NALAIYA THIRAN PROJECT -2022

PERSONAL EXPENSE TRACKER APPLICATION

IBM – DOCUMENTATION

UNDER THE GUIDANCE OF

INDUSTRY MENTOR	:	KUSBOO
SPOC	:	Dr. G. PRADEEP
FACULTY EVALUTOR	:	Dr. J. SUDHA
FACULTY MENTOR	:	R. RAMYA

TEAM ID : PNT2022TMID46374

Submitted by

MONIKA T	820319104024
FALILA BANU H	820319104015
MOUNIKA R	820319104025
SINDHUJA M.K	820319104039

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A.V.C. COLLEGE OF ENGINEERING

ANNA UNIVERSITY :: 2019 – 2023

ABSTRACT

Personal Expense tracker allows the user to maintain a computerized diary. This application will keep a track of expenses of a user on a day-to-day basis. This application keeps a record of your expenses and also will give you a category wise distribution of your expenses. With the help of this application user can track their daily/weekly/monthly expenses. This application will also have a feature which will help you stay on budget because you know your expenses. Expense tracker application will generate report at the end of month to show Expense via a graphical representation. We also have added a special feature which will distributes your expenses in different categories suitable for the user. An expense history will also be provided in application. Some of the concerns maintaining a personal expense is a BIG problem, in daily expenses many times we don't know where the money goes.

Some of the conventional methods used to tackle this problem in normal circumstances are like making use of a sticky notes by common users, Proficient people deals with this kind of problems by using spreadsheets to record expense and using a ledger to maintains the large amounts data by especially by expert people. As this shows that it is various methods used by different people. This makes using this data contrary. There is still complication in areas like there is no assurance for data compatible, there are chances of crucial inputs can be missed and the manual errors may sneak in.

1. INTRODUCTION

1.1 PROJECT OVERVIEW

Daily Expense Tracker System is a system which will keep a track of Income-Expense of a House-Wife on a day to day basics, This System takes Income and divides in daily expense allowed, also has a limit feature which can be used to set our monthly budget limit. If limit exceeds, then alert will be sent through mail. Daily expense tracking System will generate report at the end of month to show Income-Expense Curve. It will let you add the savings amount which you had saved for some particular Festivals or day like Birthday or Anniversary.

1.2.PURPOSE

Expense tracking is an important part of creating a budget for your small business. Keeping a daily record of your expenses by tracking receipts, invoices and other outgoing expenses improves the financial health of your budget. Tracking expenses can help you stay on top of your cash flow and prepare you for tax season.

2. LITERATURE SURVEY

2.1.Existing Problem

The Expense tracker existing system does not provide the user portable device management level, existing system only used on desktop software so unable to update anywhere expenses done and unable to update the location of the expense details disruptive that the proposed system provides [6]. In existing, we need to maintain the Excel sheets, CSV files for the user daily, weekly and monthly expenses. In existing, there is no as such complete solution to keep a track of its daily expenses easily. To do so a person as to keep a log in a diary or in a computer system, also all the calculations need to be done by the user which may sometimes results in mistakes leading to losses. The existing system is not user friendly because data is not maintained perfectly.

EXPENSE MANAGER APPLICATION

Mobile applications are top in user convenience and have overpasses the web applications in terms of popularity and usability. In this paper, we develop a mobile application developed for the android platform that keeps record of user personal expenses, his/her contribution in group expenditures, top investment options, view of the current stock market, read authenticated financial news and grab the best ongoing offers in the market in popular categories. The proposed application would eliminate messy sticky notes, spreadsheets confusion and data handling inconsistency problems while offering the best overview of your expenses. With our application can manage their expenses and decide on their budget more effectively.

link:https://www.researchgate.net/publication/347972162_Expense_Manager_Application

EXPENSE TRACKER APPLICATION

Expense tracker is an android based application. This application allows the user to maintain a computerized diary. Expense tracker application which will keep a track of Expenses of a user on a day-to-day basis. This application keeps a record of your expenses and also will give you a category wise distribution of your expenses. With the help of this application user can track their daily/weekly/monthly expenses. This application will also has a feature which will help you stay on budget because you know your expenses. Expense tracker application will generate report at the end of month to show Expense via a graphical representation. We also have added a special feature which will distributes your expenses in different categories suitable for the user. An expense history will also be provided in application.

link: https://ijirt.org/Article?manuscript=150860

2.2.References:

S.No	**AUTHOR**	**PAPER TITLE**	**YEAR**	**JOURNAL**	**FINDINGS (Technologies used)**
1.	Velmurugan A, Albert Mayan, Niranjana P and Richard Francis	**EXPENSE MANAGER APPLICATI ON**	2020	Journal-of Physics: Conference Series	1.Android Platform 2.Methodology used Android Studio, Kotlin and Java, SQLite, Android OS, Fig ma Designing Tool.
2.	Nidhi Jitendra Jadhaw,Rutuja Vijay	**EXPENSE TRACKER**	2022	International Research	1.Mobile Application 2.Using Database Layer

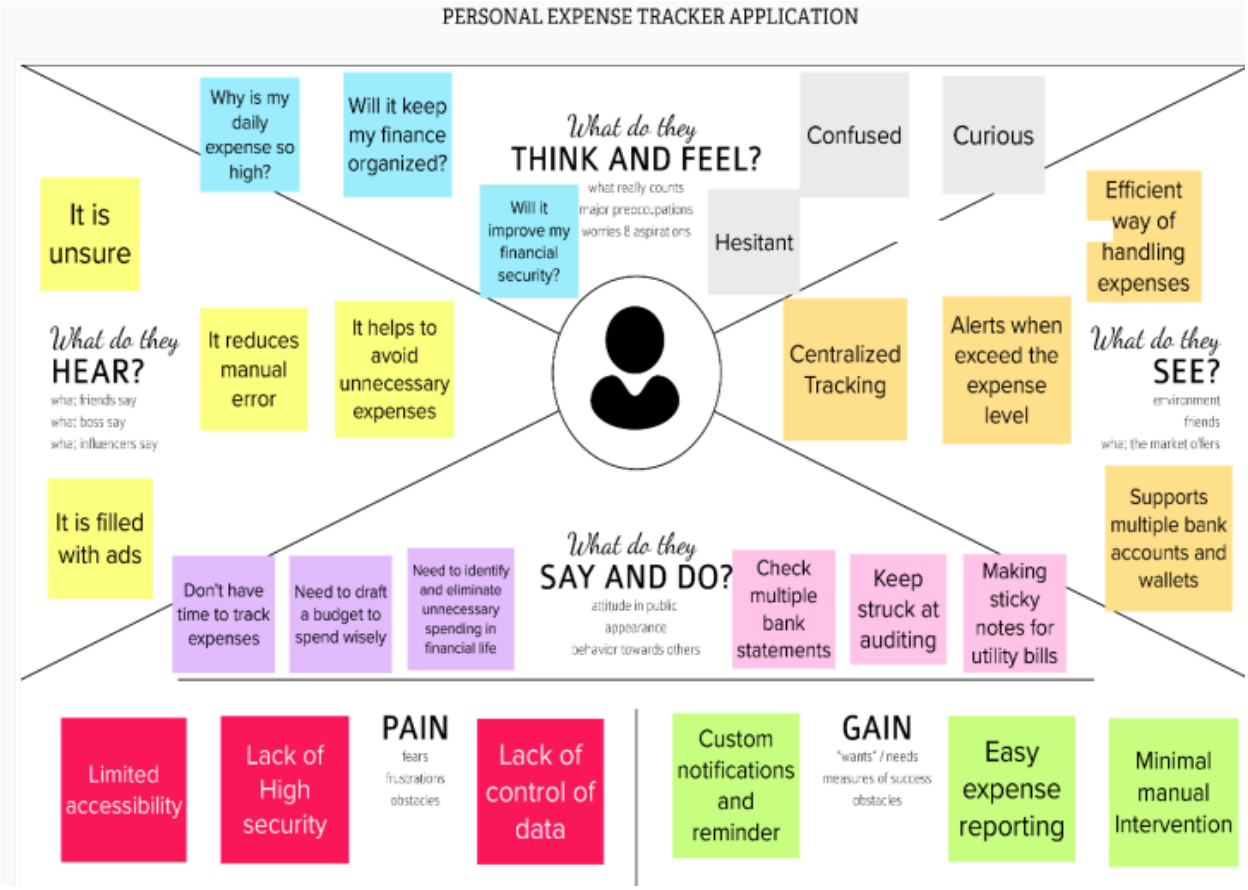
	Chakor, Trupti Mahesh, Damayanti, D.Pawar			Journal of Modernization in Engineering Technology and Science	which hold all of the data and Financial Information 3.Supported by User Interface .
3.	Dr.V.Geetha,G Nikhitha,H.Srilaya Dr.C.K.Gomathy	**EXPENDITURE MANAGEMENT**	2022	Journal of Computing &Architecture	1.It is based Web Application. 2. Android App which runs on all AndroidPlatforms.
4.	TamiaRuvimbo Masendu,Aanajey ManiTripath	**DAILY EXPENSE TRACKER**	2022	International Journal of Research in Engineering, Science and Management.	1.Technology used Java(Apache NetBeans IDE 13) and MySQL Workbench . 2.Application is Based Graphical User Interface(GUI)
5.	Hezerto,Malikberdi	**BUDGET TRACKER HIGHLY CUSTOMIZAB LE BUDGETING MOBILE APPLICATION**	2021	Master of Information Technology	1.Technology can be used React Native, Expo, Redux, Recompose, Ramda and Async Storage (Global memory of the device) 2.Visual Studio Code, Android Studio

2.3. Problem Statement Definition

Tracking expenses is one of the key factors in making budget work. At the instant, there is no as such complete solution present easily which enables a person to keep a track of its daily expenditure easily. To do so a person has to keep a log in a diary or in a computer, also all the calculations need to be done by the user which may sometimes results in errors leading to losses. Due to lack of a complete tracking system, there is a constant overload to rely on the daily entry of the expenditure.

3. IDEATION & PROPOSED SOLUTION

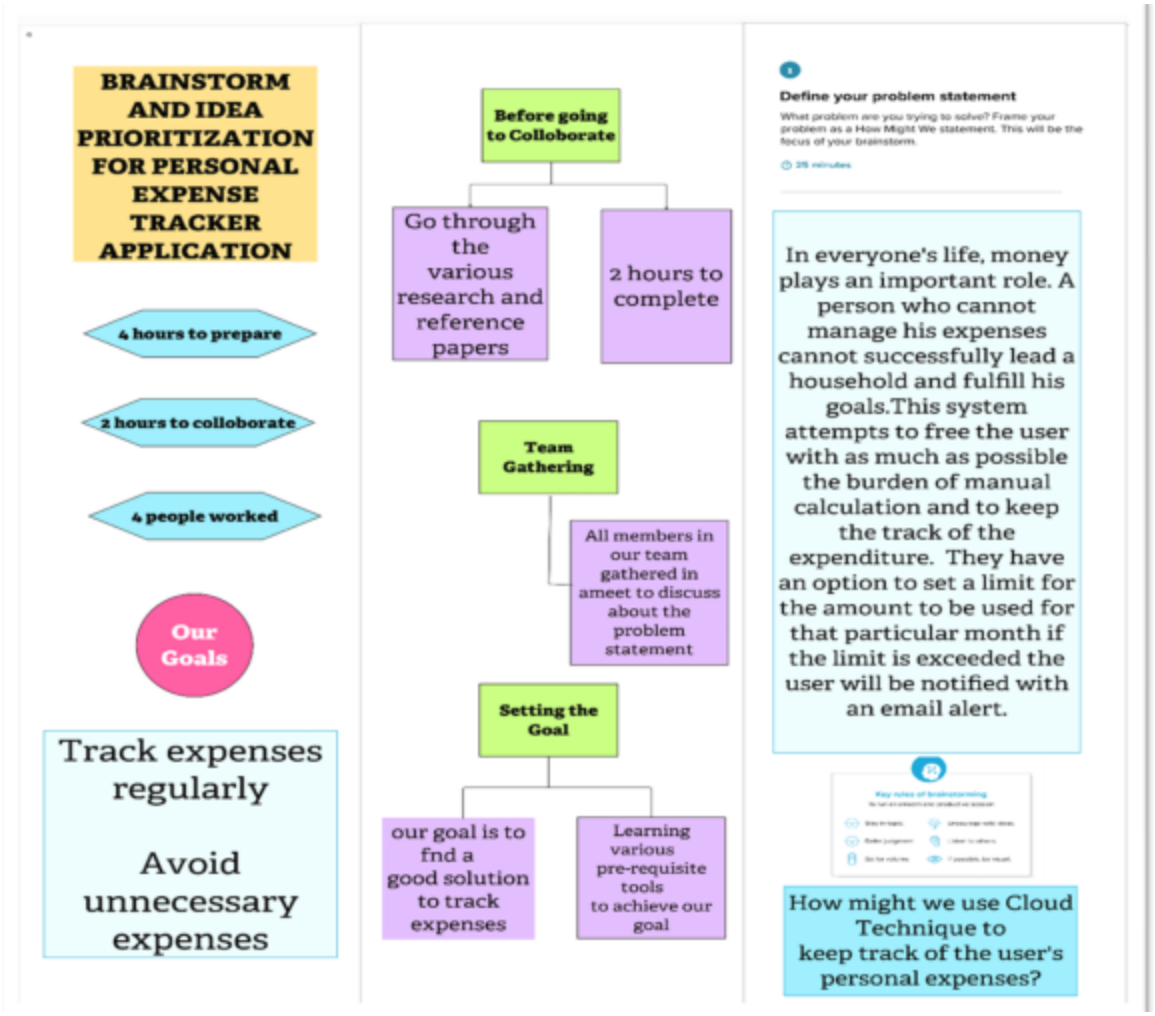
3.1.Empathy Map Canvas



3.2.Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping

Step-3: Idea Prioritization

3.3.Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Tracking expenses is one of the key factors in making budget work. At the instant, there is no as such complete solution present easily which enables a person to keep a track of its daily expenditure easily. To do so a person has to keep a log in a diary or in a computer, also all the calculations need to be done by the user which may sometimes results in errors leading to losses. Due to lack of a complete tracking system, there is a constant overload to rely on the daily entry of the expenditure.
2.	Idea / Solution description	In this project, we developed a cloud-based web application which keeps track of user's personal expenses. This system attempts to free the user with as much as possible the burden of manual calculation and to keep the track of the expenditure. This system also eliminates sticky notes, bills.
3.	Novelty / Uniqueness	This personal expense tracker Application has features that enables the user to have an option to set a limit for the amount to be used. If the limit is exceeded the user will be notified with an Email and SMS alert. This tracker doesn't have annoying ads.
4.	Social Impact / Customer Satisfaction	The user will be able to Stick to their Spending Limits. They can able to scan their bills anytime thus data loss is avoided. Users can also able to get an analysis of their expenditure in graphical forms.
5.	Business Model (Revenue Model)	This application will generate revenue by offering premium features to the user. Advertising through app is the easy way to earn money. Users may pay

		to remove the app advertisements. Through subscription the users can able to connect their bank account.
6.	Scalability of the Solution	Since this application is deployed on IBM Cloud, it can handle multiple users at a time. With our application, the users can be able to manage their expenses more effectively and know about their budget Vs income.

3.4.Problem Solution Fit

Project Design Phase-I - Solution Fit Template

Project Title: Personal Expense Tracker Application

Team ID: PNT2022TMID46374

Define CS, fit into CC

1. CUSTOMER SEGMENT(S)

- Working peoples
- Organizations
- Students and families
- Common people with all ages can able to track their expenses.

6. CUSTOMER CONSTRAINTS

- Network Issues
- Data Privacy
- Spending power
- Available devices

5. AVAILABLE SOLUTIONS

People makes use of sticky notes or diary for calculating their expenditure.

Pros:
1. Didn't need any devices for calculations.

Cons:
1. Time consuming.
2. Manual errors occur sometimes.

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS

- People have to track their expenses regularly.
- They need to keep their receipts and bills which shows their amount they spent.
- Also they need to manually add or remove the desired categories.

9. PROBLEM ROOT CAUSE

- The root cause for this problem is the delay in the budget.
- There may be a chance of getting errors in human calculations.
- No one alerts if their spending exceeds particular limit.
- They do not have enough time for calculating their expenditure.

7. BEHAVIOUR

- People should know their budget for each month and set appropriate saving goals.
- Collect receipts regularly without fail.

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

3. TRIGGERS

- Realizing that excessive spending leading to lack of money in case of emergencies.
- Lack of Budgeting knowledge.

4. EMOTIONS: BEFORE / AFTER

Before

- Excessive expenditure
- Afraid of spending

After

- Being aware of what they are spending.
- Satisfied and happy with their budget expenditure.
- There will not be any frustrations any more since the process is quick and flexible.

10. YOUR SOLUTION

- A cloud-based web application which keeps track of user's personal expenses. This system attempts to free the user with as much as possible the burden of manual calculation and to keep the track of the expenditure.
- User just need to enter their day-to-day expenses. They also have an option to set the limit. If their expenditure exceeds that limit, notification will be sent through mail.
- This system also eliminates sticky notes, bills.

8.CHANNELS OF BEHAVIOUR

ONLINE

- Provide the details of day-to-day expenses.
- Select the area where customers use.
- Maintain the expenses for budgeting.

OFFLINE

- Maintain the required documents regularly.
- Inspect the expenses for budgeting.

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 by giving appropriate details.
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP.
FR-3	User Login	Login after registering properly.
FR-4	Track Expense	Tracking the expenses regularly.
FR-5	Dashboard panel	Managing the summary of expenses and income.
FR-6	Alert Notification	If the expenses exceed the limit that the user entered, notification will be sent through mail.

4.2. Non-Functional Requirements:

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

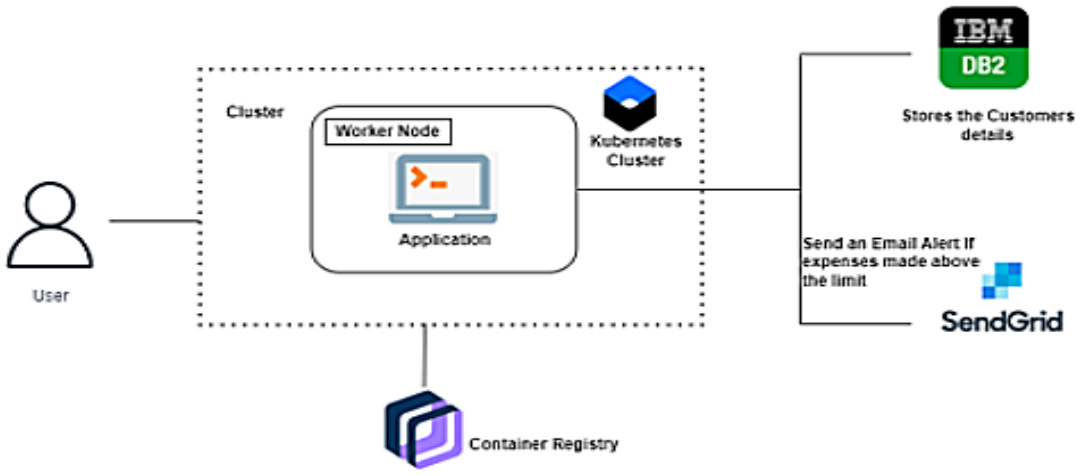
FR No.	**Non-Functional Requirement**	**Description**
NFR-1	Usability	This system will usable by anyone who are willing to manage their expenses and aims to save for future investments.
NFR-2	Security	This system assures all data inside the system or its part will be protected against malware attacks or unauthorized access.
NFR-3	Reliability	Using this report expenses, store receipts, and track spending on an individual or departmental level is a painless process.
NFR-4	Performance	Calculates the overall expenses fastly and also generate it in the form of pdf documents.
NFR-5	Availability	Users can also able to add and calculate expenses in offline mode.
NFR-6	Scalability	This system has better storage capacity and also it provides flexibility to a product to appropriately react to growth.

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.

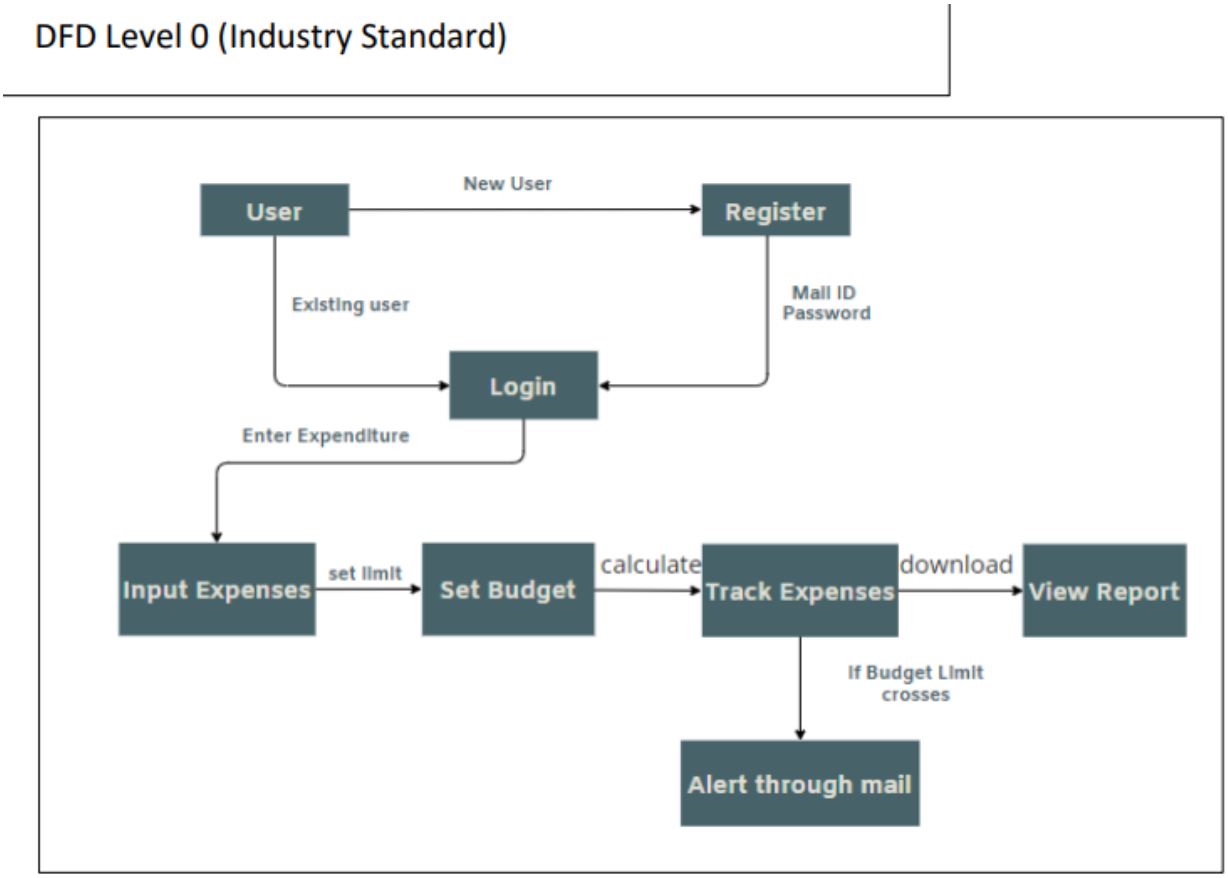
Example:(Simplified)



Project Workflow:

- The user interacts with the application.
- Registers by giving proper details.
- Enter the expenses regularly and set budget limit.
- The database will have all the details and track expenses.
- If the limit is crossed, notification will be sent through mail.

Example: DFD Level 0(Industry Standard)

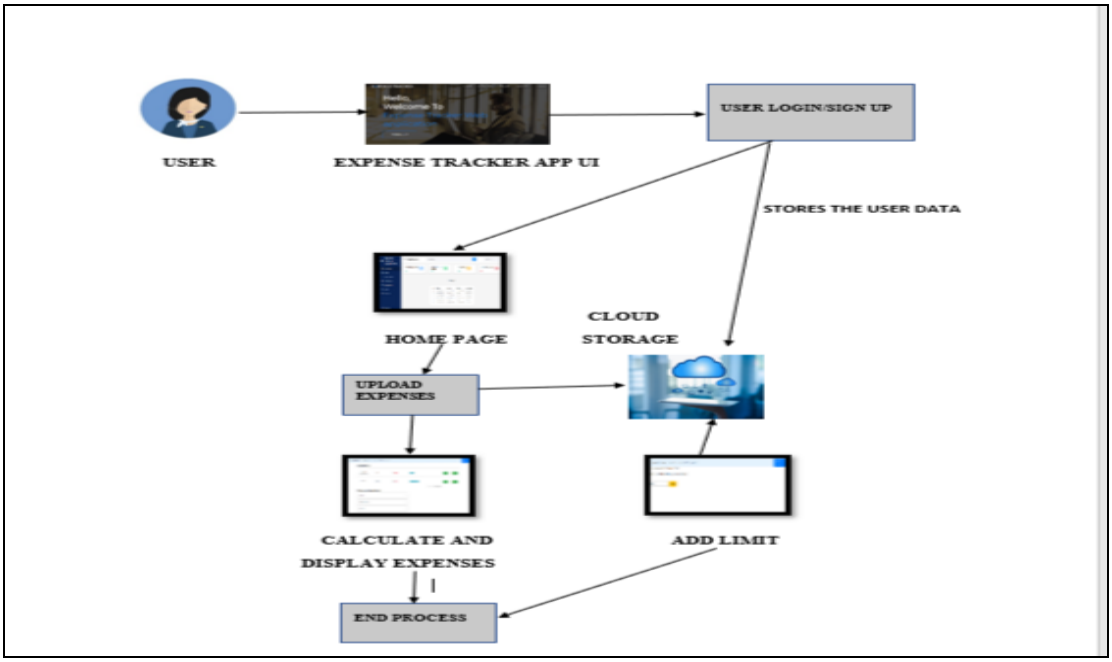


5.2.Solution &Technical Architecture:

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

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

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



5.3.User Stories:

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

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	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
	Registration	USN-2	As a user, I will receive confirmation	I can receive confirmation email & click confirm.	High	Sprint-1

			email once I have registered for the application			
	Registration	USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login.	Low	Sprint-2
	Registration	USN-4	As a user, I can register for the application through Gmail	I can register for the app through Gmail login.	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can register & access the dashboard with Gmail Login.	High	Sprint-1
	Dashboard	USN-6	As a user, I can add my day-to-day expenses regularly.	I can track my expenses perfectly.	High	Sprint-2
Customer (Web user)	Dashboard	USN-7	As a user, I can see login page and registration page for which the user logins and input expenses.	I can login through Gmail and register for expense tracking.	Medium	Sprint-2
Customer Care Executive	Dashboard	USN-8	As a customer care executive, I can solve the queries of	I can reply to their queries and solve their problems.	High	Sprint-3

			users.			
Administrator	Registration	USN-9	As an Administrator, I can view the basic details of user.	I can provide the login details	Medium	Sprint-4
	Dashboard	USN-10	As an administrator, I can able to view the overall progress of a user.	I can give rewards based on their progress.	Low	Sprint-4

6. PROJECT PLANNING & SCHEDULING

6.1.Sprint Planning & Estimation

Use the below template to create product backlog and sprint schedule.

Sprints	Functional Requirement(Epic)	User Story Number	User Story/Task	Story Points	Priority	Team Members
Sprint-1	User Panel	USN-1	The user can register and login into the website, add expenses and see the progress of their expenses.	20	High	MONIKA T SINDHUJA M.K FALILA BANU H MOUNIKA R
Sprint-2	Admin Panel	USN-2	The role of the admin is to check about the progress of customers to provide any rewards if savings is increased.	20	High	MONIKA T SINDHUJA M.K FALILA BANU H MOUNIKA R
Sprint-3	Chat Bot	USN-3	The user can directly talk to chatbot regarding the products. Get the suggestions based on information provided by the user.	20	High	MONIKA T SINDHUJA M.K FALILA BANU H MOUNIKA R
Sprint-4	Final delivery	USN-4	Container of applications using docker Kubernetes and deployment of application. Create the documentation and final submit of the	20	High	MONIKA T SINDHUJA M.K FALILA BANU H MOUNIKA R

			application			

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	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

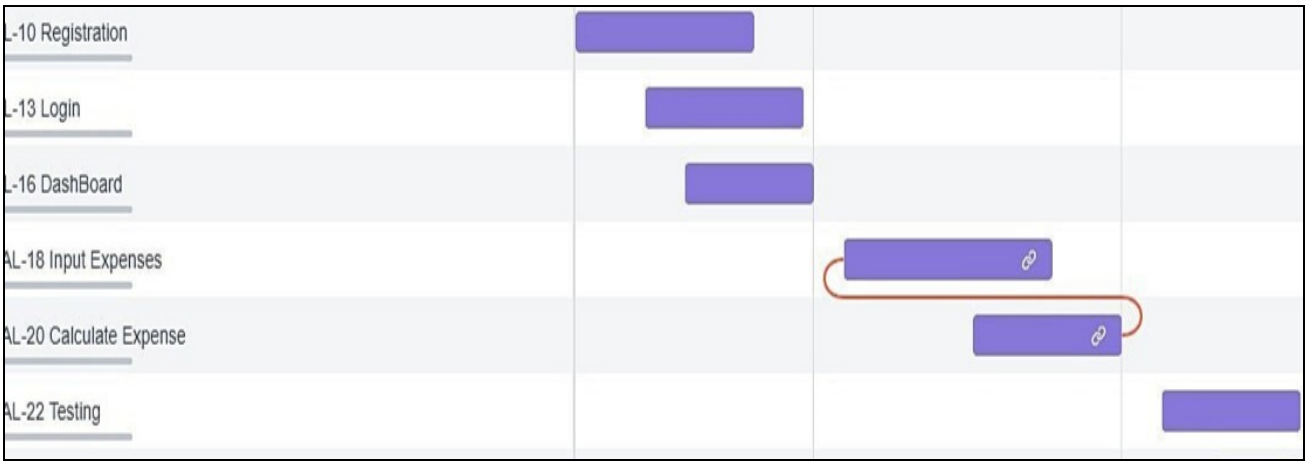
$$AV = \frac{sprint\ duration}{velocity}$$

Total Average Velocity=7.5

6.3.Reports from JIRA

Burndown chart:

A burndown chart is a visual representation of the remaining work versus the time required to complete it. By estimating the time it takes to complete tasks, issues, and testing, you can determine the project completion date.



Components and technology stack:

S.no	Component	Description	Technology
1.	User Interface	The user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript, Flask, Python
2.	Creating account	The user can able to create an account. The user details are stored in IBM DB2 securely.	Flask app using Kubernetes cluster, IBM DB2
3.	Login to account	The user interacts with the website to login into account. The user details are verified by comparing it with details	Flask app using Kubernetes cluster, IBM DB2

		stored in IBM DB2.	
4.	Add Expenses	The user interacts with the website to add expense. The user can choose a certain category and enters the amount spent	
5.	Wallet Dashboard	IBM Cloud Kubernetes Service provides a native Kubernetes experience that is secure and easy to use. This tool is used to load-balance, scale, and monitor the containers.	IBM Cloud Kubernetes Services
6.	Tracking of Expenses	IBM Container Registry enables to store and distribute Docker images in a managed, private registry.	IBM Cloud Container Registry
7.	Setting budget limit	The user can be able to set the limit based on which notifications are created.	IBM DB2
8.	Cloud Database	Database Service on Cloud	IBM DB2
9.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local File-system

10.	External API-1	To send email alerts when the expenses are made above the wallet limit.	SendGrid

Application Characteristics:

S.no	Characteristics	Description	Technology
1.	Open-Source Frameworks	Flask is an open-source framework written in Python. Similarly, Docker is also used	Flask, Docker
2.	Security Implementations	Only registered users who have specific privileges has access to the website.	IBM DB2
3.	Scalable Architecture	Three-tier architecture- Presentation tier, Application tier, Data tier	Python, IBM Cloud Services
4.	Availability	The application can be available for user at any time.	Kubernetes and Docker
5.	Performance	The application can handle multiple requests per second.	Kubernetes cluster, IBM Container Registry

7. CODING AND SOLUTIONING

7.1 Feature 1 - Backend

i) app.py

```
from flask import Flask,render_template,request, redirect, url_for, session
import ibm_db
conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=9938aec0-8105-433e-8bf9-
0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32459;SECURITY=SS
L;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=lwk74677;PWD=CnqHgzoxQOU3eV
Ty",",")
app = Flask(__name__)
print(conn)
print("DATABASE CONNECTED SUCESSFULLY")
```

7.2 Feature 2 - Database

```
DATABASE=bludb
HOSTNAME=9938aec0-8105-433e-8bf9-
0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;
PORT=32459;
SECURITY=SSL
SSLServerCertificate=DigiCertGlobalRootCA.crt
UID=lwk74677;
PWD=CnqHgzoxQOU3eVTy
```

SendGrid Coding:

```
import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail

message = Mail(
        from_email='personalexpensetracker24@gmail.com',
```

```
            to_emails='monika.anj202@gmail.com',
            subject='Hello there! Welcome to PERSONAL EXPENSE TRACKER APPLICATION',
            html_content='<strong>PDA warmly welcomes YOU!!!</strong</strong>')
try:
sg = SendGridAPIClient('SG.agdL93_hTXSf1Pi8EGC9xw.zlxSPuwGvwW0zz9CaFoG1kqF-
Cq9fPLBROX-_ALVk_g')
            response = sg.send(message)
            print(response.status_code)
            print(response.body)
            print(response.headers)

except Exception as e:
           print(str(e))
```

Chatbot

```
<script>
  window.watsonAssistantChatOptions = {
    integrationID: "238a365b-6981-4baf-b761-3555f7f9f4c9", // The ID of this integration.
    region: "jp-tok", // The region your integration is hosted in.
    serviceInstanceID: "c371dcde-b75b-4b8f-aa33-abd1b615fa7d", // The ID of your service instance.
    onLoad: function(instance) {
 instance.render(); }
  };
  setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
  });
</script>
```

7.3 DATABASE SCHEME :

Database Used : IBM Cloud Database

8. TESTING

8.1.TEST CASES

A test case has components that describe input, action and an expected response, in order to determine if a feature of an application is working correctly.A test case is a set of instructions on “HOW” to validate a particular test objective/target, which when followed will tell us if the expected behavior of the system is satisfiedor not.

Characteristics of a good test case:

i. Accurate: Exacts-the purpose.
ii. Economical:No unnecessary steps or words.
iii. Traceable: Capable of being traced to requirements.
iv. Repeatable: Can be used to perform the test over and over.
v. Reusable: Can be reused if necessary.

S. NO	**Scenario**	**Input**	**Excepted output**	**Actual output**
1	Login Form	User name and password	Login	Login success.
2	Registration Form	User basic details	Registration	User registration details stored in database.
3	Add Expense Form	Expense Details	Expense Added Successfully	User's expense details stored in database

8.2.USER ACCEPTANCE TESTING

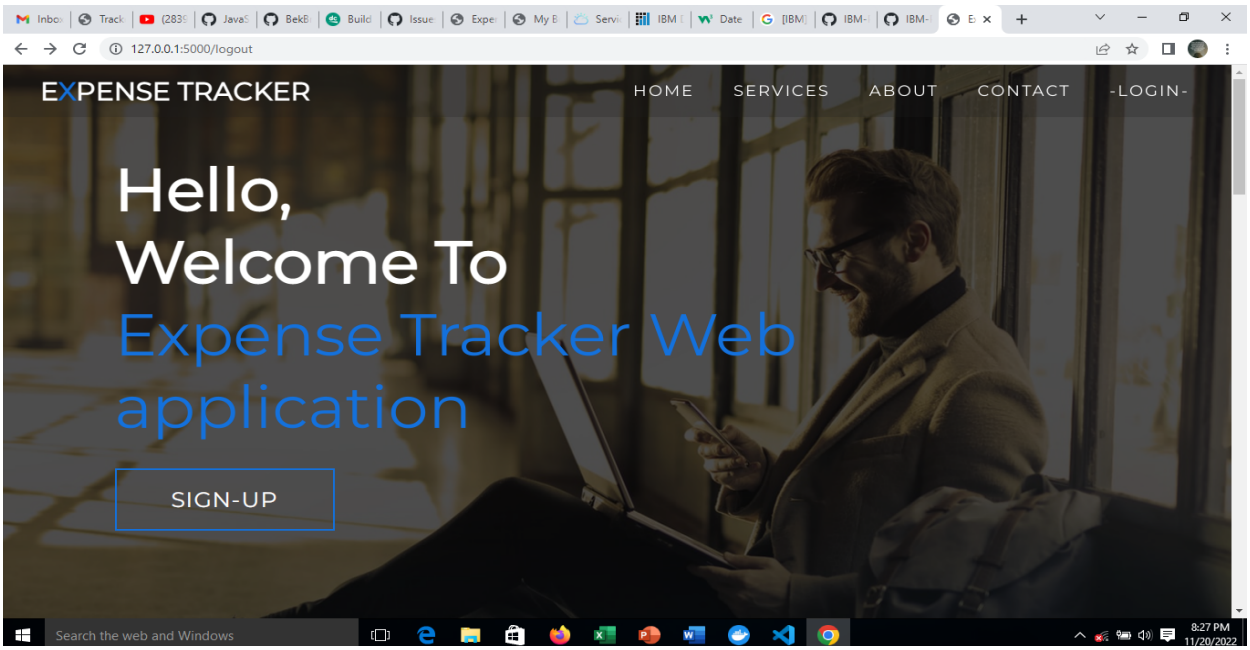
This is a type of testing done by users, customers, or other authorized entities to determine application/software needs and business processes. Acceptance testing is the most important phase of testing as this decides whether the client approves the application/software or not. It may involve functionality, usability, performance, and U.I of the application. It is also known as user acceptance testing (UAT), operational acceptance testing(OAT), and end-user-testing.

S.no	Parameter	Screenshot / Values
1.	Dashboard design	No of Visualizations – login and register page, add expense page, view report page for user.
2.	Data Responsiveness	The time taken for the response is maximum of 5 to 10 seconds.
3	Utilization of Data Filters	Not recommended
4.	Effective User Story	No of Scenes Added – 4 to 5 redirecting pages are provided.
5.	Descriptive Reports	Descriptive Report is an organized summary of testing objectives, activities, and results. It is effectively used to know about the expenditure of people.

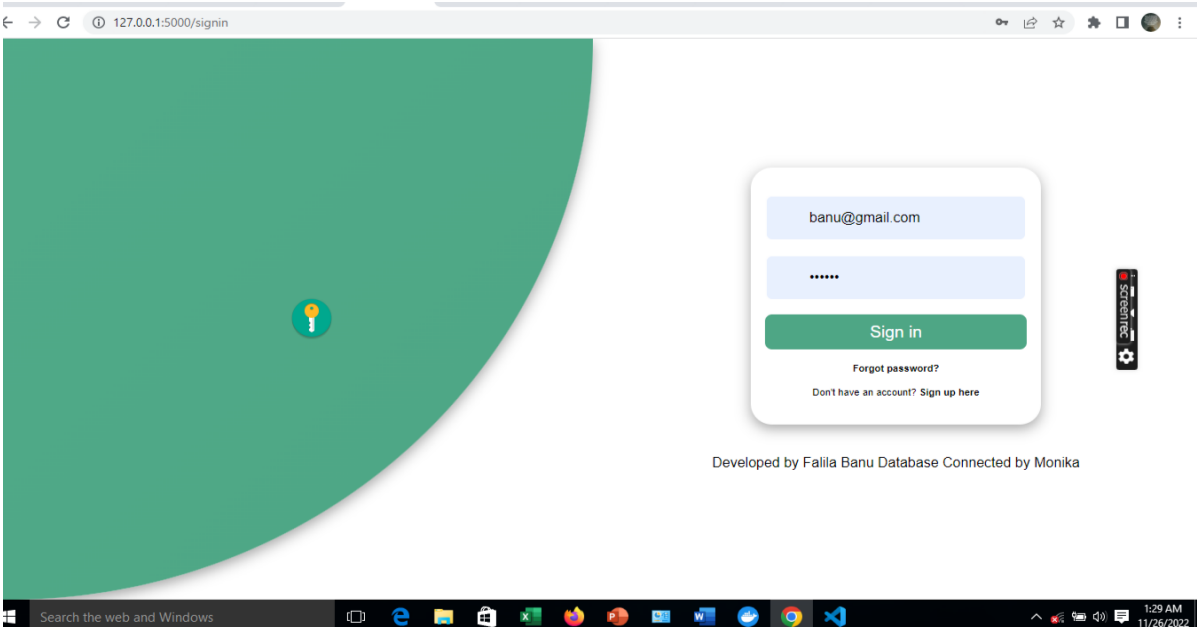
9. RESULTS

9.1.PERFORMANCE METRICS

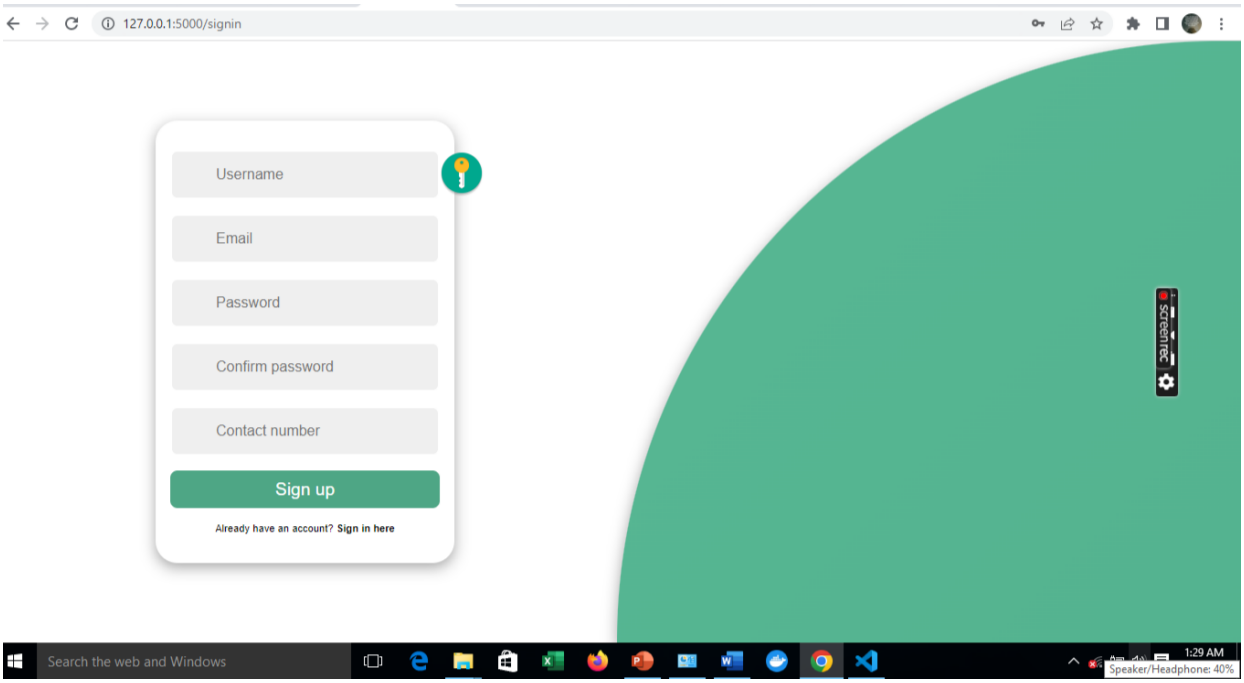
9.1.1.HOME PAGE



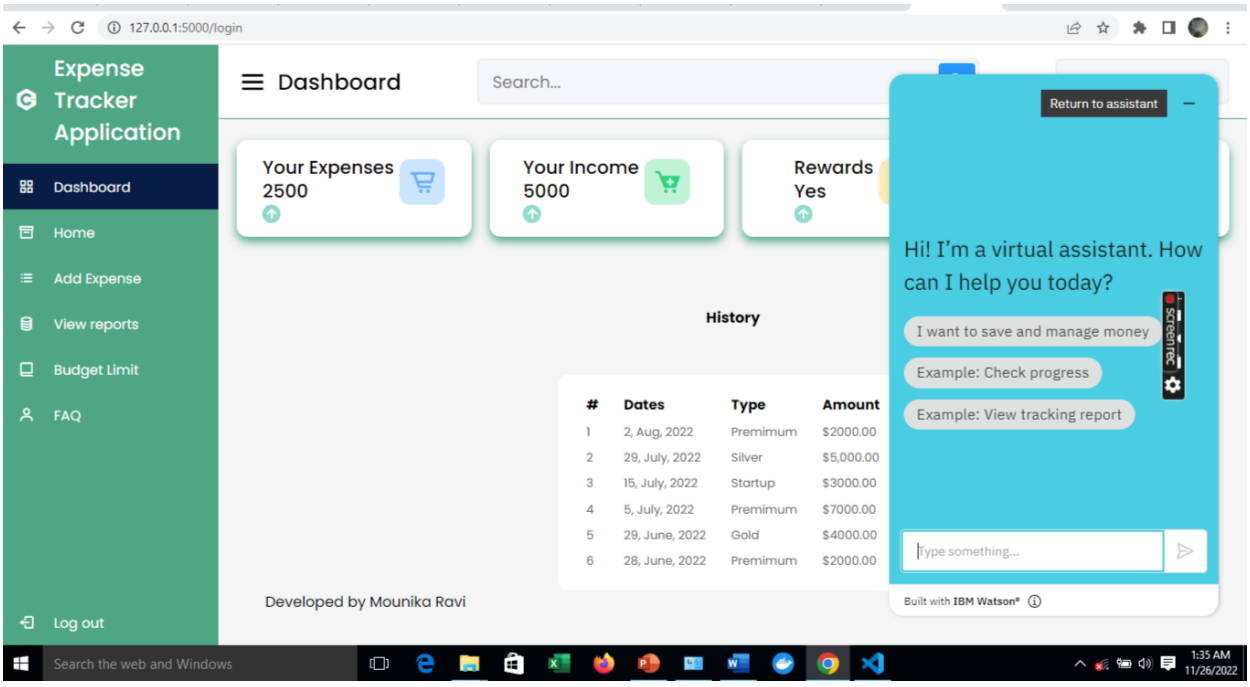
9.1.2.LOGIN PAGE



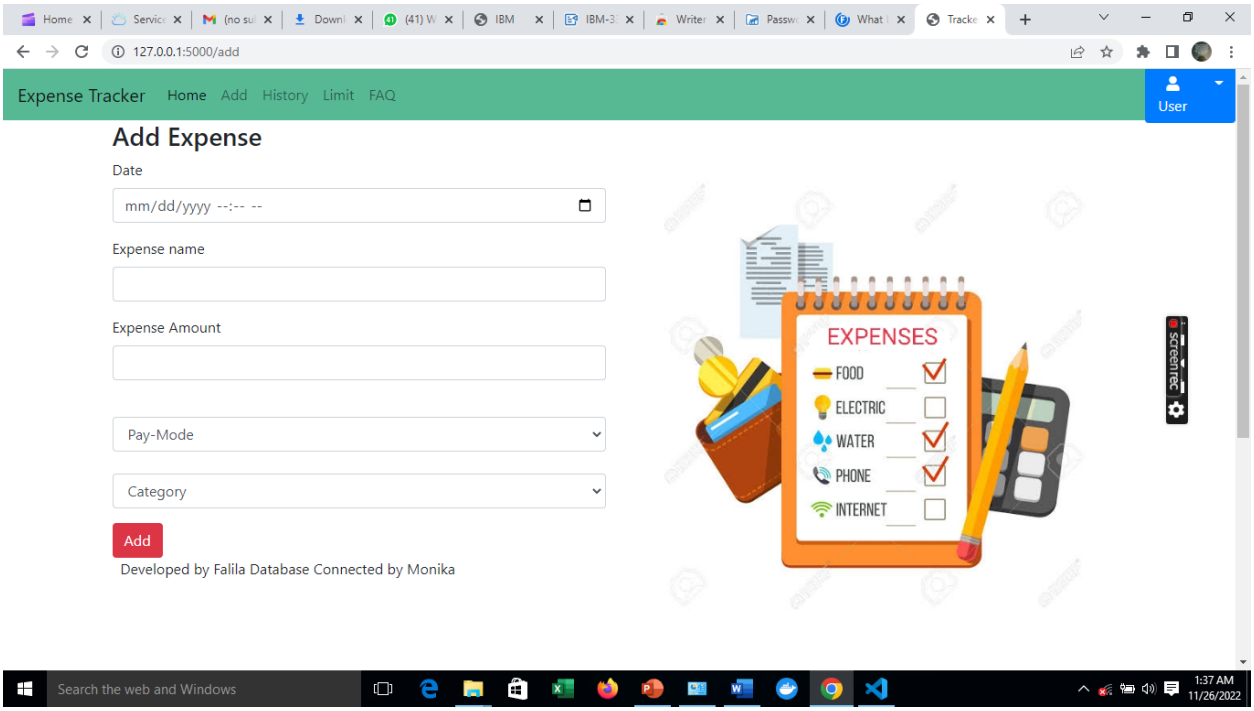
9.1.3.SIGN UP PAGE



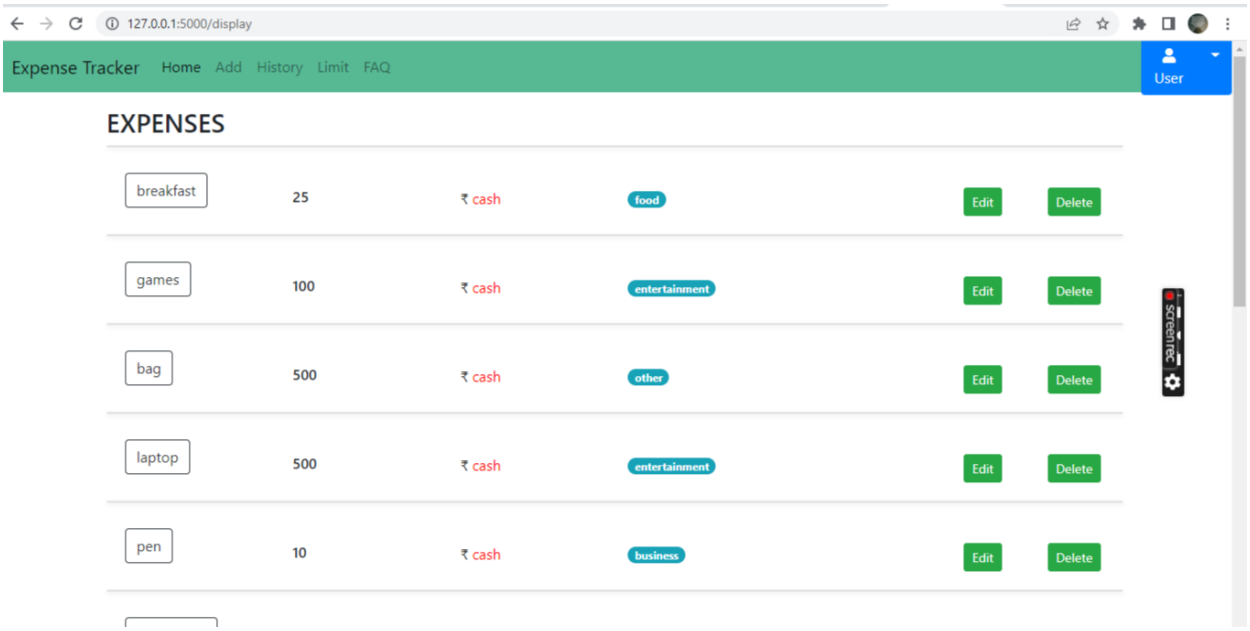
9.1.4.DASHBOARD PAGE(After Successful Login)

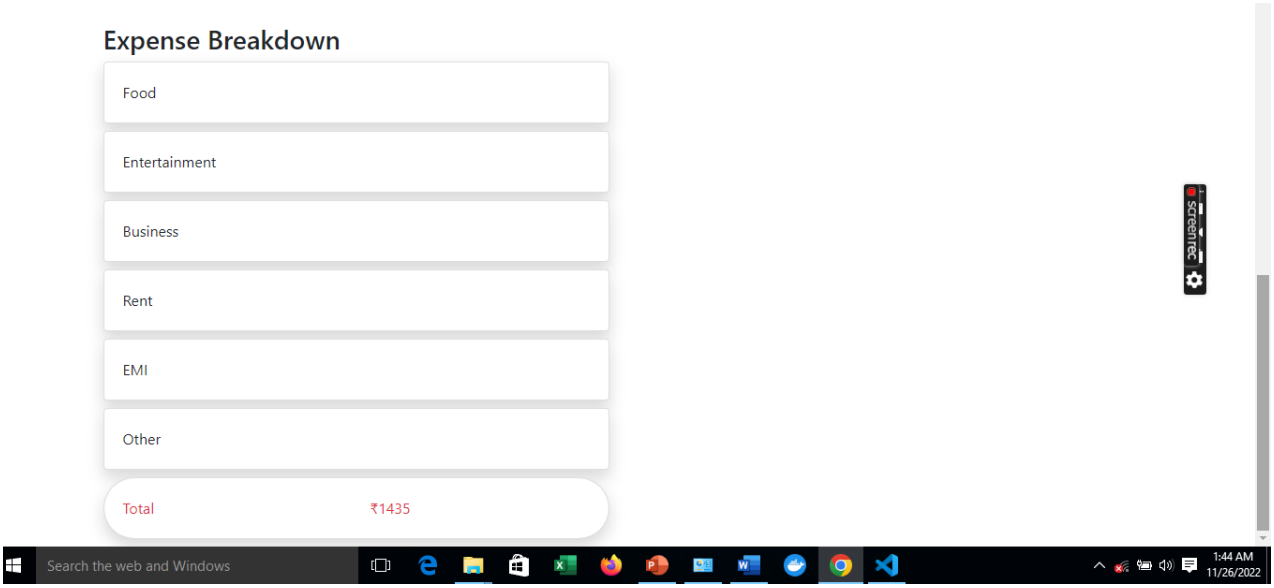


9.1.5.ADD EXPENSE PAGE

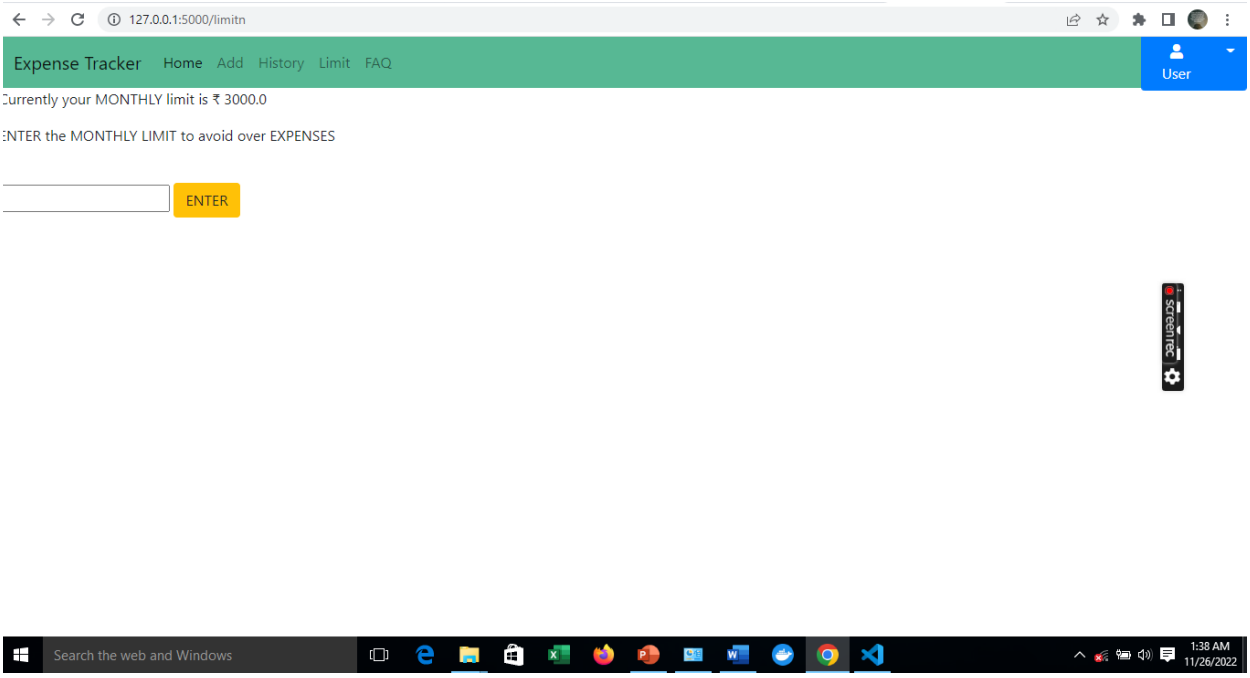


9.1.6.DISPLAY EXPENSES

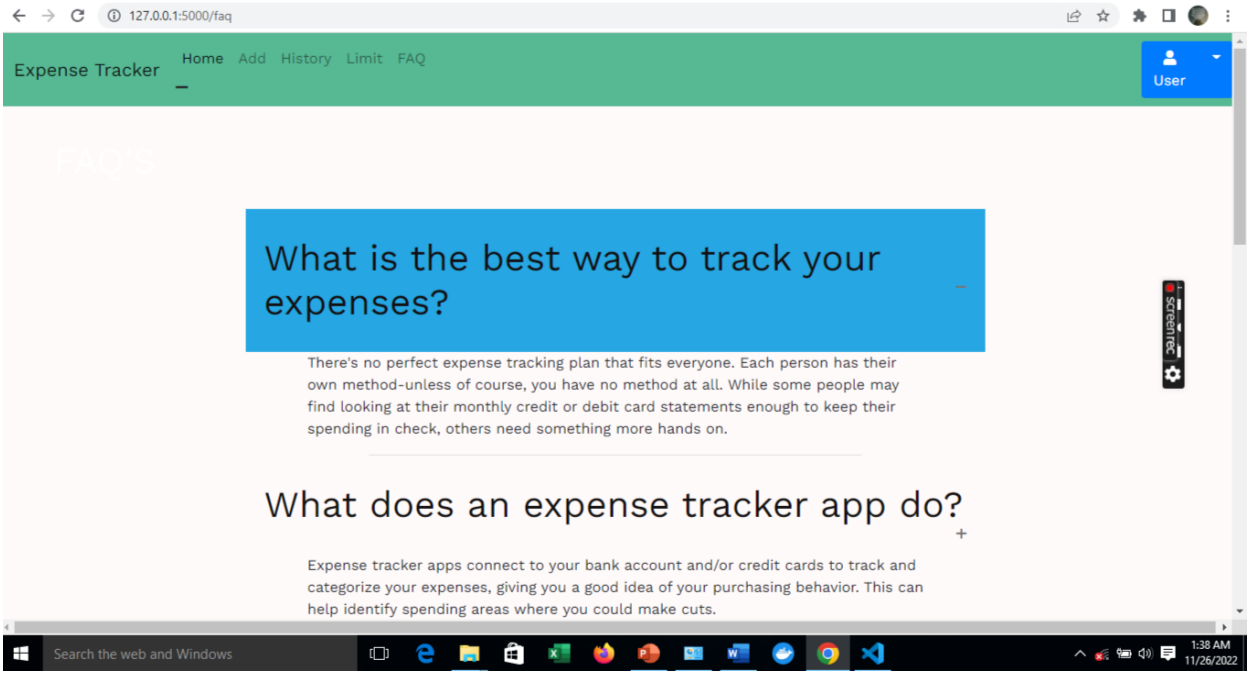




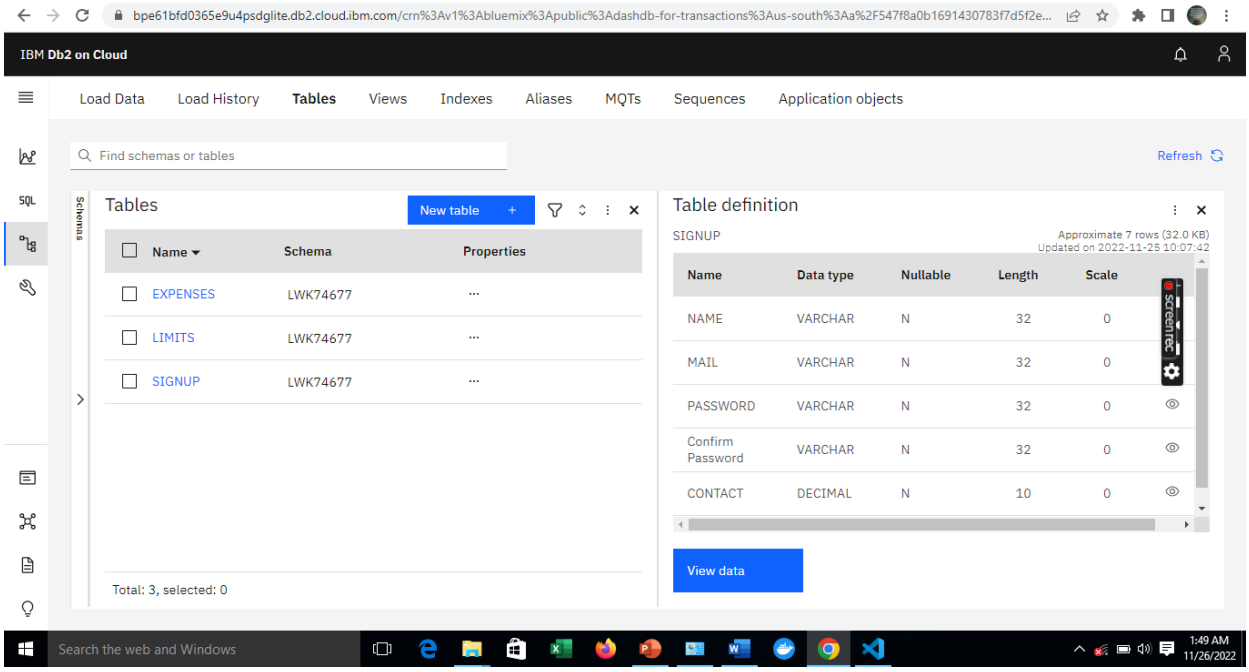
9.1.7.ADD LIMIT



9.1.8.FAQ



9.1.9.DATABASE



10. ADVANTAGES & DISADVANTAGES

10.1.ADVANTAGES

1. It is a user-friendly application.
2. We can Identify problem areas.
3. Using this system user can keep their finances organized.

10.2.DISADVANTAGES

i. It is time consuming.

ii. It leads to error-prone results.

iii. It lacks of data security.

11. CONCLUSION

This project is designed for successful completion of project on Personal Expense Tracker Application. The basic building aim is to manage expenses of users. Personal Expense Tracker Application System is a Web based application that is designed to store, process,retrieve and analyze information concerned with the uses of expenses. This project aims at maintaining all the information pertaining expenditures. To do all this we require high quality Web Application to manage those jobs. It provides a platform to keep an accurate record of your money inflow and outflow.

12. FUTURE SCOPE

This system is developed such a way that additional enhancement can be done without much difficulty. The renovation of the project would increase the flexibility of the system. In future, we can develop this project in android platform. We will add extra features like backup features (GPS), Feedback form, and automatic report generation option.

13. APPENDIX

13.1. SOURCE CODE:

app.py

```
from flask import Flask,render_template, request, redirect, url_for, session
import ibm_db
import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=9938aec0-8105-433e-8bf9-
0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32459;SECURITY=SS
L;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=lwk74677;protocol=tcpip;PWD=Cnq
HgzoxQOU3eVTy",",")
app = Flask(__name__)

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

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

```
def signup():
    return render_template('index.html')


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

    if request.method == 'POST' :
        name = request.form['name']
        mail=request.form['mail']
        pwd=request.form['pwd']
        cpwd=request.form['cpwd']
        cno=request.form['cno']

        sql = "SELECT * FROM signup WHERE mail =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,mail)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        if account:
            return render_template('index.html', msg="You are already a member, please login using
your details")
        else:
            insert_sql = "INSERT INTO signup VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, mail)
            ibm_db.bind_param(prep_stmt, 3, pwd)
            ibm_db.bind_param(prep_stmt, 4, cpwd)
            ibm_db.bind_param(prep_stmt, 5, cno)
            ibm_db.execute(prep_stmt)
        return render_template('home.html', msg="Data saved successfully..")

@app.route("/signin",methods=['post','get'])
def signin():
    if request.method=="post":
```

```
        return render_template("index.html")
    return render_template("index.html")

@app.route('/login',methods=['POST'])
def login():
   global userid
    msg = ''
   if request.method == 'POST' :
        mail = request.form['mail']
        pwd = request.form['pwd']
        sql = "SELECT * FROM signup WHERE mail =? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,mail)
        ibm_db.bind_param(stmt,2,pwd)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        param = "SELECT * FROM signup WHERE mail = '" + "\'" + mail + "\'" + " and password
= '" + "\'" + pwd + "\'"
        res = ibm_db.exec_immediate(conn, param)
        dictionary = ibm_db.fetch_assoc(res)
        if account:
            session['loggedin'] = True
            # session["id"] = dictionary["ID"]
            # userid = dictionary["ID"]
            session['name'] = dictionary["NAME"]
            session['mail'] = dictionary["MAIL"]

            return render_template('dashboard.html')
            # return redirect('/homepage')
        else:
           msg = 'Incorrect username / password !!'
    return render_template('index.html', msg = msg)
@app.route("/add")
def adding():
    return render_template('add.html')
```

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


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

    print(date)
    p1 = date[0:10]
    p2 = date[11:13]
    p3 = date[14:]
    p4 = p1 + "-" + p2 + "." + p3 + ".00"
    print(p4)
    # date=datetime.datetime.now().date()
    sql = "INSERT INTO expenses (userid, date, expensename, amount, paymode, category)
VALUES (?,?, ?, ?, ?, ?)"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, session['mail'])
    ibm_db.bind_param(stmt, 2, p4)
    ibm_db.bind_param(stmt, 3, expensename)
    ibm_db.bind_param(stmt, 4, amount)
    ibm_db.bind_param(stmt, 5, paymode)
    ibm_db.bind_param(stmt, 6, category)
    ibm_db.execute(stmt)

    print("Expenses added")

 # email part

 # email part

    param = "SELECT * FROM expenses WHERE MONTH(date) = MONTH(current
timestamp) AND YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
```

```
res = ibm_db.exec_immediate(conn, param)
print("passed")
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(total)
param = "SELECT mail, limit FROM limits WHERE mail =mail"
# param = "SELECT mail, limit FROM limits WHERE mail = " + str(session['mail'])
res = ibm_db.exec_immediate(conn, param)
dictionary = ibm_db.fetch_assoc(res)
row = []
s = 0
while dictionary != False:
    temp = []
    temp.append(dictionary["LIMIT"])
    row.append(temp)
    dictionary = ibm_db.fetch_assoc(res)
    s = temp[len(temp)-1]
```

```
    if total > int(s):
        msg = "Hello " + session['name'] + " , " + "you have crossed the monthly limit of Rs. " +
str(s) + "/- !!!" + "\n" + "Thank you, " + "\n" + "Team Personal Expense Tracker."
        sendmail(msg,session['mail'])

    return redirect("/display")

#DISPLAY---graph

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

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

@app.route("/limit" )
def limit():
```

```
        return redirect('/limitn')

@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
    if request.method == "POST":
        number= request.form['number']
        sql = "INSERT INTO limits (mail, limit) VALUES (?, ?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, session['mail'])
        ibm_db.bind_param(stmt, 2, number)
        ibm_db.execute(stmt)

        return redirect('/limitn')


@app.route("/limitn")
def limitn():
    param = "SELECT mail,limit FROM limits WHERE mail = mail"
    res = ibm_db.exec_immediate(conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    print(dictionary)
    row = []
    s = "/-"
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMIT"])
        print(temp)
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[len(temp)-1]

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

    #delete---the--data

@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
```

```
def delete(id):
    param = "DELETE FROM expenses WHERE  userid = mail"
    res = ibm_db.exec_immediate(conn, param)

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


#UPDATE---DATA

@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    param = "SELECT * FROM expenses WHERE  userid = mail"
    res = ibm_db.exec_immediate(conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    while dictionary != False:
        temp = []
        # temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        row.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

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




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

```
def update(id):
  if request.method == 'POST' :

      date = request.form['date']
      expensename = request.form['expensename']
      amount = request.form['amount']
      paymode = request.form['paymode']
      category = request.form['category']
      p1 = date[0:10]
      p2 = date[11:13]
      p3 = date[14:]
      p4 = p1 + "-" + p2 + "." + p3 + ".00"

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

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

13.2.GITHUB & PROJECT DEMO LINK :

Github : https://github.com/IBM-EPBL/IBM-Project-33789-1660226812

Project Demo Link : https://youtu.be/GcmP6ZeJ3LA