GOVERNMENT COLLEGE OF ENGINEERING
(Formerly IRTT)
ERODE-638 316

BONAFIDE CERTIFICATE

Certified that this project titled **"Personal Expense Tracker Application"** is the bonafide work of **"GOPI G(731119104014), GANAPATHI I(731119104012), JAYAPRAKASH R(731119104020),ANBUSELVAM A(731119104005)"** who carried out the project work under my supervision.

SIGNATURE OF HOD
Dr.A.SARADHA, M.E.,Ph.D
HEAD OF THE DEPARTMENT
DEPARTMENT OF CSE,
GOVERNMENT COLLEGE OF
ENGINEERING, ERODE – 638316

SIGNATURE OF SPOC
Dr.G.GOWRISON, M.E.,Ph.D.,
ASSISTANT PROFESSOR(SR)
DEPARTMENT OF ECE,
GOVERNMENT COLLEGE OF
ENGINEERING,ERODE - 638316

SIGNATURE OF FACULTY MENTOR
DR.A.KAVIDHA, M.E.,Ph.D

ASSISTANT PROFESSOR(SR)
DEPARTMENT OF CSE,
GOVERNMENT COLLEGE OF
ENGINEERING, ERODE – 638316

SIGNATURE OF FACULTY EVALUATOR
DR.A.SARADHA, M.E.,Ph.D
HEAD OF THE DEPARTMENT
DEPARTMENT OF CSE,
GOVERNMENT COLLEGE OF
ENGINEERING, ERODE - 63831

PERSONAL EXPENSE TRACKER

Source Code GitHub & Project Demo Link

1 INTRODUCTION

In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management.

1.1 Project Overview

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.

1.2 Purpose

- **No Need to install web application:** the problem of installing web application avoided on any device. So, reducing space and time related problems.
- **Remotely Accessible:** a web-based application can be used remotely via a network connection that is platform independent.
- **Movability and Repository:** to reduce the problem of movability and repository field by using to make the concept of web-based application.
- **Cloud Storage:** Cloud Storage allows developers to store and retrieve data from the cloud database. The data stored in cloud will not expire. This means that data will persist even if the tab or the browser window is closed.

- **Voice Features:** we have used Speechly for building real-time multimodal voice user interfaces. It enables developers and designers to enhance their current touch user interface with voice functionalities for better user experience.

2. LITERATURE SURVEY

2.1. Existing problem

- Lack of proper planning of out income.
- Person has to keep a log in a diary or in a computer.
- All the calculations need to be done by the user.
- Overload to rely on the daily entry of the expenditure
- At the end of the month, we start to have money crisis.

2.2. Reference

S. NO	TITLE OF THE PAPER	AUTHOR	PUBLISHED YEAR	ABSTRACT

1.	Expense Tracker	Aman Garg, Mukul Goel, Sagar Mittal, Mr.Shekhar Singh	April 2021	This Expense Tracker is a web application that facilitates the users to keep track and manage their personal as well as business expenses. This application helps the users to keep a digitaldiary. It will keep track of a user's income and expenses daily. Tracking your expenses daily can not only save your amount, but it can also assist you set financial goals for the longer term. If you know exactly where your amount goes every month, you will easily see where some cutbacks and compromises can be made.
2.	Family Expense Manager Applicat io	Rajaprabh a M N	2017	The user can make use of this application in his/her daily life. After being used it can be a part of daily life to update

	n			and view daily expenses and family expenses. This helps to keep track of expenses & manage it for the user as they are busy in their daily routine, they are not able to keep track of their incomes & expenses
3.	Daily Expense Tracker Mobile Applicat io n	Nuura Najati Binti Mustafa	2021	This application is an easier alternative to keeping track of users' use of money than the traditional way of writing their expenses in their diary. This application implements least squares method which helps to predict an outcome by finding the best fit line for a set of data. The use of the least squares method will help users in obtaining a successful budget planned with the prediction of the outcome of the budget based on expenses

4.	Daily Expense Tracker	Shivam Mehra, Prabhat Parashar	2021	This paper represents a Daily Expense Tracker is a tool that resides on a remote server and is accessible via browsers. It creates a digital record of the income and expense of the user. It input from the user a income, source of this income and the date of earning that income and creates a transaction entry under income category sums to the total amount of income and making real time changes. The web application will also be voice powered and all the functionalities can be used with voice commands.
5.	Expenditur e manage me	Dr.V.Geetha, G. Nikhitha.	2022	In this method this device can be utilized by any individual to govern their income expenditure from each day to annual basis

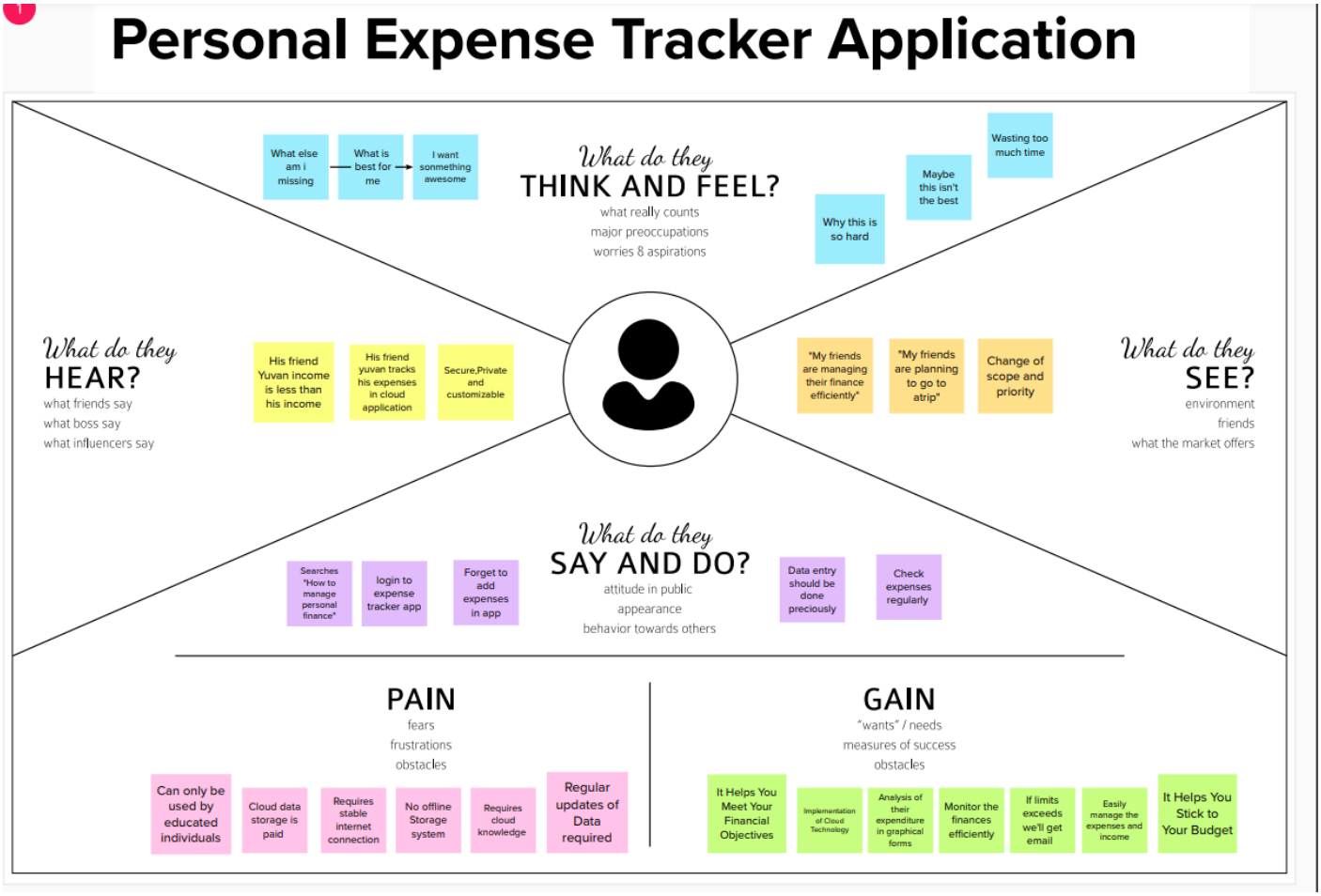
	nt system			and to hold an eye on their spending, Including the person to whom the payments were made and the purpose for the payment. On a weekly, monthly, and yearly basis, details of expenses will be displayed in the form of a pie chart. It aids us in remembering and adding information about what money we receive from others and what costs or payments we must make on a given date or month.

2.3. Problem Statement Definition

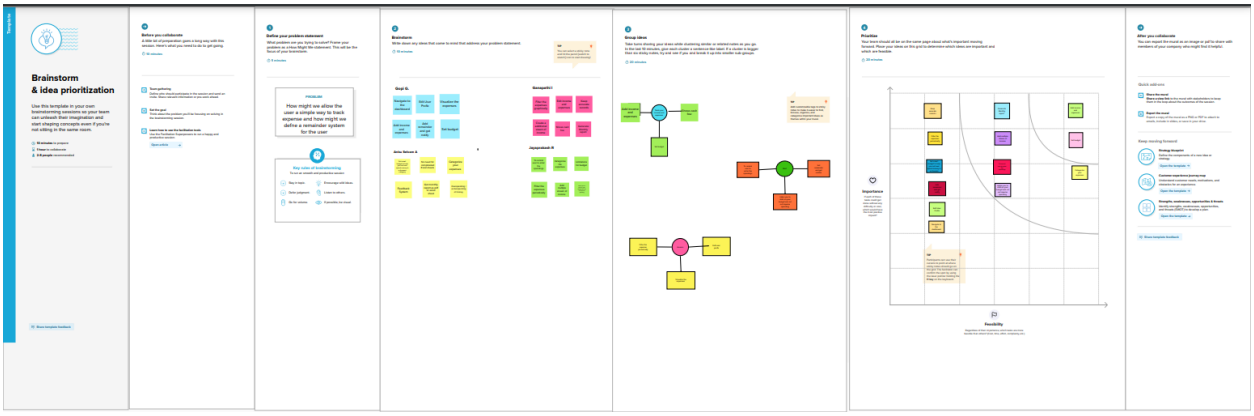
Many organizations have their own system to record their income and expenses, which they feel is the main key point of their business progress. It is a good habit for a person to record daily expenses and earning but due to unawareness and lack of proper applications to suit their privacy, lacking decision making capacity people are using traditional note keeping methods to do so. Due to lack of a complete tracking system, there is a constant overload to rely on the dailyentry of the expenditure and toestimation till the end of the month.

3. IDEATION & PROPOSED SOLUTION

3.1. Empathy Map Canvas



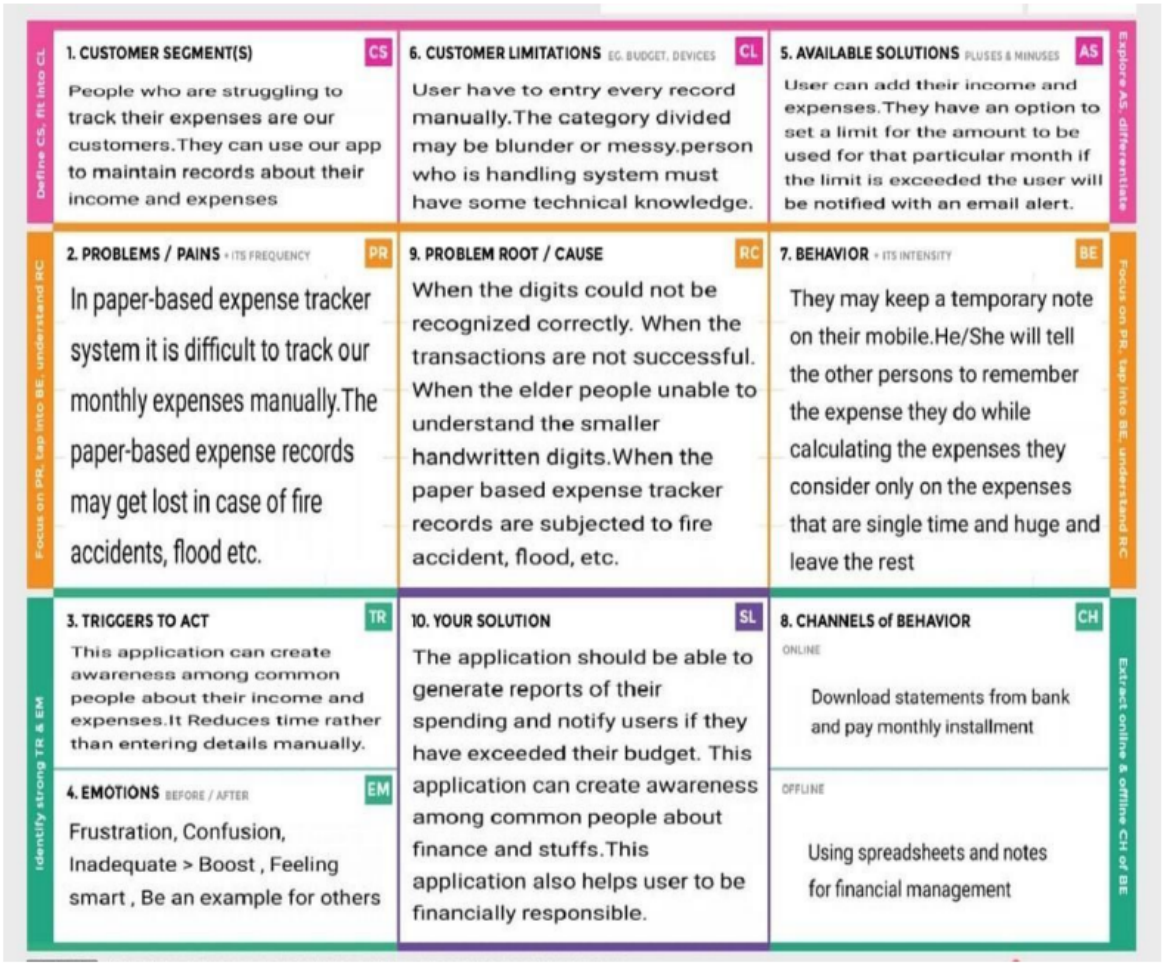
3.2. Ideation & Brainstorming



3.3. Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	In paper-based expense tracker system it is difficult to track our monthly expenses manually. The paper-based expense records may get lost in case of fire accidents, flood etc.
2.	Idea / Solution description	This app makes your life easier by helping you to manage your finances efficiently. This personal expense app will not only help you with budgeting and accounting but also give you helpful insights about financial management.
3.	Novelty / Uniqueness	The user gets notified when their expense exceeds the limit and also it reminds the user when they forgot to make entry.
4.	Social Impact / Customer Satisfaction	It will help the people to track their expenses and also alerts when they exceed the limit of their budget.
5.	Business Model (Revenue Model)	We can provide the application in a subscription based.
6.	Scalability of the Solution	This application can handle large number of users simultaneously.

3.4 Problem Solution Fit



4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Financial Accounts	Account Details Verification of Details
FR-4	User Dashboard	Expense Data Data Records
FR-5	User Notifications	System Access Real time Alerting
FR-6	Security of User Data	Secured Database Data Security Algorithms

4.2 Non-functional Requirements

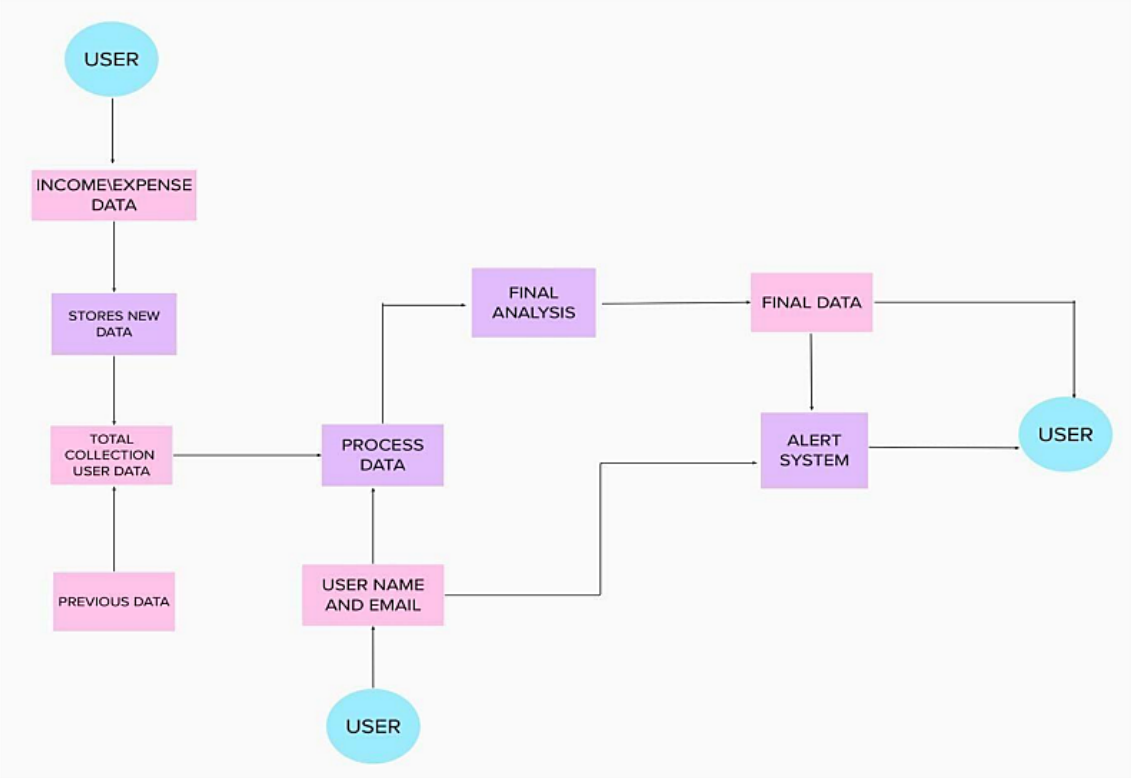
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	By using this application, the user can keep track of their expenses and can ensure that user's money is used wisely.
NFR-2	**Security**	Maintain user personal details in a encrypted manner by using data security algorithms .
NFR-3	**Reliability**	It will maintain a proper tracking of day-to-day expenses in an efficient manner.
NFR-4	**Performance**	By enter our incoming and departing cash, and the software can help you keep and monitor it with at-most quality and security with high performance.
NFR-5	**Availability**	Using charts and graphs may help you monitor your budgeting and assets.
NFR-6	**Scalability**	Rely on your budgeting app to track, streamline, and automate all the recurrent expenses and remind you on a timely basis.

5. Project Design

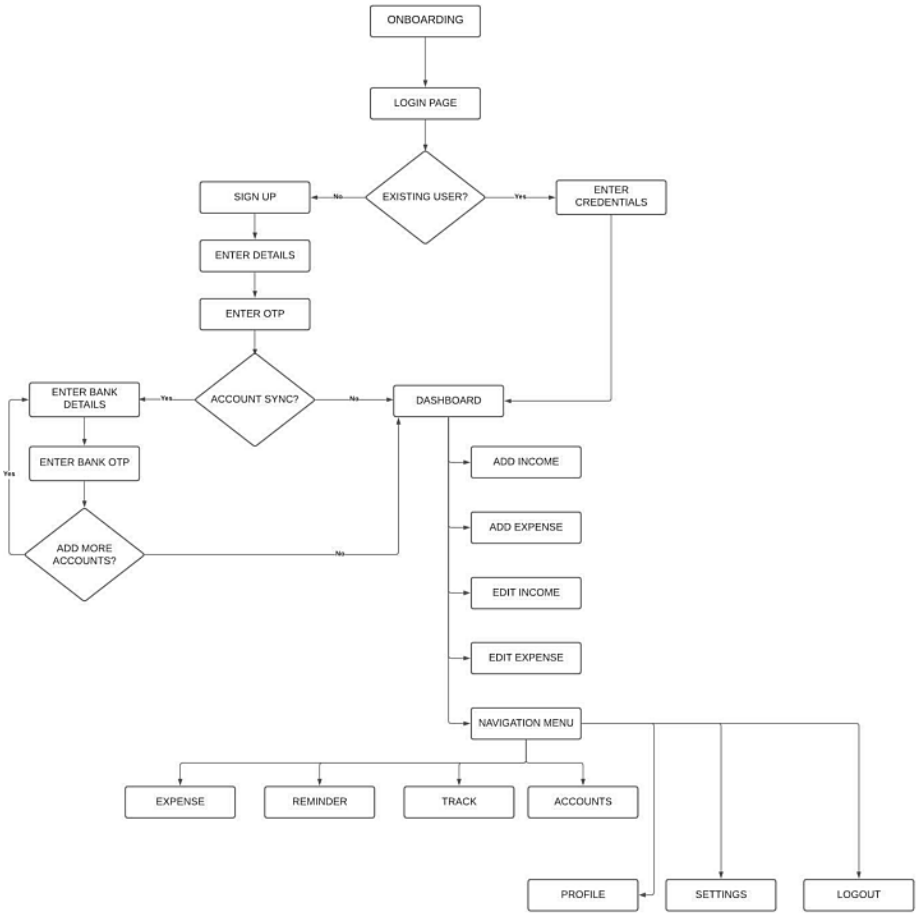
5.1. Data Flow Diagrams

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.

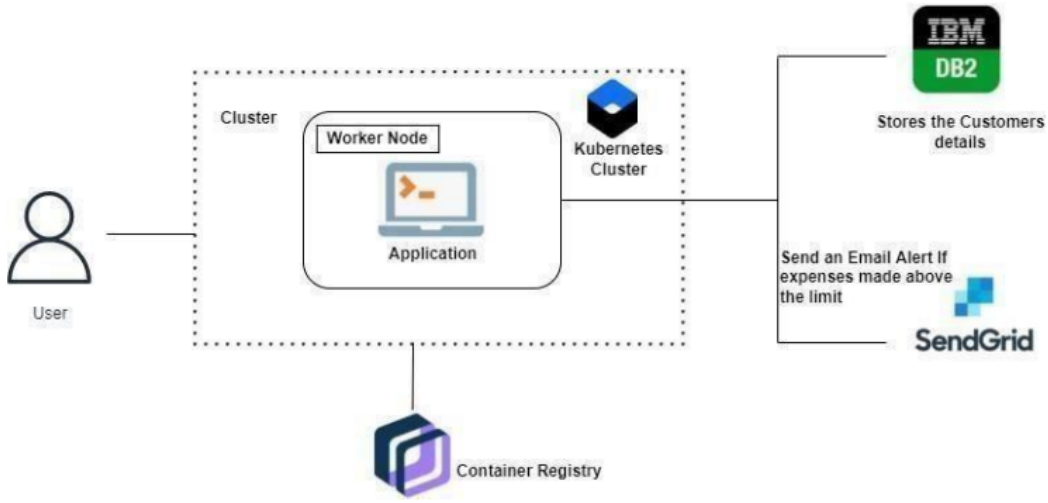


5.2. Solution & Technical Architecture

Solution architecture:



Technical Architecture



5.3. User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user & Web user)	Registration	USN - 1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard.	High	Sprint - 1

	Login	USN - 2	As a user, I can login to user dashboard and see the information about my incomes and expenses.	I can login to user dashboard and see the information.	High	Sprint - 1
	Dashboard	USN - 3	As a user, I can enter my income and expenditure details.	I can view my daily expenses.	High	Sprint – 2
	Expense Update	USN - 4	As a user, I can track my expenses and manage my monthly budget.	I can track my expenses and manage my monthly budget.	High	Sprint – 3
	Email Alert	USN - 5	As a user, I can see if there is an excessive expense and if there is such condition, I will be notified via e-mail.	I can receive e mail, if there is an excessive expense.	Medium	Sprint – 3
Customer Care Executive	Customer Care	USN – 6	As a customer care executive, I can solve the log in issues and other issues of the application.	I can provide support or solution at any time 24*7	Medium	Sprint – 4
Administrator	Application	USN - 7	As an administrator, I can upgrade or update the application.	I can fix the bug which arises for the customers and users of the application.	Medium	Sprint – 4

6. PROJECT PLANNING & SCHEDULING

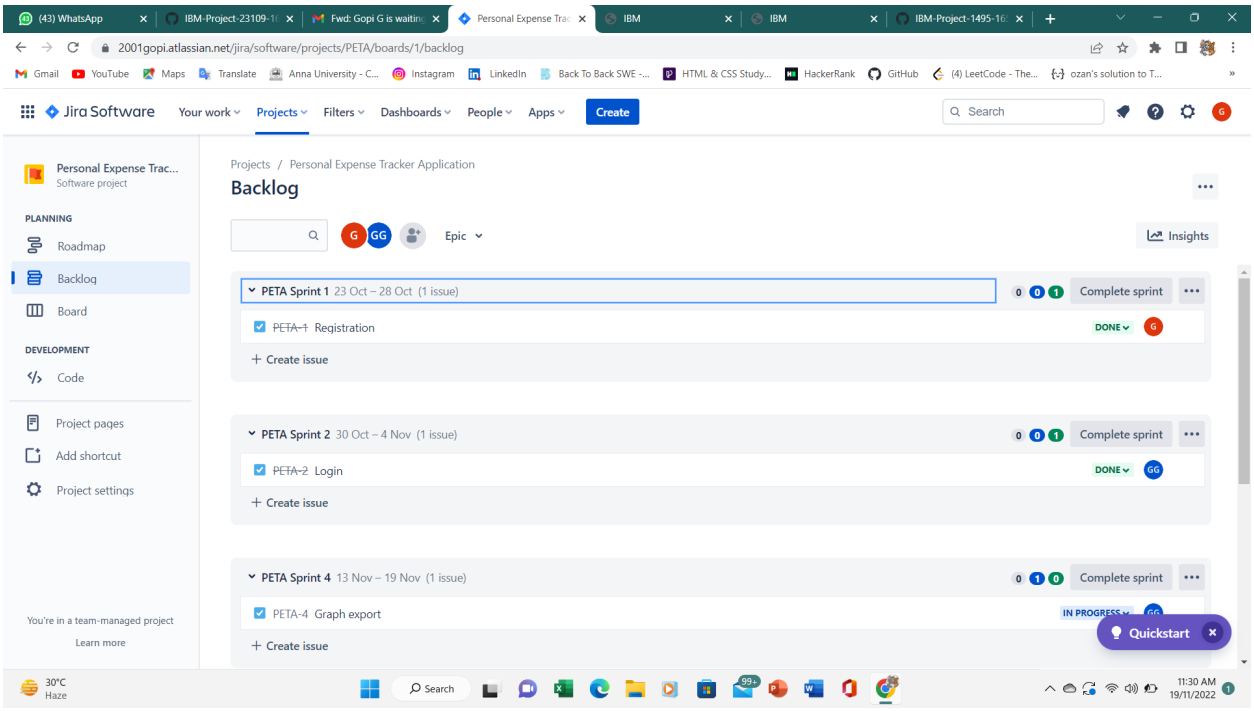
6.1. Sprint Planning & Estimation

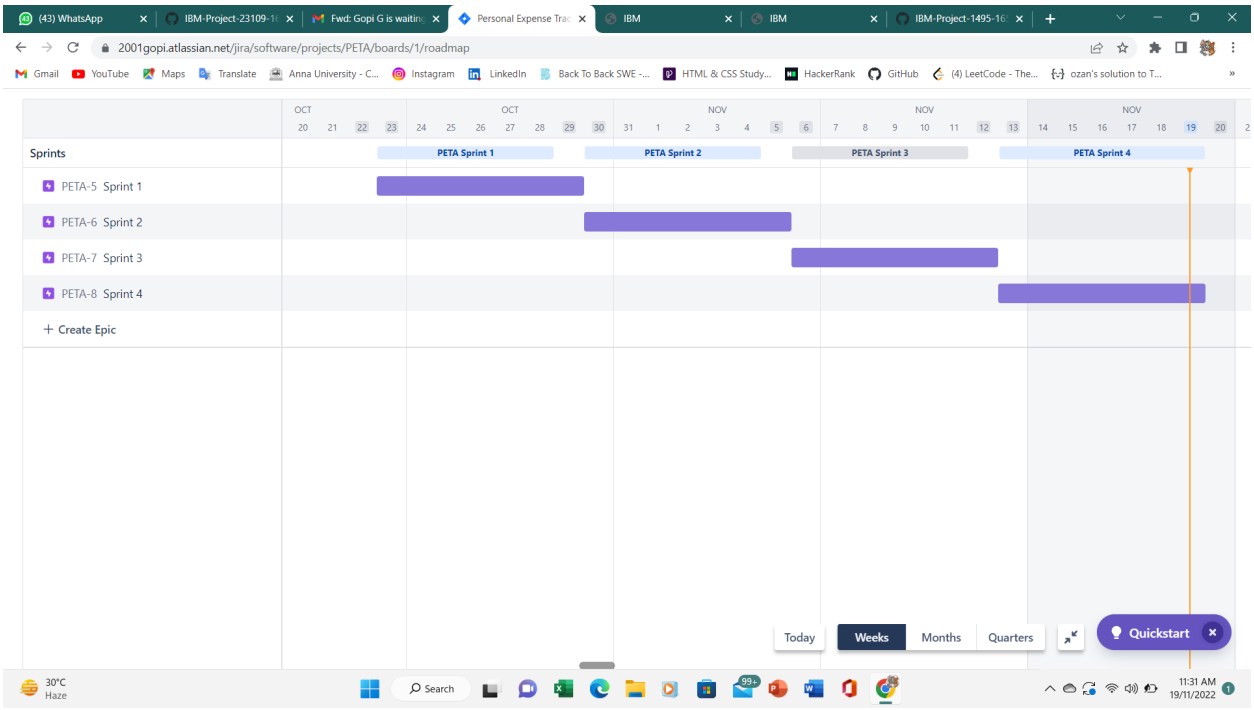
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Homepage	USN-1	AS a user I can view the index page to see the about of the Expense tracker	10	High	Gopi
Sprint-1	Registration	USN-2	As a User, I need to register user id and passcode for every workers over there in municipality	10	High	Ganapathi
Sprint-1	Login	USN-3	As a user, I need to login with user id and password to get in to the website	10	High	Jayaprakash
Sprint-2	Dashboard	USN-4	As a User, I will follow Co-Admin's instruction to reach the filling bin in short roots and save time	20	Low	Gopi Jayaprakash
Sprint-3	Add Expenses	USN-5	As a User I will add my expense throughout the month I spend on	20	Medium	Ganapathi Anbuselvam
Sprint-3	Total Expense Graph	USN-6	As a User I can view my expense in a graph of overview of the expense I spend.	20	Medium	Gopi
Sprint-4	Deployment in cloud	USN-7	As a User I can access the cloud to store my data of expense	20	High	Gopi Ganapathi Jayaprakash Anbuselvam

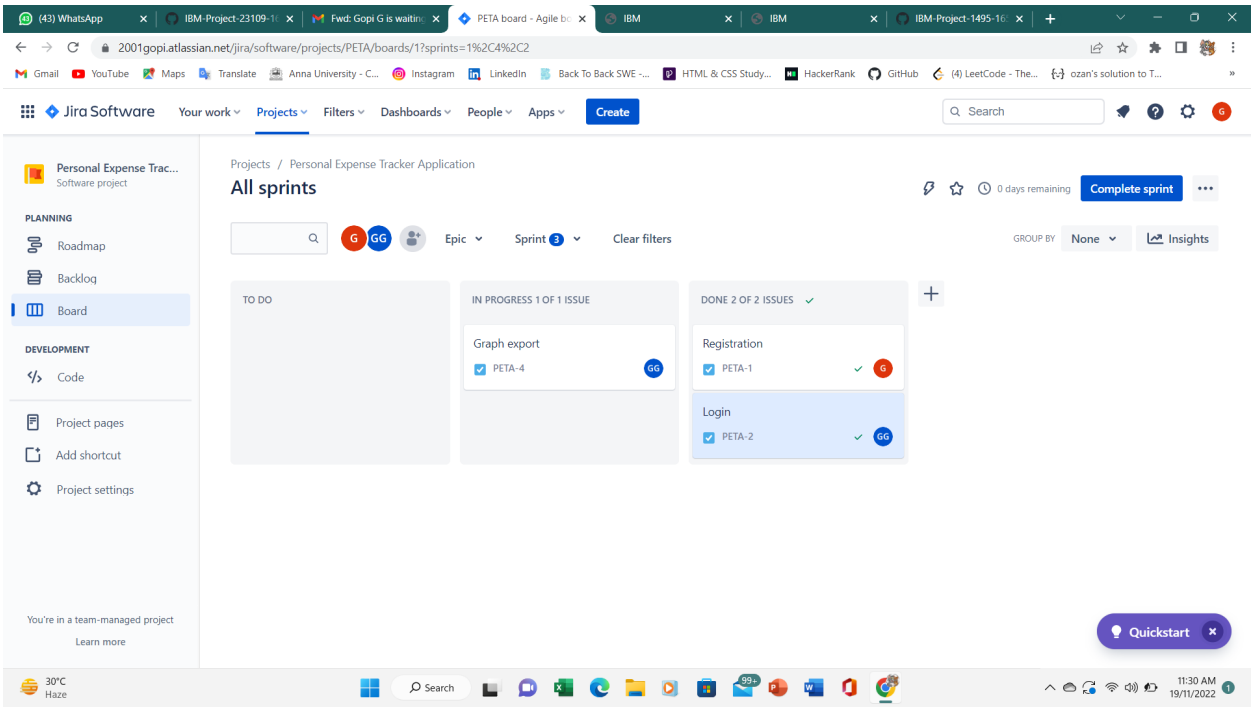
6.2. Sprint Delivery Schedule

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

6.3. Reports from JIRA


2001gopi.atlassian.net/jira/software/projects/PETA/boards/1/backlog
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Personal Expense Trac...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / Personal Expense Tracker Application
Backlog
Epic
Insights
PETA Sprint 1 23 Oct – 28 Oct (1 issue)
Complete sprint
PETA-1 Registration
DONE
Create issue
PETA Sprint 2 30 Oct – 4 Nov (1 issue)
Complete sprint
PETA-2 Login
DONE
Create issue
PETA Sprint 4 13 Nov – 19 Nov (1 issue)
Complete sprint
PETA-4 Graph export
IN PROGRESS
Create issue
Quickstart
30°C
Haze
Search
11:30 AM
19/11/2022


2001gopi.atlassian.net/jira/software/projects/PETA/boards/1/roadmap
OCT
OCT
NOV
NOV
NOV
Sprints
PETA Sprint 1
PETA Sprint 2
PETA Sprint 3
PETA Sprint 4
PETA-5 Sprint 1
PETA-6 Sprint 2
PETA-7 Sprint 3
PETA-8 Sprint 4
Create Epic
Today
Weeks
Months
Quarters
Quickstart
30°C
Haze
Search
11:31 AM
19/11/2022


(43) WhatsApp
IBM-Project-23109-1
Fwd: Gopi G is waiting
PETA board - Agile b
IBM
IBM
IBM-Project-1495-16
2001gopi.atlassian.net/jira/software/projects/PETA/boards/1?sprints=1%2C4%2C2
Gmail
YouTube
Maps
Translate
Anna University - C...
Instagram
LinkedIn
Back To Back SWE -...
HTML & CSS Study...
HackerRank
GitHub
(4) LeetCode - The...
ozan's solution to T...
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Personal Expense Trac...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / Personal Expense Tracker Application
All sprints
0 days remaining
Complete sprint
Epic
Sprint 3
Clear filters
GROUP BY
None
Insights
TO DO
IN PROGRESS 1 OF 1 ISSUE
DONE 2 OF 2 ISSUES
Graph export
PETA-4
Registration
PETA-1
Login
PETA-2
Quickstart
30°C
Haze
Search
11:30 AM
19/11/2022

7. CODING & SOLUTIONING

7.1. Feature 1

In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management.

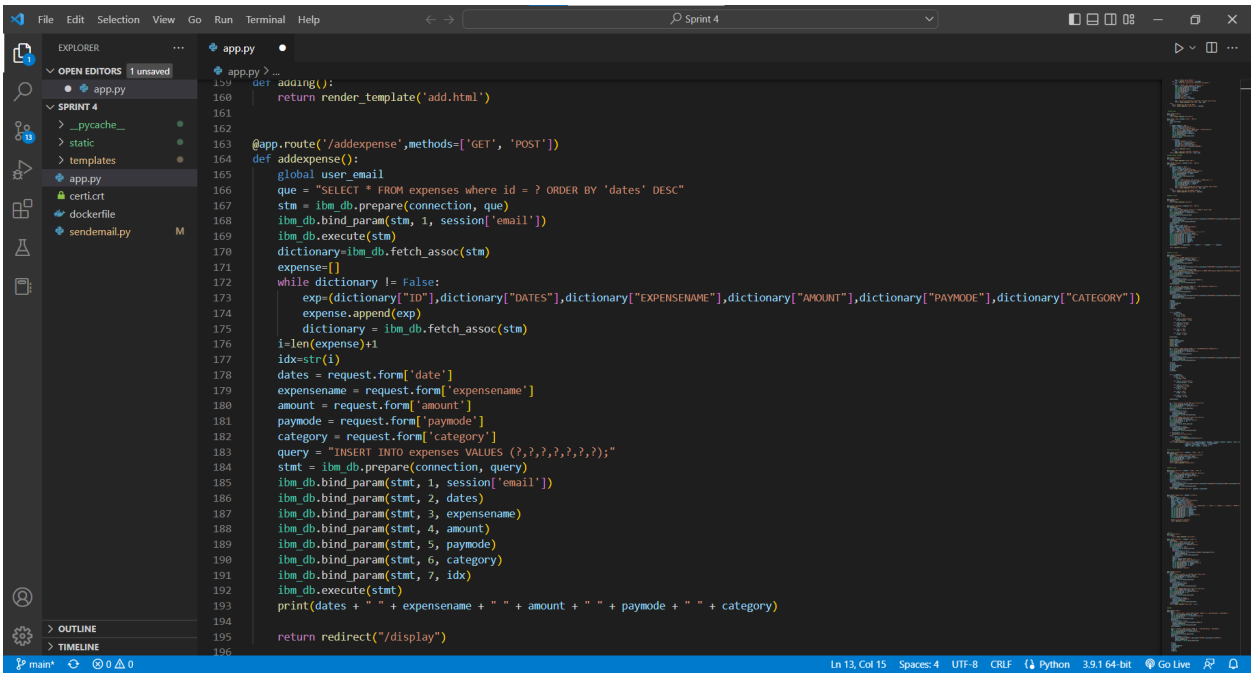
Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that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

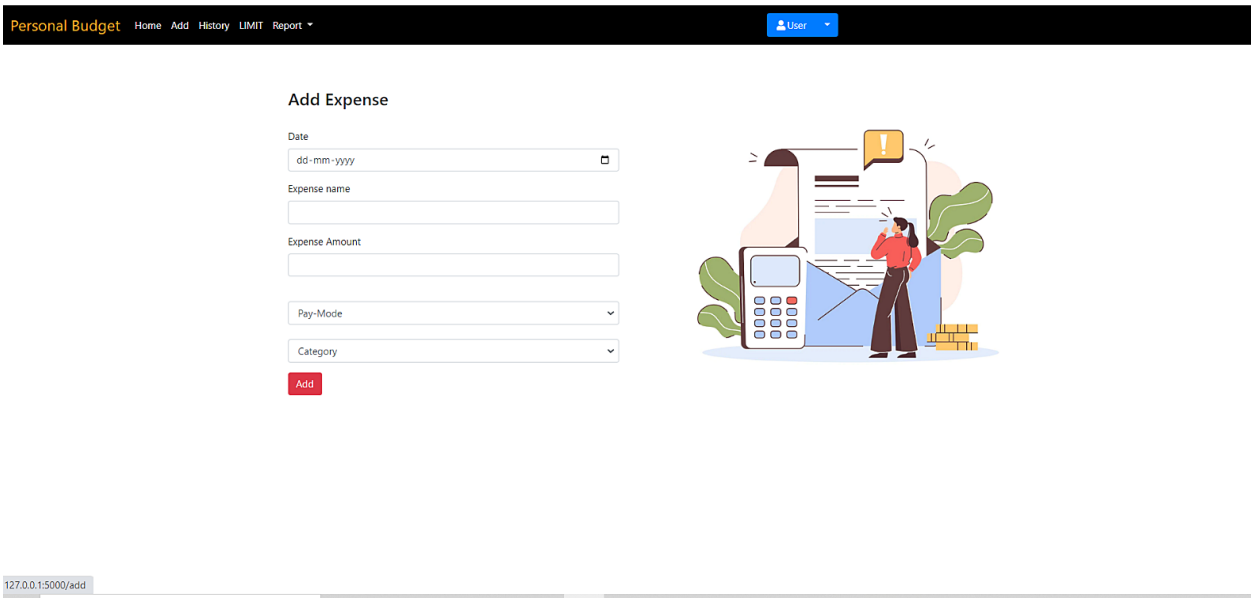
Software Required

- Python
- Flask
- Docker
- Kubernetes
- IBM DB2

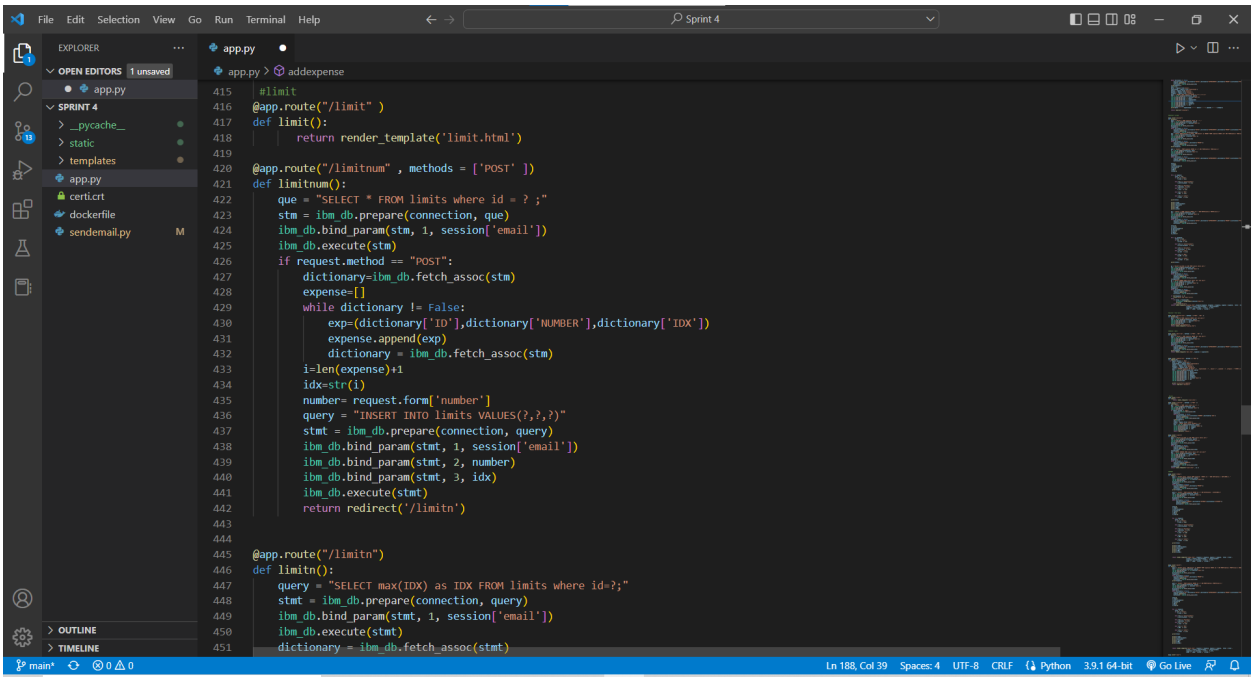
System Required

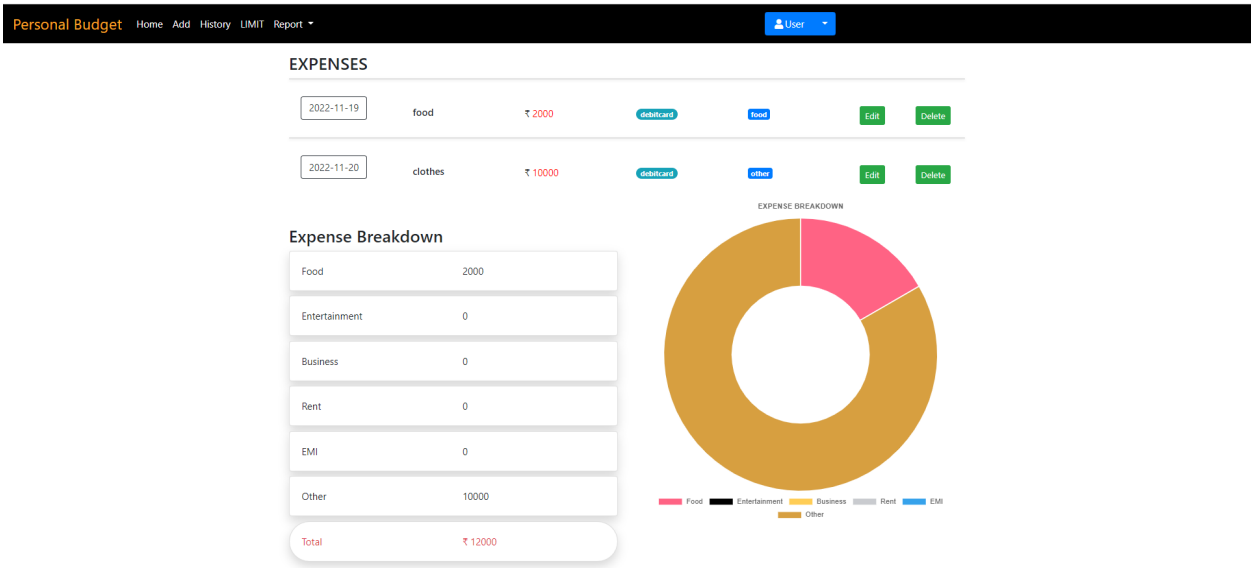
- 8GB RAM.
- Intel Core i3.
- OS-Windows/Linux/MAC .
- Laptop or Desktop

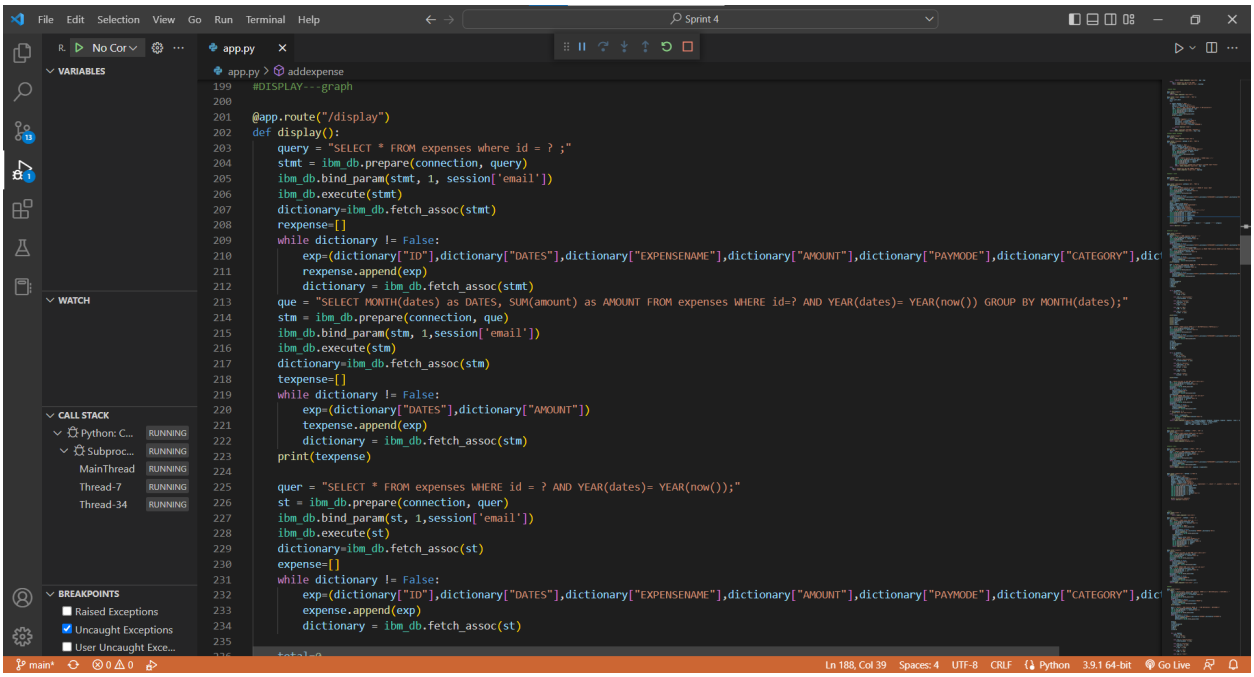




127.0.0.1:5000/add


File Edit Selection View Go Run Terminal Help
Sprint 4
EXPLORER
OPEN EDITORS 1 unsaved
app.py
SPRINT 4
__pycache__
static
templates
app.py
certi.crt
dockerfile
sendemail.py
app.py
app.py > addexpense
#limit
@app.route("/limit")
def limit():
return render_template('limit.html')
@app.route("/limitnum" , methods = ['POST'])
def limitnum():
que = "SELECT * FROM limits where id = ? ;"
stm = ibm_db.prepare(connection, que)
ibm_db.bind_param(stm, 1, session['email'])
ibm_db.execute(stm)
if request.method == "POST":
dictionary=ibm_db.fetch_assoc(stm)
expense=[]
while dictionary != False:
exp=(dictionary['ID'],dictionary['NUMBER'],dictionary['IDX'])
expense.append(exp)
dictionary = ibm_db.fetch_assoc(stm)
i=len(expense)+1
idx=str(i)
number= request.form['number']
query = "INSERT INTO limits VALUES(?,?,?)"
stmt = ibm_db.prepare(connection, query)
ibm_db.bind_param(stmt, 1, session['email'])
ibm_db.bind_param(stmt, 2, number)
ibm_db.bind_param(stmt, 3, idx)
ibm_db.execute(stmt)
return redirect('/limitn')
@app.route("/limitn")
def limitn():
query = "SELECT max(IDX) as IDX FROM limits where id=?;"
stmt = ibm_db.prepare(connection, query)
ibm_db.bind_param(stmt, 1, session['email'])
ibm_db.execute(stmt)
dictionary = ibm_db.fetch_assoc(stmt)
OUTLINE
TIMELINE
main*
Ln 188, Col 39 Spaces: 4 UTF-8 CRLF Python 3.9.1 64-bit Go Live


Personal Budget
Home
Add
History
LIMIT
Report
User
EXPENSES
2022-11-19
food
₹ 2000
debitcard
food
Edit
Delete
2022-11-20
clothes
₹ 10000
debitcard
other
Edit
Delete
Expense Breakdown
Food 2000
Entertainment 0
Business 0
Rent 0
EMI 0
Other 10000
Total ₹ 12000
EXPENSE BREAKDOWN
Food
Entertainment
Business
Rent
EMI
Other


File Edit Selection View Go Run Terminal Help
Sprint 4
app.py
app.py > addexpense
VARIABLES
WATCH
CALL STACK
Python: C... RUNNING
Subproc... RUNNING
MainThread RUNNING
Thread-7 RUNNING
Thread-34 RUNNING
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exce...
#DISPLAY---graph
@app.route("/display")
def display():
query = "SELECT * FROM expenses where id = ? ;"
stmt = ibm_db.prepare(connection, query)
ibm_db.bind_param(stmt, 1, session['email'])
ibm_db.execute(stmt)
dictionary=ibm_db.fetch_assoc(stmt)
rexpense=[]
while dictionary != False:
exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dic
rexpense.append(exp)
dictionary = ibm_db.fetch_assoc(stmt)
que = "SELECT MONTH(dates) as DATES, SUM(amount) as AMOUNT FROM expenses WHERE id=? AND YEAR(dates)= YEAR(now()) GROUP BY MONTH(dates);"
stm = ibm_db.prepare(connection, que)
ibm_db.bind_param(stm, 1,session['email'])
ibm_db.execute(stm)
dictionary=ibm_db.fetch_assoc(stm)
texpense=[]
while dictionary != False:
exp=(dictionary["DATES"],dictionary["AMOUNT"])
texpense.append(exp)
dictionary = ibm_db.fetch_assoc(stm)
print(texpense)
quer = "SELECT * FROM expenses WHERE id = ? AND YEAR(dates)= YEAR(now());"
st = ibm_db.prepare(connection, quer)
ibm_db.bind_param(st, 1,session['email'])
ibm_db.execute(st)
dictionary=ibm_db.fetch_assoc(st)
expense=[]
while dictionary != False:
exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dic
expense.append(exp)
dictionary = ibm_db.fetch_assoc(st)
main*
Ln 188, Col 39 Spaces: 4 UTF-8 CRLF Python 3.9.1 64-bit Go Live


Personal Budget
Home
Add
History
LIMIT
Report
User
Currently your MONTHLY limit is ₹ 4355
ENTER the MONTHLY LIMIT to avoid over EXPENSES
Submit

7.2. Feature 2

Tracking your expenses can save your amount, but it can also help you set and work for financial goals for the future. If you know exactly where your amount is going every month, you can easily see where some cutbacks and compromises can be made and are possible. The project what we have developed words more efficient than the other available income and expense tracker. The project successfully avoids the manual calculation for avoiding calculation the income and expense per month and save time of user. The modules are developed with efficient, reliable and also in an attractive manner.

8. TESTING

8.1. Test case

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
1	Functional	Login Page	Verify user is able to Login into the Application		1) Open the Personal expense tracker application. 2) Login with user Credentials 3) Verify logged in to user account	Username: Varun Password: test	Login Successful	Working as expected	Pass		N		Gopi
2	Functional	Signup Page	Verify user is able to Signup in the Application		1) Open thePersonal expense tracker 2) Enter theDetails and Create a new User 3) Verify if useris created and inserted into DB Table	Username: Sudharshan Password: test Name: Ayshu	Account Created Successfully	Working as expected	Pass		N		Ganapathi
3	Functional	Dashboard page	Verify if all the user details are stored in Database		1) Open thePersonal expense tracker application. 2) Enter theDetails and Create a new User 3) Verify if useris created and inserted into DB Table	Username: Sakthi@gmail.com password: Testing123	User should navigate to user account homepage	Working as expected	Pass				Jayaprakash
4	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL and click go 2.Click on Sign IN button 3.Enter InValid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: chalam@gmail password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass				Gopi
5	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL and click go 2.Click on Sign IN button 3.Enter InValid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: chalam@gmail.com password: Testing123678686786876876	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass				Anbuselvam

8.2. User Accepatance Testing

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

2.Defect Analysis

This reportshows the numberof resolved or closed bugs at each severity level, and howthey were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	1	0	0	0	1
Duplicate	1	0	0	0	1
External	3	1	0	0	4
Fixed	4	1	0	0	5
Not Reproduced	0	0	0	0	1
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	9	2	0	0	11

3.Testcase Analysis

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	0	0	0	0
Client Application	5	0	0	5
Security	0	0	0	0
Outsource Shipping	0	0	0	0
Exception Reporting	5	0	0	5
Final Report Output	0	0	0	0
Version Control	0	0	0	0

9. RESULT

9.1. Performance Metrics

			NFT - Risk Assessment				
S.No	Project Name	Scope/feature	Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes
1	Personal Expense Tracker Application	New	Low	No Changes	Moderate	Yes, 2hrs	>10 to 30%

			NFT - Detailed Test Plan				
			S.No	Project Overview	NFT Test approach	Assumptions/Dependencies/Risks	Approvals/SignOff
			1	Login Page	1) Open the Personal Expense Tracker Application 2) Login with user Credentials	No Risks	N/A
			2	Signup Page	1) Open the Personal Expense Tracker Application 2) Enter the Details and Create a new User	No Risks	N/A
			3	Records Page	1) Log in to Personal Expense Tracker Application 2) Enter all the pesonal details and expenses and mark it as expense or income	No Risks	N/A
			4	Dashboard	1) Log in to Personal Expense Tracker Application 2) View the Analytics	No Risks	N/A
			5	Bills Page	1) Log in to Personal Expense Tracker Application 2) Bills can be added.	No Risks	N/A
			5	Email Acknowledgement	1) Mails are Sent to the Registered user if expenses>budget	No Risks	N/A

			End Of Test Report				
S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)
1	Personal Expense Tracker Application	1) Log in to Personal Expense Tracker Application 2) Test for all Testcases 3) Log out to Personal Expense Tracker Application	YES	Test Passed	GO/NO-GO decision	N/A	None

10. ADVANTAGES & DISADVANTAGES

10.1. Advantages

Knowing what you spend will help you to:

- Create a monthly budget
- Know where you're spending more than you actually think you are
- Figure out ways to cut back on your spending
- Know how much extra payments you can make towards your debt
- Plan for future large purchases
- Create a savings plan for putting money away every month
- Plan for retirement
- Create an investment strategy with extra money

10.2. Disadvantages

- Having security issues
- Need to update often
- Lots of manual work

11. CONCLUSION

Tracking your expenses can save your amount, but it can also help you set and work for financial goals for the future. If you know exactly where your amount is going every month, you can easily see where some cutbacks and compromises can be made and are possible. The project what we have developed words more efficient than the other available income and expense tracker. The project successfully avoids the manual calculation for avoiding calculation the income and expense per month and save time of user. The modules are developed with efficient, reliable and also in an attractive manner.

12. FUTURE SCOPE

Provision to add different currencies will be added so that this application is not just limited to USA but also can be used worldwide and the currency converters will be designed and added in order to convert the different currency rates. In order to make it more user friendly and less user intensive, when the user tries to add the same category or vendor to an expense/income record, a duplicate alert will be presented showing the same category/vendor which the user entered previously for some expense/income and then he can tap on it and the entries will be automatically filled for the current record.

13. APPENDIX

13.1. Source Code

app.py

```
from flask import Flask, render_template, request, redirect, session ,url_for
import ibm_db
import re
import sendemail

app = Flask(__name__)

hostname = '1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;'
uid = 'vjx02808'
pwd = 'ZheXo0qLEishDDb0'
driver = "{IBM DB2 ODBC DRIVER}"
db_name = 'Bludb'
port = '32286'
protocol = 'TCPIP'
cert = "certi.crt"
dsn = (
```

```
    "DATABASE ={0};"
    "HOSTNAME ={1};"
    "PORT ={2};"
    "UID ={3};"
    "SECURITY=SSL;"
    "PROTOCOL={4};"
    "PWD ={6};"
).format(db_name, hostname, port, uid, protocol, cert, pwd)
connection = ibm_db.connect(dsn, "", "")
app.secret_key = 'a'


#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():
    global user_email
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
```

```
        query = "SELECT * FROM register WHERE email=?;"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account 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:
            query = "INSERT INTO register values(?,?,?);"
            stmt = ibm_db.prepare(connection, query)
            ibm_db.bind_param(stmt, 1, username)
            ibm_db.bind_param(stmt, 2, email)
            ibm_db.bind_param(stmt, 3, password)
            ibm_db.execute(stmt)
            session['loggedin'] = True
            session['id'] = email
            user_email = email
            session['email'] = email
            session['username'] = username

                msg = 'You have successfully registered ! Proceed
Login Process'
            return render_template('login.html', msg = msg)
    else:
        msg = 'PLEASE FILL OUT OF THE FORM'
        return render_template('register.html', msg=msg)



 #LOGIN--PAGE

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

```
95
96 @app.route('/login',methods =['GET', 'POST'])
97 def login():
98     global user_email
99     msg = ''
100
101       if request.method == 'POST' :
102           email = request.form['email']
103           password = request.form['password']
104               sql = "SELECT * FROM register WHERE email =? AND
   password=?;"
105           stmt = ibm_db.prepare(connection, sql)
106           ibm_db.bind_param(stmt,1,email)
107           ibm_db.bind_param(stmt,2,password)
108           ibm_db.execute(stmt)
109           account = ibm_db.fetch_assoc(stmt)
110           print (account)
111
112           if account:
113               session['loggedin'] = True
114               session['id'] = account['EMAIL']
115               user_email=  account['EMAIL']
116               session['email']=account['EMAIL']
117               session['username'] = account['USERNAME']
118
119               return redirect('/home')
120           else:
121               msg = 'Incorrect username / password !'
122       return render_template('login.html', msg = msg)
123
124   #CHANGE FORGOT PASSWORD
125
126   @app.route("/forgot")
127   def forgot():
128       return render_template('forgot.html')
129
130   @app.route("/forgotpw", methods =['GET', 'POST'])
131   def forgotpw():
132       msg = ''
133       if request.method == 'POST' :
```

```
134           email = request.form['email']
135           password = request.form['password']
136           query = "SELECT * FROM register WHERE email=?;"
137           stmt = ibm_db.prepare(connection, query)
138           ibm_db.bind_param(stmt, 1, email)
139           ibm_db.execute(stmt)
140           account = ibm_db.fetch_assoc(stmt)
141           print(account)
142           if account:
143                   query = "UPDATE register SET password = ? WHERE
    email = ?;"
144                stmt = ibm_db.prepare(connection, query)
145                ibm_db.bind_param(stmt, 1, password)
146                ibm_db.bind_param(stmt, 2, email)
147                ibm_db.execute(stmt)
148                 msg = 'Successfully changed your password ! Proceed
    Login Process'
149                return render_template('login.html', msg = msg)
150       else:
151           msg = 'PLEASE FILL OUT THE CORRECT DETAILS'
152           return render_template('forgot.html', msg=msg)
153
154
155   #ADDING----DATA
156
157
158   @app.route("/add")
159   def adding():
160       return render_template('add.html')
161
162
163   @app.route('/addexpense',methods=['GET', 'POST'])
164   def addexpense():
165       global user_email
166        que = "SELECT * FROM expenses where id = ? ORDER BY 'dates'
    DESC"
167       stm = ibm_db.prepare(connection, que)
168       ibm_db.bind_param(stm, 1, session['email'])
169       ibm_db.execute(stm)
170       dictionary=ibm_db.fetch_assoc(stm)
```

```
171       expense=[]
172       while dictionary != False:
173 
   exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"
   ],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"
   ])
174           expense.append(exp)
175           dictionary = ibm_db.fetch_assoc(stm)
176       i=len(expense)+1
177       idx=str(i)
178       dates = request.form['date']
179       expensename = request.form['expensename']
180       amount = request.form['amount']
181       paymode = request.form['paymode']
182       category = request.form['category']
183       query = "INSERT INTO expenses VALUES (?,?,?,?,?,?,?);"
184       stmt = ibm_db.prepare(connection, query)
185       ibm_db.bind_param(stmt, 1, session['email'])
186       ibm_db.bind_param(stmt, 2, dates)
187       ibm_db.bind_param(stmt, 3, expensename)
188       ibm_db.bind_param(stmt, 4, amount)
189       ibm_db.bind_param(stmt, 5, paymode)
190       ibm_db.bind_param(stmt, 6, category)
191       ibm_db.bind_param(stmt, 7, idx)
192       ibm_db.execute(stmt)
193        print(dates + " " + expensename + " " + amount + " " +
   paymode + " " + category)
194 
195       return redirect("/display")
196 
197 
198 
199   #DISPLAY---graph
200 
201   @app.route("/display")
202   def display():
203       query = "SELECT * FROM expenses where id = ? ;"
204       stmt = ibm_db.prepare(connection, query)
205       ibm_db.bind_param(stmt, 1, session['email'])
206       ibm_db.execute(stmt)
```

```
        dictionary=ibm_db.fetch_assoc(stmt)
        rexpense=[]
        while dictionary != False:
    exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"
    ],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"
    ],dictionary["IDX"])
            rexpense.append(exp)
            dictionary = ibm_db.fetch_assoc(stmt)
         que = "SELECT MONTH(dates) as DATES, SUM(amount) as AMOUNT
    FROM expenses WHERE id=? AND YEAR(dates)= YEAR(now()) GROUP BY
    MONTH(dates);"
        stm = ibm_db.prepare(connection, que)
        ibm_db.bind_param(stm, 1,session['email'])
        ibm_db.execute(stm)
        dictionary=ibm_db.fetch_assoc(stm)
        texpense=[]
        while dictionary != False:
            exp=(dictionary["DATES"],dictionary["AMOUNT"])
            texpense.append(exp)
            dictionary = ibm_db.fetch_assoc(stm)
        print(texpense)

        quer = "SELECT * FROM expenses WHERE id = ? AND YEAR(dates)=
    YEAR(now());"
        st = ibm_db.prepare(connection, quer)
        ibm_db.bind_param(st, 1,session['email'])
        ibm_db.execute(st)
        dictionary=ibm_db.fetch_assoc(st)
        expense=[]
        while dictionary != False:
    exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"
    ],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"
    ],dictionary["IDX"])
            expense.append(exp)
            dictionary = ibm_db.fetch_assoc(st)

        total=0
        t_food=0
```

```
238       t_entertainment=0
239       t_business=0
240       t_rent=0
241       t_EMI=0
242       t_other=0
243
244
245       for x in expense:
246             total += x[3]
247             if x[5] == "food":
248                 t_food += x[3]
249
250             elif x[5] == "entertainment":
251                 t_entertainment  += x[3]
252
253             elif x[5] == "business":
254                 t_business  += x[3]
255             elif x[5] == "rent":
256                 t_rent  += x[3]
257
258             elif x[5] == "EMI":
259                 t_EMI  += x[3]
260
261             elif x[5] == "other":
262                 t_other  += x[3]
263
264       print(total)
265
266       print(t_food)
267       print(t_entertainment)
268       print(t_business)
269       print(t_rent)
270       print(t_EMI)
271       print(t_other)
272
273       qur = "SELECT * FROM expenses WHERE id = ? AND MONTH(dates)=
    MONTH(now());"
274       stt = ibm_db.prepare(connection, qur)
275       ibm_db.bind_param(stt, 1, session['email'])
276       ibm_db.execute(stt)
```

```
277       dictionary=ibm_db.fetch_assoc(stt)
278       lexpense=[]
279       while dictionary != False:
280
   exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"
   ],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"
   ],dictionary["IDX"])
281           lexpense.append(exp)
282           dictionary = ibm_db.fetch_assoc(stt)
283
284       ttotal=0
285       to_food=0
286       to_entertainment=0
287       to_business=0
288       to_rent=0
289       to_EMI=0
290       to_other=0
291
292
293       for x in lexpense:
294             ttotal += x[3]
295             if x[5] == "food":
296                 to_food += x[3]
297
298             elif x[5] == "entertainment":
299                 to_entertainment  += x[3]
300
301             elif x[5] == "business":
302                 to_business  += x[3]
303             elif x[5] == "rent":
304                 to_rent  += x[3]
305
306             elif x[5] == "EMI":
307                 to_EMI  += x[3]
308
309             elif x[5] == "other":
310                 to_other  += x[3]
311
312       print(ttotal)
313
```

```
314
315       qy = "SELECT max(IDX) as IDX FROM limits where id=?;"
316       smt = ibm_db.prepare(connection, qy)
317       ibm_db.bind_param(smt, 1, session['email'])
318       ibm_db.execute(smt)
319       dictionary = ibm_db.fetch_assoc(smt)
320       uexpense=[]
321       while dictionary != False:
322           exp=(dictionary["IDX"])
323           uexpense.append(exp)
324           dictionary = ibm_db.fetch_assoc(smt)
325       k=uexpense[0]
326       qu = "SELECT NUMBER FROM limits where id=? and idx=?"
327       sm = ibm_db.prepare(connection, qu)
328       ibm_db.bind_param(sm, 1, session['email'])
329       ibm_db.bind_param(sm, 2, k)
330       ibm_db.execute(sm)
331       dictionary = ibm_db.fetch_assoc(sm)
332       fexpense=[]
333       while dictionary != False:
334           exp=(dictionary["NUMBER"])
335           fexpense.append(exp)
336           dictionary = ibm_db.fetch_assoc(stmt)
337
338       if len(fexpense) <= 0:
339           print("Enter the limit First")
340       else:
341           if ttotal > fexpense[0]:
342               m=sendemail.sendgridmail(session["email"])
343               print(m)
344           else: print("Error")
345           return  render_template("display.html",rexpense=rexpense,
   texpense = texpense, expense = expense,  total = total ,
346                                   t_food = t_food,t_entertainment =
   t_entertainment,
347                                  t_business = t_business,  t_rent =
   t_rent,
348                                t_EMI =  t_EMI,  t_other =  t_other )
349
350
```

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

@app.route('/delete/<idx>', methods = ['POST', 'GET' ])
def delete(idx):
    query = "DELETE FROM expenses WHERE id=? and idx=?;"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, session["email"])
    ibm_db.bind_param(stmt, 2, idx)
    ibm_db.execute(stmt)
    print('deleted successfully')
    return render_template("display.html")



#UPDATE---DATA

@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    query = "SELECT * FROM expenses WHERE id=? and idx=?;"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, session['email'])
    ibm_db.bind_param(stmt, 2, id)
    ibm_db.execute(stmt)
    dictionary=ibm_db.fetch_assoc(stmt)
    expense=[]
    while dictionary != False:

exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"
],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"
],dictionary["IDX"])
        expense.append(exp)
        dictionary = ibm_db.fetch_assoc(stmt)
    print(expense)
    return render_template('edit.html', expenses = expense[0])



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

```
388     if request.method == 'POST' :
389         dates = request.form['date']
390         expensename = request.form['expensename']
391         amount = request.form['amount']
392         paymode = request.form['paymode']
393         category = request.form['category']
394         query = "UPDATE expenses SET dates = ? , expensename = ? ,
    amount = ?, paymode = ?, category = ? WHERE id = ? and idx=?;"
395         stmt = ibm_db.prepare(connection, query)
396         ibm_db.bind_param(stmt, 1, dates)
397         ibm_db.bind_param(stmt, 2, expensename)
398         ibm_db.bind_param(stmt, 3, amount)
399         ibm_db.bind_param(stmt, 4, paymode)
400         ibm_db.bind_param(stmt, 5, category)
401         ibm_db.bind_param(stmt, 6, session['email'])
402         ibm_db.bind_param(stmt, 7, id)
403         ibm_db.execute(stmt)
404 
405         print('successfully updated')
406         return redirect("/display")
407 
408 
409 
410 
411 
412 
413 
414 
415     #limit
416   @app.route("/limit" )
417   def limit():
418           return render_template('limit.html')
419 
420   @app.route("/limitnum" , methods = ['POST' ])
421   def limitnum():
422       que = "SELECT * FROM limits where id = ? ;"
423       stm = ibm_db.prepare(connection, que)
424       ibm_db.bind_param(stm, 1, session['email'])
425       ibm_db.execute(stm)
426       if request.method == "POST":
```

```
427           dictionary=ibm_db.fetch_assoc(stm)
428           expense=[]
429           while dictionary != False:
430
   exp=(dictionary['ID'],dictionary['NUMBER'],dictionary['IDX'])
431               expense.append(exp)
432               dictionary = ibm_db.fetch_assoc(stm)
433           i=len(expense)+1
434           idx=str(i)
435           number= request.form['number']
436           query = "INSERT INTO limits VALUES(?,?,?)"
437           stmt = ibm_db.prepare(connection, query)
438           ibm_db.bind_param(stmt, 1, session['email'])
439           ibm_db.bind_param(stmt, 2, number)
440           ibm_db.bind_param(stmt, 3, idx)
441           ibm_db.execute(stmt)
442           return redirect('/limitn')
443
444
445   @app.route("/limitn")
446   def limitn():
447       query = "SELECT max(IDX) as IDX FROM limits where id=?;"
448       stmt = ibm_db.prepare(connection, query)
449       ibm_db.bind_param(stmt, 1, session['email'])
450       ibm_db.execute(stmt)
451       dictionary = ibm_db.fetch_assoc(stmt)
452       expense=[]
453       while dictionary != False:
454           exp=(dictionary["IDX"])
455           expense.append(exp)
456           dictionary = ibm_db.fetch_assoc(stmt)
457       k=expense[0]
458       que = "SELECT NUMBER FROM limits where id=? and idx=?"
459       stmt = ibm_db.prepare(connection, que)
460       ibm_db.bind_param(stmt, 1, session['email'])
461       ibm_db.bind_param(stmt, 2, k)
462       ibm_db.execute(stmt)
463       dictionary = ibm_db.fetch_assoc(stmt)
464       texpense=[]
465       while dictionary != False:
```

```
466           exp=(dictionary["NUMBER"])
467           texpense.append(exp)
468           dictionary = ibm_db.fetch_assoc(stmt)
469       s=texpense[0]
470       return render_template("limit.html" , y= s)
471
472
473   #REPORT
474
475   @app.route("/today")
476   def today():
477           query = "SELECT dates, amount FROM expenses  WHERE id = ?
    AND DATE(dates) = DATE(NOW()); "
478           stmt = ibm_db.prepare(connection, query)
479           ibm_db.bind_param(stmt, 1, str(session['email']))
480           ibm_db.execute(stmt)
481           dictionary=ibm_db.fetch_assoc(stmt)
482           texpense=[]
483           while dictionary != False:
484             exp=(dictionary["DATES"],dictionary["AMOUNT"])
485             texpense.append(exp)
486             dictionary = ibm_db.fetch_assoc(stmt)
487           print(texpense)
488
489               query = "SELECT * FROM expenses WHERE id = ? AND
    DATE(dates) = DATE(NOW())"
490           stmt = ibm_db.prepare(connection, query)
491           ibm_db.bind_param(stmt, 1, session['email'])
492           ibm_db.execute(stmt)
493           dictionary=ibm_db.fetch_assoc(stmt)
494           expense=[]
495           while dictionary != False:
496
    exp=(dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGO
    RY"])
497                 expense.append(exp)
498                 dictionary = ibm_db.fetch_assoc(stmt)
499
500           total=0
501           t_food=0
```

```
502         t_entertainment=0
503         t_business=0
504         t_rent=0
505         t_EMI=0
506         t_other=0
507 
508 
509         for x in expense:
510             total += x[0]
511             if x[2] == "food":
512                 t_food += x[0]
513 
514             elif x[2] == "entertainment":
515                 t_entertainment  += x[0]
516 
517             elif x[2] == "business":
518                 t_business  += x[0]
519             elif x[2] == "rent":
520                 t_rent  += x[0]
521 
522             elif x[2] == "EMI":
523                 t_EMI  += x[0]
524 
525             elif x[2] == "other":
526                 t_other  += x[0]
527 
528         print(total)
529 
530         print(t_food)
531         print(t_entertainment)
532         print(t_business)
533         print(t_rent)
534         print(t_EMI)
535         print(t_other)
536 
537 
538 
539          return render_template("today.html", texpense = texpense,
   expense = expense,  total = total ,
540                                 t_food = t_food,t_entertainment =
```

```
   t_entertainment,
541                                t_business = t_business,  t_rent =
   t_rent,
542                              t_EMI =  t_EMI,  t_other =  t_other )
543
544
545   @app.route("/month")
546   def month():
547         query = "SELECT dates, SUM(amount) as AMOUNT FROM expenses
   WHERE id= ? AND MONTH(dates)= MONTH(now()) GROUP BY dates ORDER BY
   dates;"
548         stmt = ibm_db.prepare(connection, query)
549         ibm_db.bind_param(stmt, 1, str(session['email']))
550         ibm_db.execute(stmt)
551         dictionary=ibm_db.fetch_assoc(stmt)
552         texpense=[]
553         while dictionary != False:
554           exp=(dictionary["DATES"],dictionary["AMOUNT"])
555           texpense.append(exp)
556           dictionary = ibm_db.fetch_assoc(stmt)
557         print(texpense)
558
559             query = "SELECT * FROM expenses WHERE id = ? AND
   MONTH(dates)= MONTH(now());"
560         stmt = ibm_db.prepare(connection, query)
561         ibm_db.bind_param(stmt, 1, session['email'])
562         ibm_db.execute(stmt)
563         dictionary=ibm_db.fetch_assoc(stmt)
564         expense=[]
565         while dictionary != False:
566
   exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"
   ],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"
   ],dictionary["IDX"])
567           expense.append(exp)
568           dictionary = ibm_db.fetch_assoc(stmt)
569
570         total=0
571         t_food=0
572         t_entertainment=0
```

```
573         t_business=0
574         t_rent=0
575         t_EMI=0
576         t_other=0
577 
578 
579         for x in expense:
580             total += x[3]
581             if x[5] == "food":
582                 t_food += x[3]
583 
584             elif x[5] == "entertainment":
585                 t_entertainment  += x[3]
586 
587             elif x[5] == "business":
588                 t_business  += x[3]
589             elif x[5] == "rent":
590                 t_rent  += x[3]
591 
592             elif x[5] == "EMI":
593                 t_EMI  += x[3]
594 
595             elif x[5] == "other":
596                 t_other  += x[3]
597 
598         print(total)
599 
600         print(t_food)
601         print(t_entertainment)
602         print(t_business)
603         print(t_rent)
604         print(t_EMI)
605         print(t_other)
606 
607 
608 
609          return render_template("month.html", texpense = texpense,
    expense = expense,  total = total ,
610                                  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():
           query = "SELECT MONTH(dates) as DATES, SUM(amount) as
  AMOUNT FROM expenses WHERE id=? AND YEAR(dates)= YEAR(now()) GROUP
  BY MONTH(dates);"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1,session['email'])
        ibm_db.execute(stmt)
        dictionary=ibm_db.fetch_assoc(stmt)
        texpense=[]
        while dictionary != False:
          exp=(dictionary["DATES"],dictionary["AMOUNT"])
          texpense.append(exp)
          dictionary = ibm_db.fetch_assoc(stmt)
        print(texpense)

            query = "SELECT * FROM expenses WHERE id = ? AND
  YEAR(dates)= YEAR(now());"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1,session['email'])
        ibm_db.execute(stmt)
        dictionary=ibm_db.fetch_assoc(stmt)
        expense=[]
        while dictionary != False:

  exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"
  ],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"
  ],dictionary["IDX"])
          expense.append(exp)
          dictionary = ibm_db.fetch_assoc(stmt)

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

```
644         t_EMI=0
645         t_other=0
646 
647 
648         for x in expense:
649             total += x[3]
650             if x[5] == "food":
651                 t_food += x[3]
652 
653             elif x[5] == "entertainment":
654                 t_entertainment  += x[3]
655 
656             elif x[5] == "business":
657                 t_business  += x[3]
658             elif x[5] == "rent":
659                 t_rent  += x[3]
660 
661             elif x[5] == "EMI":
662                 t_EMI  += x[3]
663 
664             elif x[5] == "other":
665                 t_other  += x[3]
666 
667         print(total)
668 
669         print(t_food)
670         print(t_entertainment)
671         print(t_business)
672         print(t_rent)
673         print(t_EMI)
674         print(t_other)
675 
676 
677 
678           return render_template("year.html", texpense = texpense,
    expense = expense,  total = total ,
679                                    t_food = t_food,t_entertainment =
    t_entertainment,
680                                   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)
   return render_template('home.html')


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

```

13.2. GitHub & Project Demo Link

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-1495-1658391120

Demo Video Links

Drive Link:

https://drive.google.com/file/d/1-yFL56dfBdAHzvzEc1zaH1ujG74Y7d_O/view?usp=sharing

YouTube Link:

https://youtu.be/tpUqKJgdGKA