PERSONAL EXPENSE TRACKER APPLICATION

IBM PROJECT REPORT

Submitted by

SRINIVASAN M A	**-**	**813819205057**
NARAESH ARCHUN K	**-**	**813819205039**
RAJAMURUGAN M	**-**	**813819205048**
SABARISHAN M	**-**	**813819205051**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

SARANATHAN COLLEGE OF ENGINEERING, TIRUCHIRAPALLI

ANNA UNIVERSITY, CHENNAI 600 025

ACKNOWLEDGEMENT

We sincerely thank **Shri. S. Ravindran, Secretary, Saranathan College of Engineering,** for giving us a platform to achieve our project.

We express our sincere thanks to **Dr. D. Valavan, Principal, Saranathan College of Engineering,** for giving us an opportunity and immense support for the successful completion of the project.

We are obliged to **Dr. R. Thillaikarasi, Professor & Head of the Department, Information Technology, Saranathan College of Engineering,** for her valuable suggestion and encouragement to our project. We express our heartfelt thanks to our project coordinator **Ms. J. Sangeethapriya, M.Tech., Assistant Professor, Department of Information Technology** for providing us with their valuable ideas.

We would like to thank all our department faculty members and technical assistant for their support and help rendered by them in completion of this project.

We would like to thank our parents for their constant encouragement and support without which this project would not be possible. Last but not the least we would like to thank our friends who have been instrumental in providing idea and material for the construction of our project.

Above all, we thank the God almighty for his bountiful blessing.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Project Overview

The art of money management is about turning money into riches by changing perspectives; instead of thinking of money as an expense, think of it as an investment instrument. A well-defined money management strategy incorporates wealth accumulation, protection, and preservation. These fundamental financial concepts relate to individual needs, goals, financial targets, priorities, and risk factors.

This paper describes a cloud-based expense-tracking application. This application makes it simple for users to keep track of their expenses, and the user can set a limit. The application sends an email notification when the limit is reached. Python, Flask, and Docker were used to develop this application. The information about the user is stored in the database via the IBM cloud. SendGrid, a cloud-based SMTP provider, is used to send email alerts to users.

1.2 Purpose

Users can track their daily spending and monthly income using the Personal Expense Tracker application, which also generates a monthly expense report. The user of this application can manage their expenses to achieve financial stability by keeping track of all their expenditures. The categorization of expenses by week, month, and year makes it easier to see where more money is being spent. To use the expense tracker, users must first register by entering their name, email address, username, and password, as well as a password confirmation.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing problem

TITLE	Expense Manager Application	Expense Tracker	Tracking Personal Finances
METHODOLOGY USED	User Registration and Creation, Adding Income and Expenses, Category Master, Management View- Date Wise and category wise, and Remainder.	User Registration and Creation, Adding Income and Expenses, Category Master, Management View- Date Wise and category wise, and Remainder.	Financial Touch, Paper systems, Digital systems, and Credit Scores
ADVANTAGES	This project has shown the emotional components of the decisions, the wide variety of tools developed and used to keep track and the ways people engage with the unknown and unpredictable parts of their financial existence	This project is for keeping the day-to-day expenditures and helps to keep record of people's money daily. It effectively keeps away from the manual figuring for trying not to ascertain the pay and cost each month.	People can include this application in their daily routine and they can be disciplined about their expenses, get better at saving, and utilise the money on other useful things.
DISADVANTAGES	It is not set out to fully characterize all of personal finance, and is looking at financial practices within a limited population.	No plan was made to reduce unwanted spending of money and some options to keep record were not added.	Merging of the application with credit/debit or any of the smart card was not implemented.

Table 2.1

2.2 References

1. Expense Tracker Application

1. Velmurugan A, Associate Professor, School of Computing, Sathyabama Institute of Science and Technology, Chennai.
2. Albert Mayan J, Associate Professor, School of Computing, Sathyabama Institute of Science and Technology, Chennai.
3. Niranjana P, U.G Student, Department of CSE, Sathyabama Institute of Science and Technology, Chennai
4. Richard Francis, U.G Student, Department of CSE, Sathyabama Institute of Science and Technology, Chennai

2. Expense Tracker

1. Atiya Kazi, Professor, Department of Information Technology, Finolex Academy of Management and Technology, Ratnagiri, Maharashtra, India.
2. Praphulla S. Kherade, Department of Information Technology, Finolex Academy of Management and Technology, Ratnagiri, Maharashtra, India.
3. Raj S. Vilankar, Department of Information Technology, Finolex Academy of Management and Technology, Ratnagiri, Maharashtra, India.
4. Parag M. Sawant, Department of Information Technology, Finolex Academy of Management and Technology, Ratnagiri, Maharashtra, India.

3. Tracking Personal Finances

1. Joseph ‘Jofish’ Kaye, Yahoo Labs, Sunnyvale, CA, USA, jofish@yahoo-inc.com
2. Mary McCuistion, Essential Anthropology, San Jose, CA, USA, marymccuistion@att.ne
3. Rebecca Gulotta, HCII, CMU, Pittsburgh, PA, USA, rgulotta@cs.cmu.edu

4. David A. Shamma, Yahoo Labs, San Francisco, CA, shamma@yahoo-inc.com

2.3 Problem Statement Definition

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

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

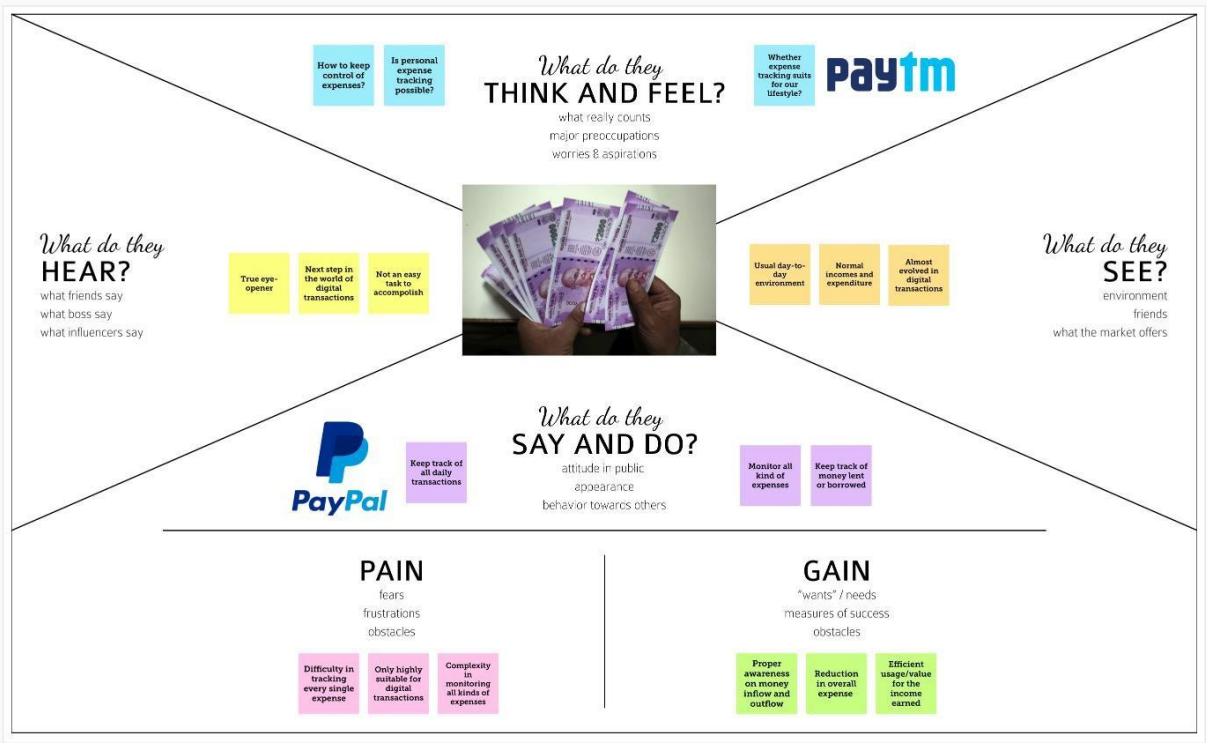


Fig 3.1

3.2 Ideation & Brainstorming

Ideation refers to the whole creative process of coming up with and communicating new ideas. It can take many different forms, from coming up with a totally new idea to combining multiple existing ideas to create a new process or organizational system. Ideation is similar to a practice known as brainstorming.

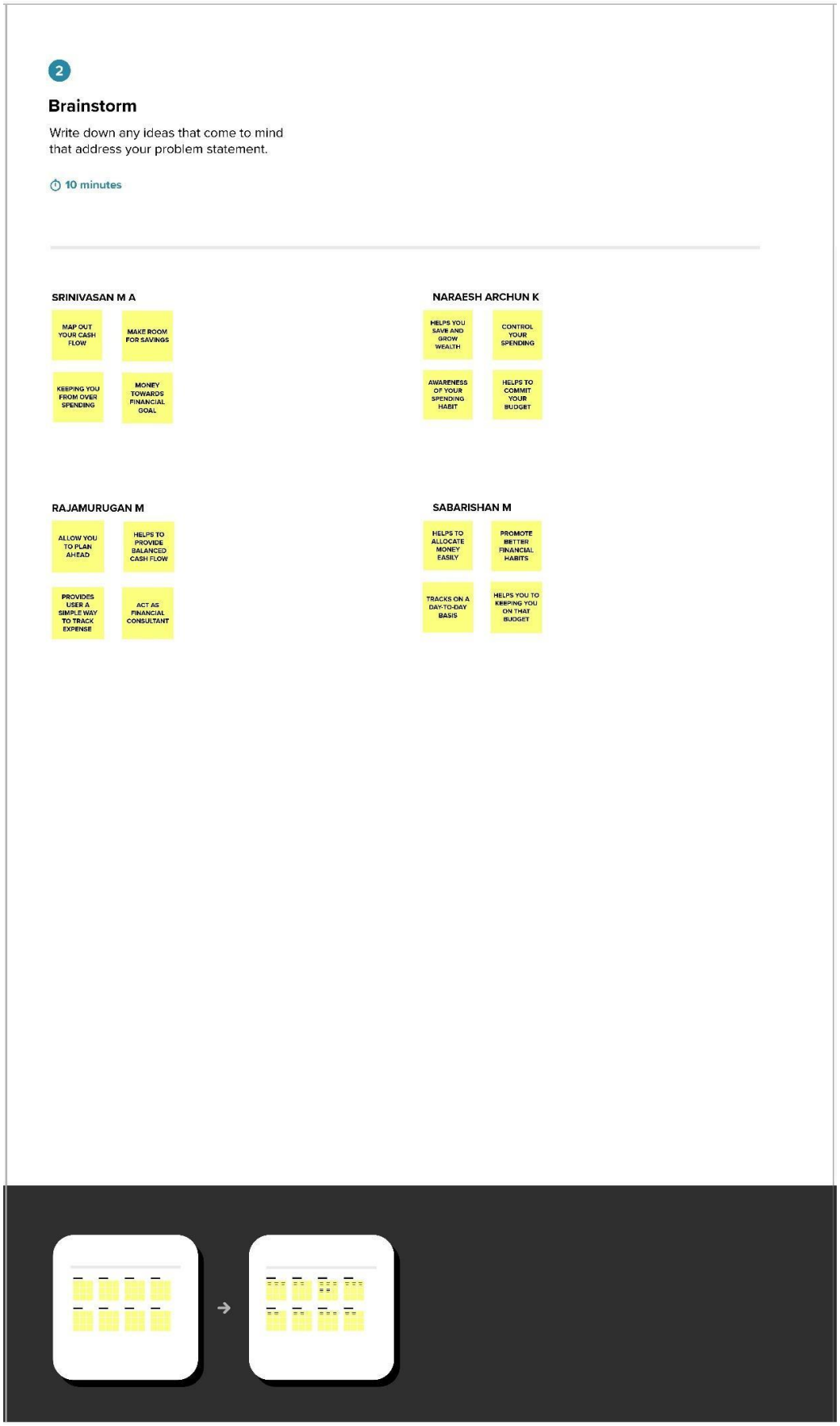

2
Brainstorm
Write down any ideas that come to mind that address your problem statement.
10 minutes
SRINIVASAN M A
MAP OUT YOUR CASH FLOW
MAKE ROOM FOR SAVINGS
KEEPING YOU FROM OVER SPENDING
MONEY TOWARDS FINANCIAL GOAL
NARAESH ARCHUN K
HELPS YOU SAVE AND GROW WEALTH
CONTROL YOUR SPENDING
AWARENESS OF YOUR SPENDING HABIT
HELPS TO COMMIT YOUR BUDGET
RAJAMURUGAN M
ALLOW YOU TO PLAN AHEAD
HELPS TO PROVIDE BALANCED CASH FLOW
PROVIDES USER A SIMPLE WAY TO TRACK EXPENSE
ACT AS FINANCIAL CONSULTANT
SABARISHAN M
HELPS TO ALLOCATE MONEY EASILY
PROMOTE BETTER FINANCIAL HABITS
TRACKS ON A DAY-TO-DAY BASIS
HELPS YOU TO KEEPING YOU ON THAT BUDGET

Fig 3.2

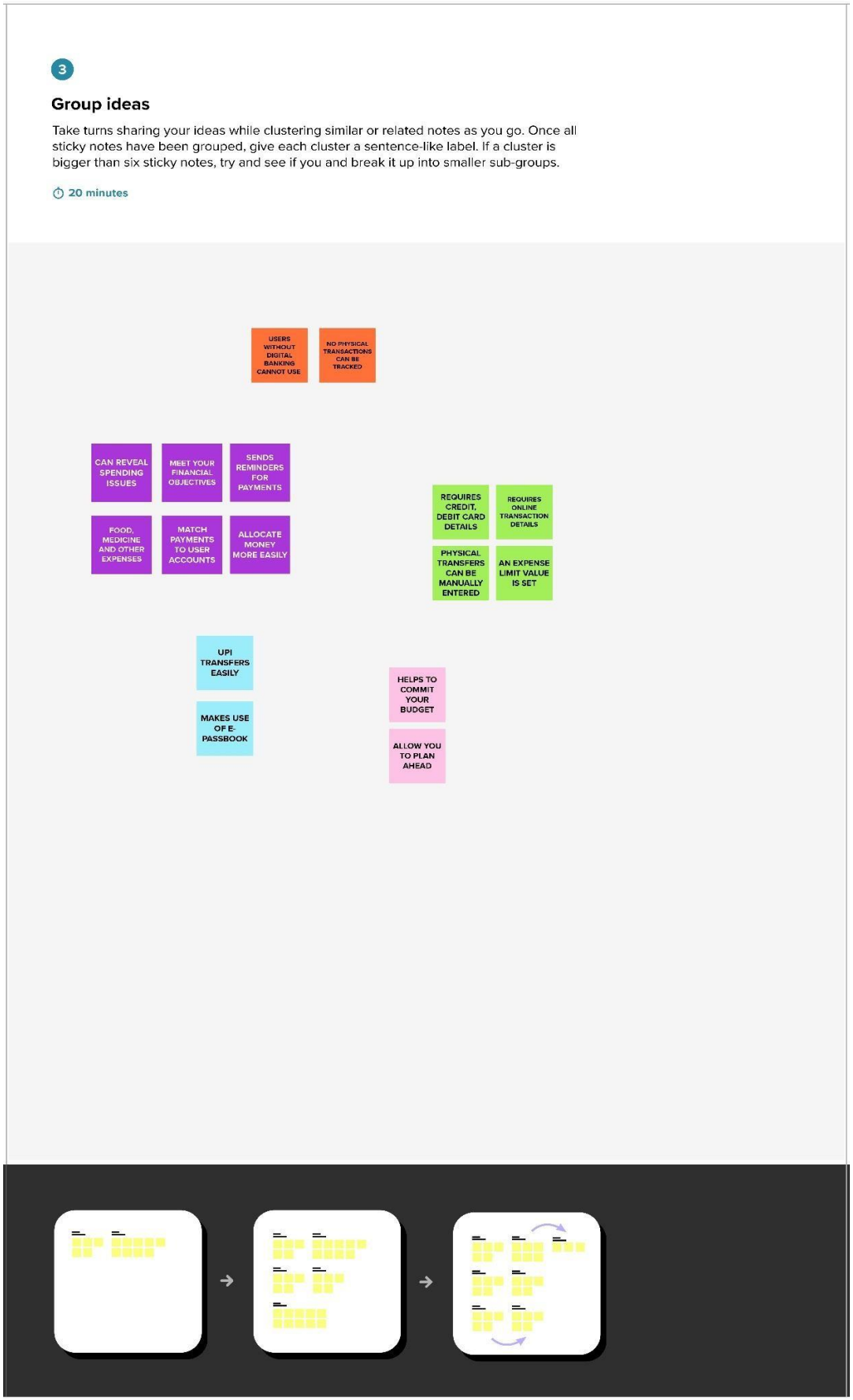

3
Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.
20 minutes
USERS WITHOUT DIGITAL BANKING CANNOT USE
NO PHYSICAL TRANSACTIONS CAN BE TRACKED
CAN REVEAL SPENDING ISSUES
MEET YOUR FINANCIAL OBJECTIVES
SENDS REMINDERS FOR PAYMENTS
FOOD, MEDICINE AND OTHER EXPENSES
MATCH PAYMENTS TO USER ACCOUNTS
ALLOCATE MONEY MORE EASILY
REQUIRES CREDIT, DEBIT CARD DETAILS
REQUIRES ONLINE TRANSACTION DETAILS
PHYSICAL TRANSFERS CAN BE MANUALLY ENTERED
AN EXPENSE LIMIT VALUE IS SET
UPI TRANSFERS EASILY
MAKES USE OF E-PASSBOOK
HELPS TO COMMIT YOUR BUDGET
ALLOW YOU TO PLAN AHEAD

Fig 3.3


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
USERS WITHOUT DIGITAL BANKING CANNOT USE
REQUIRES CREDIT, DEBIT CARD DETAILS
SENDS REMINDERS FOR PAYMENTS
AN EXPENSE LIMIT VALUE IS SET
CAN REVEAL SPENDING ISSUES
ALLOCATE MONEY MORE EASILY
MAKES USE OF E-PASSBOOK
REQUIRES ONLINE TRANSACTION DETAILS
HELPS TO COMMIT YOUR BUDGET
PHYSICAL TRANSFERS CAN BE MANUALLY ENTERED
NO PHYSICAL TRANSACTIONS CAN BE TRACKED
MATCH PAYMENTS TO USER ACCOUNTS
ALLOW YOU TO PLAN AHEAD
MEET YOUR FINANCIAL OBJECTIVES
UPI TRANSFERS EASILY
FOOD, MEDICINE AND OTHER EXPENSES
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

Fig 3.4

3.3 Proposed Solution

Proposed solution should relate the current situation to a desired result and describe the benefits that will accrue when the desired result is achieved. So, begin your proposed solution by briefly describing this desired result.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Personal finance applications will ask users to add their expenses and based on their expenses, wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert.
2.	Idea / Solution description	The solution to this problem is, the people who gets regular payments can able to track their payments and avoid unwanted expenses.
3.	Novelty / Uniqueness	This application tracks your every expenses anywhere and anytime without using the paper work. Just click and enter your expenditure. to avoid data loss, quick settlements and reduce human error. To alert the user through notification message about the expenditure of the user weekly.
4.	Social Impact / Customer Satisfaction	Using this application one can track their personal expenses and frame a monthly/annual budget. If your expense exceeded than specified limit, the application will show you an alert message.
5.	Business Model (Revenue Model)	Business people can use subscription/premium feature of this application to gain revenue.
6.	Scalability of the Solution	Unsubscribed users will be notified through alert message if the limit is exceeded. Subscribed users can have the additional facility to set the remainder for the upcoming payments to be paid on-time.

Table 3.1

3.4 Problem Solution fit

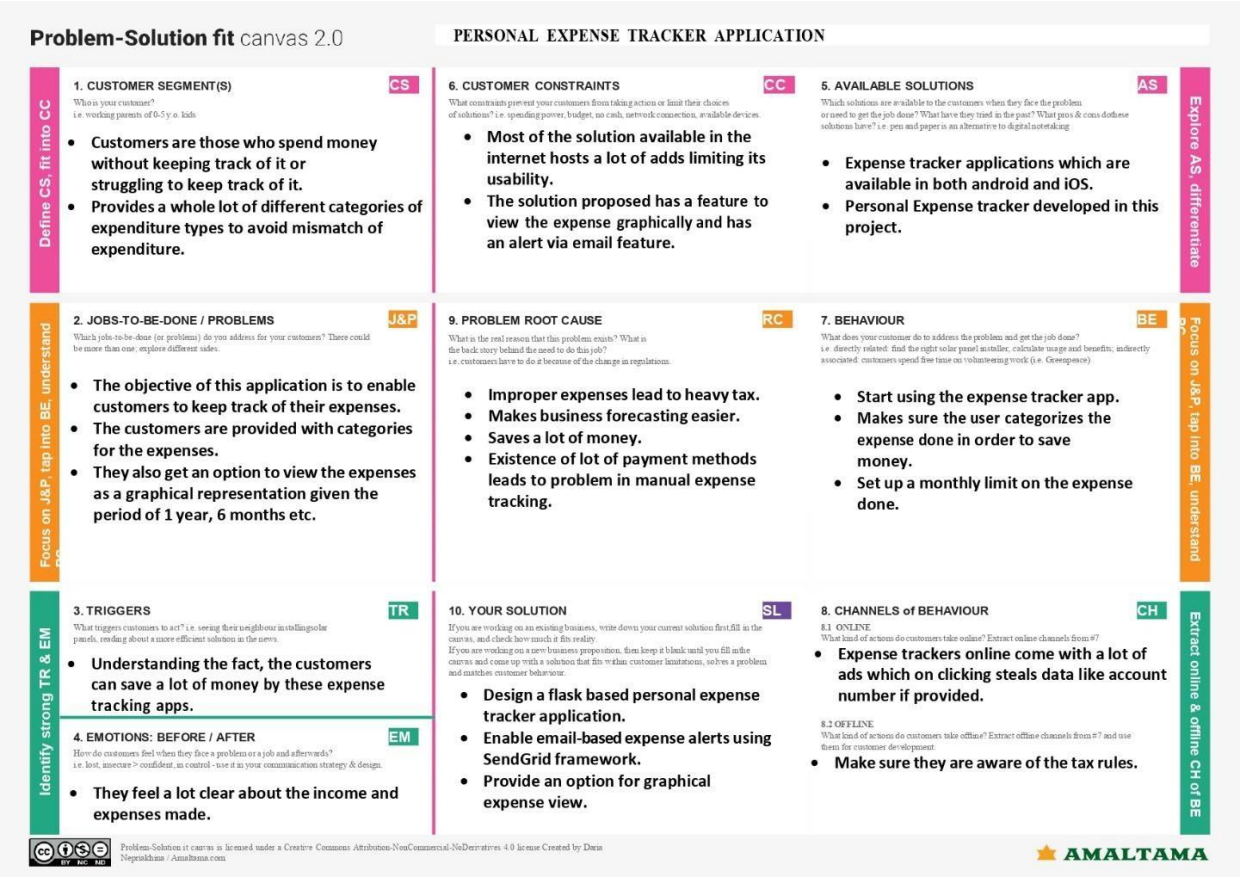


Fig 3.5

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional requirement

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. So, it's important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behavior under specific conditions. 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
FR-2	User Confirmation	Confirmation via Email
FR-3	Tracking Expense	Helpful insights about money management
FR-4	Alert Message	Give alert mail if the amount exceeds the budget limit

Table 4.1

4.2 Non-functional requirement

Non-functional requirements, not related to the system functionality, rather define how the system should perform. Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	You will able to allocate money to different priorities and also help you to cut down on unnecessary spending
NFR-2	Security	It employs the latest security and technology measures to keep customers personal and financial information safe
NFR-3	Reliability	Used to manage his/her expense so that the user is the path of financial stability. It is categorized by week, month, and year and also helps to see more expenses made. Helps to define their own categories.
NFR-4	Performance	Help to gain control of your finance, pay down debt, grow your net worth, help to upload receipts, track mileage
NFR-5	Availability	Able to track business expense and monitor important for maintaining healthy cash flow but also qualifying for deductions that could reduce your taxable income
NFR-6	Scalability	To know where money goes and you can ensure that money is used widely

Table 4.2

CHAPTER 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 can be manual, automated, or a combination of both.

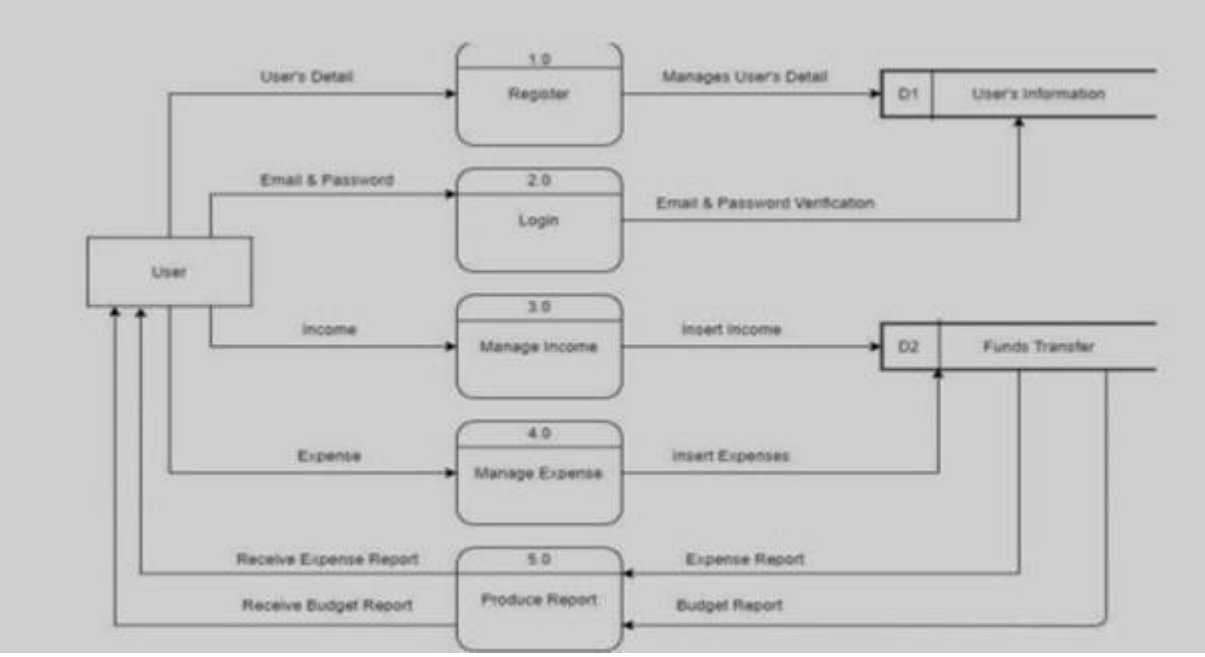


Fig 5.1

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:

1. Find the best tech solution to solve existing business problems.

2. Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.

3. Define features, development phases, and solution requirements.

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

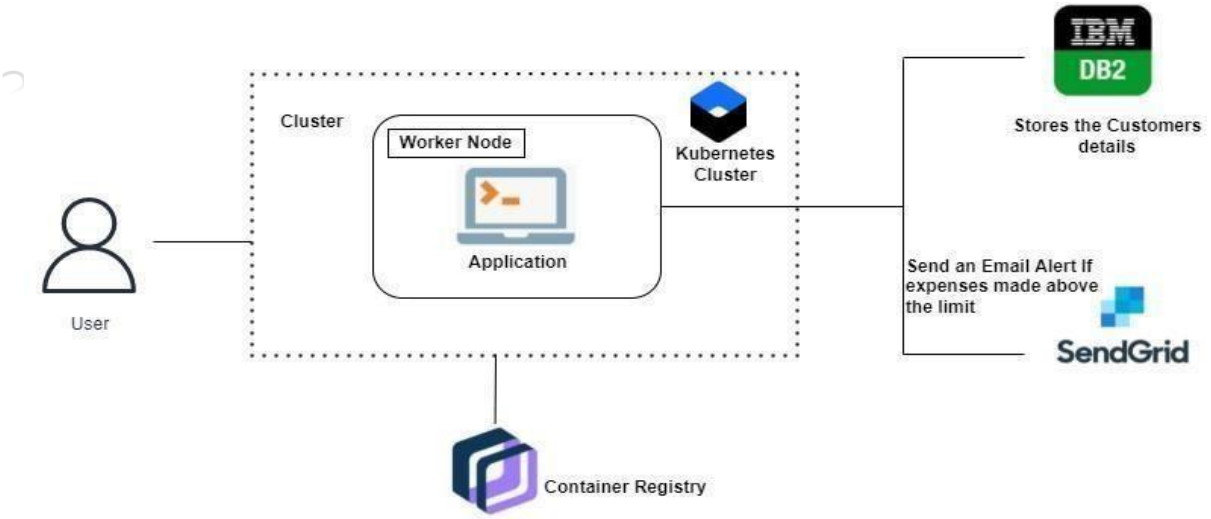


Fig 5.2

Components & Technologies:

S. No.	Component	Description	Technology
1.	User Interface	User interface interacts with host 5000	HTML, bootstrap
2.	Application Logic-1	Coding platform for developing application	Python
3.	Application Logic-2	Let you to build conversational interfaces into any application, device or channel	IBM Watson Assistant
4.	Application Building	It can be built for web application	Flask
5.	Cloud Database	Used to store and retrieve data	IBM DB2
6.	Infrastructure (Server / Cloud)	Helping to orchestrate different types of containers and deploying them to clusters	Kubernetes

Table 5.1

Application Characteristics

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	It provides libraries to build light weight application	Flask
2.	Security Implementations	Simulates human conversation or chatter through text or voice interactions.	Chatbot
3.	Scalable Architecture	It provides no isolate the internal code dependencies.	Python
4.	Availability	Runs everywhere and user friendly	Docker
5.	Performance	Orchestrate containerized application to run the cluster of hosts	Kubernetes

Table 5.2

5.3 User Stories

A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobileuser andweb user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirmingmy password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for theapplication through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for theapplication through form	I can register by entering the details	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access my dashboard	High	Sprint-1
	Dashboard	USN-6	As a user, I can log into the dashboard and manage income	I can add, delete and modify the income	High	Sprint-1
		USN-7	As a user, I can log into the dashboard and manage expense	I can add, delete and modify the expenses	High	Sprint-1
		USN-8	As a user, I can get a report isbased on the details	I can manage my money by viewing this report	Medium	Sprint-1
Administrator	Alert message	USN-9	As a user, I can get an email ifthe money level is above the limit	I can receive alert email	High	Sprint-1
	Database	USN-10	As a user, I can't able to see the database but the details are automatically stored on the database	Based on the details on the database, I can get the details of money monthly through email	High	Sprint-1

Table 5.3

CHAPTER 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	Srinivasan
		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Naraesh Archun
	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Rajamurugan
	Dashboard	USN-4	Logging in takes to the dashboard for the logged user.	2	High	Sabarishan
Sprint 2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Sabarishan
	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Naraesh Archun
	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Rajamurugan
		USN-4	Making dashboard interactive with JS	2	High	Srinivasan

Table 6.1

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 3		USN-1	Wrapping up the server side works of frontend	1	Medium	Naraesh Archun
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user's query	1	Medium	Srinivasan
	SendGrid	USN-3	Using SendGrid to send mail to the user about their expenses	1	Low	Sabarishan
		USN-4	Integrating both frontend and backend	2	High	Rajamurugan
Sprint 4	Docker	USN-1	Creating image of website using docker/	2	High	Rajamurugan
	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Srinivasan
	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Sabarishan
	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Naraesh Archun

Table 6.2

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	20	6 Days	23 Oct 2022	28 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	30 Oct 2022	04 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	06 Nov 2022	11 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	13 Nov 2022	18 Nov 2022	20	19 Nov 2022

Table 6.3

6.3 Reports from JIRA

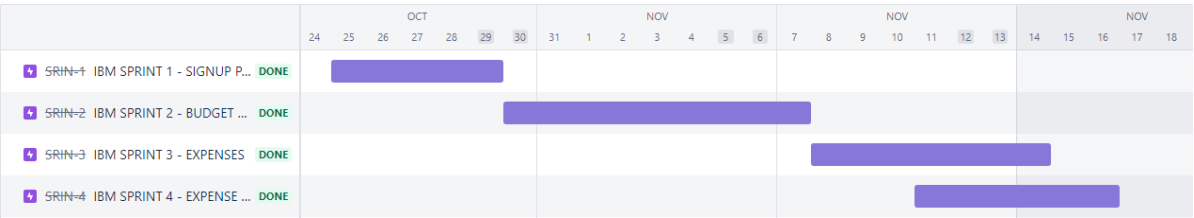


Fig 6.1

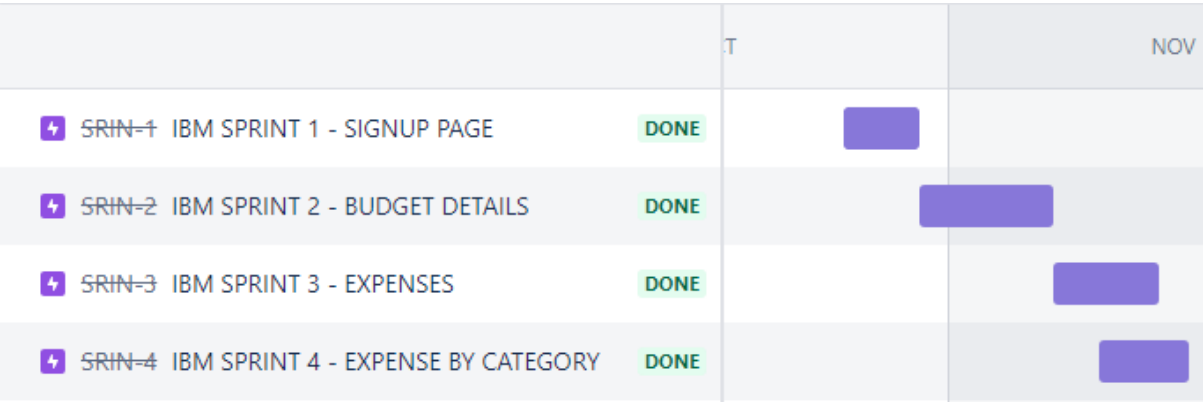
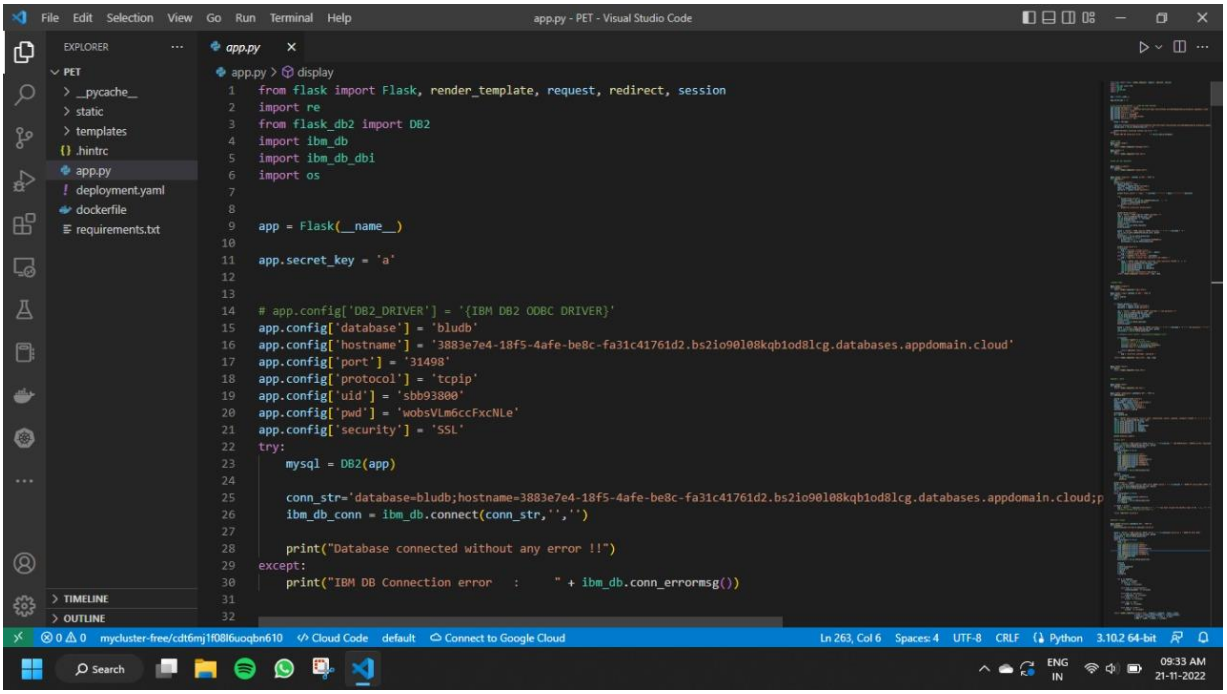


Fig 6.2

CHAPTER 7

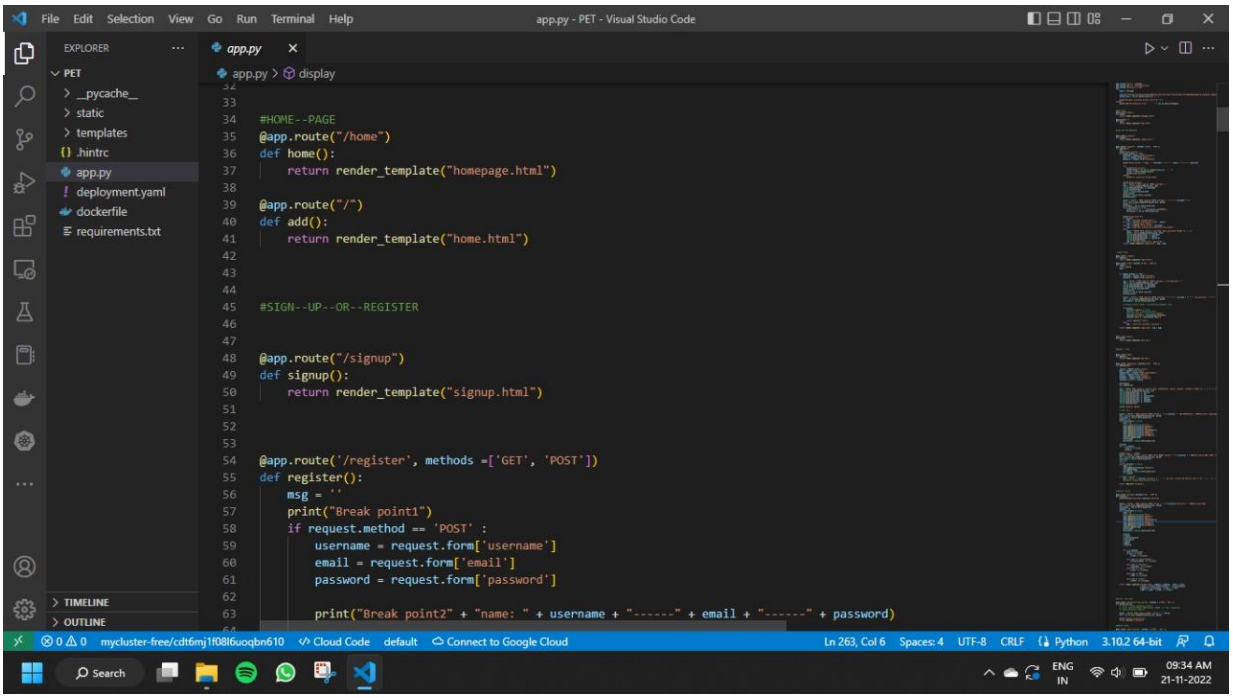
CODING & SOLUTIONING

7.1 Source Code



```python
from flask import Flask, render_template, request, redirect, session
import re
from flask_db2 import DB2
import ibm_db
import ibm_db_dbi
import os


app = Flask(__name__)

app.secret_key = 'a'


# 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;p
    ibm_db_conn = ibm_db.connect(conn_str,'','')

    print("Database connected without any error !!")
except:
    print("IBM DB Connection error   :     " + ibm_db.conn_errormsg())
```

Fig 7.1



```python
#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)
```

Fig 7.2

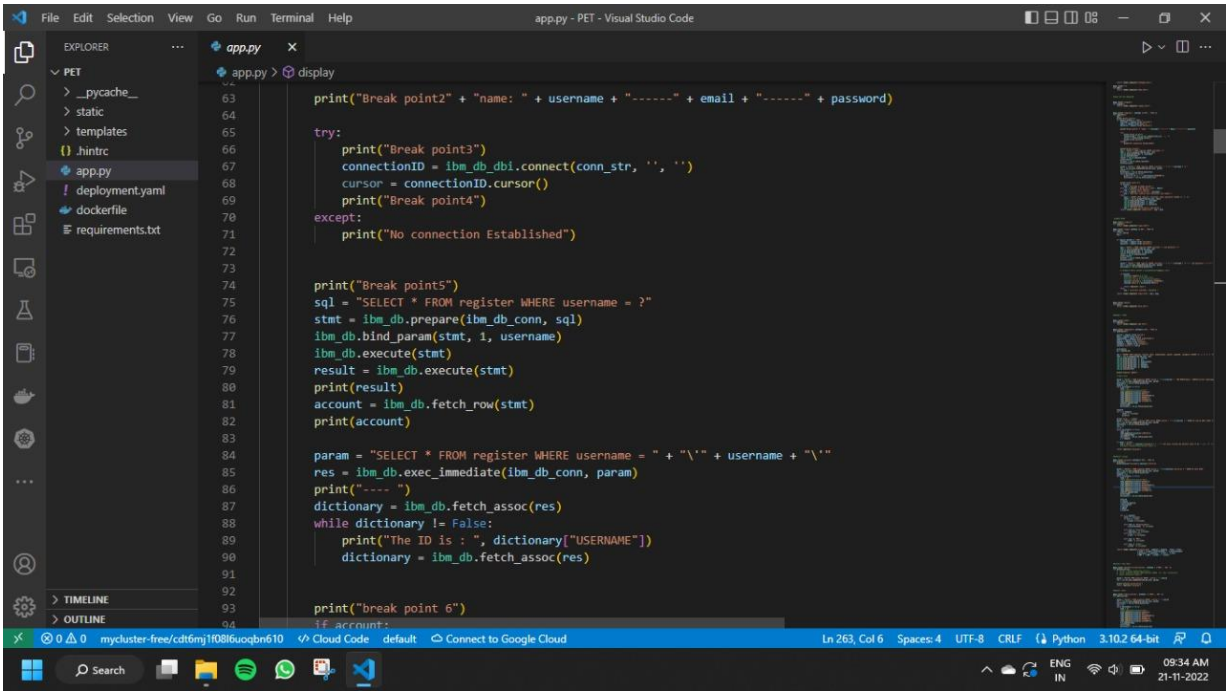

app.py - PET - Visual Studio Code
app.py > display
print("Break point2" + "name: " + username + "------" + email + "------" + password)
try:
print("Break point3")
connectionID = ibm_db_dbi.connect(conn_str, '', '')
cursor = connectionID.cursor()
print("Break point4")
except:
print("No connection Established")
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)
print("break point 6")

Fig 7.3

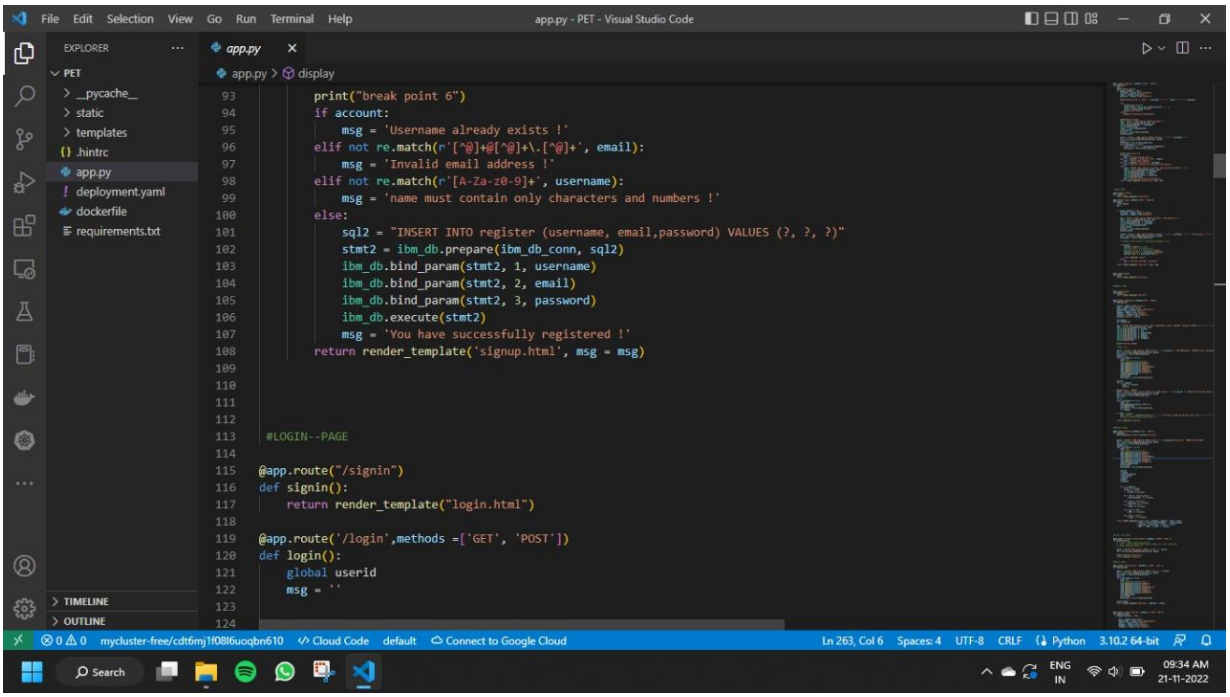

app.py - PET - Visual Studio Code
app.py > display
print("break point 6")
if account:
msg = 'Username already exists !'
elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
msg = 'Invalid email address !'
elif not re.match(r'[A-Za-z0-9]+', username):
msg = 'name must contain only characters and numbers !'
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)
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 = ''

Fig 7.4

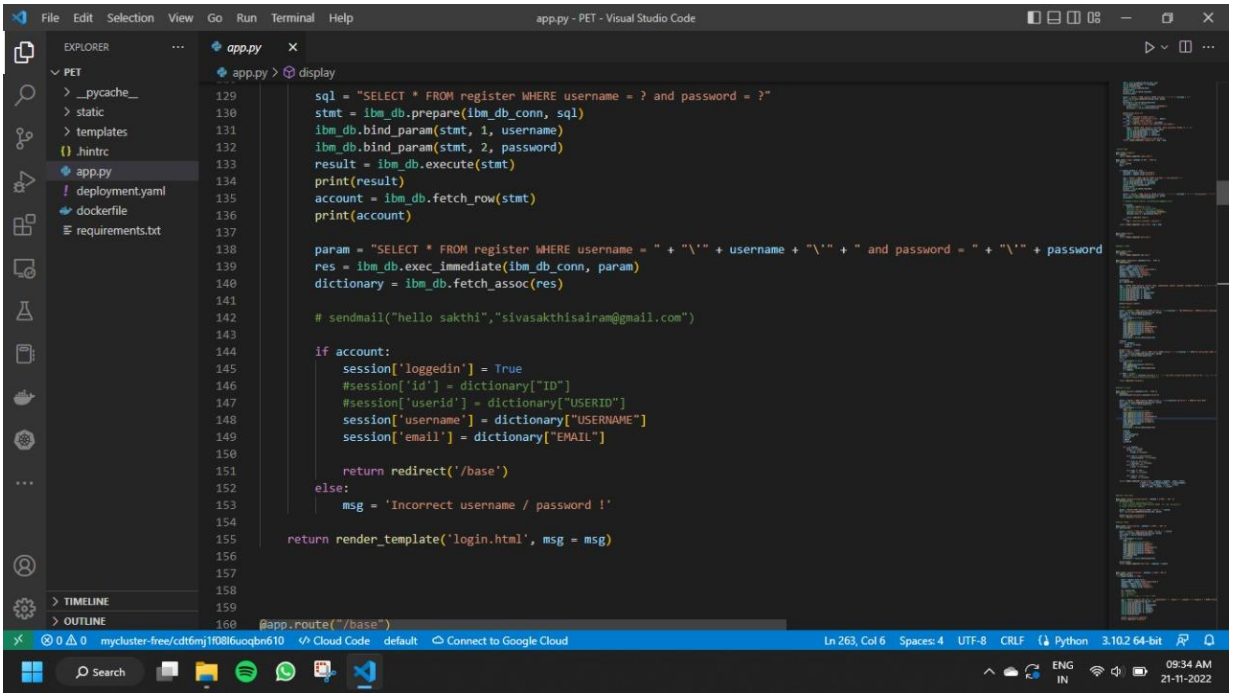

app.py - PET - Visual Studio Code
app.py > display
sql = "SELECT * FROM register WHERE username = ? and password = ?"
stmt = ibm_db.prepare(ibm_db_conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.bind_param(stmt, 2, password)
result = ibm_db.execute(stmt)
print(result)
account = ibm_db.fetch_row(stmt)
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"]
#session['userid'] = dictionary["USERID"]
session['username'] = dictionary["USERNAME"]
session['email'] = dictionary["EMAIL"]
return redirect('/base')
else:
msg = 'Incorrect username / password !'
return render_template('login.html', msg = msg)
@app.route("/base")

Fig 7.5

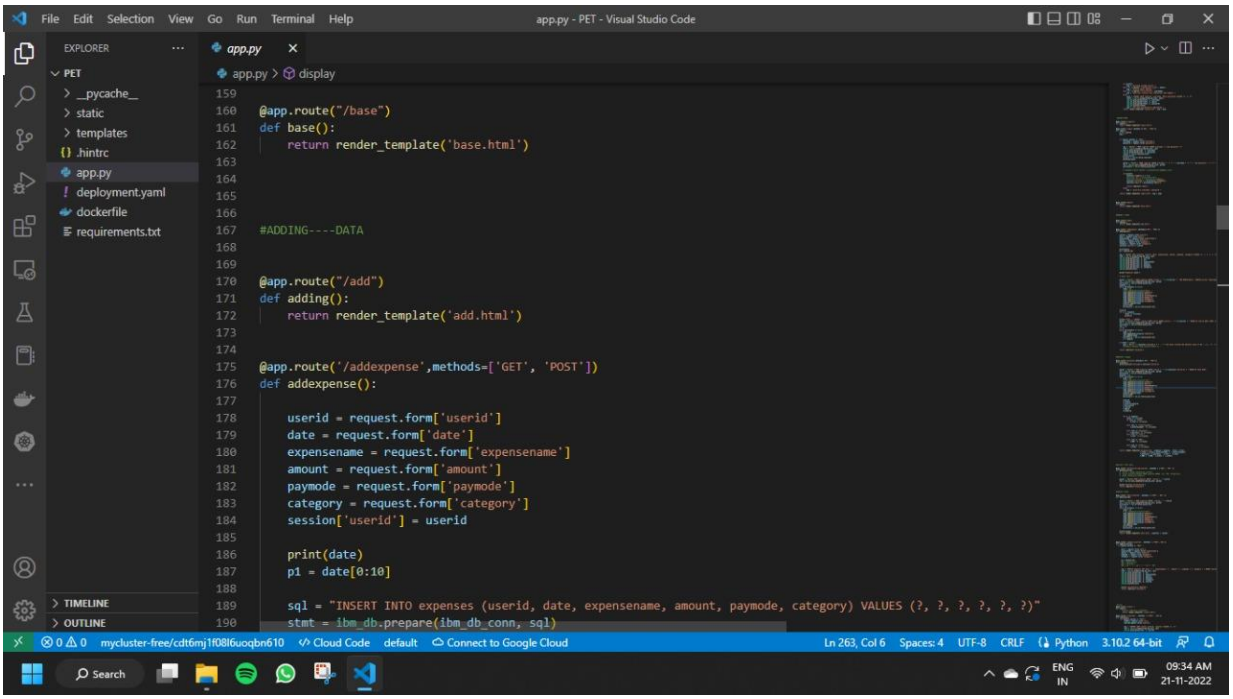

app.py - PET - Visual Studio Code
app.py > display
@app.route("/base")
def base():
return render_template('base.html')
#ADDING----DATA
@app.route("/add")
def adding():
return render_template('add.html')
@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():
userid = request.form['userid']
date = request.form['date']
expensename = request.form['expensename']
amount = request.form['amount']
paymode = request.form['paymode']
category = request.form['category']
session['userid'] = userid
print(date)
p1 = date[0:10]
sql = "INSERT INTO expenses (userid, date, expensename, amount, paymode, category) VALUES (?, ?, ?, ?, ?, ?)"
stmt = ibm_db.prepare(ibm_db_conn, sql)

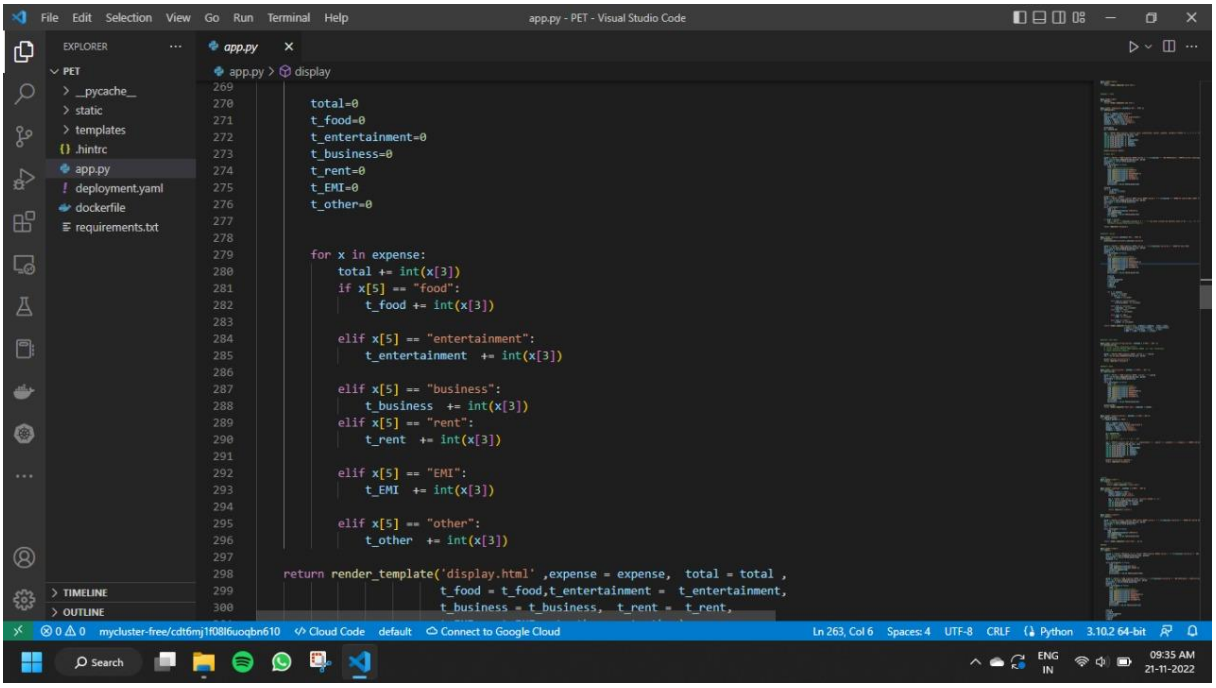
Fig 7.6

```
    print("Expenses added")

    # email part

    param = "SELECT * FROM expenses WHERE userid = " + str(userid) + " AND MONTH(date) = MONTH(current timestamp) AND YEAR(
    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 += int(x[3])
        print(x)

    print("total : ",total)
    param = "SELECT userid, limitss FROM limits WHERE userid = " + str(userid) + " ORDER BY userid DESC LIMIT 1"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
```

Fig 7.7

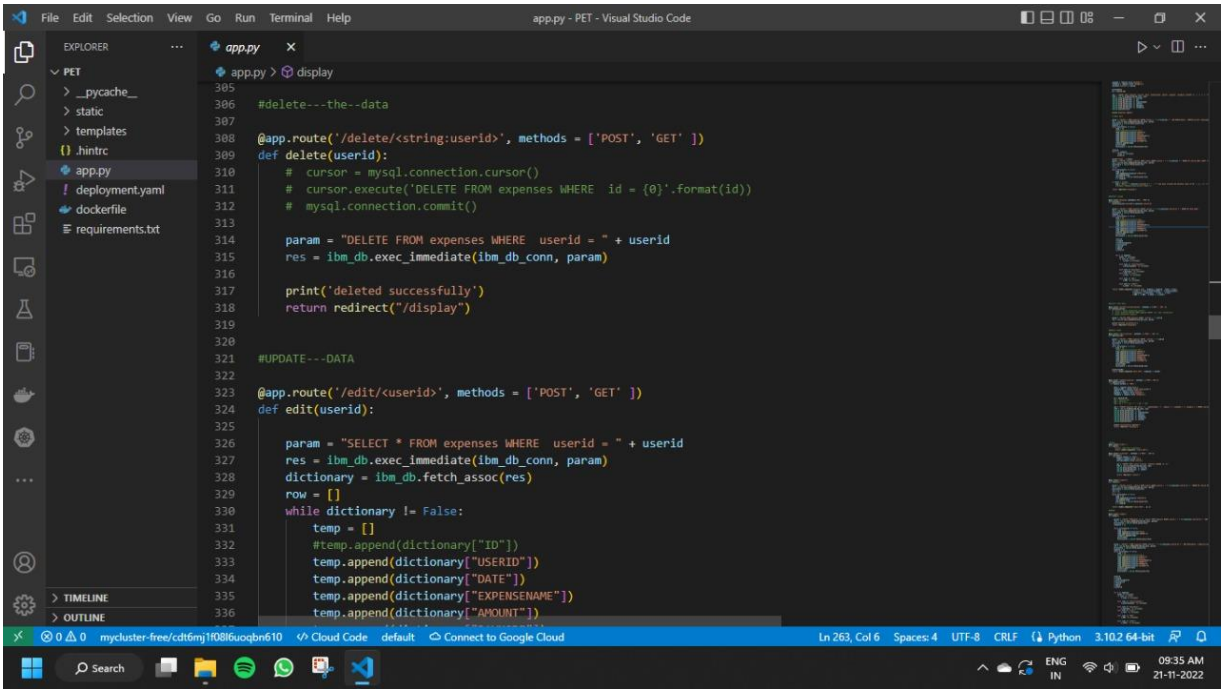
```
    s = 0
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMITSS"])
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[0]

    if total > int(s):
        msg = "Hello " + session['username'] + " , " + "you have crossed the monthly limit of Rs. ", s , "/- !!!" + "\n" +
        #sendmail.Sendmail(msg,session['email'])

    return redirect("/display")



#DISPLAY---graph

@app.route("/display",methods=['GET', 'POST'])
def display():
    print(session["username"],session['userid'])


    param = "SELECT * FROM expenses WHERE userid = " + str(session['userid']) + " 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"])
```

Fig 7.8



```
        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 += int(x[3])
            if x[5] == "food":
                t_food += int(x[3])

            elif x[5] == "entertainment":
                t_entertainment  += int(x[3])

            elif x[5] == "business":
                t_business  += int(x[3])
            elif x[5] == "rent":
                t_rent  += int(x[3])

            elif x[5] == "EMI":
                t_EMI  += int(x[3])

            elif x[5] == "other":
                t_other  += int(x[3])

    return render_template('display.html' ,expense = expense,  total = total ,
                                t_food = t_food,t_entertainment =  t_entertainment,
                                t_business = t_business,  t_rent =  t_rent,
```

Fig 7.9



```
#delete---the--data

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

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

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


#UPDATE---DATA

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

    param = "SELECT * FROM expenses WHERE  userid = " + userid
    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"])
```

Fig 7.10

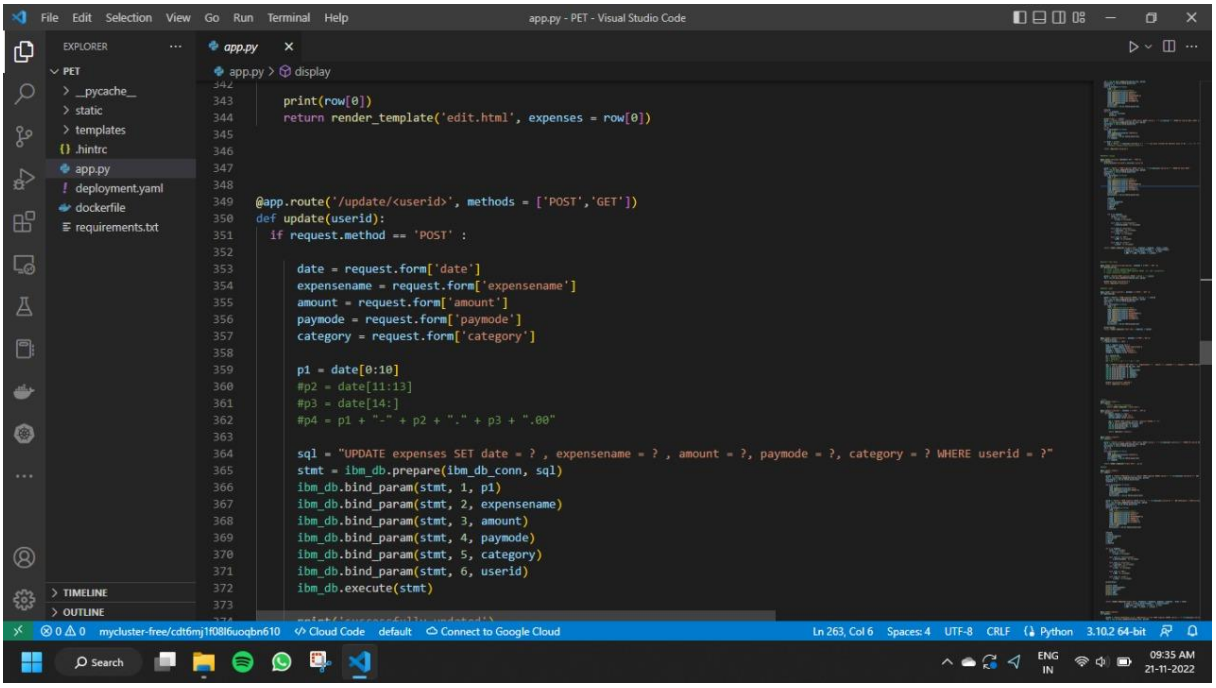


Fig 7.11

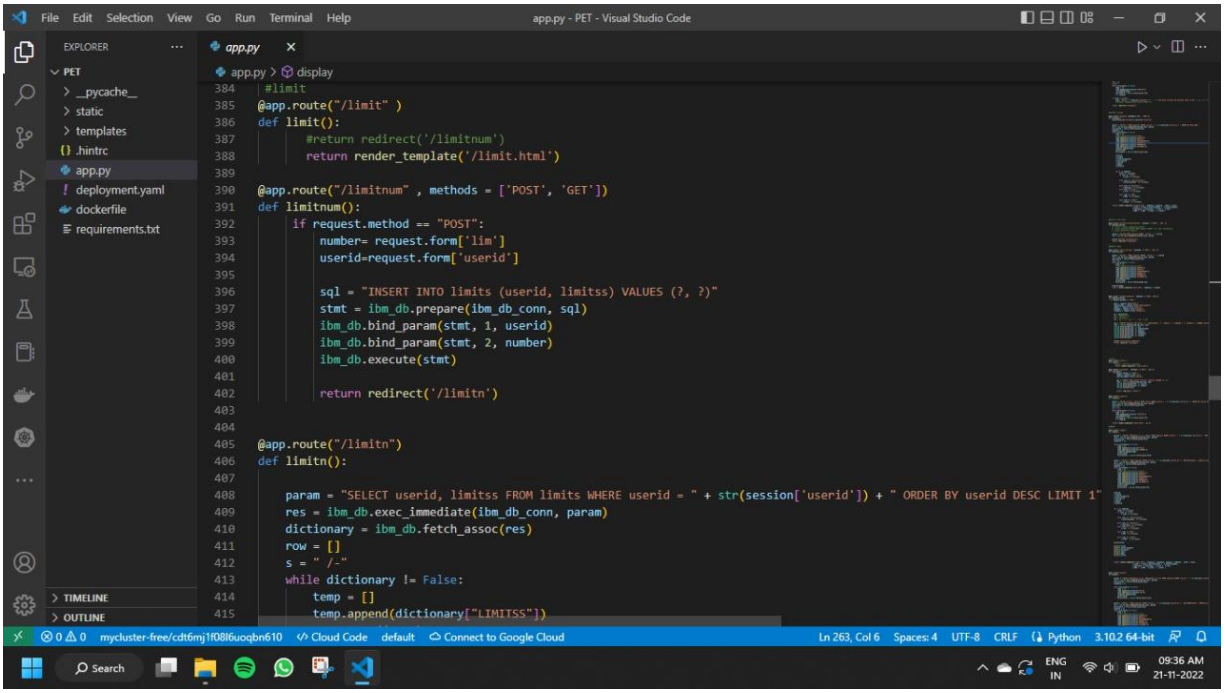


Fig 7.12

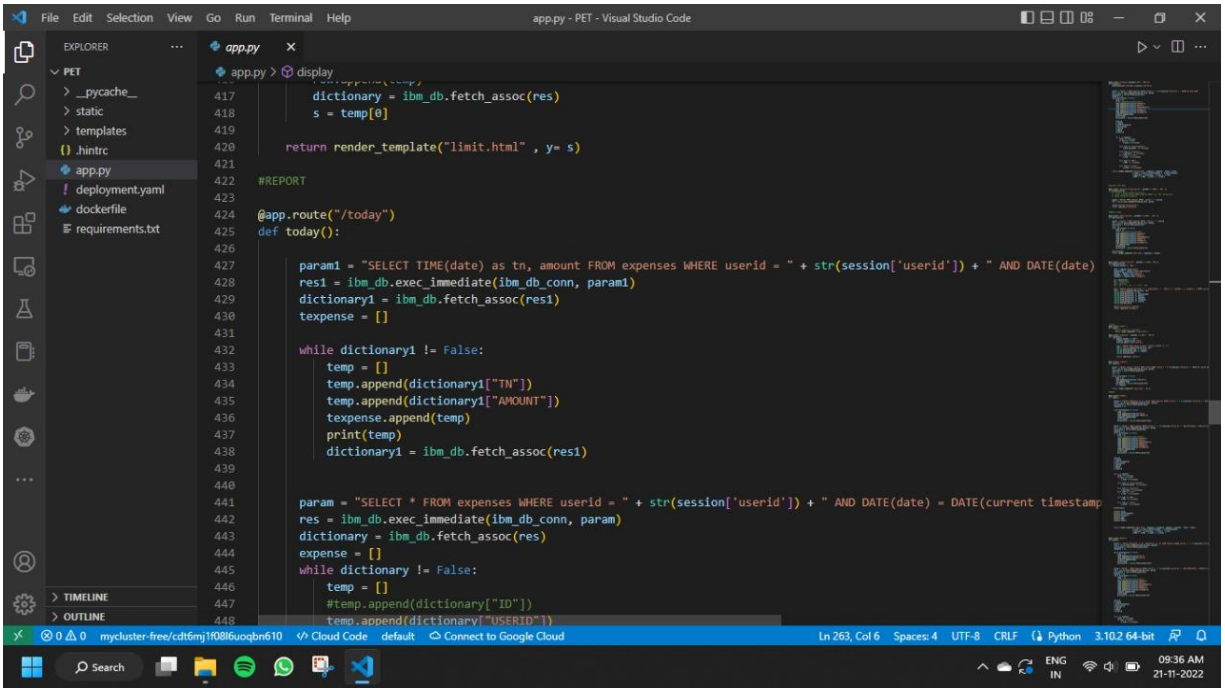


Fig 7.13

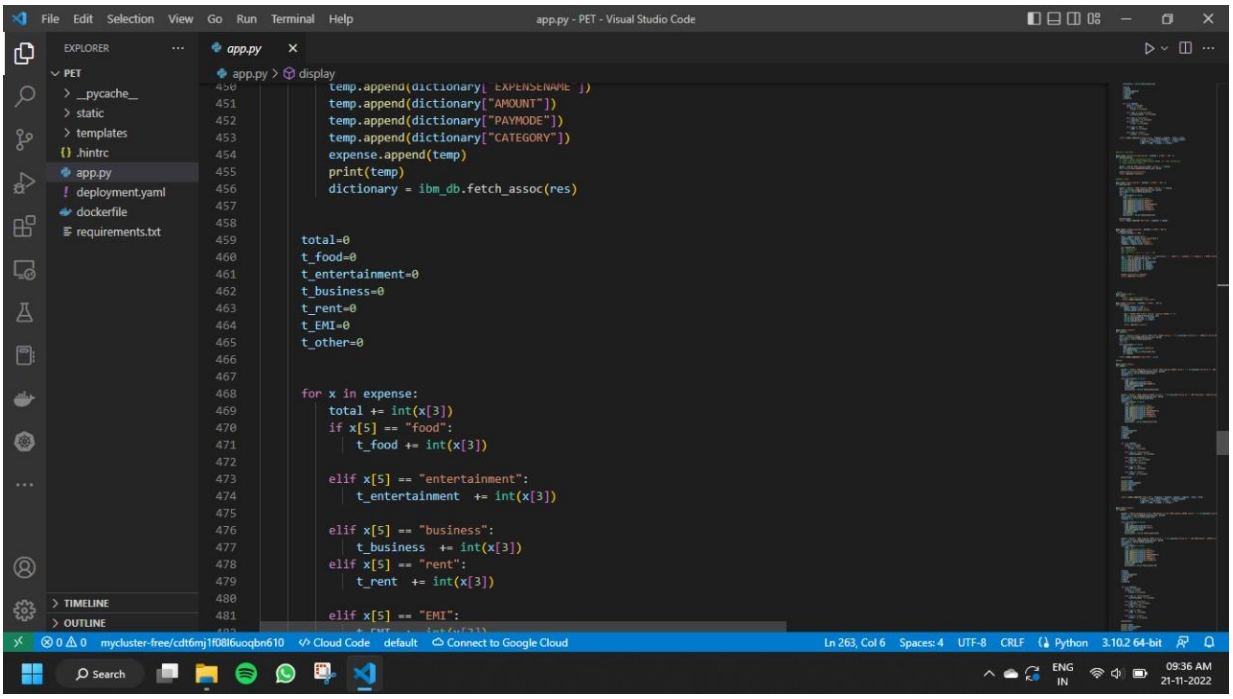


Fig 7.14

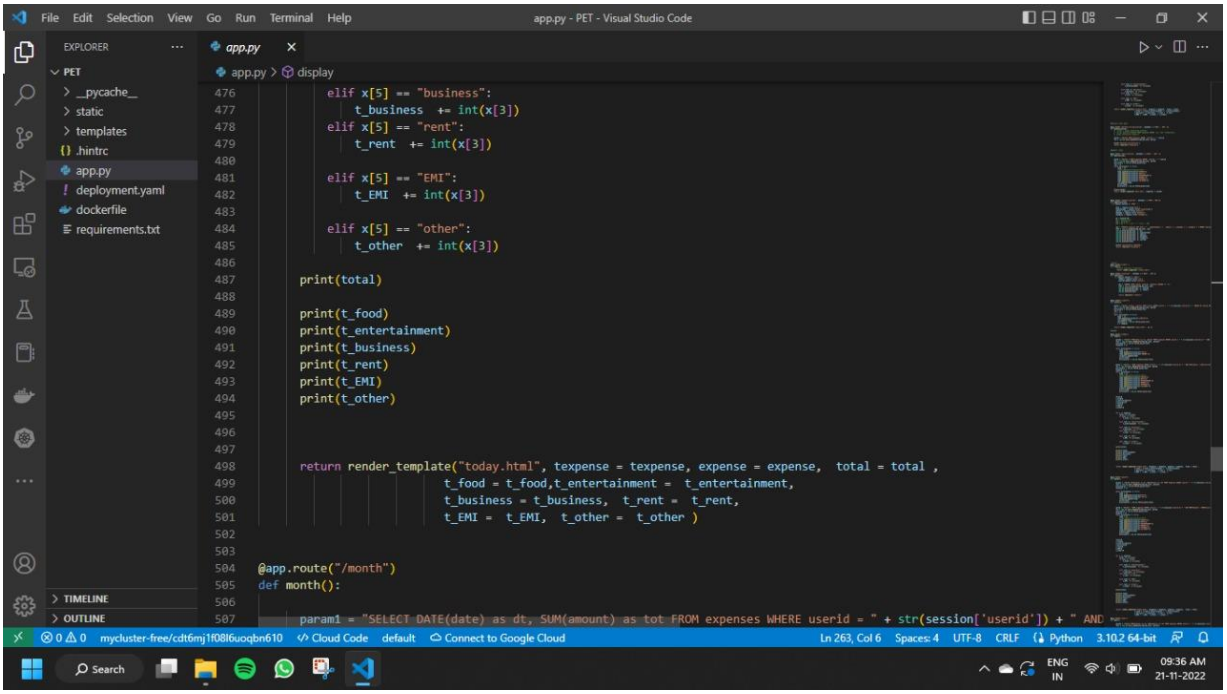


Fig 7.15

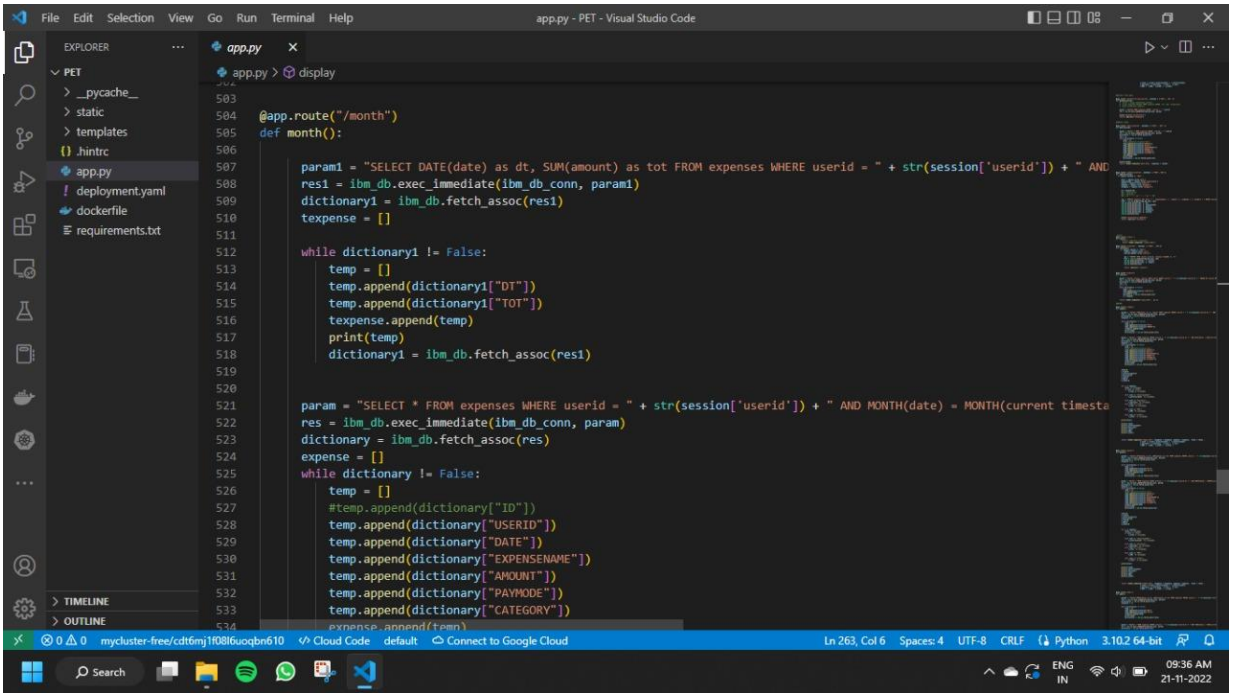


Fig 7.16

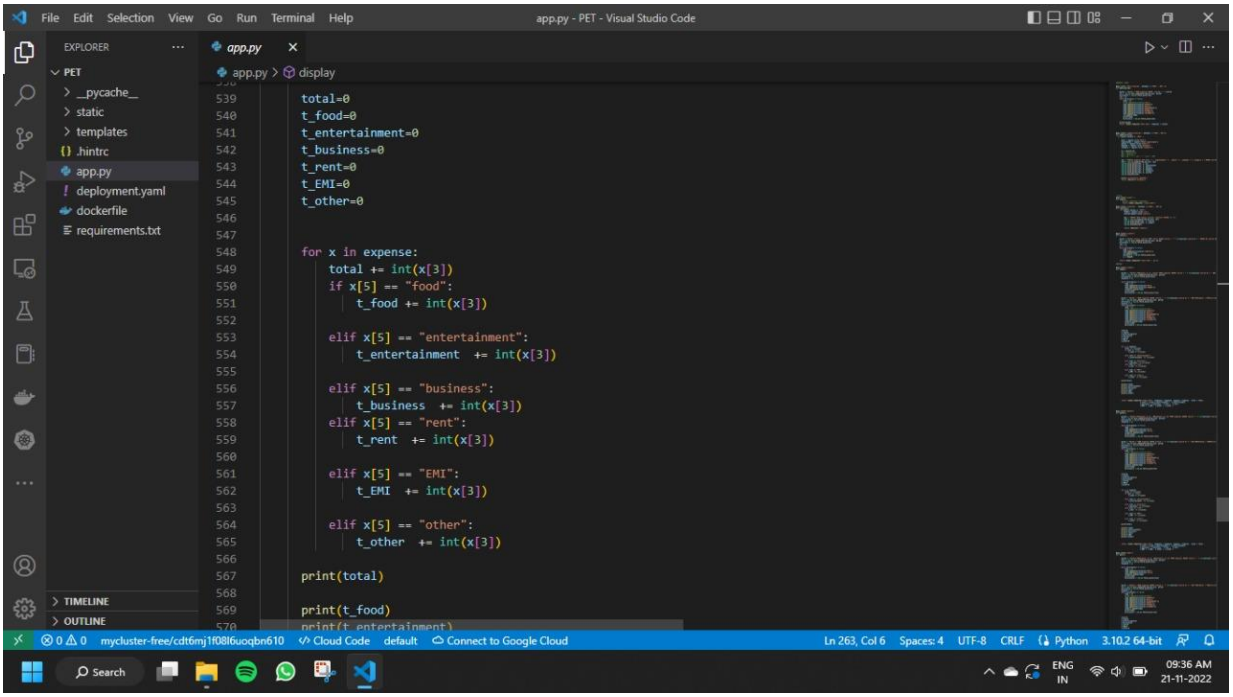


Fig 7.17

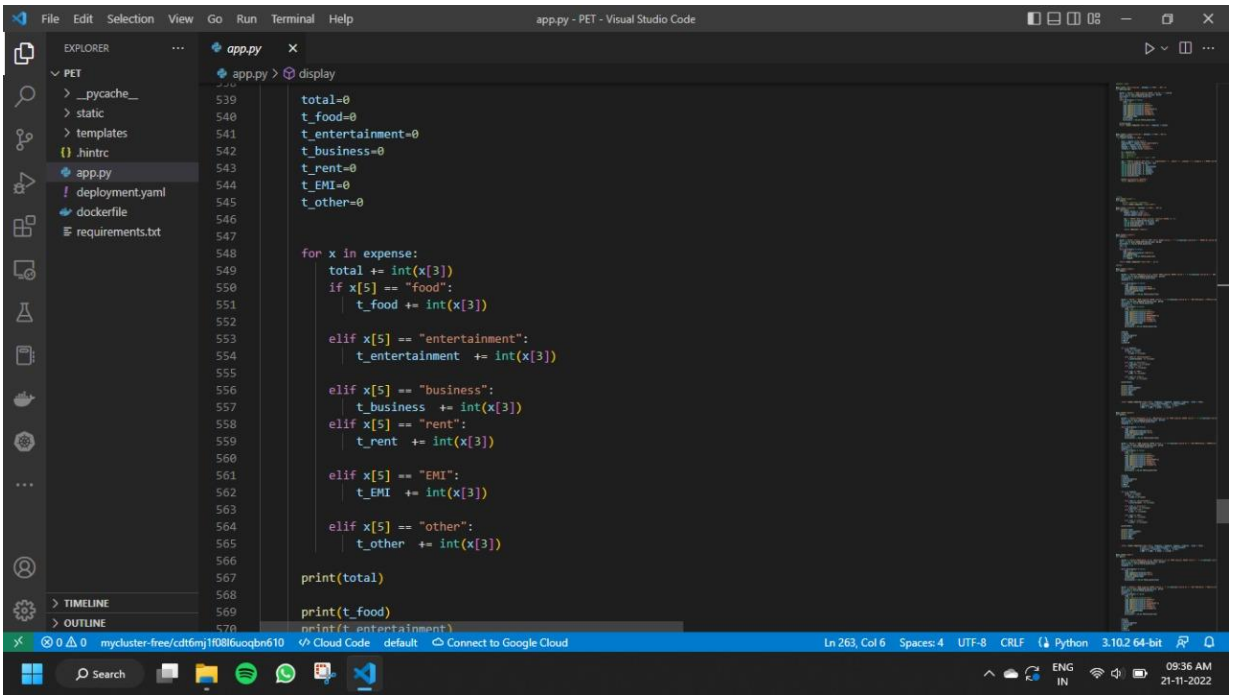


Fig 7.18

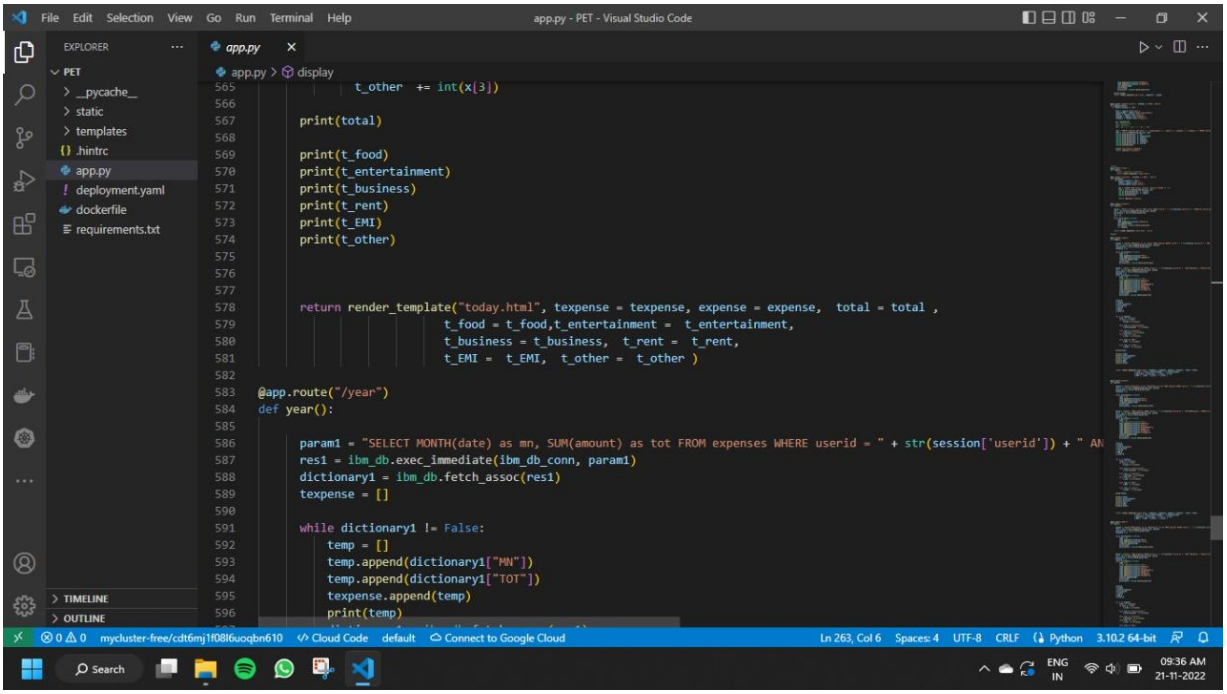


Fig 7.19

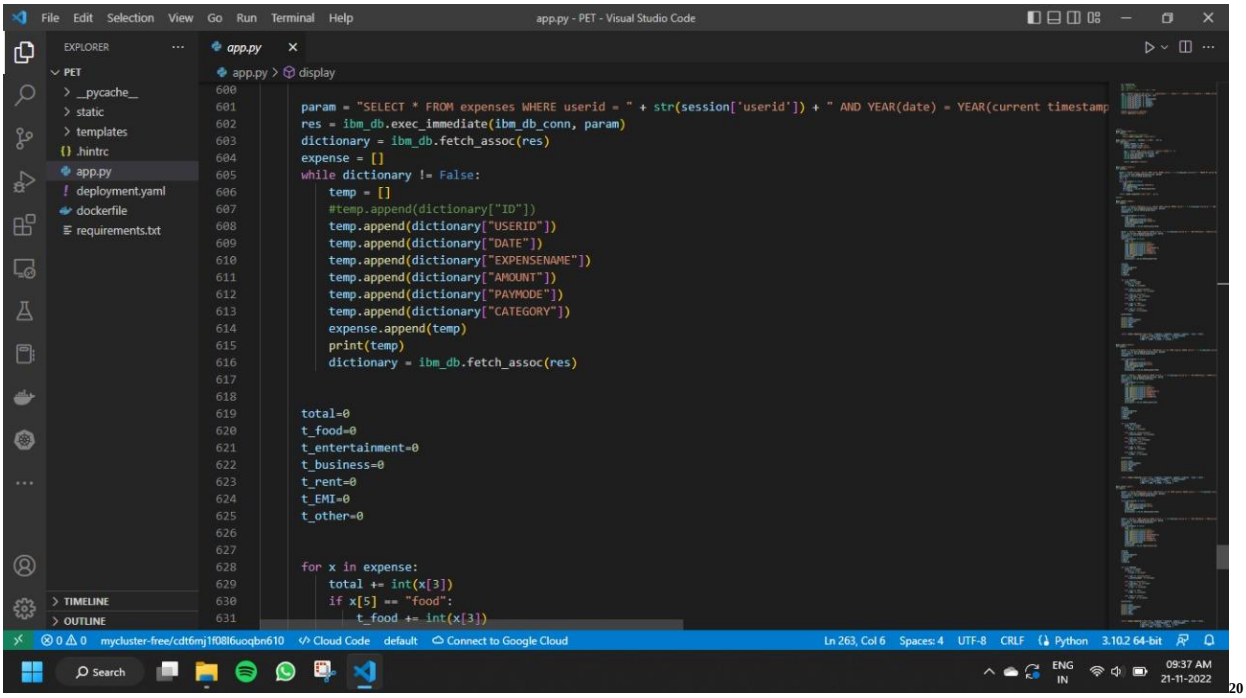


Fig 7.20

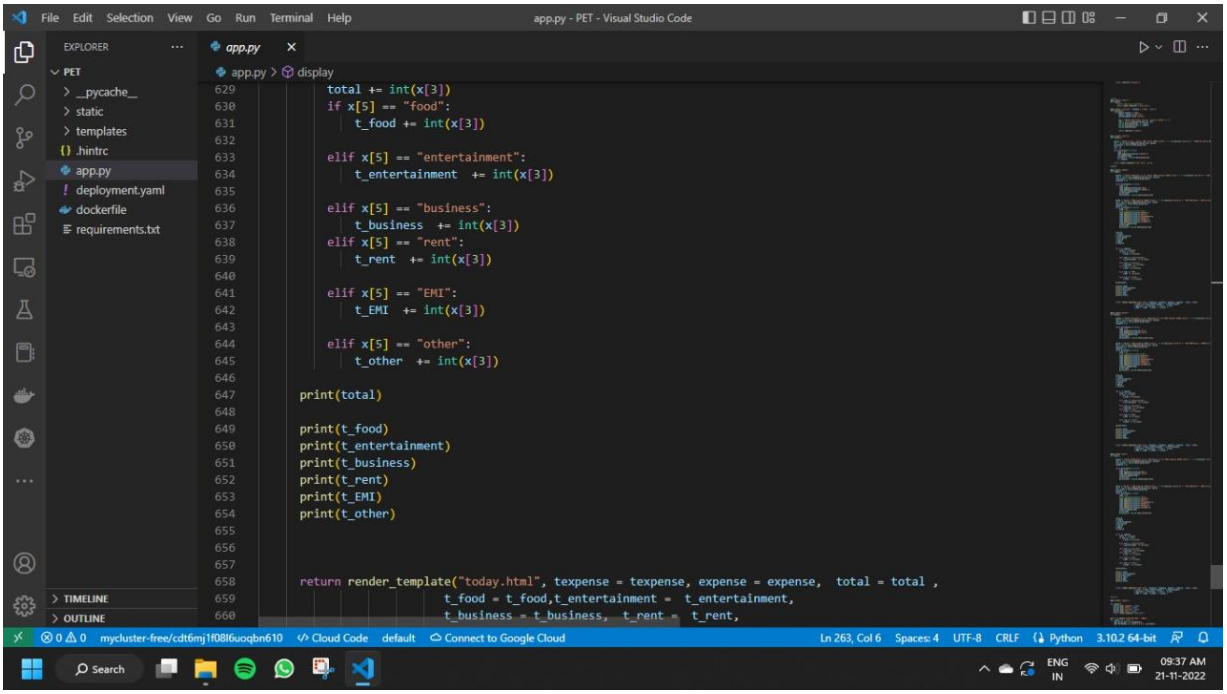


Fig 7.21

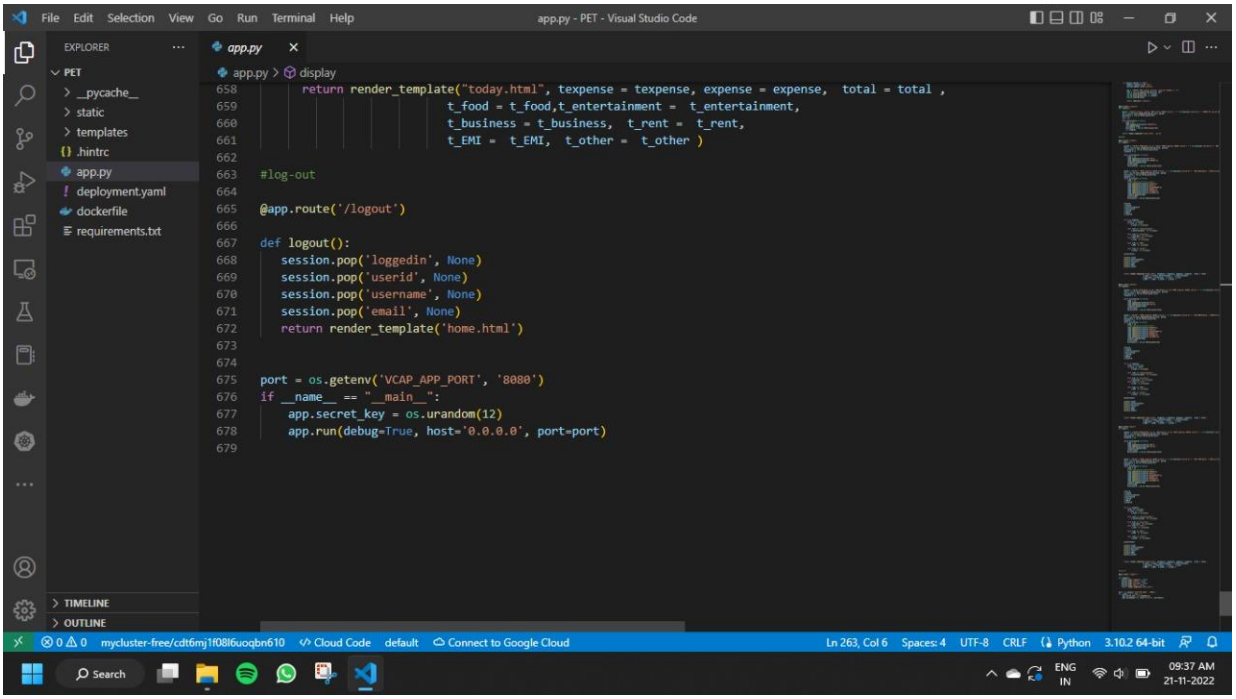


Fig 7.22

7.2 Output

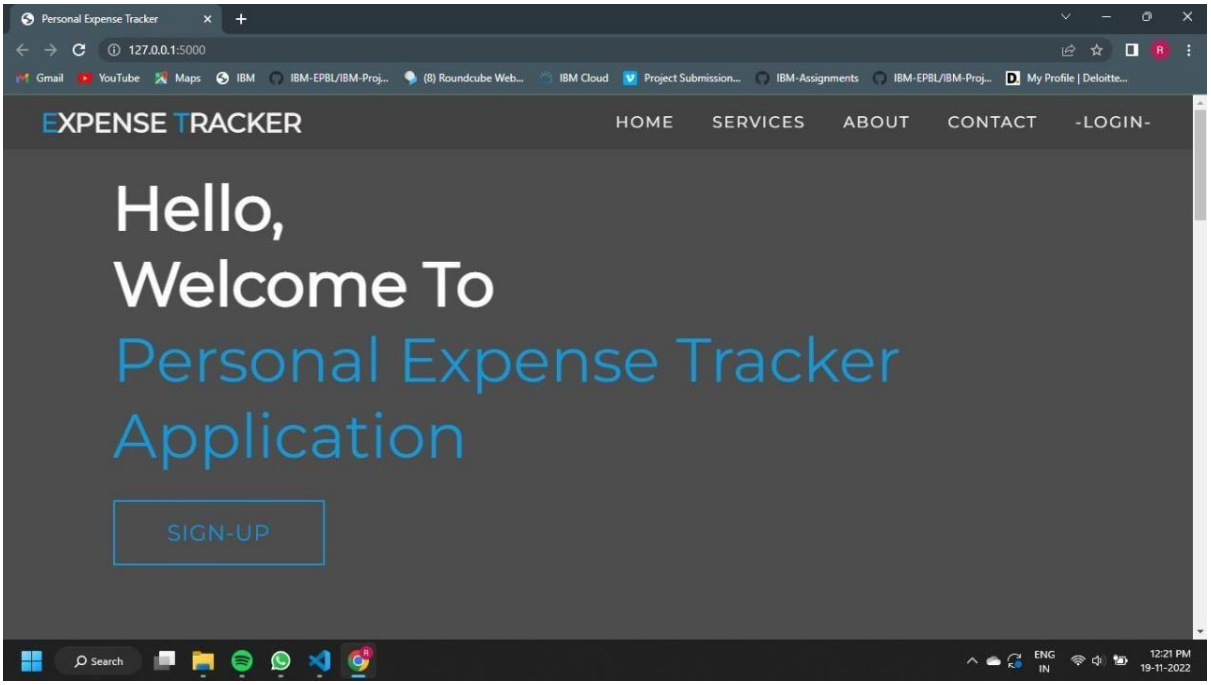


Fig 7.23

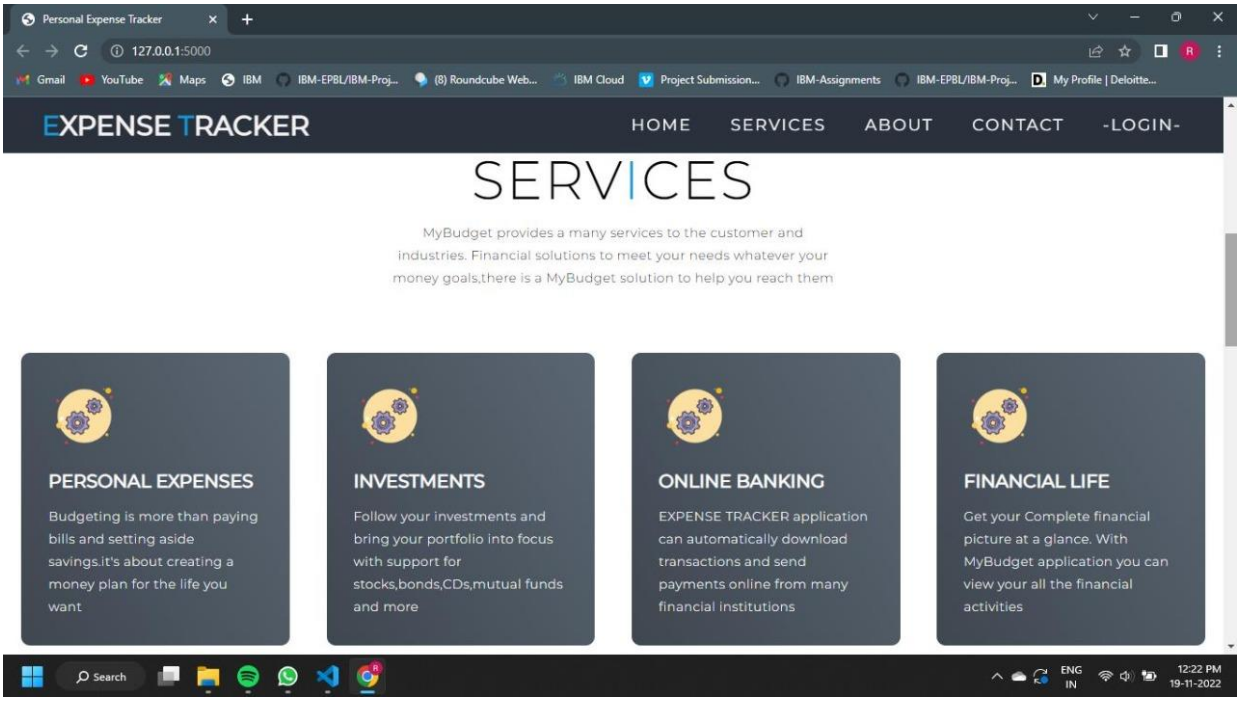


Fig 7.24

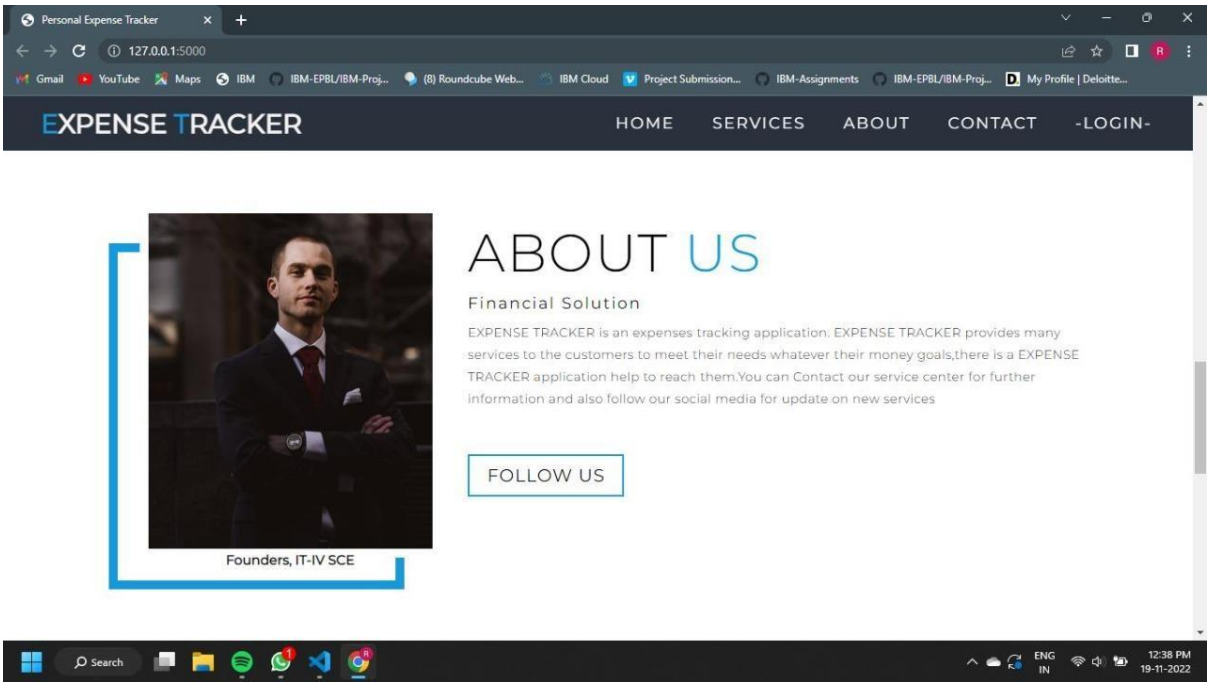


Fig 7.25

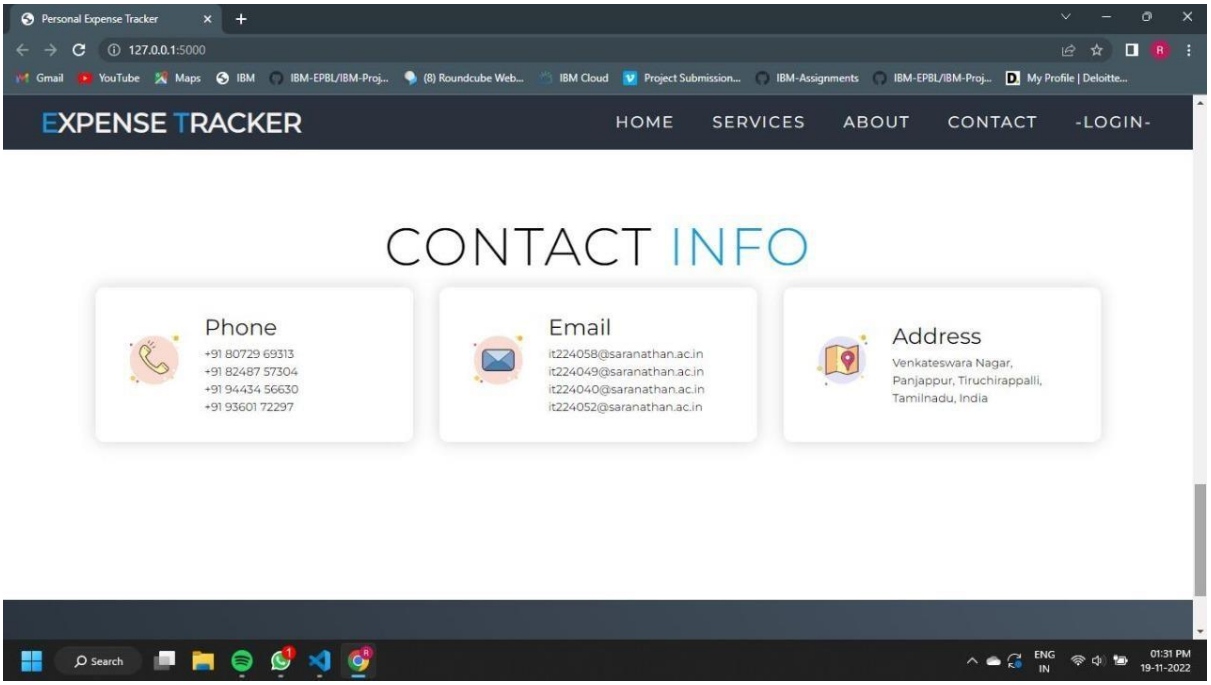


Fig 7.26

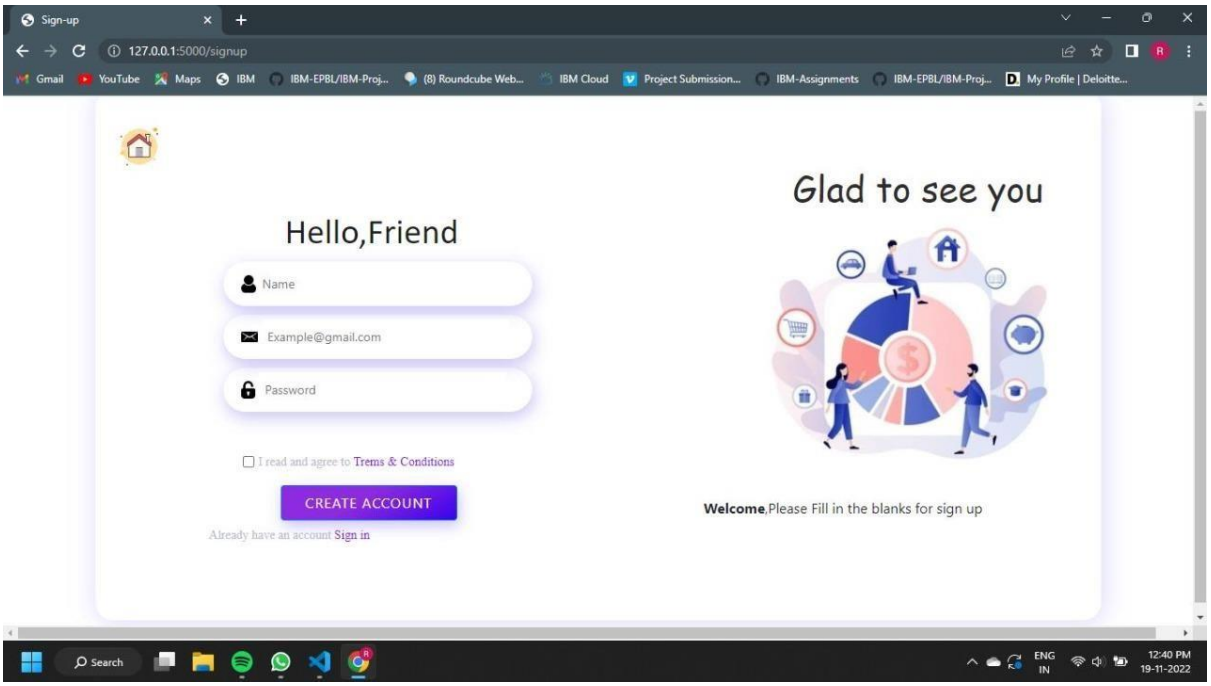


Fig 7.27

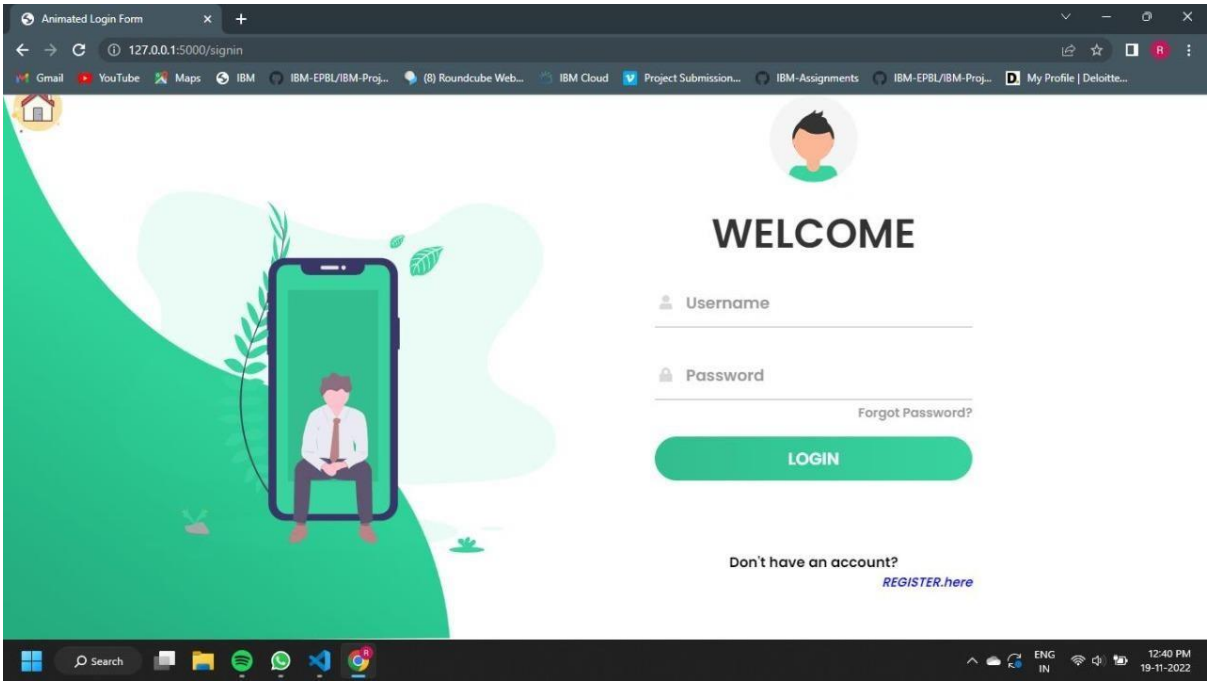


Fig 7.28

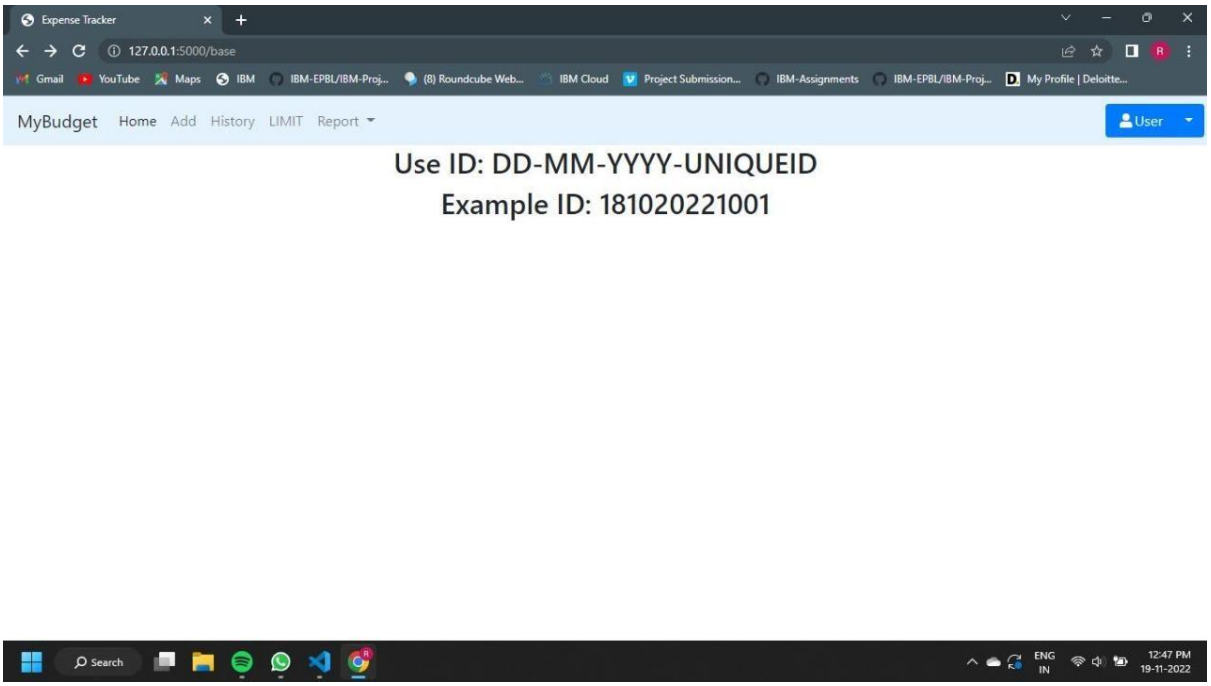


Fig 7.29

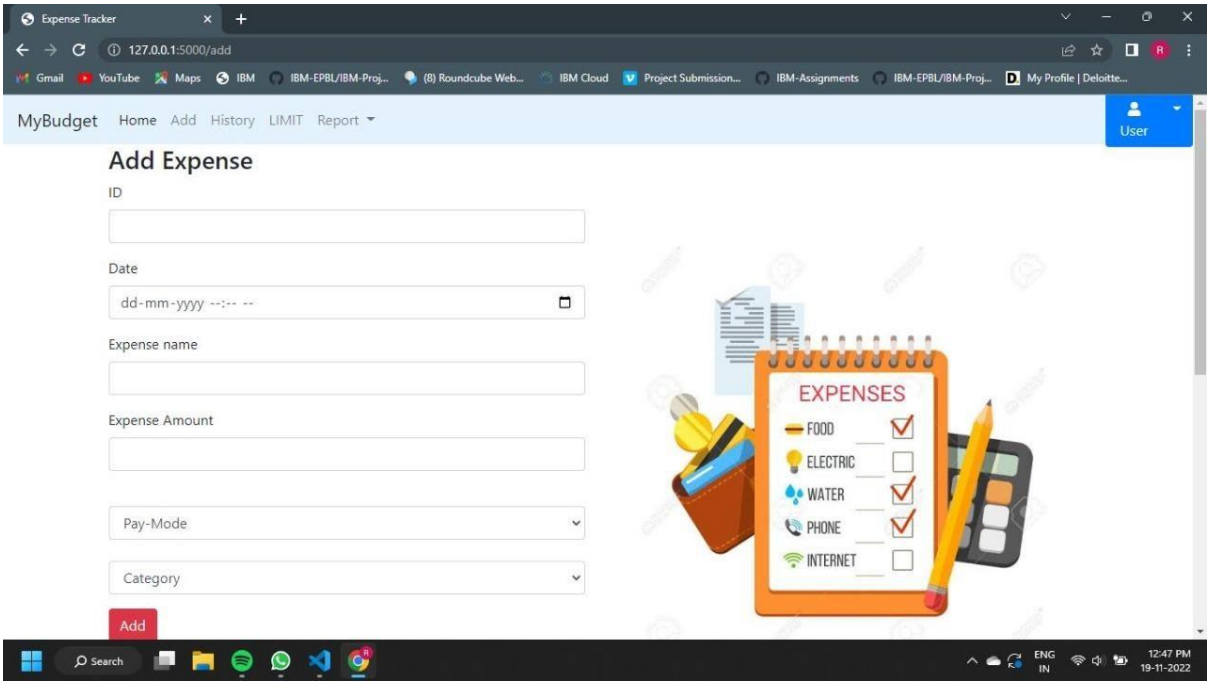


Fig 7.30

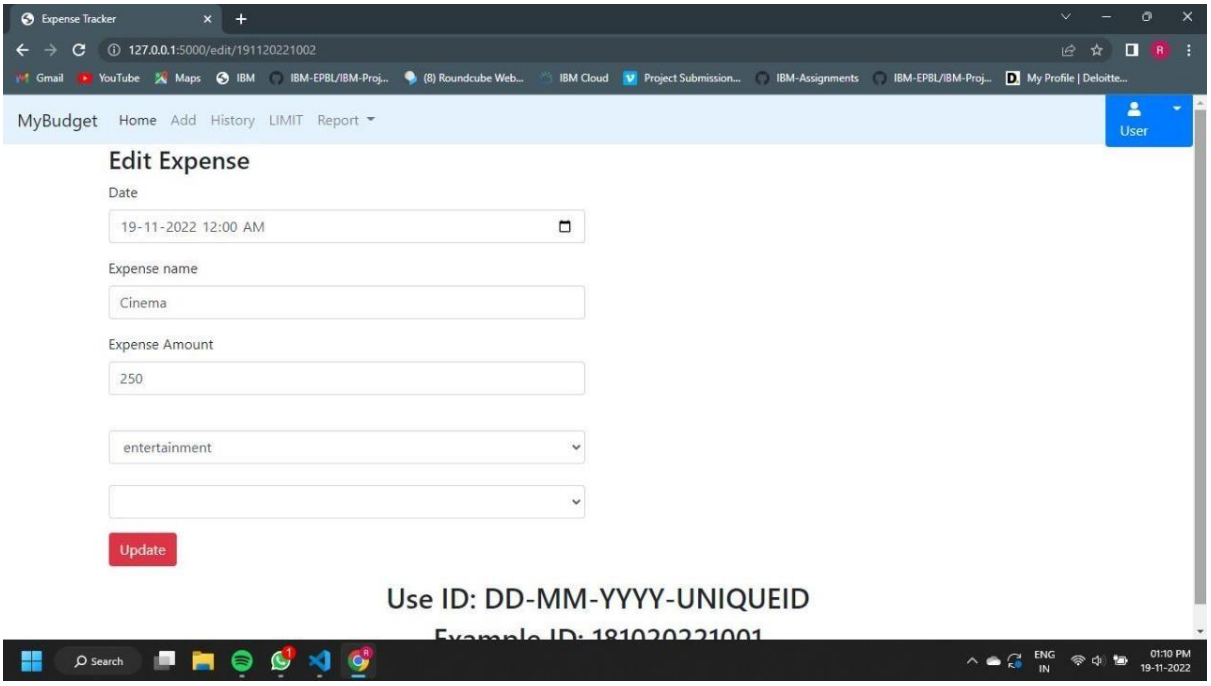


Fig 7.31

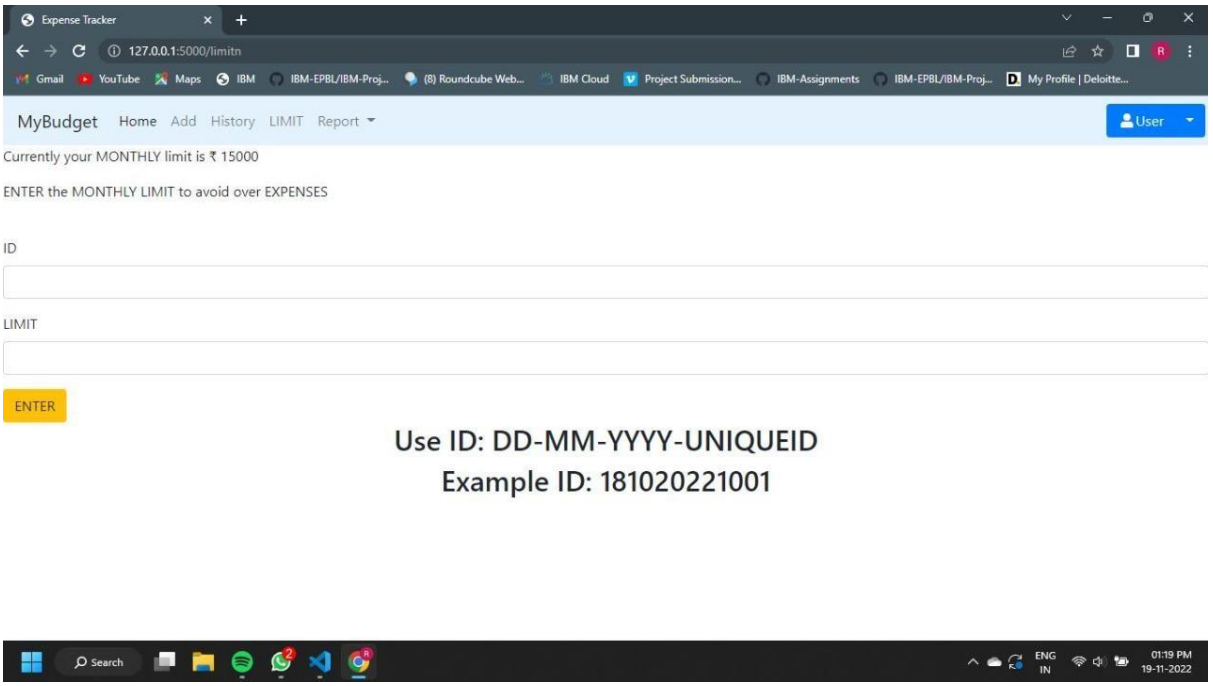


Fig 7.32

CHAPTER 8

TESTING

8.1 Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. This testing methodology is done during the development process by the software developers and sometimes QA staff.

Unit testing is a type of testing in which individual units or functions of software testing. Its primary purpose is to test each unit or function. A unit is the smallest testable part of an application. It mainly has one or a few inputs and produces a single output.

8.2 Integration Testing

Integration testing is also known as integration and testing (I&T), is a type of software testing in which the different units, modules or components of a software application are tested as a combined entity. However, these modules may be coded by different programmers.

Integration Testing is a type of software testing, which is performed on software to determine the flow between two or more modules by combining them. Integration testing makes sure that the interactions between different components of the software is completed smoothly without any complication.

The purpose of the integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit tested, integration testing is performed.

8.3 Test Cases

S.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Sign Up	User name, Email ID and password	Signed Up successfully	Signed Up successfully	PASS

Table 8.1

S.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Login	User name and password	Logged in successfully	Logged in successfully	PASS

Table 8.2

.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Adding expense	Expense ID, expense name, date, amount, payment method and category	Expense added	Expense added	PASS

Table 8.3

S.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Setting up the limit	Expense ID, Monthly limit and amount	Limit set up is Successful	Limit set up is Successful	PASS

Table 8.4

S.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Editing expense	Expense ID and date	Edited successfully	Edited successfully	PASS

Table 8.5

S.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Expense report	Month	Report generated in a pie chart	Report generated in a pie chart	PASS

Table 8.6

CHAPTER 9

RESULTS

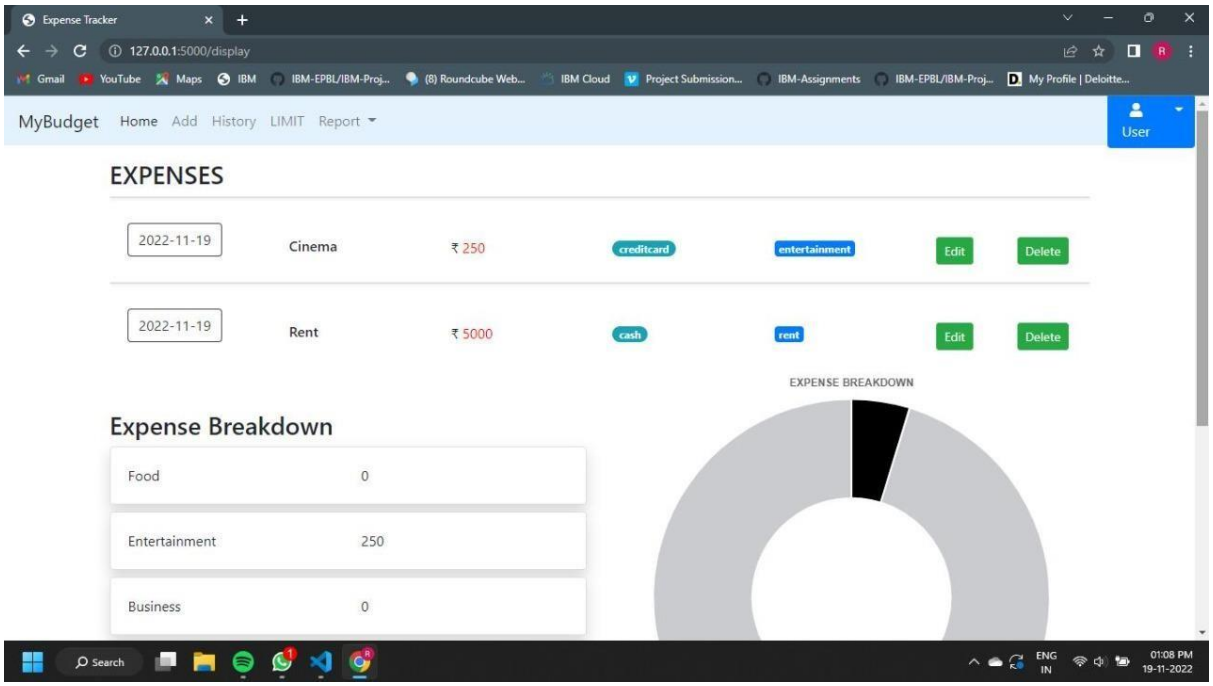


Fig 9.1

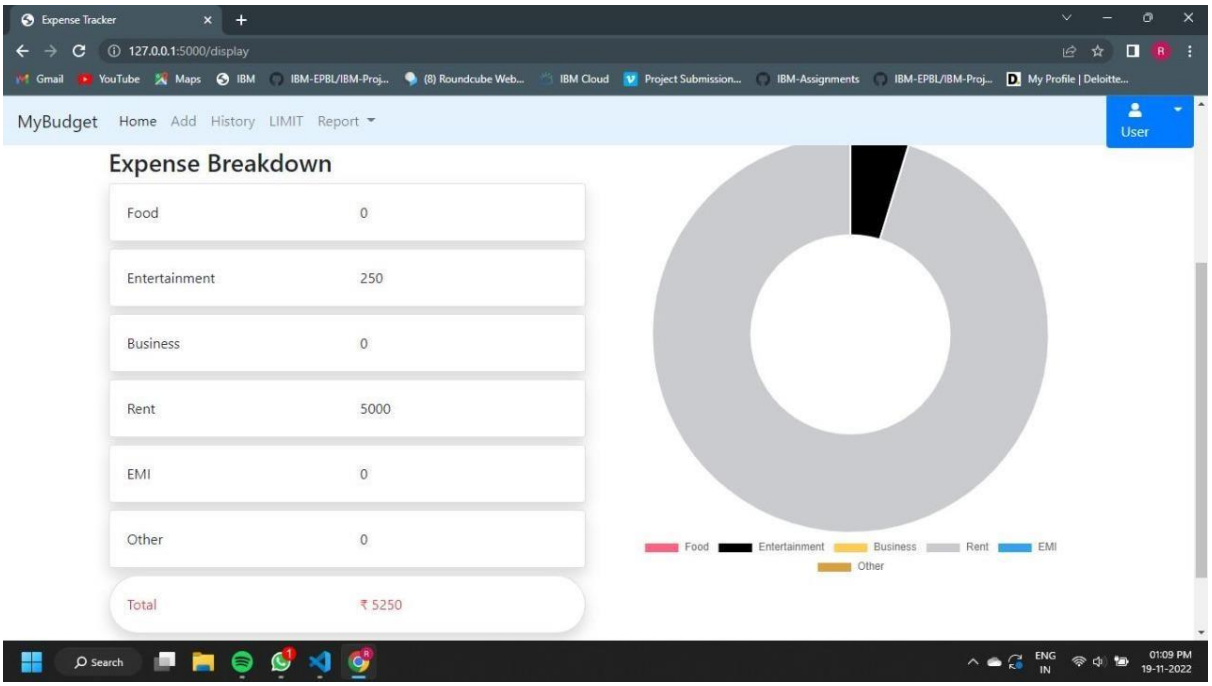


Fig 9.2

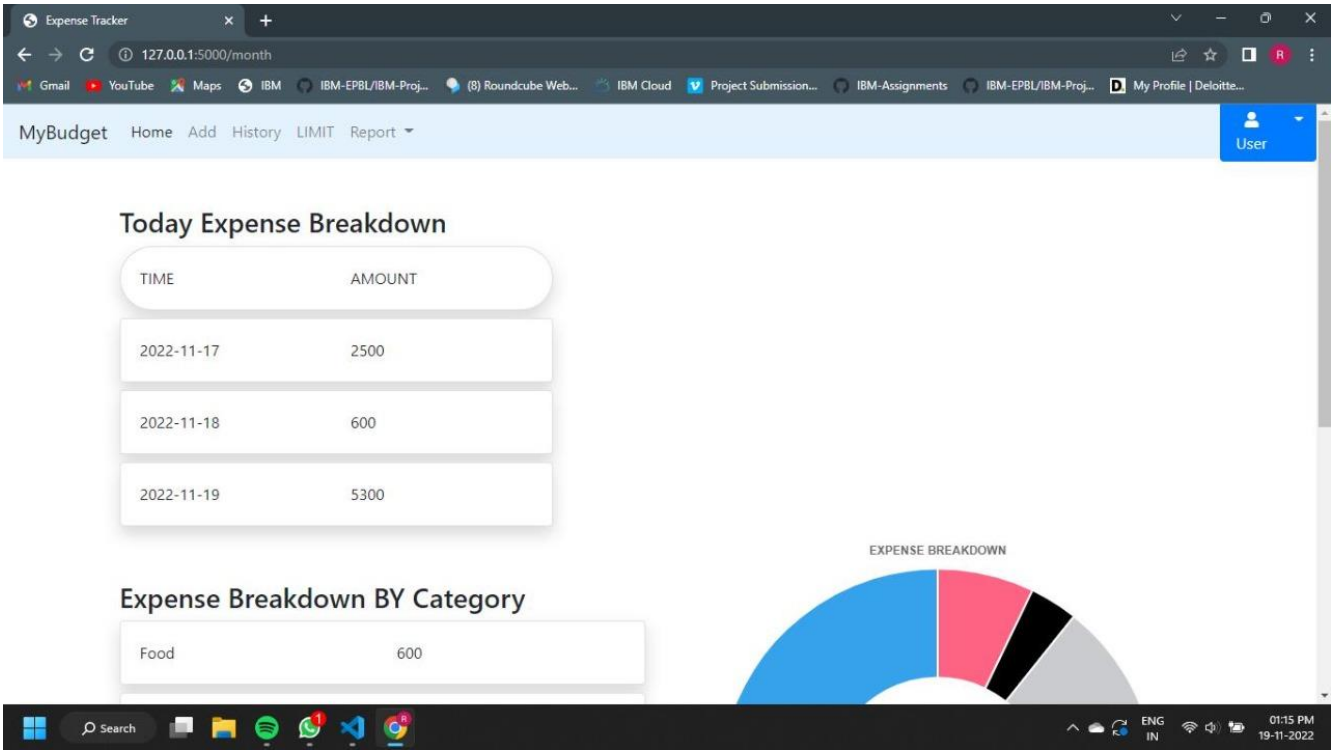

Expense Tracker
127.0.0.1:5000/month
MyBudget
Home
Add
History
LIMIT
Report
User
Today Expense Breakdown
TIME
AMOUNT
2022-11-17
2500
2022-11-18
600
2022-11-19
5300
EXPENSE BREAKDOWN
Expense Breakdown BY Category
Food
600

Fig 9.3

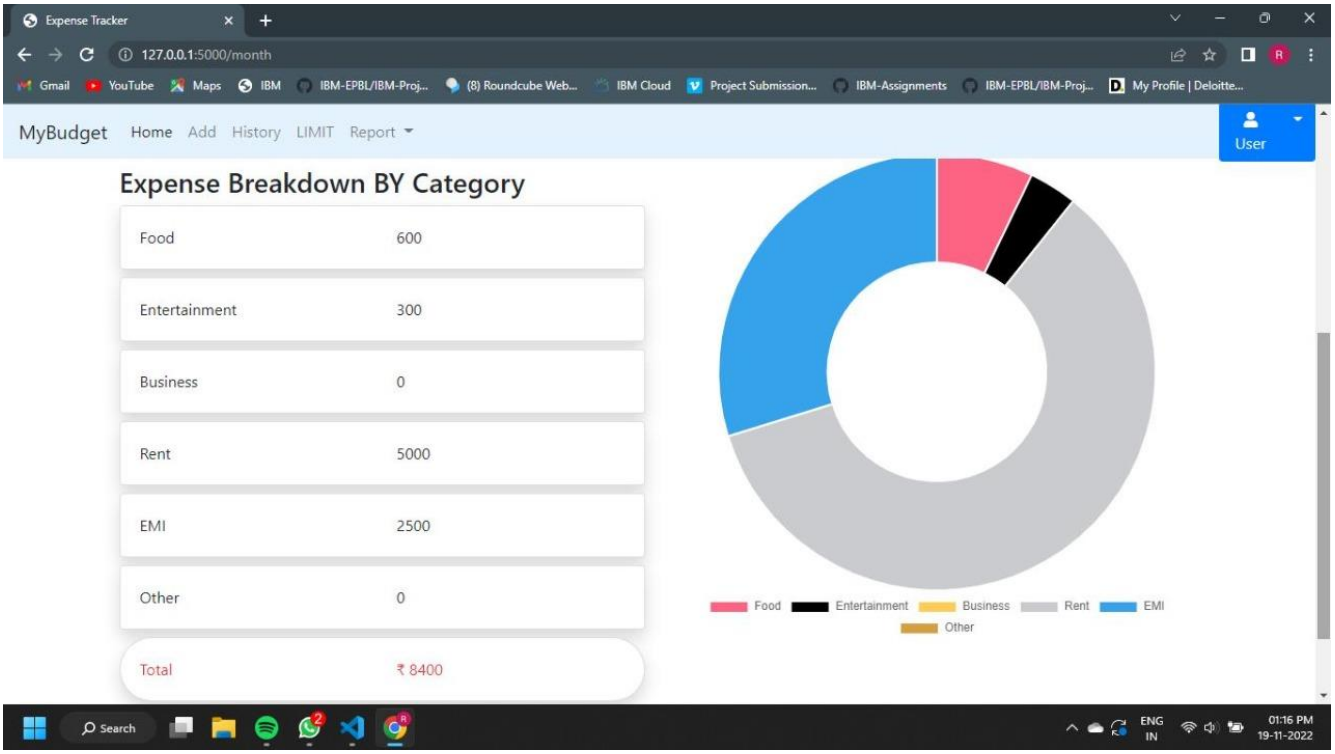

Expense Tracker
127.0.0.1:5000/month
MyBudget
Home
Add
History
LIMIT
Report
User
Expense Breakdown BY Category
Food
600
Entertainment
300
Business
0
Rent
5000
EMI
2500
Other
0
Total
₹ 8400
Food
Entertainment
Business
Rent
EMI
Other

Fig 9.4

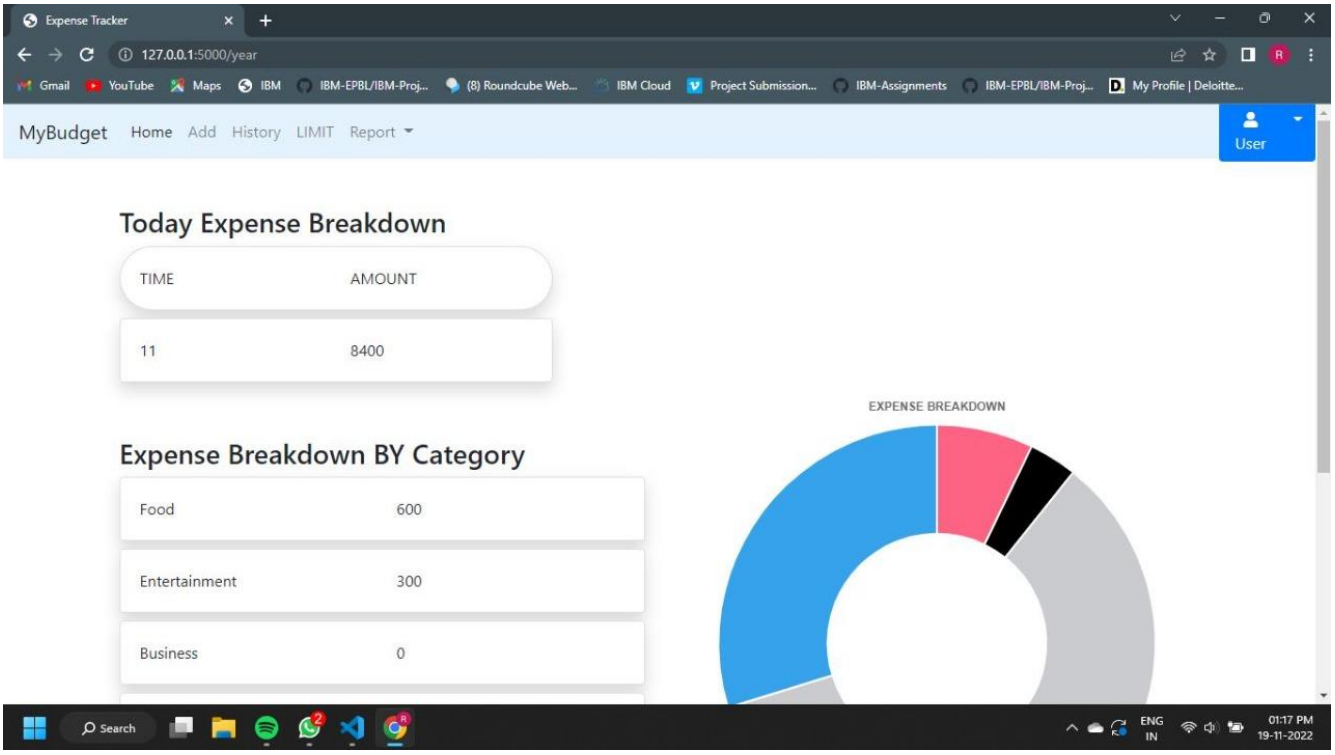

Expense Tracker
127.0.0.1:5000/year
MyBudget
Home
Add
History
LIMIT
Report
User
Today Expense Breakdown
TIME
AMOUNT
11
8400
Expense Breakdown BY Category
Food
600
Entertainment
300
Business
0
EXPENSE BREAKDOWN

Fig 9.5

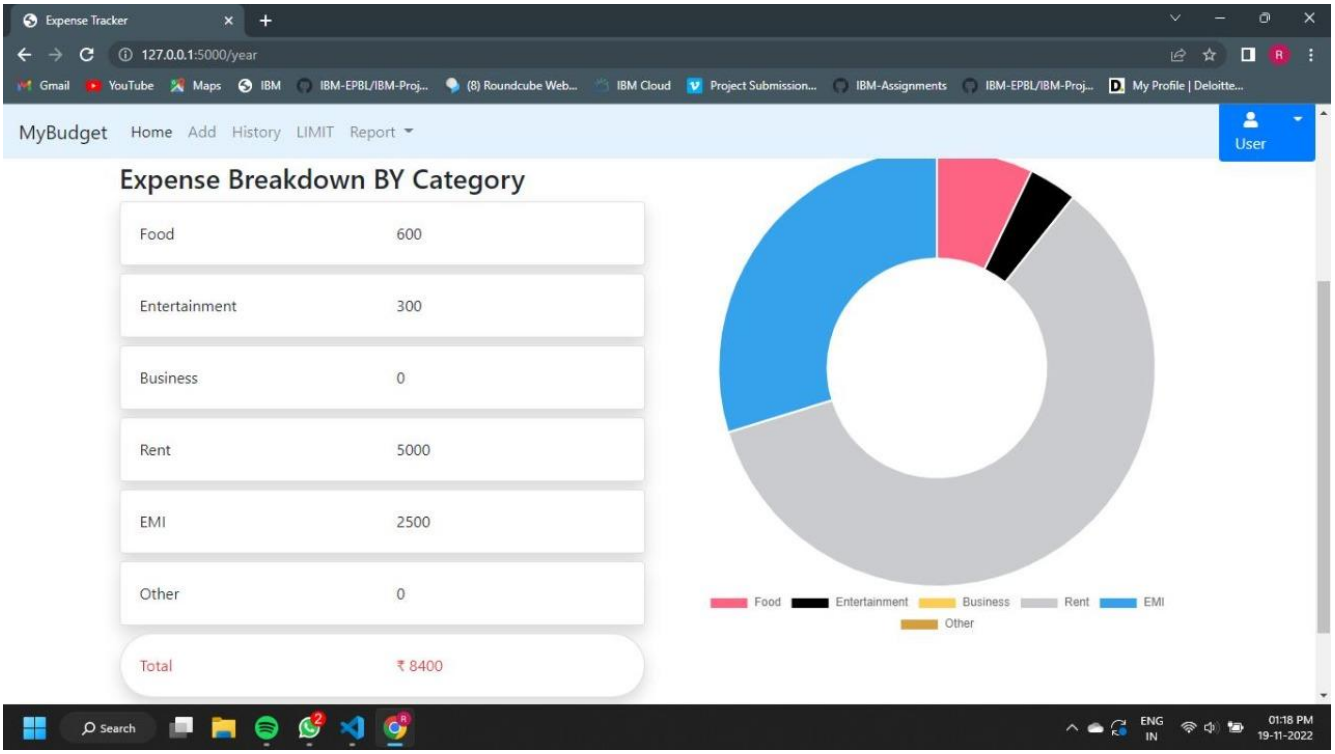

Expense Tracker
127.0.0.1:5000/year
MyBudget
Home
Add
History
LIMIT
Report
User
Expense Breakdown BY Category
Food
600
Entertainment
300
Business
0
Rent
5000
EMI
2500
Other
0
Total
₹ 8400
Food
Entertainment
Business
Rent
EMI
Other

Fig 9.6

CHAPTER 10

CONCLUSION

- Our Expense Tracker application allows users to easily track their expenses by entering daily, monthly, and annual expenses.
- The monthly expenditure report can be generated, and the reports are displayed in the form of a pie chart.
- The user has control over how much money they spend each month.
- Paper and pencil are not required because this application can perform all calculations.
- Users can add, edit, and delete expenses as necessary.

CHAPTER 11

FUTURE SCOPE

- Paper and pencil are not required because this application can perform all calculations.
- Money can be saved and helps the user to reduce spending money on unwanted things.
- People can understand the value of the money and importance of saving money.

CHAPTER 12

APPENDIX

12.1 Source Code

https://github.com/IBM-EPBL/IBM-Project-16152-1659608395/tree/main/Project%20Development%20Phase

12.2 GitHub & Project Demo Link

https://github.com/IBM-EPBL/IBM-Project-16152-1659608395/tree/main/Final%20Deliverables