PERSONAL EXPENSE TRACKER APPLICATION

1. INTRODUCTION:

1.1. PROJECT OVERVIEW:

Daily Expense Tracker System is a system which will keep a track of Income-Expense of a House-Wife on a day to day basics, This System takes Income from House-Wife and divides in daily expense allowed, If u exceed that days expense it will cut if from your income and give new daily expense allowed Amt, and if that days expense is less it will add it in savings. Daily expense tracking System will generate report at the end of month to show Income-Expense Curve. It will let you add the savings amt which you had saved for some particular Festivals or day like Birthday or Anniversary.

1.2. PURPOSE:

Also known as expense manager and money manager, an expense tracker is a software or application that helps to keep an accurate record of your money inflow and outflow. Many people in India live on a fixed income, and the find that towards the end of the month don’t have sufficient money to meet their needs.

2. LITERATURE SURVEY:

2.1. EXISTING PROBLEM:

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.

If you are spending money frivolously, you will not have money to set financial goals. However, when you have a daily expense manager, you will be

able to work with limited resources and use your money in a wise manner so that you can create financial goals and ensure you meet them.

2.2. REFERANCE:

1. eExpense: A Smart Approach to Track Everyday Expense. Publisher:

IEEE 2018 Conference

AUTHORS: Shahed Anzarus Sabab, Sadman Saumik Islam, Md. Jewel Rana, Monir Hossain

2. Expense Manager Application. Publisher: IEEE 2020 Conference

AUTHORS: Velmurugan A, Albert Mayan J, Niranjana P and Richard Francis

3. Expense Tracker Application. Publisher: IEEE 2021 Conference

AUTHORS: Velmurugan. R , Mrs. P. Usha

4. Expense Tracker: A Smart Approach to Track Everyday Expense. Publisher: IEEE 2020 Conference

AUTHORS: Hrithik Gupta, Anant Prakash Singh, Navneet Kumar and J. Angelin Blessy

5. Expense Tracker. Publisher: IEEE 2021 Conference

AUTHORS: Atiya kazi, Praphulla S. Kherade, Raj S. Vilankar, Parag M.Sawant

2.3. PROBLEM STATEMENT DEFINITION:

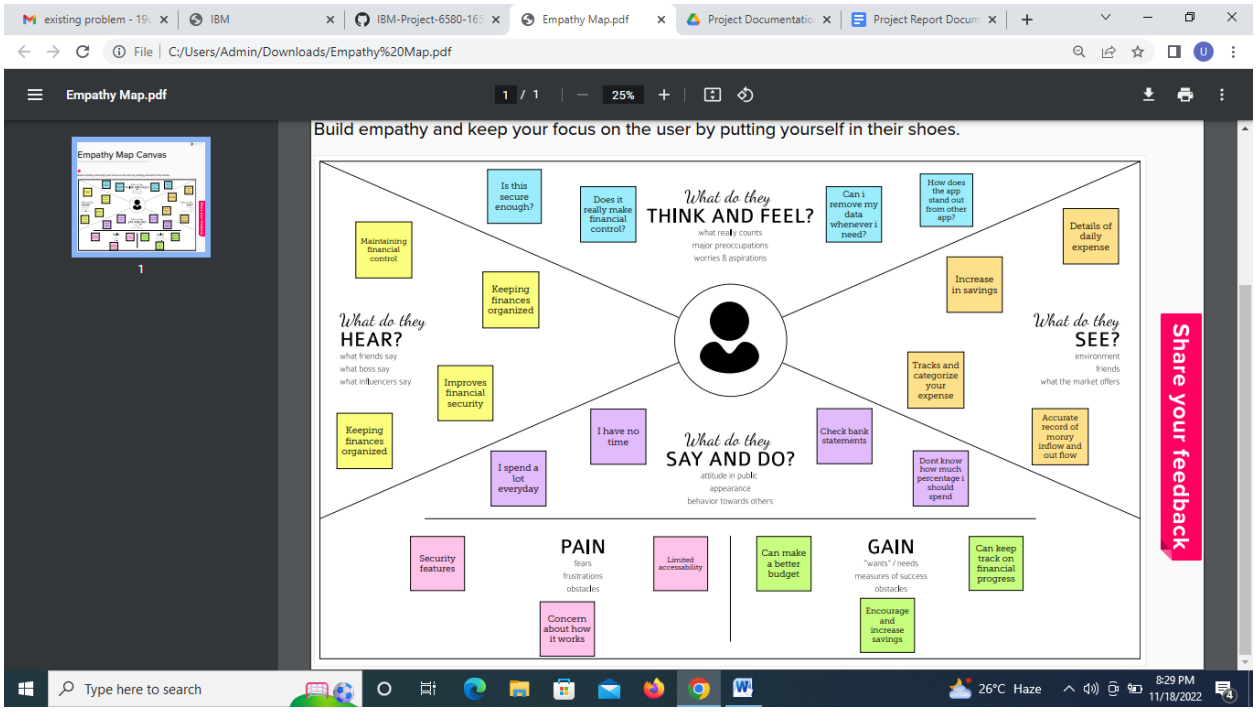
Now-a-days many peoples are not able to keep track of their daily expenses. They can't even have a control over their daily expense. In such a case personal expense tracker application will help them to have a control over their daily expense.

Problem Statement (PS)	**I am (Customer)**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	Software Engineer	Budgeting allmy expense	I have lack of spend	I don't have any expense tracker in my	Unplanned

			visibility	mobile phone	
PS-2	Financer	Do expense calculation without using calculator	I don't know how todo it	I don't haveany idea about calculation without calulator	Insecured

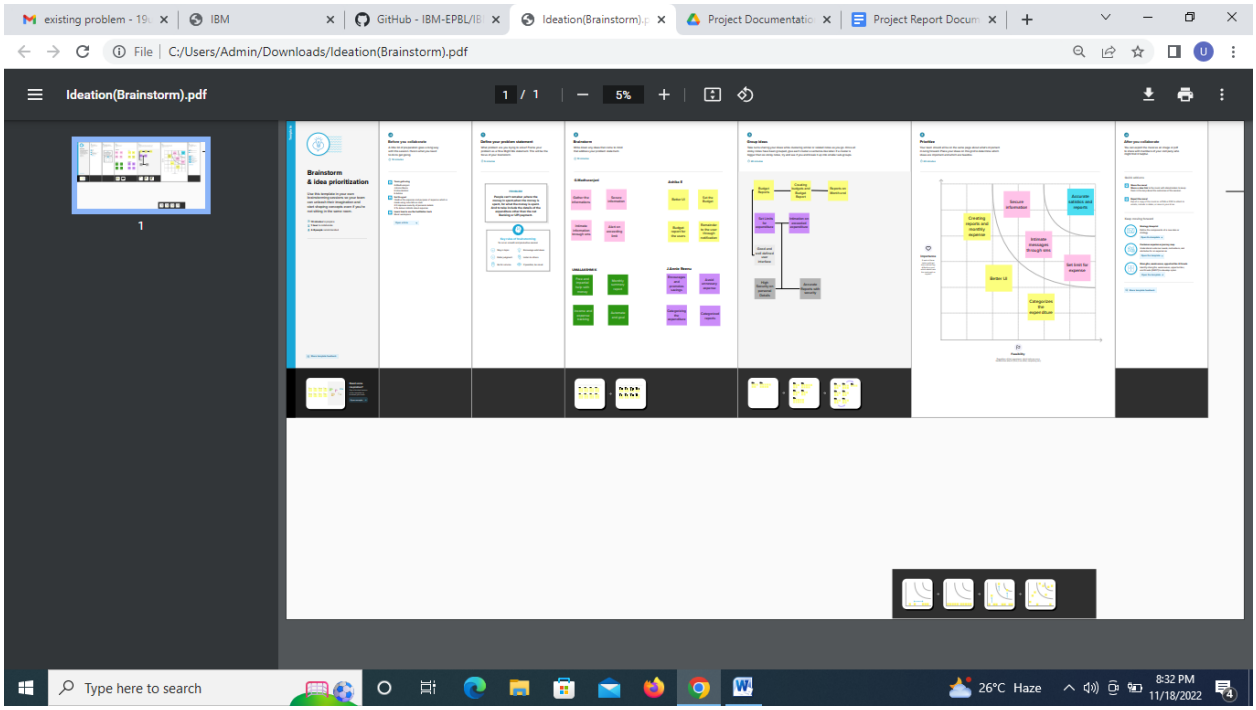
3. IDEATION & PROPOSED SOLUTION:

3.1. EMPATHY MAP CANVAS:



3.2. IDEATION AND BRAINSTROMING:

To list by organizing brainstorming sessions and prioritize the top three ideas based on feasibility and importance.



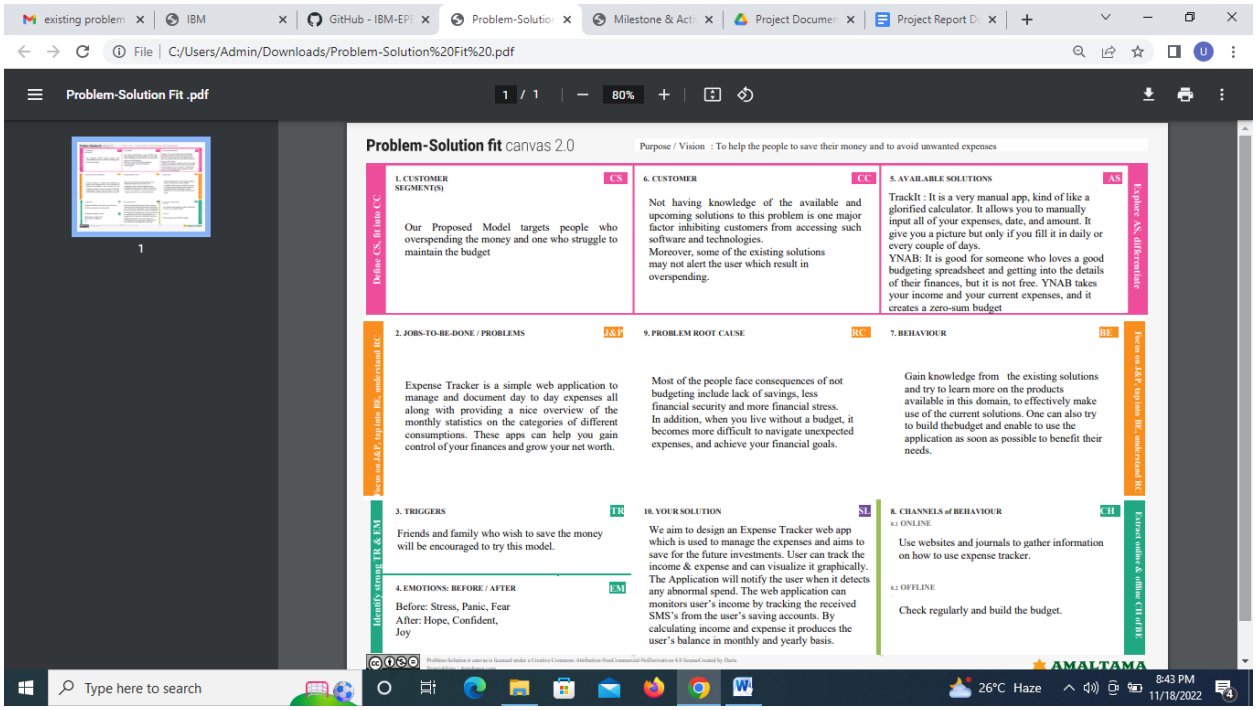
3.3. PROPOSED SOLUTION:

To prepare the proposed solution documents, which includes the novelty, feasibility of ideas, business model, social impact, scalability of the solution, etc.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To track regular expense of a individual person to maintain budget
2.	Idea / Solution description	Our application will track the regular expense of a person which help them to live a budget friendly life. The daily and monthly expense of an individual will be tracked using this application.
3.	Novelty / Uniqueness	This application is collaborated with cloud. All the informations in the application will be stored in the cloud

		which can be easily retrived whenever we want.
4.	Social Impact / Customer Satisfaction	The personal expense tracker tracks the daily expense and calculate the daily expense which helps a person to avoid unnecessary expense and makem them to save more money.Which will lead to a happy life
5.	Business Model (Revenue Model)	Free trial for 1 month can be given to the users, so that a significant user base is created. Following the free trial, the users can be given subscription for 3 months, 6 months or 1 year.
6.	Scalability of the Solution	More number of users can be managed effectively, since entire application is hosted on cloud. It also help us to review the expense of the previous month.

3.4. PROPOSED SOLUTION FIT:



4. REQUIRED ANALYSIS:

4.1. FUNCTIONAL REQUIREMENT:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	Monthly refreshments	Personal expense tracker application shall allow the user to add the data to the expanses.
FR-3	Planner	It helps to show the graphical representation to the users about previous month expense results.
FR-4	Tracker	To track the expense flow is to decreased or increased compared to the previous month and current month.
FR-5	Category	It will help the user to add new categories or

		to delete the existing categories in the application

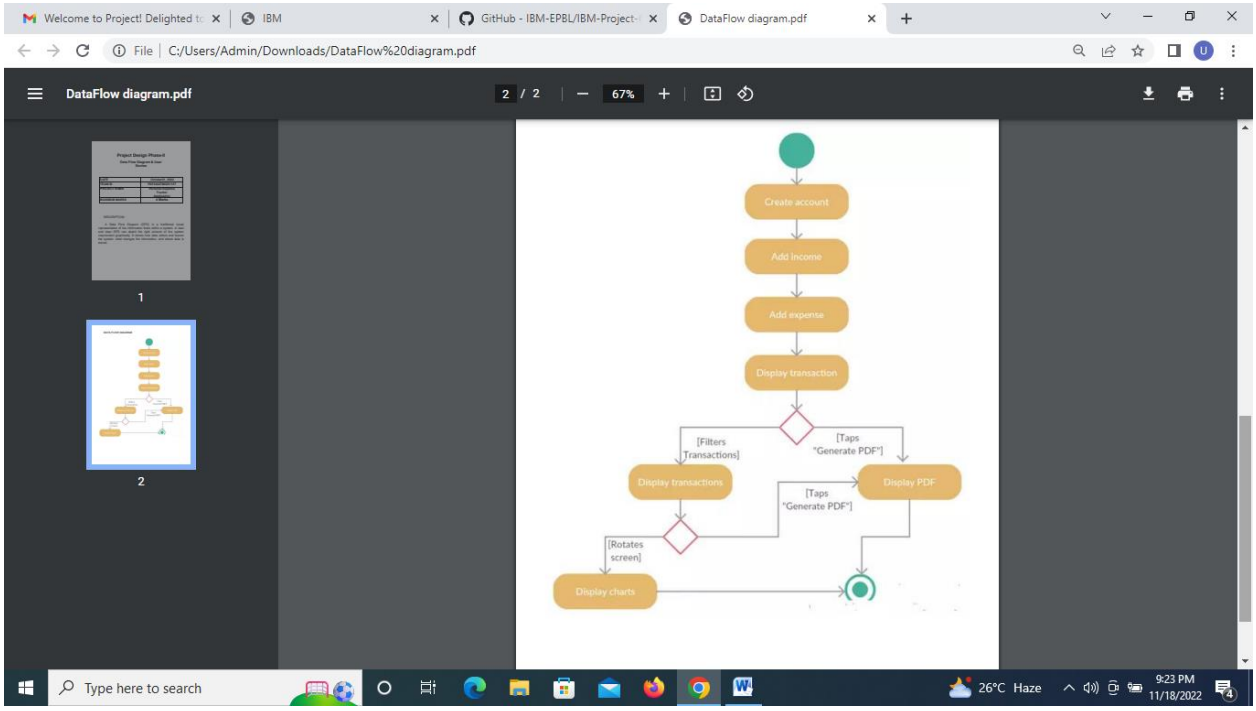
4.2. NON FUNCTIONAL REQUIREMENTS:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	It is very mush user friendly one. It is easy to calculate our expenses and track the money flow.
NFR-2	**Security**	More data security of the user bank account and payment data sheet.
NFR-3	**Reliability**	Each data is stored in the well formed database sheet with heavy security.
NFR-4	**Performance**	It has to give the users to the options to add or delete the categories and to track the money flow. And then to alert the users if more money flow in the particular category.
NFR-5	**Availability**	It is the application to be available in offline and to track our money flow in monthly.
NFR-6	**Scalability**	It has the ability to add more than 25 categories in one month with backup option.

5. PROJECT DESIGN:

5.1. DATA FLOW DIAGRAMS:

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

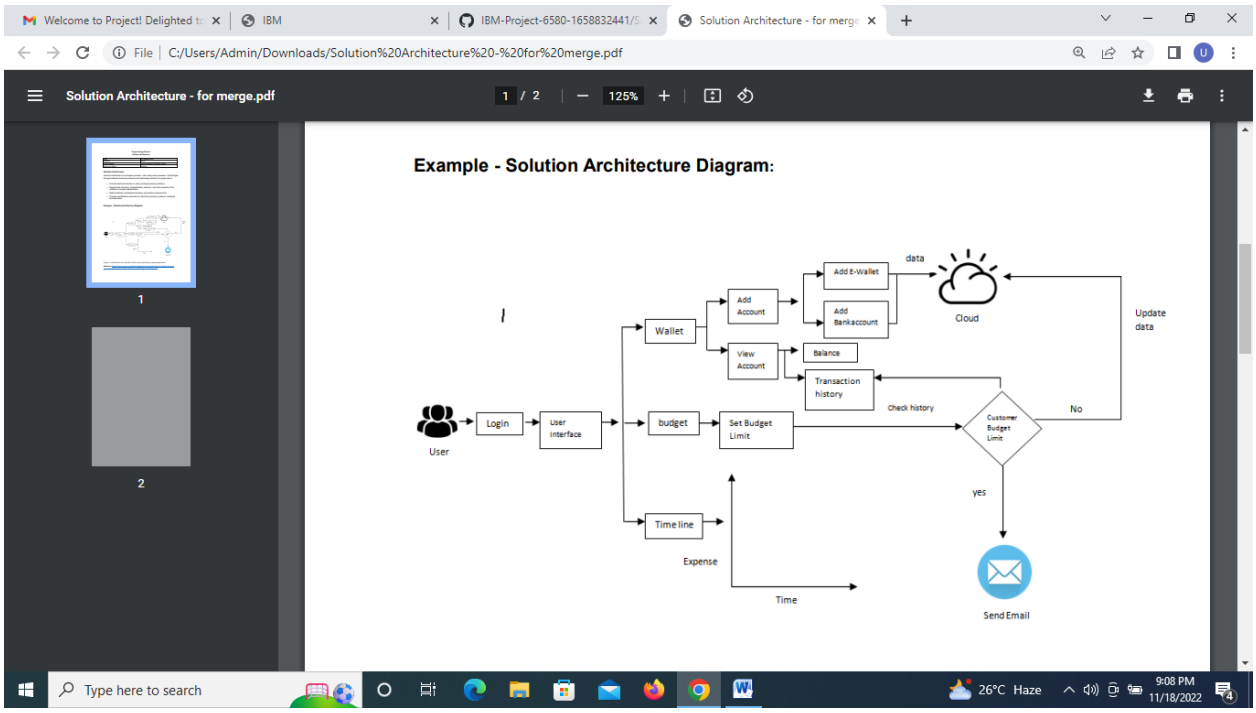


5.2. SOLUTION & TECHNICAL ARCHITECTURE:

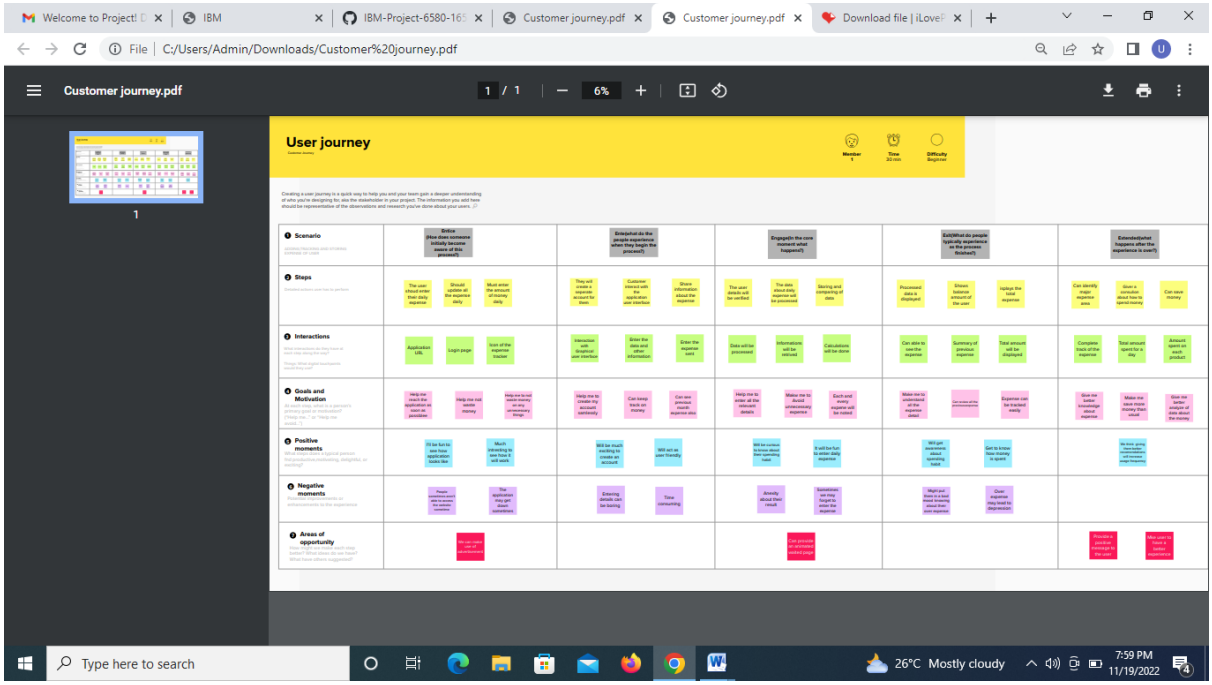
Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.

- Provide specifications according to which the solution is defined, managed, and delivered.



5.3. USER STORIES:



6. PROJECT PLANNING & SCHEDULING:

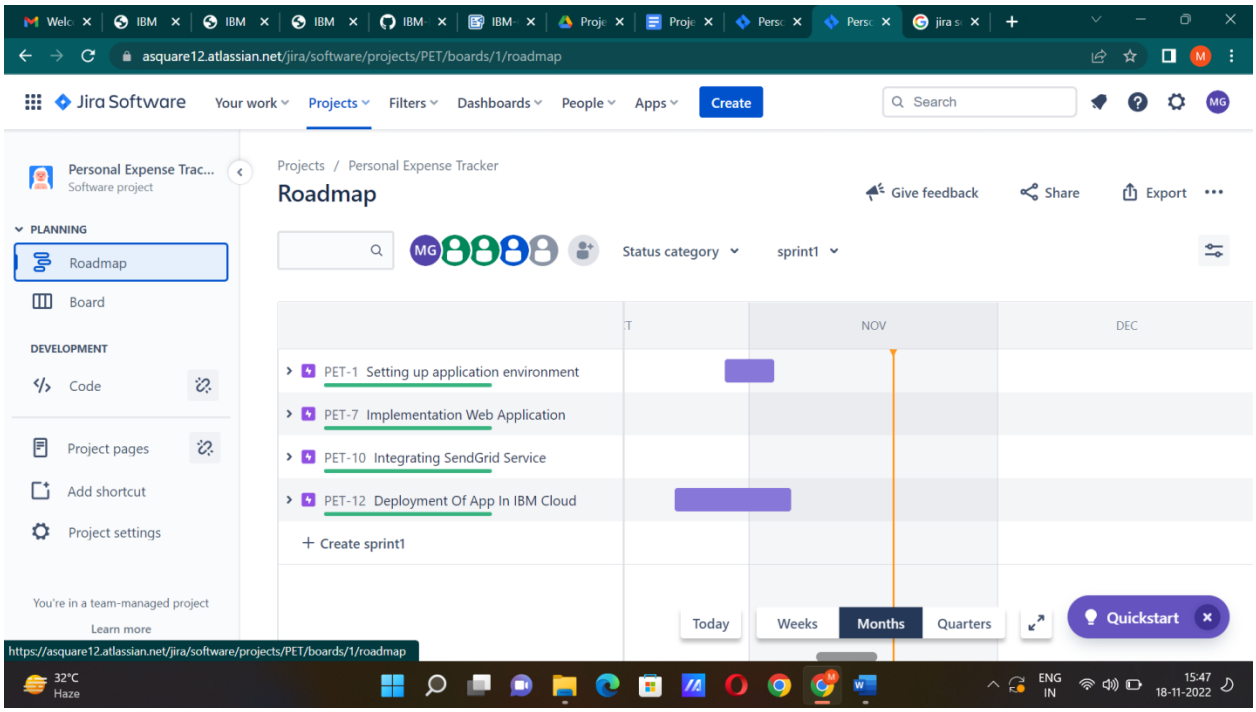
6.1. SPRINT PLANNING & ESTIMATION:

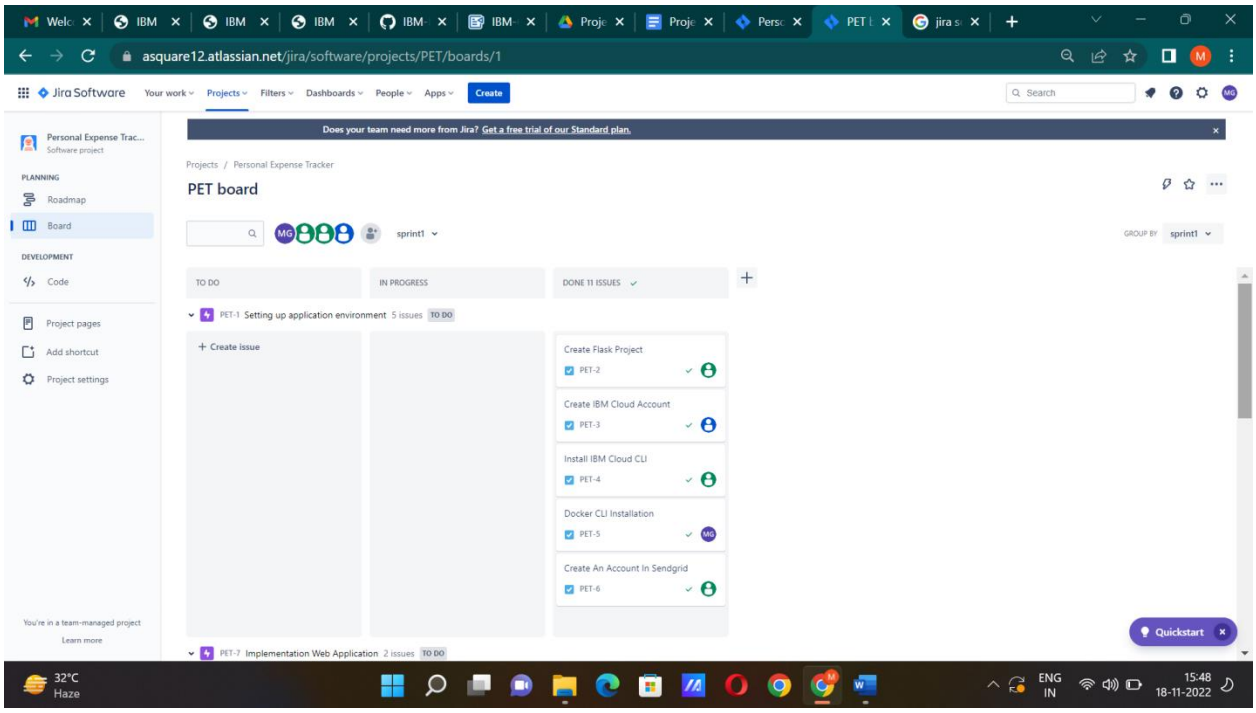
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	Registration	USN-1	As a user, I can register for the application byentering my email, password, and confirmingmy password.	2	High
Sprint-1		USN-2	As a user, I will receive confirmation emailonce I have registered for the application	1	High
Sprint-2		USN-3	As a user, I can register for the application through Facebook	2	Low
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium
Sprint-1	Login	USN-5	As a user, I can log into the application byentering email & password	1	High
Sprint-3	Dashboard	USN-6	As a user I can see the expenditure details on the application	3	High
Sprint-3	Limits	USN-6	As a user I can set my monthly expense limit so that I receive a mail on exceeding that	4	High
Sprint-4	Reports	USN-6	As a user I can view the graphical form of my expenses category wise	5	Medium

6.2. 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	6	6 Days	24 Oct 2022	29 Oct 2022	6	29 Oct 2022
Sprint-2	2	6 Days	31 Oct 2022	05 Nov 2022	2	05 Nov 2022
Sprint-3	7	6 Days	07 Nov 2022	12 Nov 2022	7	12 Nov 2022
Sprint-4	5	6 Days	14 Nov 2022	19 Nov 2022	5	19 Nov 2022

6.3. REPORTS FROM JIRA:





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

7.1. FEATURE 1:

dockerfile.txt:

```
FROM
python:3.6
        WORKDIR /app
        ADD . /app
        COPY requirements.txt /app
        RUN python3 -m pip install -r requirements.txt
        RUN python3 -m pip install ibm_db
        EXPOSE 5000
        CMD ["python","app.py"]
        # FROM python:3.10-alpine
        # WORKDIR /app
        # ADD . /app
        # RUN set -e; \
        # apk add --no-cache --virtual .build-deps \
        # gcc \
        # libc-dev \
        # linux-headers \
        # mariadb-dev \
        # python3-dev \
        # ;
        # COPY requirements.txt /app
        # RUN pip3 install -r requirements.txt
        # CMD ["python3","app.py"]
```

7.2. FEATURE 2:

flask-service.yaml:

```
apiVersion:
v1
```

kind: Service

metadata:

name: flask-app-service

spec:

selector:

app: flask-app

ports:

- name: http

protocol: TCP

port: 80

targetPort: 5000

type: LoadBalancer

7.3. DATABASE SCHEMA:

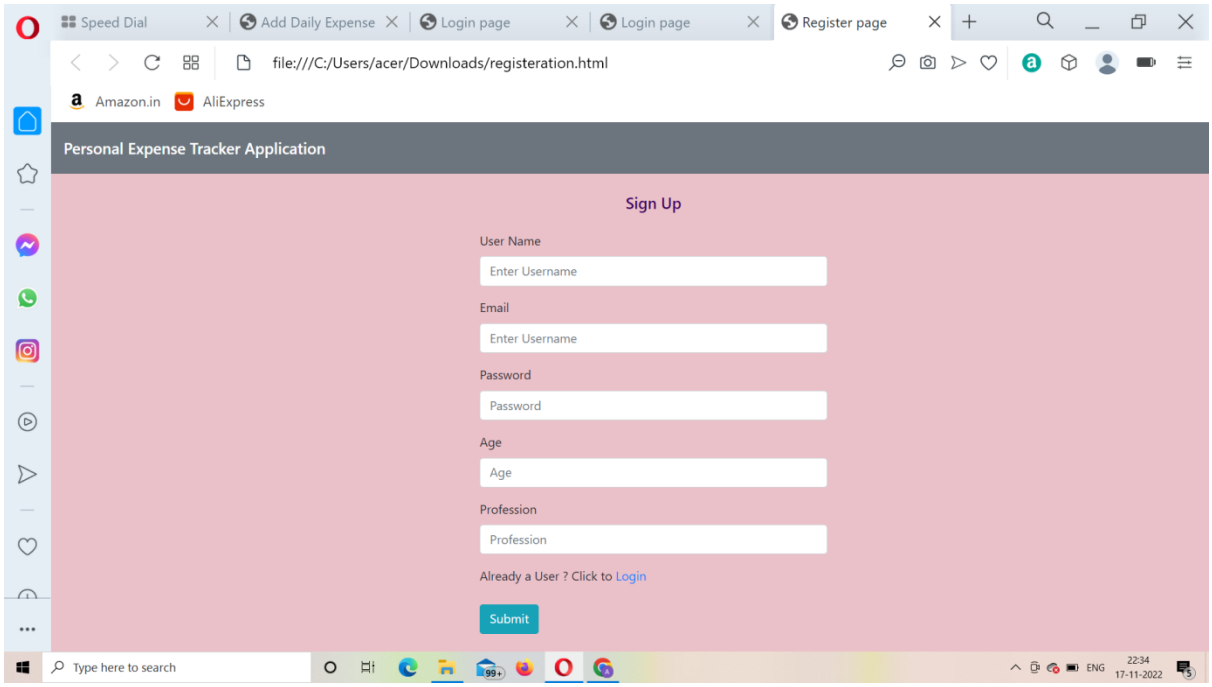
8. TESTING:

8.1. TEST CASES:

Test case ID	Purpose	Test Cases	Result
TC1	Authentication	User name with length less than 2 characters	User name cannot be less than 2 characters
TC2	Authentication	Valid user name with minimum 2 characters	User name accepted
TC3	Authentication	User name left blank	User name cannot be less than 2 characters
TC4	Authentication	Password field	Password cannot

		left blank	be empty
TC5	Authentication	Password with length less than 4 characters	Password cannot be less than 4 characters
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
TC9	Authentication	Security question with length less than 3 characters	Security question cannot be less than 3 characters

8.2. USER ACCEPTANCE TESTING:



9. RESULTS:

9.1. PERFORMANCE METRICS:

The art of money management is all about turning your money into wealth by reframing your mindset; instead of thinking of managing money in terms of just expenses, you should also think of money as an investment tool. A defined money management plan incorporates wealth accumulation, protection of accumulated wealth, and preservation of wealth. These key financial concepts are tied to your specific needs, objectives, financial goals, priorities, and risk factors.

In a B2B scenario, businesses often find it hard to focus on money management due to varied cash flows. Therefore, businesses shift their focus to behavioral influences (spending, savings, investments) that affect their decision-making strategies for managing their money.

10. ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

With cloud storage, the finance team can easily access digital receipts and expense reports on any device at any time. The application will track all of your data for you. It doesn't do it once a week or once a month, like you might if you were doing it manually. That you have a day by day way of checking to ensure that you're on track and moving in the right direction.

Ability to monitor costs incrementally:

Tracking expenses throughout a project provides you the ability to view various expense categories and time periods. This can help you understand how much money you can spend for the rest of the project while staying within your budget.

Allows for budgeting on future projects:

If you record your expenses for a project, you can use that information to budget for similar future projects. For example, if you license software to complete a project, you can include that budget if you need to use that software for other initiatives.

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.

11. CONCLUSION:

Personal expense tracker application will help of keep track of your daily expense which will help you to save more money. We see how much money we've spent for each and everything. So that we can avoid unnecessary expenditure. It will help you to save money by tracking the expense.

12. FUTURE SCOPE:

It will have various options to keep record (for example food, Traveling fuel salary etc). Automatically it will keep on sending notifications for our daily expenditure.

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

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

13. APPENDIX

SOURCE CODE:

ADD.HTML:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93
hXpG5KkN" crossorigin="anonymous"></script>

<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js"
integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvf
a0b4Q" crossorigin="anonymous"></script>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PV
CmYl" crossorigin="anonymous"></script>

    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
```

```
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/d
AiS6JXm" crossorigin="anonymous">

 <title>Add Daily Expense</title>

    <style type="text/css"

  Body

 background-color:rgb(235,193,201);

        form{

            margin-left: 550px;

            width: 30%;

        }

    </style>

</head>

<body>

     <nav class="h1 navbar navbar-dark bg-secondary p-3 mb-2 bg-primary text-
white justify-content-between">

      <a class="navbar-brand">Personal Expense Tracker Application</a>

  </nav>

    <br>

    <blockquote class="blockquote text-center">

      <p class="mb-0">We care your share (Add your daily expense).</p>
```

```
    <footer class="blockquote-footer">Save money for secure future<cite title="Job__Finder"></cite></footer>
  </blockquote>
  <br>
<div class="container">
  <div class="row">
    <div class="col-md-6">
      <h3>Add Expense</h3>
      <form action="/addexpense" method="POST">
        <div class="form-group">
          <label for="">Date</label>
          <input class="form-control" type="datetime-local" name="date" id="date">
        </div>
        <div class="form-group"> <label for="">Expense name</label>
          <input class="form-control" type="text" name="expensename" id="expensename">
        </div>
        <div class="form-group">
          <label for="">Expense Amount</label>
          <input class="form-control" type="number" min="0" name="amount" id="amount">
```

```
            </div>
            <div class="form-group">
               <label for=""></label>
               <select class="form-control" name="paymode" id="paymode">
                  <option  selected hidden>Pay-Mode</option>
                  <option name="cash" value="cash">cash</option>
                  <option name="debitcard" value="debitcard">debitcard</option>
                  <option name="creditcard" value="creditcard">creditcard</option>
                  <option name="epayment" value="epayment">UPI</option>
                  <option name="onlinebanking" value="onlinebanking">onlinebanking</option>
                   </select>
             <div class="form-group">
               <label for=""></label>
               <select class="form-control" name="category" id="category">
                   <option selected hidden>Category</option>
                  <option   name = "food" value="food">food</option>
                  <option name = "entertainment" value="entertainment">Entertainment</option>
                  <option name = "business" value="business">Business</option>
```

```
                    <option name ="rent" value="rent">Rent</option>
                    <option name = "EMI" value="EMI">EMI</option>
                    <option name = "other" value="other">other</option>

                </select>
            </div>
            <input class="btn btn-info" type="submit" value="Add" id="">
            </div>
        </form>
    </div>
  </div>
</div>
</body>
</html>
```

DASHBOARD.HTML:

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/html" xmlns="http://www.w3.org/1999/html"
    xmlns="http://www.w3.org/1999/html" xmlns="http://www.w3.org/1999/html" xmlns="http://www.w3.org/1999/html"
    xmlns="http://www.w3.org/1999/html">
```

```
<head>
    <meta charset="UTF-8">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
    <title>Login page</title>
  <style type="text/css">
        body{
            background-color: rgb(235,193,201);
```

```
        }
        .msg,.login{
            text-align: center;
            margin-top: 25px;
            margin-bottom:25px;
        }
        form{
            margin-left: 550px;
            width: 30%;
        }
    </style>
</head>
<body>
  <nav class="h1 navbar navbar-dark bg-secondary p-3 mb-2 bg-primary text-white justify-content-between">
      <a class="navbar-brand">Personal Expense Tracker Application</a>
  </nav>
  </br>
  </br>
<div>
  <h1>
```

```
    Personal Details

     <! display detail >

  </h1>

</div>

  </br>

  </br>

<div >

  <center><button type="submit" formaction="/add" class="btn btn-info">Add Expense</button></center>

</div>

</br>

</br>

<div>

    <! display expenses>

</div>

</body>

</html>
```

LOGIN.HTML:

```
<!DOCTYPE html>

<html lang="en">

<head>
```

```
    <meta charset="UTF-8">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
    <title>Login page</title>
  <style type="text/css">
        body{
            background-color: rgb(235,193,201);
        }
```

```
        .msg,.login{
            text-align: center;
            margin-top: 25px;
            margin-bottom:25px;
        }
        form{
            margin-left: 550px;
            width: 30%;
        }
    </style>
</head>
<body>
  <nav class="h1 navbar navbar-dark bg-secondary p-3 mb-2 bg-primary text-white justify-content-between">
      <a class="navbar-brand">Personal Expense Tracker Application</a>
  </nav>
  <form action="/login" method="POST">
        <h5 class="msg" style="color:rgb(65, 0, 102);">Log In</h5>
        <div class="form-group">
            <label for="exampleInputEmail">Email</label>
```

```
        <input type="text" name="email" class="form-control" id="exampleInputEmail" aria-describedby="emailHelp" placeholder="Enter Username" required>
    </div>
    <div class="form-group">
        <label for="exampleInputPassword">Password</label>
        <input type="password" name="password" class="form-control" id="exampleInputPassword" placeholder="Password" required>
    </div>
    <div>
        New User ? Click to <a href="{{url_for('register')}}">Register</a>
    </div>
    <br>
    <button type="submit" class="btn btn-info">Submit</button>
  </form>
</body>
</html>
```

REGISTRATION.HTML:

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

```
    <meta charset="UTF-8">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
    <title>Register page</title>
  <style type="text/css">
        body{
            background-color: rgb(235,193,201);
        }
```

```
        .msg,.login{
            text-align: center;
            margin-top: 25px;
            margin-bottom:25px;
        }
        form{
            margin-left: 550px;
            width: 30%
        }
    </style>
</head>
<body>
  <nav class="h1 navbar navbar-dark bg-secondary p-3 mb-2 bg-primary text-white justify-content-between">
      <a class="navbar-brand">Personal Expense Tracker Application</a>
  </nav>
  <form action="/register" method="POST">
        <h5 class="msg" style="color:rgb(65, 0, 102);">Sign Up</h5>
        <div class="form-group">
            <label for="exampleInputUsername">User Name</label>
```

```
        <input type="text" name="username" class="form-control" id="exampleInputUsername" aria-describedby="emailHelp" placeholder="Enter Username" required>
    </div>
    <div class="form-group">
        <label for="exampleInputEmail">Email</label>
        <input type="text" name="email" class="form-control" id="exampleInputEmail" aria-describedby="emailHelp" placeholder="Enter Username" required>
    </div>
    <div class="form-group">
        <label for="exampleInputPassword">Password</label>
        <input type="password" name="password" class="form-control" id="exampleInputPassword" placeholder="Password" required>
    </div>
    <div class="form-group">
        <label for="exampleInputAge">Age</label>
        <input type="age" name="age" class="form-control" id="exampleInputage" placeholder="Age" required>
    </div>
    <div class="form-group">
        <label for="exampleInputAge">Profession</label>
```

```
        <input type="profession" name="profession" class="form-control" id="exampleInputprofession" placeholder="Profession" required>
      </div>
      <div>
        Already a User ? Click to <a href="{{url_for('login')}}">Login</a>
      </div>
      <br>
      <button type="submit" class="btn btn-info">Submit</button>
  </form>
</body>
</html>
```

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)
```

DOCKERFILE.TXT:

```
FROM python:3.6
WORKDIR /app
ADD . /app
COPY requirements.txt /app
RUN python3 -m pip install -r requirements.txt
RUN python3 -m pip install ibm_db
```

```
EXPOSE 5000

CMD ["python","app.py"]

# FROM python:3.10-alpine

# WORKDIR /app

# ADD . /app

# RUN set -e; \

#         apk add --no-cache --virtual .build-deps \

#                 gcc \

#                 libc-dev \

#                 linux-headers \

#                 mariadb-dev \

#                 python3-dev \

#         ;

# COPY requirements.txt /app

# RUN pip3 install -r requirements.txt

# CMD ["python3","app.py"]
```

SENDEMAIL.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)
```

GITHUB & PROJECT DEMO LINK:

https://github.com/IBM-EPBL/IBM-Project-6580-1658832441