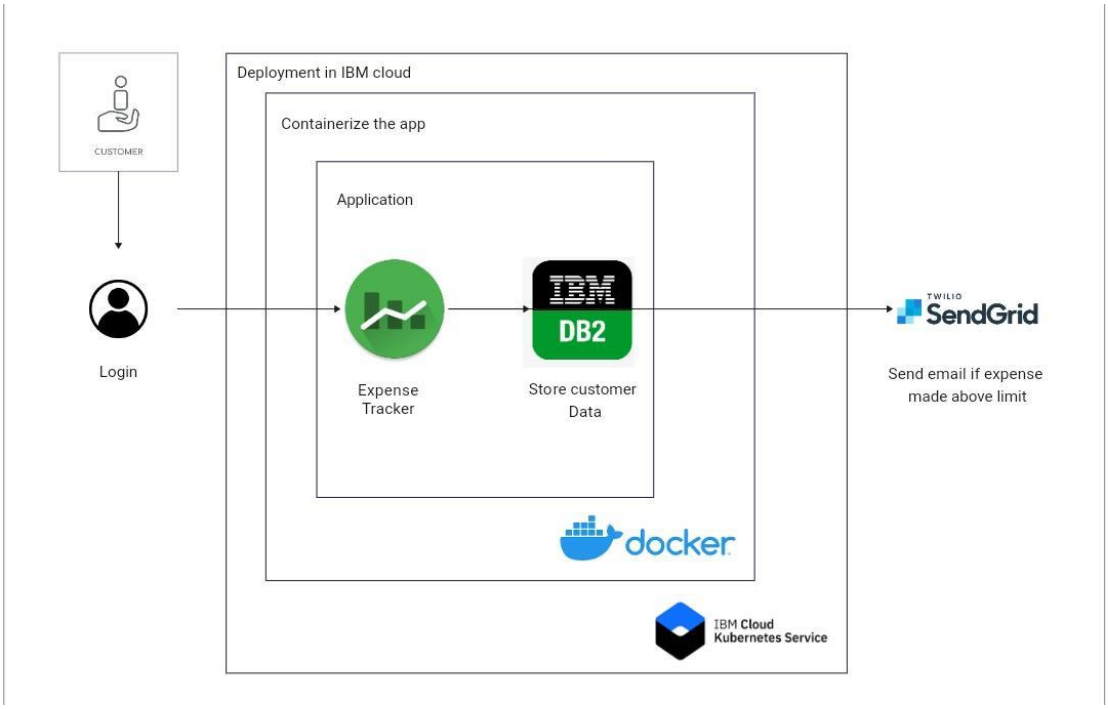
5.2 Solution & Technical Architecture

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g., Web UI, Mobile App, etc.	HTML, CSS, Python flask
2	Registration	User register in the application to start the process	HTML, CSS, Python flask, IBM cloud, IBM Container registry
3	Login	User login to their account	HTML, CSS, Python flask, IBM cloud, IBM Container registry

4	Wallet page	User can add their expenses in the wallet	HTML, CSS, Python flask, IBM cloud, IBM Container registry
5	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
6	Email alert	User can be notified when their expenses cross the limit in the wallet	Kubernetes, IBM container registry, SendGrid
7	Graphical view	User can able to see their monthly expenses in a graph format	IBM cloud object storage, IBM container registry, HTML, CSS

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Docker and Kubernetes are the open source frameworks	Docker, Kubernetes
2	Security Implementations	IBM DB2 is used for the security control	IBM DB2
3	Scalable Architecture	This architecture connects the three dimensions like processing, storage, and connectivity between the user and the system	Python flask, IBM container registry
4	Availability	It is always available	Python flask and IBM cloud
5	Performance	The application can perform well user can experience the fast while using the application	Python flask and IBM cloud


CUSTOMER
Login
Deployment in IBM cloud
Containerize the app
Application
Expense Tracker
IBM DB2
Store customer Data
docker
IBM Cloud Kubernetes Service
TWILIO SendGrid
Send email if expense made above limit

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	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 confirming my password.	2	High	Shakthi Dharun
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Vishnu
			As a user, I can register for the application through Gmail	1	High	Shakthi Dharun
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Ranganathan
Sprint-1	Dashboard	USN-4	As a user, I can add income and expenses in the application	2	High	Raja Guru
Sprint-2	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Shakthi Dharun
Sprint-2	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Vishnu
Sprint-2		USN-4	Making dashboard interactive with JS	2	High	Raja Guru

Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Ranganathan
Sprint-3	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user's query	1	Medium	Shakthi Dharun
Sprint-3	SendGrid	USN-3	Using SendGrid to send mail to the user about	1	Low	Vishnu
Sprint-3		USN-4	Integrating both frontend and backend	2	High	Raja Guru
Sprint-4	Docker	USN-1	Creating image of website using docker	2	High	Ranganathan
Sprint-4	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Shakthi Dharun
Sprint-4	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Vishnu
Sprint-4	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Raja Guru

6.1 Sprint Delivery Schedule

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

7. CODING & SOLUTIONING

Feature 1
User Login:
The **login page** allows a user to gain access to an application by entering their username and password or by authenticating using a social media login.

Code:

```
{% extends "newlayout.html" %}
{% block content %}

<div class="row d-flex justify-content-center align-items-center h-100">

<div class="col-md-9 col-lg-6 col-xl-5">
<img src="https://mdbcdn.b-cdn.net/img/Photos/new-templates/bootstrap-login-form/draw2.webp" class="img-fluid"
alt="Sample image">
</div>

<div class="col-md-8 col-lg-6 col-xl-4 offset-xl-1">
<form method="post">

<!-- username input -->
<div class="form-outline mb-4">
<input required type="username" name="username" class="form-control form-control-lg"
placeholder="Enter username" value="{{ request.form['username'] }}" />
<label class="form-label" for="username">Username</label>
</div>

<!-- Password input -->
<div class="form-outline mb-3">
<input required type="password" name="password" class="form-control form-control-lg"
placeholder="Enter password" value="{{ request.form['password'] }}" />
<label class="form-label" for="password">Password</label>
</div>

<div class="text-center text-lg-start mt-4 pt-2">
<!-- {% if msg!=" " %}
<div class="alert alert-danger" role="alert">
{{ msg }}
</div>
```

```
{% endif %} -->
<button type="submit" class="btn btn-primary btn-lg"
style="padding-left: 2.5rem; padding-right: 2.5rem;">Login</button>
<p class="small fw-bold mt-2 pt-1 mb-0">Don't have an account? <a href="/register"
class="link-danger">Register</a></p>
</div>

</form>
</div>
</div>
{% endblock content %}
```

User Register

A signup page (also known as a registration page) **enables users and independently register and gain access to the application**.

```
{% extends "newlayout.html" %}
{% block content %}


<div class="row justify-content-center">
<div class="col-md-10 col-lg-6 col-xl-5 order-2 order-lg-1">

<form class="mx-1 mx-md-4" method="post">

<div class="d-flex flex-row align-items-center mb-4">
<i class="fas fa-user fa-lg me-3 fa-fw"></i>
<div class="form-outline flex-fill mb-0">
<input required type="text" name="username" class="form-control"
value="{{ request.form['username'] }}" />
<label class="form-label" for="username">Your Username</label>
</div>
</div>

<div class="d-flex flex-row align-items-center mb-4">
<i class="fas fa-envelope fa-lg me-3 fa-fw"></i>
<div class="form-outline flex-fill mb-0">
<input required type="email" name="email" class="form-control"
value="{{ request.form['email'] }}" />
<label class="form-label" for="email">Your Email</label>
</div>
</div>

<div class="d-flex flex-row align-items-center mb-4">
```

```
<i class="fas fa-lock fa-lg me-3 fa-fw"></i>
<div class="form-outline flex-fill mb-0">
<input required type="password" name="password" class="form-control"
value="{{ request.form['password'] }}" />
<label class="form-label" for="password">Password</label>
</div>
</div>

<div class="d-flex flex-row align-items-center mb-4">
<i class="fas fa-key fa-lg me-3 fa-fw"></i>
<div class="form-outline flex-fill mb-0">
<input required type="password" name="password1" class="form-control"
value="{{ request.form['password1'] }}" />
<label class="form-label" for="password1">Repeat your password</label>
</div>
</div>
<div class="text-center text-lg-start mt-4 pt-2">
<button type="submit" class="btn btn-primary btn-lg"
style="padding-left: 2.5rem; padding-right: 2.5rem;">Register</button>
<p class="small fw-bold mt-2 pt-1 mb-0">Have an account? <a href="/login"
class="link-danger">Log
In</a></p>
</div>
</form>
</div>
<div class="col-md-10 col-lg-6 col-xl-7 d-flex align-items-center order-1 order-lg-2">

<img src="https://mdbcdn.b-cdn.net/img/Photos/new-templates/bootstrap-
registration/draw1.webp" class="img-fluid"
alt="Sample image">

</div>
</div>
{% endblock content %}
```

Dashboard:

```
{% extends "navblock.html" %}
{% block content %}

{% if session.username %}
<p class="text-center h3 fw-bold mb-5 mx-1 mx-md-4 mt-4">Welcome
{{ session.username }}</p><br>

Name : {{ session['username'] }}<br>
```

```
Email : {{ session['email'] }}<br>
Budget : {{ session['budget'] }}<br>
<br>
<button type="button" onclick="displayingDiv('passDiv')">Change Password</button>
{%if session.budget%}
<button type="button" onclick="displayingDiv('budDiv')">Change Budget</button>
{%else%}
<button type="button" onclick="displayingDiv('addbudDiv')">Add Budget</button>
{%endif%}
<br>
<br>
<div id="passDiv" style="display:none;">
<form method="post" id="passwordChangeForm" action="/changePassword/"
method="post">
<label for="pass1">Enter new password</label>
<input name="pass1" type="password" value="{{request.form['pass1']}}"
placeholder="Password"><br><br>
<label for="pass2">Re-enter new password</label>
<input name="pass2" type="password" value="{{request.form['pass2']}}"
placeholder="Re-enter Password"><br><br>
<input type="submit">
<br>
</form>
</div>
<br>
<div id="budDiv" style="display:none;">
<form method="post" id="budgetChangeForm" action="/changeBudget/"
method="post">
<label for="budgetAmount">Enter new Budget</label>
<input name="budgetAmount" type="number" step="0.01" min="0"
placeholder="Budget"><br>
<input type="submit">
<br>
</form>
</div>
<div id="addbudDiv" style="display:none;">
<form method="post" id="budgetAddForm" action="/addBudget/" method="post">
<label for="budgetAmountToAdd">Enter the Budget</label>
<input name="budgetAmountToAdd" step="0.01" min="0" type="number"
placeholder="Budget"><br><br>
<input type="submit">
<br>
</form>
</div>
```

```
<br>
{% else %}
You are not logged in.
<a href="/login">login</a>.
{%endif%}
{% endblock content %}
```

Home Page:

```
{% extends "layout.html" %}
{%block basecontent%}
<section class="vh-100" style="background-color: #eee;">
<div class="container h-100">
<div class="card text-black" style="border-radius: 25px;">
<div class="card-body p-md-5">
<p class="text-center h1 fw-bold mb-5 mx-1 mx-md-4 mt-4">Cash Book</p>
<div class='parent'>
<div class="center">
<div class='child inline-block-child'><a href="/login"><button type="button"
class="btn btn-primary btn-lg"
style="padding-left: 2.5rem; padding-right: 2.5rem;">Login</button></a><br></div>
<div class='child inline-block-child'><a href="/register"><button type="button"
class="btn btn-primary btn-lg"
style="padding-left: 2.5rem; padding-right:
2.5rem;">Register</button></a><br></div>
</div></div>
</div>
<center class="card-body p-md-5">
<p><b>Team ID: PNT2022TMID03042</b></p>
<p><b>Batch: </b>B4-4M6E</p>
<p><b>Team Members: </b></p>
&#9;<p>Shakthi Dharun - 727720EUIT510</p>
&#9;<p>Raja Guru J - 727719EUIT123</p>
&#9;<p>Ranganathan R - 727719EUIT125</p>
&#9;<p>Vishnu R - 727719EUIT177</p>
<p><b>Industry Mentor Name: </b>Ms. Kushboo</p>
<p><b>Faculty Mentor Name: </b>Mr. M Mohammed Mustafa</p>
</center>
</div>
</div>
</section>
{% endblock basecontent %}
```

Layout page:

```
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>FinBud {{ title }}</title>
<link    href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-
iYQeCzEYFbKjA/T2uDLTpkwGzCiq6soy8tYaI1GyVh/UjpbCx/TYkiZhlZB6+fzT"
crossorigin="anonymous">
<link rel="stylesheet" href="/static/styles.css">
</head>

<body style="height:100%">
{% block basecontent %}
{% endblock basecontent %}
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
u1OknCvxWvY5kfmNBILK2hRnQC3Pr17a+RTT6rIHI7NnikvbZlHgTPOOmMi466
C8" crossorigin="anonymous">
</script>

<script>
window.watsonAssistantChatOptions = {
integrationID: "b3a451ac-0ddf-4af8-8c1b-fce656264a1c", // The ID of this integration.
region: "us-south", // The region your integration is hosted in.
serviceInstanceID:  "81e76571-dada-42fe-ac3d-6a9766c07453",  //  The  ID  of  your
service instance.
onLoad: function(instance) { instance.render(); }
};
setTimeout(function(){
const t=document.createElement('script');
t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/"          +
(window.watsonAssistantChatOptions.clientVersion          ||          'latest')          +
"/WatsonAssistantChatEntry.js";
document.head.appendChild(t);
});
</script>
</body>
```

</html>

Mail Alert:

This feature is used to send mail to user when the expense exceed the limit

Code:

```
import smtplib
import sendgrid
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("shakthidharun@gmail.com", "pass123")
message  = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)
s.sendmail("shakthidharun@gmail.com", email, message)
s.quit()
def sendgridmail(user,TEXT):

from_email = Email("shakthidharun@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)
```

Flask App:

Python flask framework is used to run the application

Code:

```
    from flask import (
    Flask,
    render_template,
    send_file,
    request,
```

```
    redirect,
    url_for,
    session,
    flash,
    )

    import ibm_db
    import re
    from matplotlib import pyplot as plt
    from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
    from io import BytesIO
    import random

    app = Flask(__name__)
    app.secret_key = "shakthi"

    conn = ibm_db.connect(
    "DATABASE=bludb;"
    "HOSTNAME=764264db-9824-4b7c-82df-
40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;"
    "PORT=32536;"
    "SECURITY=SSL;"
    "SSLServerCertificate=DigiCertGlobalRootCA.crt;"
    "UID=vqy21243;"
    "PWD=W7SMJiuNPhC4ElCZ;",
    "",
    "",
    )


    @app.route("/", methods=["POST", "GET"])
    @app.route("/home")
    def home():
    return render_template("home.html")


    @app.route("/login", methods=["GET", "POST"])
    def login():
    msg = ""
    if request.method == "POST":
    username = request.form["username"]
    password = request.form["password"]
```

```
    sql = "SELECT clients.*,budgets.MAXBUDGET FROM clients LEFT JOIN BUDGETS ON CLIENTs.ID=BUDGETS.ID WHERE username =? AND password =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.bind_param(stmt, 2, password)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    # print(account)
    if account:
    session["Loggedin"] = True
    session["id"] = account["ID"]
    session["email"] = account["EMAIL"]
    session["username"] = account["USERNAME"]
    session["budget"] = account["MAXBUDGET"]
    print(session["Loggedin"])
    return redirect("/dashboard")
    else:
    msg = "Incorrect login credentials"
    flash(msg)
    return render_template("login.html", title="Login")


    @app.route("/register", methods=["GET", "POST"])
    def register():
    msg = ""
    if request.method == "POST":
    username = request.form["username"]
    email = request.form["email"]
    password = request.form["password"]
    password1 = request.form["password1"]
    sql = "SELECT * FROM CLIENTS WHERE username =? or email=? "
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.bind_param(stmt, 2, email)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    print(account)
    if account:
    msg = "Account already exists"
    elif password1 != password:
    msg = "re-entered password doesnt match"
    elif not re.match(r"[A-Za-z0-9]+", username):
    msg = "Username should be only alphabets and numbers"
```

```
    else:
    sql = "INSERT INTO clients(ID,EMAIL,USERNAME,PASSWORD) VALUES
(?,?,?,?)"
    stmt = ibm_db.prepare(conn, sql)
    randNum = random.randint(1000, 10000)
    ibm_db.bind_param(stmt, 1, randNum)
    ibm_db.bind_param(stmt, 2, email)
    ibm_db.bind_param(stmt, 3, username)
    ibm_db.bind_param(stmt, 4, password)
    ibm_db.execute(stmt)
    return redirect("/dashboard")
    flash(msg)
    return render_template("register.html", title="Register")


    @app.route("/logout")
    def logout():
    session.clear()
    return redirect("/")


    def isLogged():
    return session["Loggedin"]


    @app.route("/dashboard")
    def dashboard():
    if isLogged:
    return render_template("dashboard.html", title="Dashboard")
    else:
    flash("Login to go to dashboard")
    return redirect("/login")


    @app.route("/changePassword/", methods=["POST", "GET"])
    def changePassword():
    msg = "Enter the new password"
    if request.method == "POST":
    pass1 = request.form["pass1"]
    pass2 = request.form["pass2"]
    if pass1 == pass2:
    sql = "UPDATE CLIENTS SET password=? where id=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, pass1)
```

```
ibm_db.bind_param(stmt, 2, session["id"])
if ibm_db.execute(stmt):
msg = "Successfully Changed Password!!!!"

else:
msg = "Passwords not equal"
flash(msg)
return redirect(url_for("dashboard"))


@app.route("/changeBudget/", methods=["POST", "GET"])
def changeBudget():
msg = "Enter the new budget"
if request.method == "POST":
budgetAmount = request.form["budgetAmount"]
sql = "UPDATE BUDGETS SET maxBudget=? where id=?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, budgetAmount)
ibm_db.bind_param(stmt, 2, session["id"])
if ibm_db.execute(stmt):
session["budget"] = budgetAmount
msg = "Successfully Changed Budget!!!!"
else:
msg = "Budget not changed"
flash(msg)
return redirect(url_for("dashboard"))


@app.route("/addBudget/", methods=["POST", "GET"])
def addBudget():
msg = "Enter the budget"
if request.method == "POST":
budgetAmount = request.form["budgetAmountToAdd"]
sql = "INSERT INTO BUDGETS(id,maxbudget) VALUES(?,?)"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, session["id"])
ibm_db.bind_param(stmt, 2, budgetAmount)
if ibm_db.execute(stmt):
session["budget"] = budgetAmount
msg = "Successfully Set The Budget!!!!"
else:
msg = "Budget not set yet"
flash(msg)
return redirect(url_for("dashboard"))
```

```
def fetchall(stmt):
ibm_db.bind_param(stmt, 1, session["id"])
ibm_db.execute(stmt)
results = []
result_dict = ibm_db.fetch_assoc(stmt)
results.append(result_dict)
while result_dict is not False:
result_dict = ibm_db.fetch_assoc(stmt)
results.append(result_dict)
results.pop()
return results


def getTotal(table):
sql = "SELECT SUM(AMOUNT) FROM " + table + " where USER_ID=?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, session["id"])
ibm_db.execute(stmt)
result = ibm_db.fetch_assoc(stmt)
print(result)
return result["1"]


@app.route("/log_today")
def logToday():
if isLogged():
sql = "SELECT AMOUNT,CATEGORY,NEED FROM TRANSACTIONS
WHERE USER_ID=? AND DATEADDED=CURRENT_DATE"
stmt = ibm_db.prepare(conn, sql)
expenseData = fetchall(stmt)
print(expenseData)
expenseTotal = getTotal("TRANSACTIONS")
sql = "SELECT AMOUNT FROM income WHERE ID=? AND
DATEADDED=CURRENT_DATE"
stmt = ibm_db.prepare(conn, sql)
incomeData = fetchall(stmt)
print(incomeData)
return render_template(
"logtoday.html",
title="Today's Log",
expenseData=expenseData,
incomeData=incomeData,
```

```
    expenseTotal=expenseTotal,
    )
    else:
    flash("Login First")
    return redirect("/login")


    @app.route("/addExpense/", methods=["POST", "GET"])
    def addExpense():
    msg = ""
    if request.method == "POST":
    amount = request.form["Amount"]
    need = request.form["Need/Want"]
    category = request.form["category"]
    sql = "INSERT INTO
TRANSACTIONS(USER_ID,AMOUNT,NEED,CATEGORY,DATEADDED)
VALUES(?,?,?,?,CURRENT_DATE)"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, session["id"])
    ibm_db.bind_param(stmt, 2, amount)
    ibm_db.bind_param(stmt, 3, need)
    ibm_db.bind_param(stmt, 4, category)
    if ibm_db.execute(stmt):
    msg = "Successfully Added Expense!!!!"
    else:
    msg = "Expense not added"

    flash(msg)
    return redirect(url_for("logToday"))


    @app.route("/addIncome/", methods=["POST", "GET"])
    def addIncome():
    msg = ""
    if request.method == "POST":
    amount = request.form["AmountIncome"]
    sql = "INSERT INTO INCOME(ID,AMOUNT,DATEADDED)
VALUES(?,?,CURRENT_DATE)"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, session["id"])
    ibm_db.bind_param(stmt, 2, amount)
    if ibm_db.execute(stmt):
    msg = "Successfully Added Income!!!!"
    else:
```

```
    msg = "Income not added"

    flash(msg)
    return redirect(url_for("logToday"))


    # @app.route("/Edit")
    ###Visualization functions


    @app.route("/reports")
    def reports():
    return render_template("reports.html", title="Reports")


    @app.route("/needVwant/")
    def needVwant():
    sql = "SELECT Sum(amount) AS amount, need FROM transactions WHERE DAYS(CURRENT_DATE)-DAYS(DATEADDED)<29 AND  user_id = ? GROUP BY NEED ORDER BY need"
    stmt = ibm_db.prepare(conn, sql)
    transactions = fetchall(stmt)
    values = []
    labels = []
    print(transactions)
    for transaction in transactions:
    values.append(transaction["AMOUNT"])
    labels.append(transaction["NEED"])
    fig = plt.figure(figsize=(10, 7))
    plt.pie(values)
    plt.title("Need v Want")
    plt.legend(["WANT", "NEED"])
    canvas = FigureCanvas(fig)
    img = BytesIO()
    fig.savefig(img)
    img.seek(0)
    return send_file(img, mimetype="image/png")


    @app.route("/categoriesChart/")
    def categoriesChart():
    sql = "SELECT Sum(amount) AS amount, category FROM transactions WHERE DAYS(CURRENT_DATE)-DAYS(DATEADDED)<29 AND  user_id = ? GROUP BY category ORDER BY category"
```

```
    stmt = ibm_db.prepare(conn, sql)
    transactions = fetchall(stmt)
    values = []
    labels = []
    print(transactions)
    for transaction in transactions:
    values.append(transaction["AMOUNT"])
    labels.append(transaction["CATEGORY"])
    fig = plt.figure(figsize=(10, 7))
    plt.pie(values, labels=labels)
    plt.title("Categories")
    plt.legend()
    canvas = FigureCanvas(fig)
    img = BytesIO()
    fig.savefig(img)
    img.seek(0)
    return send_file(img, mimetype="image/png")


    ##edit the legend... all visualizations workkkkkk!!!!!!!
    @app.route("/dailyLineChart/")
    def dailyLineChart():
    sql = "SELECT Sum(amount) AS amount, DAY(dateadded) as dateadded FROM transactions WHERE DAYS(CURRENT_DATE)-DAYS(DATEADDED)<29 AND user_id = ? GROUP BY dateadded ORDER BY dateadded"
    stmt = ibm_db.prepare(conn, sql)
    transactions = fetchall(stmt)
    x = []
    y = []
    print(transactions)
    for transaction in transactions:
    y.append(transaction["AMOUNT"])
    x.append(transaction["DATEADDED"])
    ##get budget
    sql = "SELECT MAXBUDGET FROM budgets WHERE id = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, session["id"])
    ibm_db.execute(stmt)
    budget = ibm_db.fetch_assoc(stmt)
    print(budget)
    fig = plt.figure(figsize=(10, 7))
    plt.scatter(x, y)
    plt.plot(x, y, "-")
    if budget:
```

```
plt.axhline(y=budget["MAXBUDGET"], color="r", linestyle="-")
plt.xlabel("Day")
plt.ylabel("Transaction")
plt.title("Daily")
plt.legend()
canvas = FigureCanvas(fig)
img = BytesIO()
fig.savefig(img)
img.seek(0)
return send_file(img, mimetype="image/png")


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

LogToday:

```
{% extends "navblock.html" %}
{% block content %}
<h3>Expenses</h3>
{%if expenseData%}
<table>
<tr>
<th>AMOUNT</th>
<th>CATEGORY</th>
<th>NEED</th>
</tr>
{%for item in expenseData%}
<tr>
{%for i in item.keys()%}
<td>{{item[i]}}</td>
{%endfor%}
</tr>
{%endfor%}
</table><br>
<div>
<h6>Total : {{expenseTotal}}</h6>
</div>
{%else%}
<i>
<p>No Data yet</p>
</i>
{%endif%}
<br>
```

```
    <!-- income table -->
    <h3>Income</h3>
    {%if incomeData%}
    <table>
    <tr>
    <th>AMOUNT</th>
    </tr>
    {%for item in incomeData%}
    <tr>
    <td>{{item['AMOUNT']}}</td>
    </tr>
    {%endfor%}
    </table>
    {%else%}
    <i>
    <p>No Data yet</p>
    </i>
    {%endif%}
    <br>
    <button type="button" onclick="displayingDiv('expenseDiv')">Add
Expense</button>
    <button type="button" onclick="displayingDiv('incomeDiv')">Add
Income</button>
    <br>
    <br>
    <div id="expenseDiv" style="display:none;">
    <form name="addexpenseForm" action="/addExpense/" method="post">
    <label for="Amount">Enter the Expense Amount</label>
    <input name="Amount" type="number" placeholder="Amount" step="0.01"
min="0"><br>
    <br>
    <label for="Need/Want">Is it a need or a want?</label>
    <select name="Need/Want" required>
    <option value="TRUE">Need</option>
    <option value="FALSE">Want</option>
    </select><br>
    <br>
    <select name="category" id="category" class="form-control">
    <option value="Miscellaneous" selected="selected">Select Category</option>
    <option value="Miscellaneous">Miscellaneous</option>
    <option value="Food">Food</option>
    <option value="Transportation">Transportation</option>
    <option value="Groceries">Groceries</option>
    <option value="Clothing">Clothing</option>
```

```
<option value="HouseHold">HouseHold</option>
<option value="Rent">Rent</option>
<option value="Bills and Taxes">Bills and Taxes</option>
<option value="Vacations">Vacations</option>
</select>
<!-- <input type="checkbox" name="options" id="education" value="education"> Education </input>
<input type="checkbox" name="options" id="entertainment" value="entertainment"> Entertainment
</input>
<input type="checkbox" name="options" id="travel" value="travel"> Travel </input>
<input type="checkbox" name="options" id="food" value="food"> Food </input>
<input type="checkbox" name="options" id="health" value="health"> Health </input>
<input type="checkbox" name="options" id="others" value="others"> Others </input> -->
<br><br>
<input type="submit">
</form>
</div>
<div id="incomeDiv" style="display:none;">
<form method="post" id="incomeForm" action="/addIncome/" method="post">
<label for="AmountIncome">Enter the Income Amount</label>
<input name="AmountIncome" type="number" placeholder="Amount" step="0.01" min="0"><br>
<br>
<input type="submit">
<br>
</form>
</div>

{% endblock content %}
```

NavBlock:

```
{% extends "newlayout.html" %}
<!-- Navbar -->
{%block nav%}
<nav class="navbar navbar-expand-lg navbar navbar-dark bg-dark"">
<div class=" collapse navbar-collapse" id="navbarNav">
<ul class="navbar-nav">
<li class="nav-item">
<a class="nav-link" href="/dashboard">Dashboard </a>
</li>
```

```
<li class="nav-item">
<a class="nav-link" href="/log_today">Log's Today</a>
</li>
<li class="nav-item">
<a class="nav-link" href="/reports">Reports</a>
</li>
<li class="nav-item">
<a class="nav-link" href="/logout">Log Out</a>
</li>
</ul>
</div>
</nav>
<!-- Navbar -->
{%endblock nav%}
{% block content %}
{% endblock content %}
```

Reports:

```
{% extends "navblock.html" %}
{% block content %}
<h3>Need Vs Want</h3>
<img src="/needVwant/" alt="Need vs. Want">
<h3>Categories Chart</h3>
<img src="/categoriesChart/" alt="categoriesChart">
<h3>Daily Chart</h3>
<img src="/dailyLineChart/" alt="dailyLineChart">
{% endblock content %}
```

8. TESTING

8.1 Test Cases

Test case ID	Feature Type	Component	Test Scenario
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with InValid credentials
AddExpensePage_TC	Functional	ld Expense Pa	whether user is able to add expense

Pre-Requisite	Steps To Execute	Test Data
None	1. Go to website 2. Home page appears	Username: shakthidharun@gmail.com password: password123
Home	1.Go to website 2.Enter details and click login	Username: shakthidharun@gmail.com password: password123
Username & password	1.Go to website 2.Enter details and click login	Username: shakthidharun@gmail.com password: password123
Username & password	1.Go to website 2.Enter details and click login	Username: shakthidharun@gmail.com password: password123
Login first	1.Go to website 2.Enter details and click login	Username: shakthidharun@gmail.com password: password123
Login first	1.Go to website 2.Enter details and click login	Username: shakthidharun@gmail.com password: password123
Have some expense to add	1. Add date, expense name and other details 2.Check if the expense gets added	add electricity bill = 6000

Expected Result	Actual Result	Status
Login/Signup popup should display	Working as expected	Pass
Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.New customer? Create account link e.Last password? Recovery password link	Working as expected	Pass
User should navigate to user account homepage	Working as expected	Pass
Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass
Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass
Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass
Application adds expenses	Working as expected	Pass

8.2 User Acceptance Testing

Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	9	3	1	2	15
Duplicate	2	1	2	1	6
External	1	2	0	3	6
Fixed	10	2	4	11	27
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	22	13	11	19	65

Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Interface	13	0	3	10
Login	20	0	3	17
Logout	5	0	0	5
Add Expense	15	0	2	13
Limit	5	0	0	5
Report	5	1	2	3
Sign up	3	0	1	2

9. RESULTS

9.1 Performance Metrics

Leadtime - 4 months

Cycle time - 10 Min

Team velocity

10. ADVANTAGES & DISADVANTAGES

Advantages

When if comes to personal finance, being out of control is not something anybody would strive for. There's nothing financially worse than feeling like you don't have any idea what's going on with your money.

The good news is, when you make an effort to record every financial transaction you make, you are essentially, taking the reins on anything and everything involving your money. At any one time, you will know exactly how much money is sitting in your bank account, and how much you can spend.

In other words, when you track your expenses, you take complete control over your finances.

If you have any plans on saving, investing, getting out of debt, or building wealth, what is holding you accountable. I mean, we can all set financial goals, and have financial dreams, but if you aren't tracking your expenses, there is nothing to hold you accountable when you make a bad financial decision.

Tracking your expenses holds you accountable to your future financial goals. And in the long run, that can be the difference between broke and wealthy.

Disadvantages

Your information is less secure, and probably being used and sold. If the service is free, then the product is you. Mint.com, like other financial apps, is a free service. They have to pay their bills somehow, so regardless of what their privacy policy may or may not say, just assume that your spending history and trends are going to be recorded and analyzed, by someone, somewhere. Now, you shouldn't have to worry about credit card fraud or **identity theft**, these companies are large enough and secure enough that you'll never have to worry about something like that. Just recognize that your information, most likely anonymous, will be used and potentially even sold. Personally, I have no problem with that, but if you do, then make sure you avoid these types of services.

11. CONCLUSION

Recording your expenses daily can ensure that you are financially aware all year long and not just during tax season. Knowing where your money is going and how much you're spending can improve your spending habits. So, using the daily expense tracker application is important to load a happy family. Daily expense tracker helps the user to avoid unexpected expenses and bad financial situations. This Project will save time and provide a responsible lifestyle.

12. FUTURE SCOPE

Year-on-year, modern expense management software undergone a continuous evolution from traditional back-office function to strategic internal set of processes. But would it be sufficient to meet the needs of next-gen companies? Have you ever thought how the next-generation software should look like? As the requirements of companies evolve continuously, the software should undergo a series of changes to meet the growing needs of next generation companies.

The next-generation travel and expense (T & E) management apps should not only just accelerate the expense management process but also should come with mobile and cloud integration capabilities that add tremendous value to the business bottom line. Future T & E management software should be able to provide greater visibility into spending and standardize critical procedures.

13. APPENDIX

Source Code

```
from flask import (
Flask,
render_template,
send_file,
request,
redirect,
url_for,
session,
flash,
)

import ibm_db
import re
from matplotlib import pyplot as plt
from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
from io import BytesIO
import random

app = Flask(__name__)
app.secret_key = "shakthi"

conn = ibm_db.connect(
"DATABASE=bludb;"
"HOSTNAME=764264db-9824-4b7c-82df-
40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;"
"PORT=32536;"
"SECURITY=SSL;"
"SSLServerCertificate=DigiCertGlobalRootCA.crt;"
"UID=vqy21243;"
"PWD=W7SMJiuNPhC4ElCZ;",
"",
"",
)


@app.route("/", methods=["POST", "GET"])
@app.route("/home")
def home():
return render_template("home.html")
```

```
@app.route("/login", methods=["GET", "POST"])
def login():
msg = ""
if request.method == "POST":
username = request.form["username"]
password = request.form["password"]
sql = "SELECT clients.*,budgets.MAXBUDGET FROM clients LEFT JOIN BUDGETS ON CLIENTs.ID=BUDGETS.ID WHERE username =? AND password =?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.bind_param(stmt, 2, password)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)
# print(account)
if account:
session["Loggedin"] = True
session["id"] = account["ID"]
session["email"] = account["EMAIL"]
session["username"] = account["USERNAME"]
session["budget"] = account["MAXBUDGET"]
print(session["Loggedin"])
return redirect("/dashboard")
else:
msg = "Incorrect login credentials"
flash(msg)
return render_template("login.html", title="Login")


@app.route("/register", methods=["GET", "POST"])
def register():
msg = ""
if request.method == "POST":
username = request.form["username"]
email = request.form["email"]
password = request.form["password"]
password1 = request.form["password1"]
sql = "SELECT * FROM CLIENTS WHERE username =? or email=? "
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.bind_param(stmt, 2, email)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)
```

```
print(account)
if account:
msg = "Account already exists"
elif password1 != password:
msg = "re-entered password doesnt match"
elif not re.match(r"[A-Za-z0-9]+", username):
msg = "Username should be only alphabets and numbers"
else:
sql = "INSERT INTO clients(ID,EMAIL,USERNAME,PASSWORD) VALUES (?,?,?,?)"
stmt = ibm_db.prepare(conn, sql)
randNum = random.randint(1000, 10000)
ibm_db.bind_param(stmt, 1, randNum)
ibm_db.bind_param(stmt, 2, email)
ibm_db.bind_param(stmt, 3, username)
ibm_db.bind_param(stmt, 4, password)
ibm_db.execute(stmt)
return redirect("/dashboard")
flash(msg)
return render_template("register.html", title="Register")


@app.route("/logout")
def logout():
session.clear()
return redirect("/")


def isLogged():
return session["Loggedin"]


@app.route("/dashboard")
def dashboard():
if isLogged:
return render_template("dashboard.html", title="Dashboard")
else:
flash("Login to go to dashboard")
return redirect("/login")


@app.route("/changePassword/", methods=["POST", "GET"])
def changePassword():
msg = "Enter the new password"
```

```
if request.method == "POST":
pass1 = request.form["pass1"]
pass2 = request.form["pass2"]
if pass1 == pass2:
sql = "UPDATE CLIENTS SET password=? where id=?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, pass1)
ibm_db.bind_param(stmt, 2, session["id"])
if ibm_db.execute(stmt):
msg = "Successfully Changed Password!!!!"

else:
msg = "Passwords not equal"
flash(msg)
return redirect(url_for("dashboard"))


@app.route("/changeBudget/", methods=["POST", "GET"])
def changeBudget():
msg = "Enter the new budget"
if request.method == "POST":
budgetAmount = request.form["budgetAmount"]
sql = "UPDATE BUDGETS SET maxBudget=? where id=?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, budgetAmount)
ibm_db.bind_param(stmt, 2, session["id"])
if ibm_db.execute(stmt):
session["budget"] = budgetAmount
msg = "Successfully Changed Budget!!!!"
else:
msg = "Budget not changed"
flash(msg)
return redirect(url_for("dashboard"))


@app.route("/addBudget/", methods=["POST", "GET"])
def addBudget():
msg = "Enter the budget"
if request.method == "POST":
budgetAmount = request.form["budgetAmountToAdd"]
sql = "INSERT INTO BUDGETS(id,maxbudget) VALUES(?,?)"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, session["id"])
ibm_db.bind_param(stmt, 2, budgetAmount)
```

```
if ibm_db.execute(stmt):
session["budget"] = budgetAmount
msg = "Successfully Set The Budget!!!!"
else:
msg = "Budget not set yet"
flash(msg)
return redirect(url_for("dashboard"))


def fetchall(stmt):
ibm_db.bind_param(stmt, 1, session["id"])
ibm_db.execute(stmt)
results = []
result_dict = ibm_db.fetch_assoc(stmt)
results.append(result_dict)
while result_dict is not False:
result_dict = ibm_db.fetch_assoc(stmt)
results.append(result_dict)
results.pop()
return results


def getTotal(table):
sql = "SELECT SUM(AMOUNT) FROM " + table + " where USER_ID=?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, session["id"])
ibm_db.execute(stmt)
result = ibm_db.fetch_assoc(stmt)
print(result)
return result["1"]


@app.route("/log_today")
def logToday():
if isLogged():
sql = "SELECT AMOUNT,CATEGORY,NEED FROM TRANSACTIONS WHERE
USER_ID=? AND DATEADDED=CURRENT_DATE"
stmt = ibm_db.prepare(conn, sql)
expenseData = fetchall(stmt)
print(expenseData)
expenseTotal = getTotal("TRANSACTIONS")
sql = "SELECT AMOUNT FROM income WHERE ID=? AND
DATEADDED=CURRENT_DATE"
stmt = ibm_db.prepare(conn, sql)
```

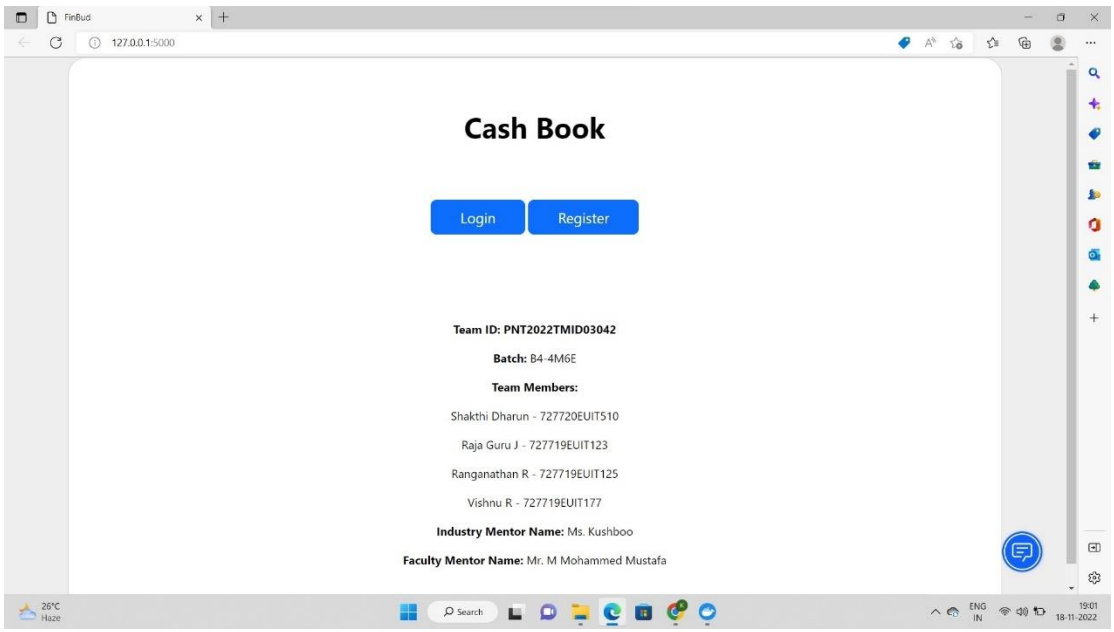
```
incomeData = fetchall(stmt)
print(incomeData)
return render_template(
"logtoday.html",
title="Today's Log",
expenseData=expenseData,
incomeData=incomeData,
expenseTotal=expenseTotal,
)
else:
flash("Login First")
return redirect("/login")


@app.route("/addExpense/", methods=["POST", "GET"])
def addExpense():
msg = ""
if request.method == "POST":
amount = request.form["Amount"]
need = request.form["Need/Want"]
category = request.form["category"]
sql = "INSERT INTO TRANSACTIONS(USER_ID,AMOUNT,NEED,CATEGORY,DATEADDED) VALUES(?,?,?,?,CURRENT_DATE)"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, session["id"])
ibm_db.bind_param(stmt, 2, amount)
ibm_db.bind_param(stmt, 3, need)
ibm_db.bind_param(stmt, 4, category)
if ibm_db.execute(stmt):
msg = "Successfully Added Expense!!!!"
else:
msg = "Expense not added"

flash(msg)
return redirect(url_for("logToday"))


@app.route("/addIncome/", methods=["POST", "GET"])
def addIncome():
msg = ""
if request.method == "POST":
amount = request.form["AmountIncome"]
```

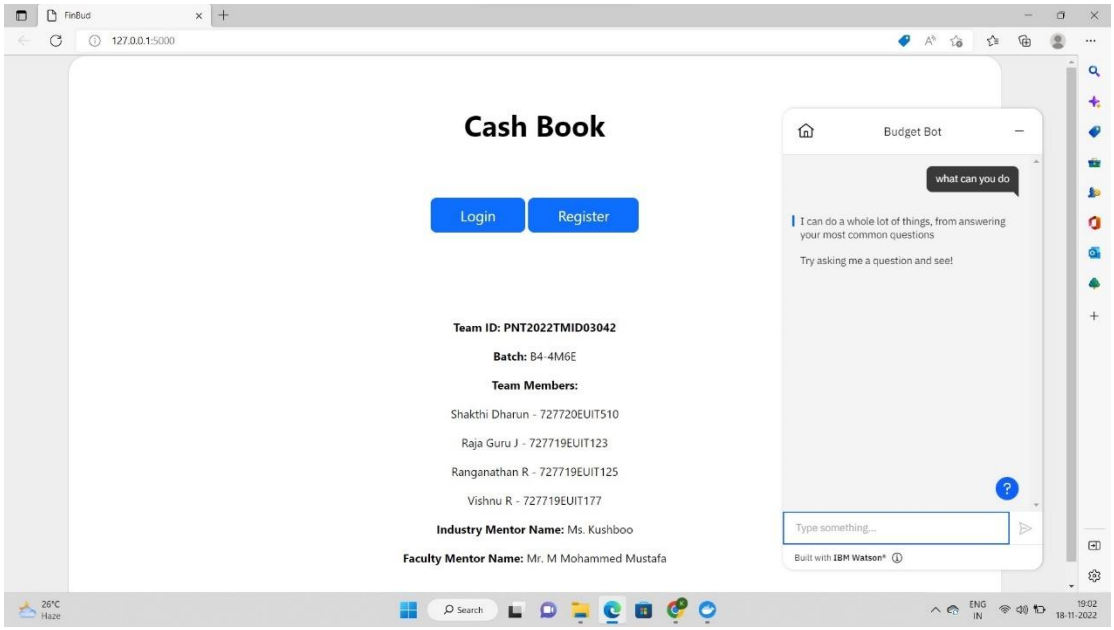
```
sql = "INSERT INTO INCOME(ID,AMOUNT,DATEADDED) VALUES(?,?,CURRENT_DATE)"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, session["id"])
ibm_db.bind_param(stmt, 2, amount)
if ibm_db.execute(stmt):
msg = "Successfully Added Income!!!!"
else:
msg = "Income not added"

flash(msg)
return redirect(url_for("logToday"))


# @app.route("/Edit")
###Visualization functions


@app.route("/reports")
def reports():
return render_template("reports.html", title="Reports")


@app.route("/needVwant/")
def needVwant():
sql = "SELECT Sum(amount) AS amount, need FROM transactions WHERE DAYS(CURRENT_DATE)-DAYS(DATEADDED)<29 AND  user_id = ? GROUP BY NEED ORDER BY need"
stmt = ibm_db.prepare(conn, sql)
transactions = fetchall(stmt)
values = []
labels = []
print(transactions)
for transaction in transactions:
values.append(transaction["AMOUNT"])
labels.append(transaction["NEED"])
fig = plt.figure(figsize=(10, 7))
plt.pie(values)
plt.title("Need v Want")
plt.legend(["WANT", "NEED"])
canvas = FigureCanvas(fig)
img = BytesIO()
fig.savefig(img)
img.seek(0)
```

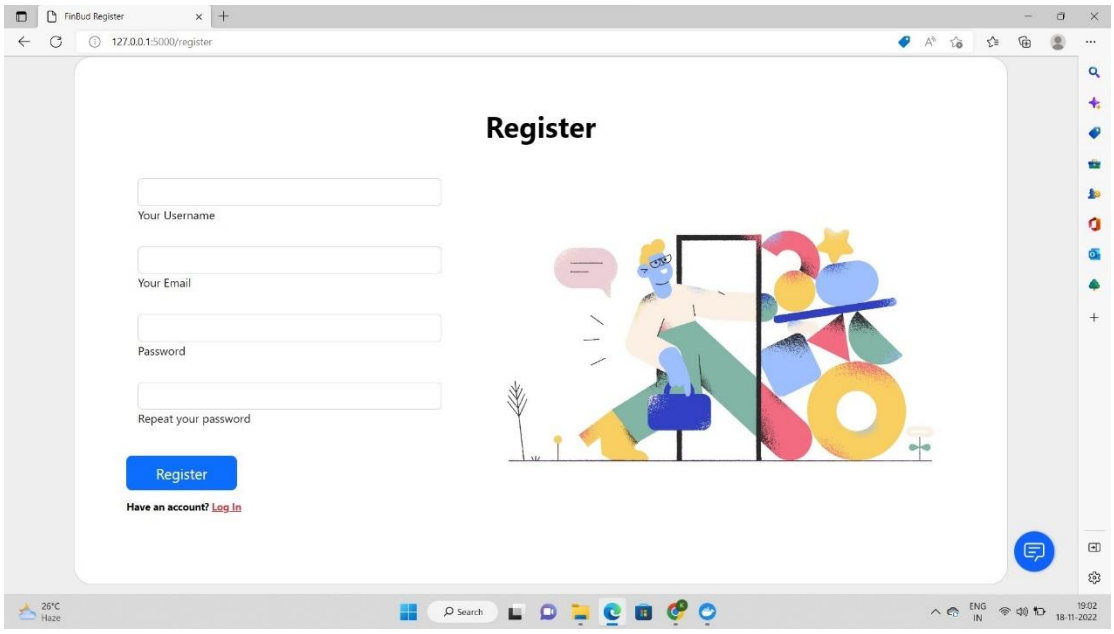
```
return send_file(img, mimetype="image/png")


@app.route("/categoriesChart/")
def categoriesChart():
sql = "SELECT Sum(amount) AS amount, category FROM transactions WHERE DAYS(CURRENT_DATE)-DAYS(DATEADDED)<29 AND  user_id = ? GROUP BY category ORDER BY category"
stmt = ibm_db.prepare(conn, sql)
transactions = fetchall(stmt)
values = []
labels = []
print(transactions)
for transaction in transactions:
values.append(transaction["AMOUNT"])
labels.append(transaction["CATEGORY"])
fig = plt.figure(figsize=(10, 7))
plt.pie(values, labels=labels)
plt.title("Categories")
plt.legend()
canvas = FigureCanvas(fig)
img = BytesIO()
fig.savefig(img)
img.seek(0)
return send_file(img, mimetype="image/png")


##edit the legend... all visualizations workkkkkk!!!!!!!
@app.route("/dailyLineChart/")
def dailyLineChart():
sql = "SELECT Sum(amount) AS amount, DAY(dateadded) as dateadded FROM transactions WHERE DAYS(CURRENT_DATE)-DAYS(DATEADDED)<29 AND user_id = ? GROUP BY dateadded ORDER BY dateadded"
stmt = ibm_db.prepare(conn, sql)
transactions = fetchall(stmt)
x = []
y = []
print(transactions)
for transaction in transactions:
y.append(transaction["AMOUNT"])
x.append(transaction["DATEADDED"])
##get budget
sql = "SELECT MAXBUDGET FROM budgets WHERE id = ?"
stmt = ibm_db.prepare(conn, sql)
```

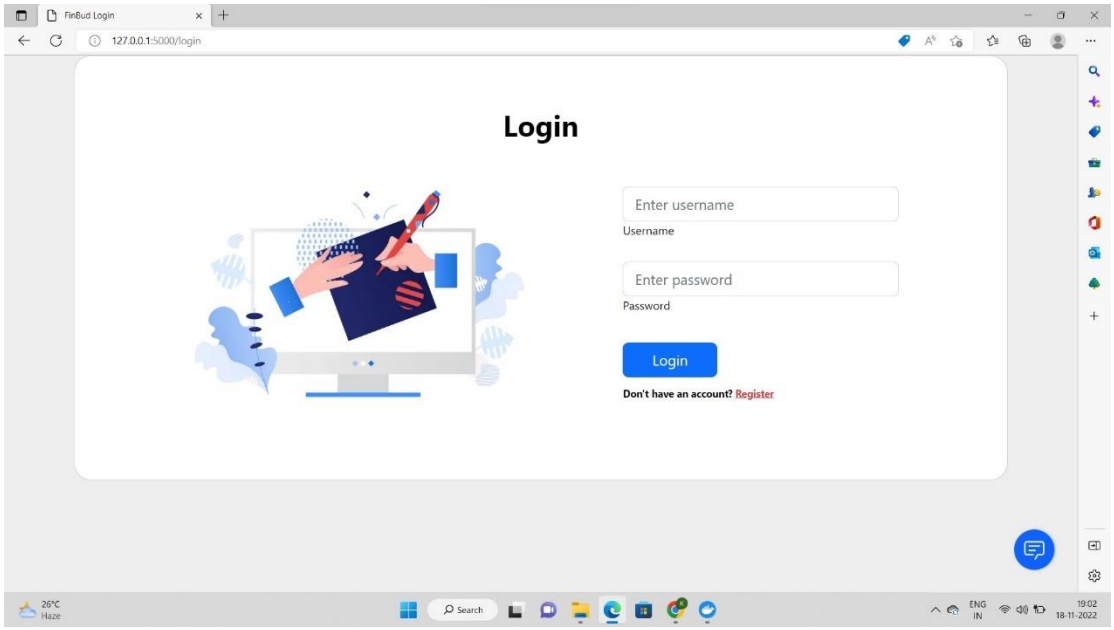
```
ibm_db.bind_param(stmt, 1, session["id"])
ibm_db.execute(stmt)
budget = ibm_db.fetch_assoc(stmt)
print(budget)
fig = plt.figure(figsize=(10, 7))
plt.scatter(x, y)
plt.plot(x, y, "-")
if budget:
plt.axhline(y=budget["MAXBUDGET"], color="r", linestyle="-")
plt.xlabel("Day")
plt.ylabel("Transaction")
plt.title("Daily")
plt.legend()
canvas = FigureCanvas(fig)
img = BytesIO()
fig.savefig(img)
img.seek(0)
return send_file(img, mimetype="image/png")


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

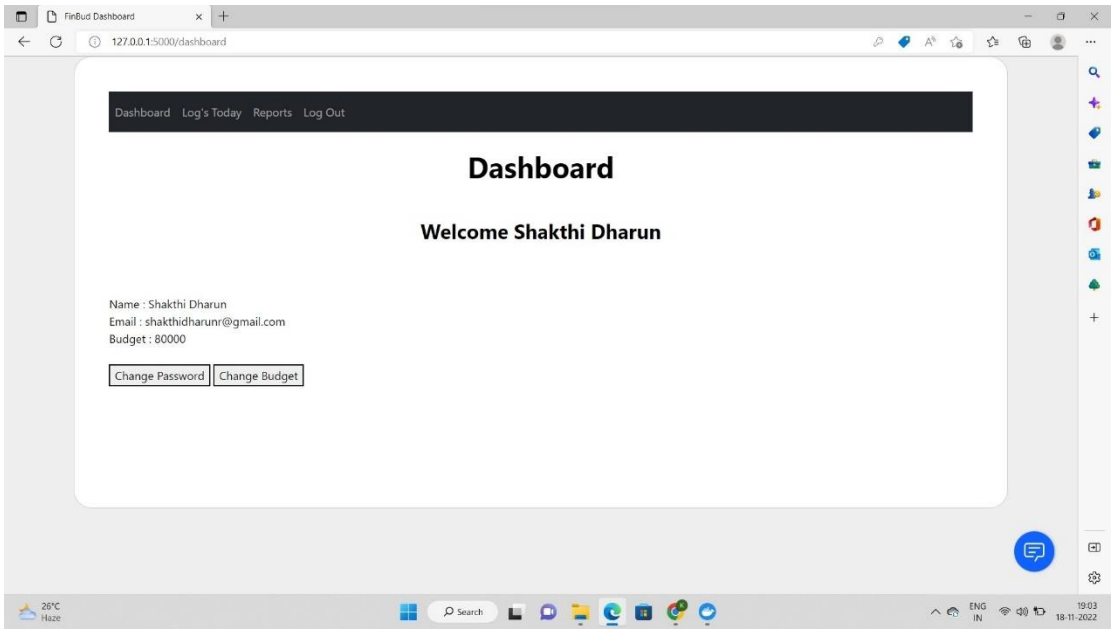
Screen Shots:

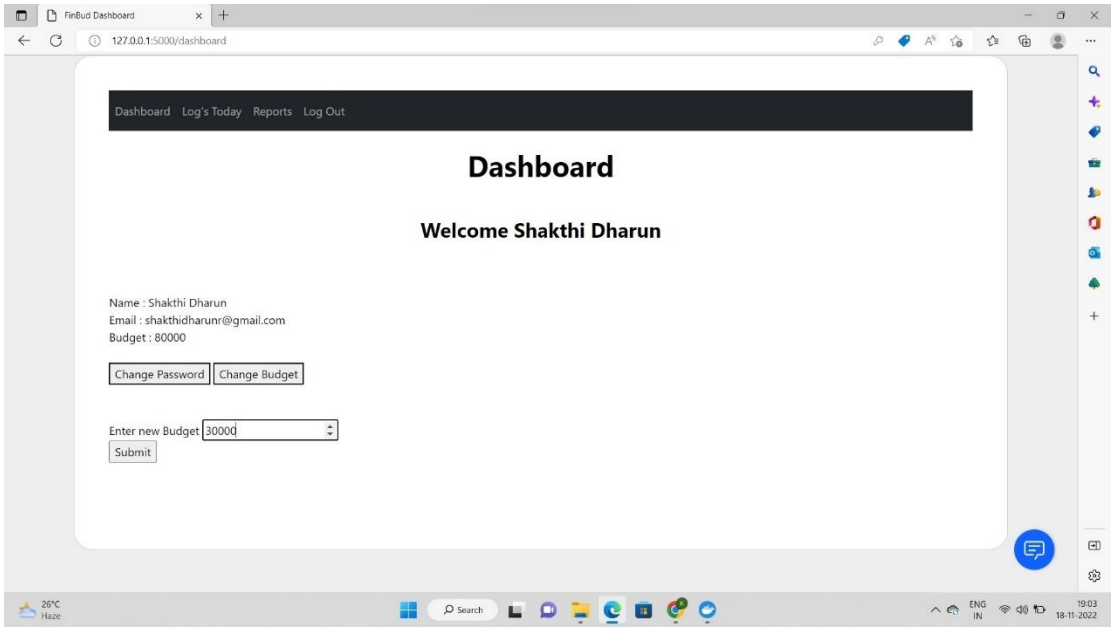


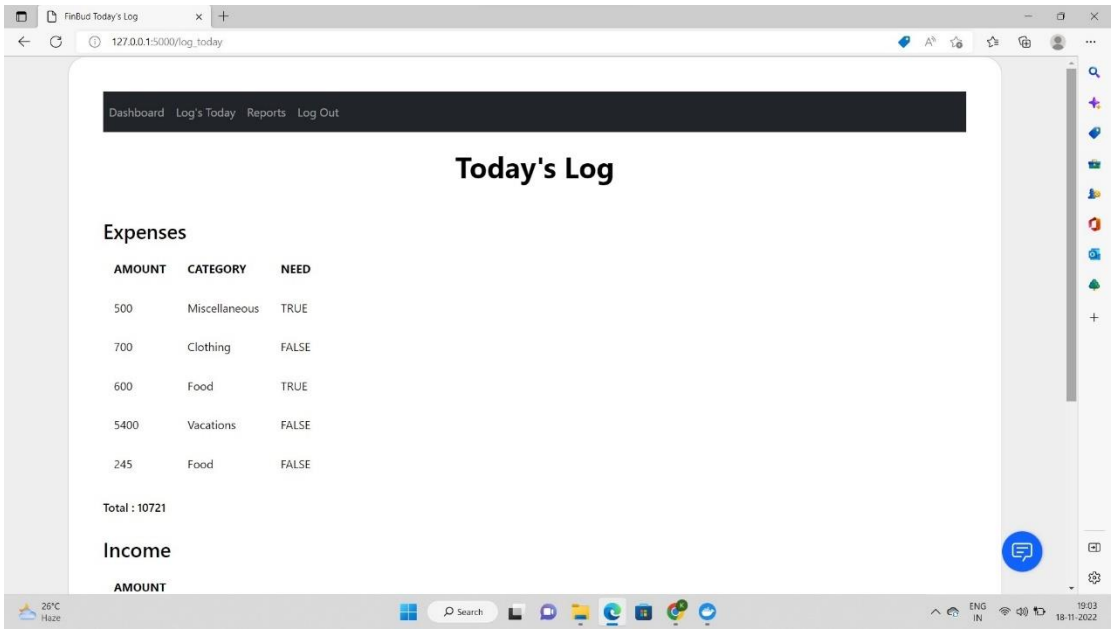

FinBud
127.0.0.1:5000
Cash Book
Login
Register
Team ID: PNT2022TMID03042
Batch: B4-4M6E
Team Members:
Shakthi Dharun - 727720EUIT510
Raja Guru J - 727719EUIT123
Ranganathan R - 727719EUIT125
Vishnu R - 727719EUIT177
Industry Mentor Name: Ms. Kushboo
Faculty Mentor Name: Mr. M Mohammed Mustafa
Budget Bot
what can you do
I can do a whole lot of things, from answering your most common questions
Try asking me a question and see!
Type something...
Built with IBM Watson®

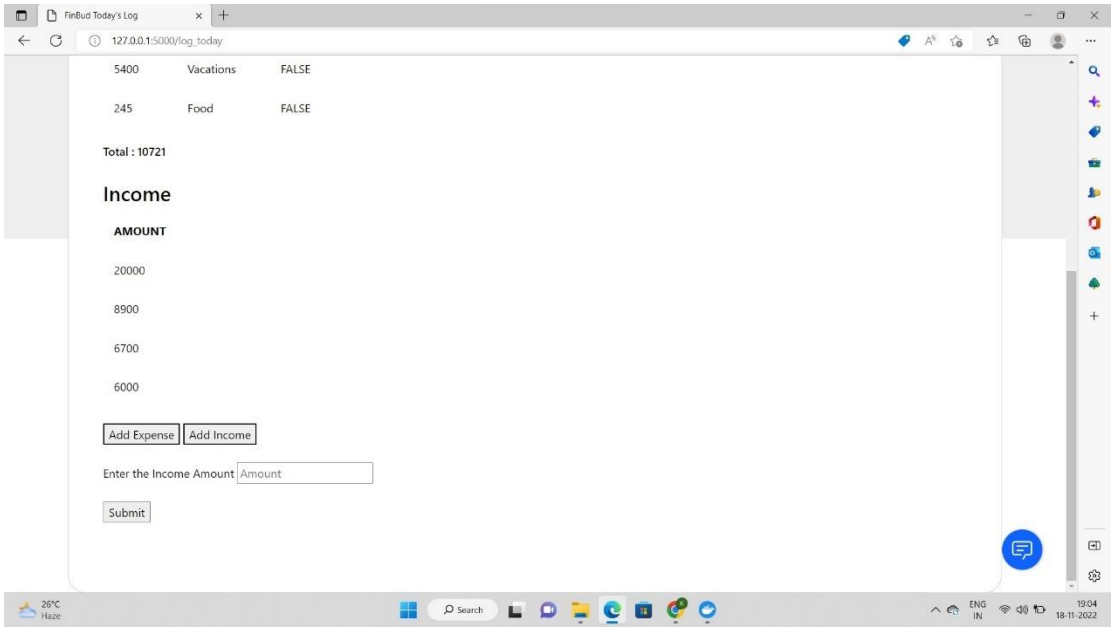

FinBud Register
127.0.0.1:5000/register
Register
Your Username
Your Email
Password
Repeat your password
Register
Have an account? Log In

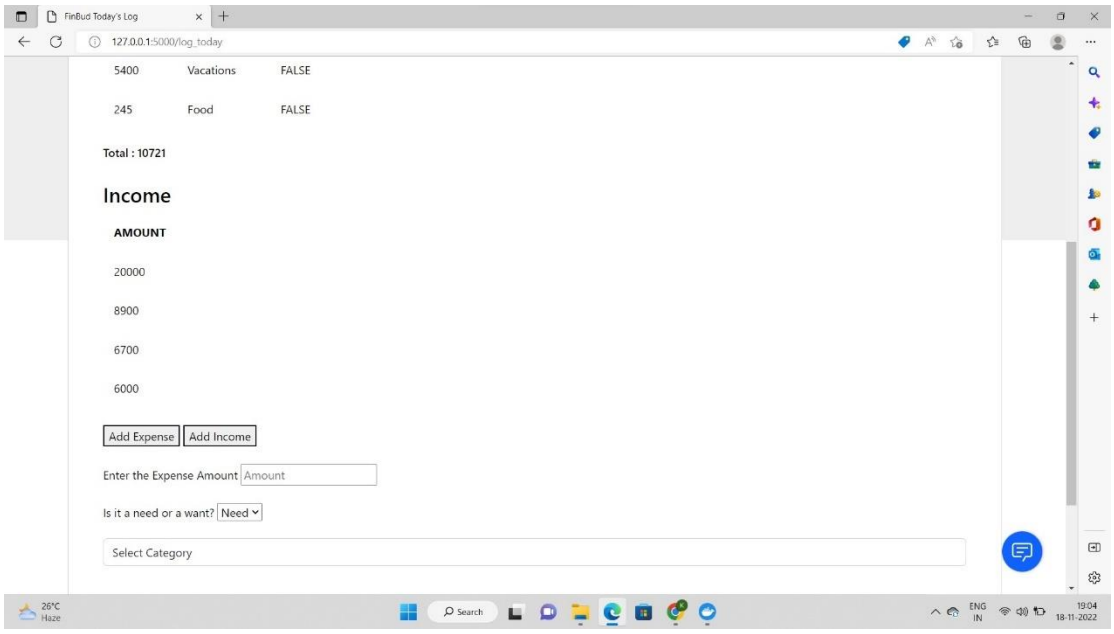

FinBud Login
127.0.0.1:5000/login
Login
Enter username
Username
Enter password
Password
Login
Don't have an account? Register

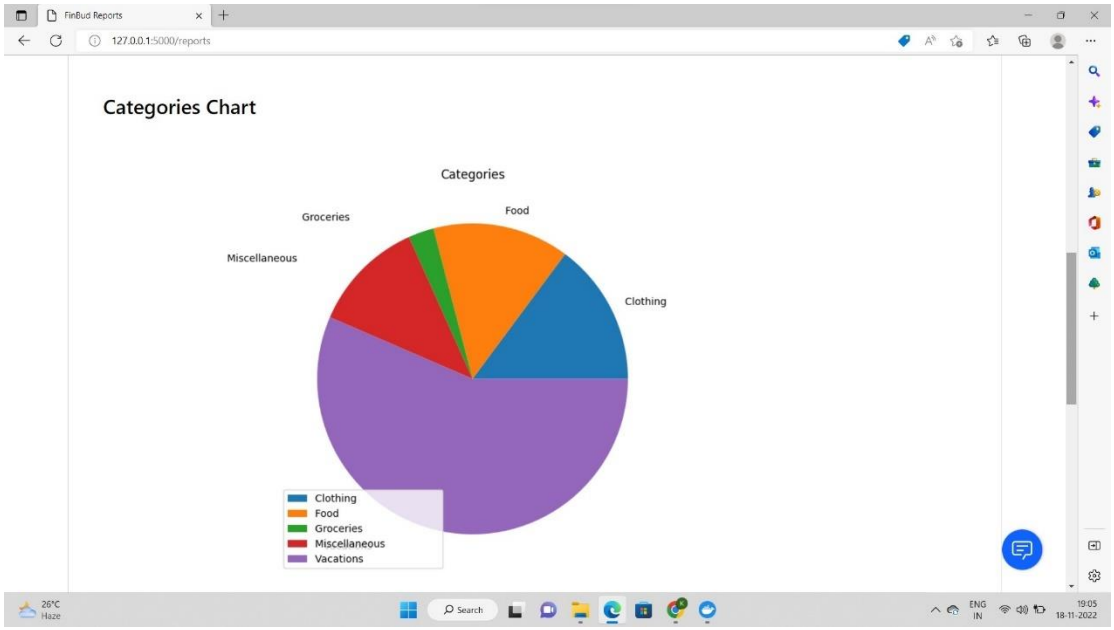

FinBud Dashboard
127.0.0.1:5000/dashboard
Dashboard Log's Today Reports Log Out
Dashboard
Welcome Shakthi Dharun
Name : Shakthi Dharun
Email : shakthidharunr@gmail.com
Budget : 80000
Change Password
Change Budget

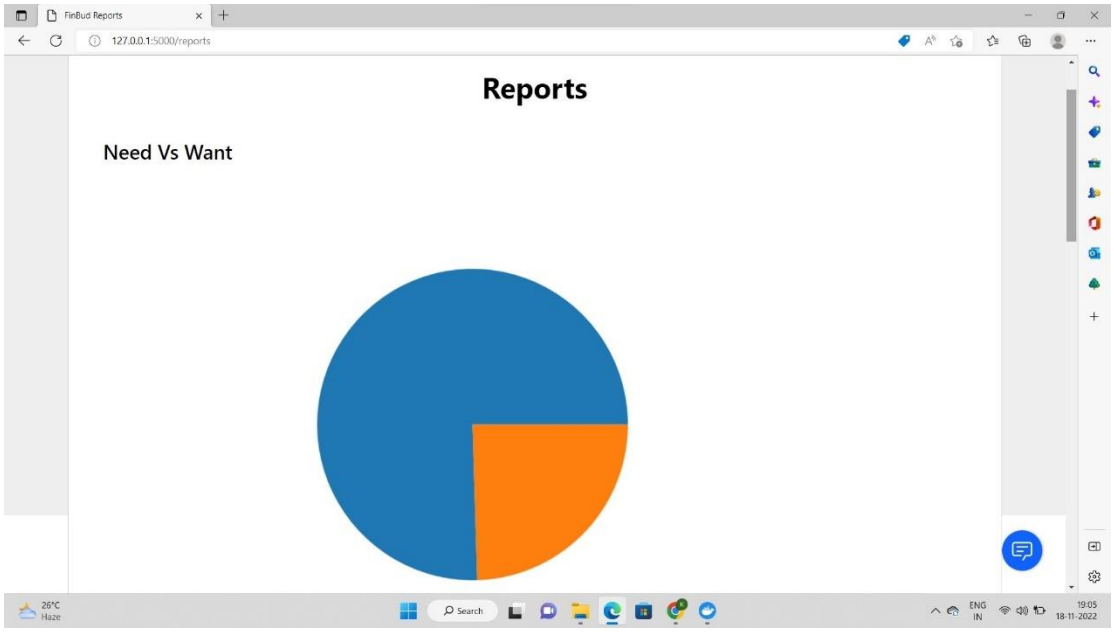

FinBud Dashboard
127.0.0.1:5000/dashboard
Dashboard Log's Today Reports Log Out
Dashboard
Welcome Shakthi Dharun
Name : Shakthi Dharun
Email : shakthidharunr@gmail.com
Budget : 80000
Change Password
Change Budget
Enter new Budget 30000
Submit


FinBud Today's Log
127.0.0.1:5000/log_today
Dashboard Log's Today Reports Log Out
Today's Log
Expenses
AMOUNT CATEGORY NEED
500 Miscellaneous TRUE
700 Clothing FALSE
600 Food TRUE
5400 Vacations FALSE
245 Food FALSE
Total : 10721
Income
AMOUNT


FinBud Today's Log
127.0.0.1:5000/log_today
5400 Vacations FALSE
245 Food FALSE
Total : 10721
Income
AMOUNT
20000
8900
6700
6000
Add Expense
Add Income
Enter the Income Amount
Amount
Submit


FinBud Today's Log
127.0.0.1:5000/log_today
5400 Vacations FALSE
245 Food FALSE
Total : 10721
Income
AMOUNT
20000
8900
6700
6000
Add Expense
Add Income
Enter the Expense Amount
Amount
Is it a need or a want?
Need
Select Category


FinBud Reports
127.0.0.1:5000/reports
Categories Chart
Categories
Groceries
Food
Miscellaneous
Clothing
Clothing
Food
Groceries
Miscellaneous
Vacations
26°C
Haze
Search
ENG
IN
19:05
18-11-2022


FinBud Reports
127.0.0.1:5000/reports
Reports
Need Vs Want
26°C
Haze
Search
ENG
IN
19:05
18-11-2022

GitHub Link:
https://github.com/IBM-EPBL/IBM-Project-24165-1659939053

Demo Video Link:
Original:
https://drive.google.com/file/d/1CTweZC7XSNU1eAV7sHtczDDPQ3z5hehY/view?usp=share_link

Compressed:
https://drive.google.com/file/d/1TrcDTMFgERwi_Np8igE3S166UB7dkmm_/view?usp=share_link