NALAIYA THIRAN - IBM PROJECT REPORT

(19CS406T Professional Readiness for Innovation, Employability and Entrepreneurship)

ON

PERSONAL EXPENSE TRACKER

Submitted by

TEAM ID: PNT2022TMID23362

NEERAJA B (113219031097)

NEHASRI R (113219031099)

MELBINA M (113219031087)

VANDANA S (113219031155)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

VELAMMAL ENGINEERING COLLEGE, CHENNAI-66.

(An Autonomous Institution, Affiliated to Anna University, Chennai)

2022-2023

VELAMMAL ENGINEERING COLLEGE
CHENNAI -66

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this NALAIYA THIRAN – IBM PROJECT REPORT **"PERSONAL EXPENSE TRACKER"** is the Bonafide work of "NEERAJA B (113219031097), NEHASRI R (113219031099), MELBINA M (113219031087), and VANDANA S(113219031155)" carried out in "PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP (NALAIYA THIRAN-IBM PROJECT)**"** during the Academic Year 2022-2023.

FACULTY EVALUATOR

Dr. S. GUNASUNDARI

Associate Professor

Dept. of Computer Science and Engineering

Velammal Engineering College

Chennai-600 066

HEAD OF THE DEPARTMENT

DR. B. MURUGESHWARI

Professor and Head

Dept. of Computer Science and Engineering

Velammal Engineering College

Chennai-600 066

TABLE OF CONTENT

CHAPTER NO.	TITLE	
1	**INTRODUCTION**	
	1.1	Project Overview
	1.2	Purpose
2	**LITERATURE SURVEY**	
	2.1	Existing System
	2.2	References
	2.3	Problem Statement Definition
3	**IDEATION AND PROPOSED SYSTEM**	
	3.1	Empathy Map Canvas
	3.2	Ideation and Brainstorming
	3.3	Proposed Solution
	3.4	Problem Solution fit
4	**REQUIREMENT ANALYSIS**	
	4.1	Functional requirement
	4.2	Non-Functional requirements
5	**PROJECT DESIGN**	
	5.1	Data Flow Diagrams
	5.2	Solution and Technical Architecture
	5.3	User Stories

6 PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning and Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

7 CODING AND SOLUTIONING

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema

8 TESTING

8.1 Test Cases

8.2 User Acceptance Testing

9 RESULTS

9.1 Performance Metrics

10 ADVANTAGES & DISADVANTAGES

11 CONCLUSION

12 FUTURE SCOPE

13 APPENDIX

1. INTRODUCTION

1.1 Project Overview

Personal Expense Tracker is a daily expense management system which is specially designed for non- salaried and salaried personnel for keeping track of their daily expenditure with easy and effective way through computerized system which tends to eliminate manual paper works. It will also manage records in systematic way and user can access the stored data conveniently. We have tried to design the project in such way that user may not have any difficulty in using this application without much effort. This software can be really used by end user who have Android running devices with them. The database which we have used for storage purpose is IBM db2.

1.2 Purpose

The idea of developing this project is to enable users to have a better experience while monitoring their expenditures. At a certain period, users do not know where they spent their money and they spend more on needless expenses beyond budgets which leads to financial crisis. Some of the conventional methods used to tackle this problem in normal circumstances are like making use of a sticky note by common users. Proficient people deal with this kind of problems by using spreadsheets to record expenses. We believe that a handy design and a good application can handle these problems easily. Our application is capable of recording the expenditure and giving a broad view with easy user interface and this application is intelligent enough to show the history of expenses.

2. LITERATURE SURVEY

2.1 Existing Problem

People can’t able to track their expenses and spending more on unnecessary expenses which leads to money crisis. Without tracking people can’t know whether they exceed the limit of their budget. Diary notes requires lots of manual calculation and It reduces the interest to track expenses. User frustrated about they can’t remember where their money goes and can’t handle their cash flow. There is no alerting system about exceeding limits. There can be many disadvantages of using a manual accounting system. Accounting, for any business, can be a complex undertaking. A manual accounting system requires you to understand the accounting process in a way that may be unnecessary with a computerized accounting system. This can be an advantage or a disadvantage, depending on the person doing the bookkeeping; often, a specially trained professional is needed to ensure that accounting is done properly. Unrevealing the complexity of your financial records by hand may be time consuming. Since it takes time to generate reports.

2.2 References

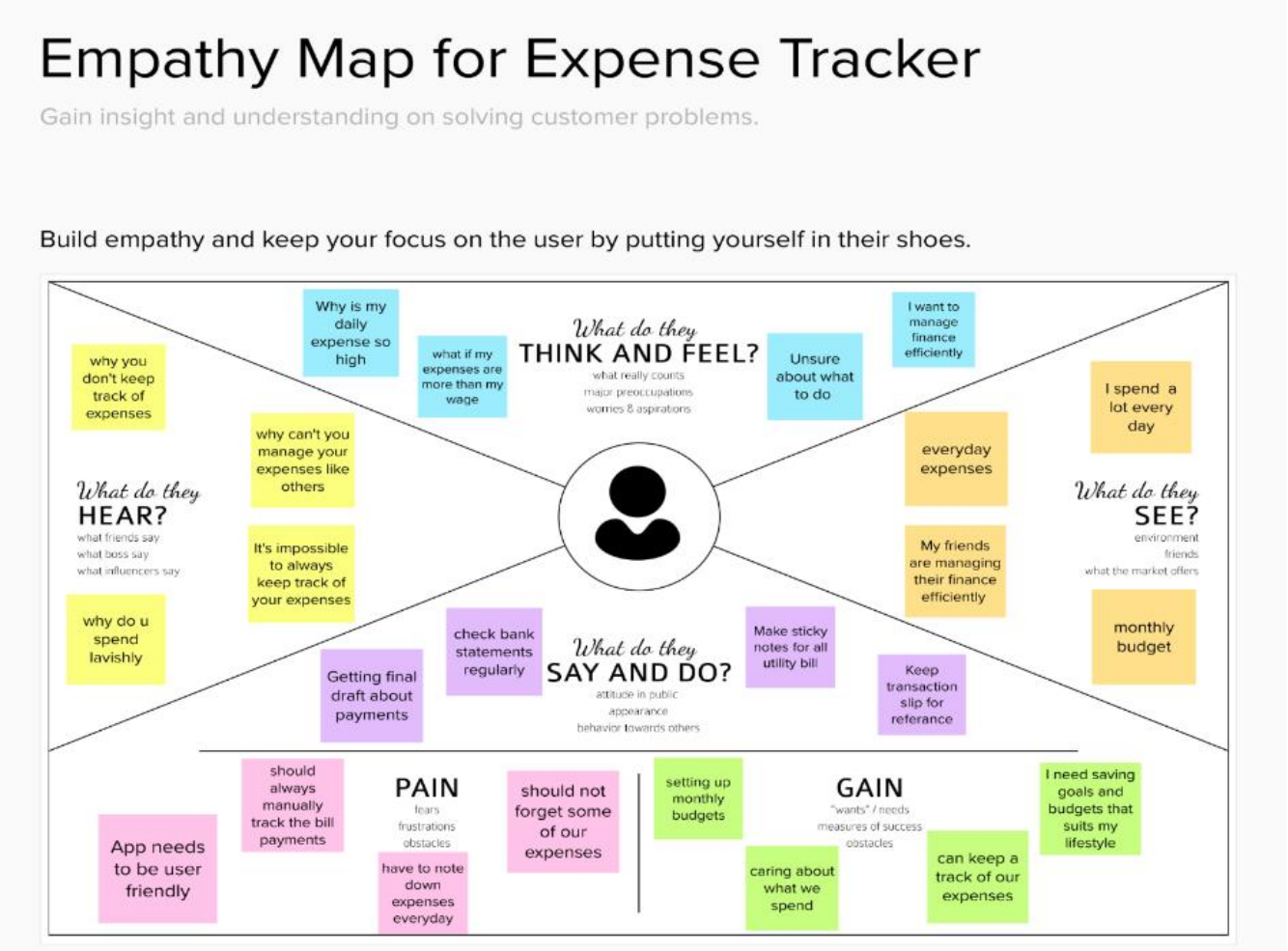
1. https://ijirt.org/master/publishedpaper/IJIRT151474_PAPER.pdf
2. https://iopscience.iop.org/article/10.1088/1757-899X/263/4/042050
3. https://www.irjet.net/archives/V6/i3/IRJET-V6I31110.pdf
4. https://www.youneedabudget.com/
5. https://www.irjet.net/archives/V6/i3/IRJET-V6I31110.pdf

2.3 Problem Statement Definition

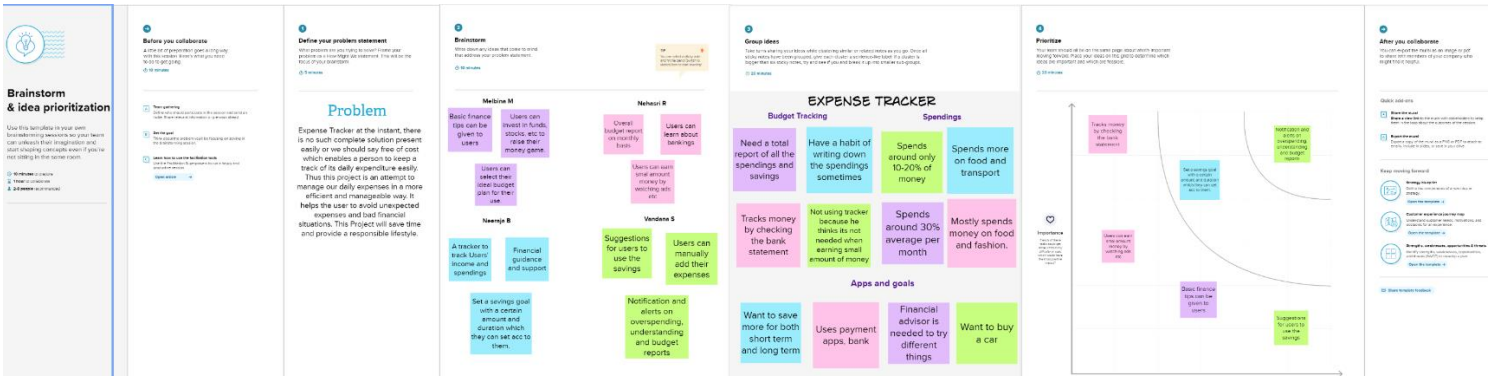
Our project helps the user to keep track their expenses and determine whether they are spending as per their set budget. Potential users need to input the required data such as the expense amount, merchant, category, and date when the expense was made. Which allows users to track their expenses daily, weekly, monthly, and yearly in terms of summary, bar graphs, and pie-charts. It is like automated diary which requires no burden of manual calculation and enables the user to not just keep the control on the expenses but also to generate and save reports. Users can insert and delete transactions. We can compare with past expenses.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



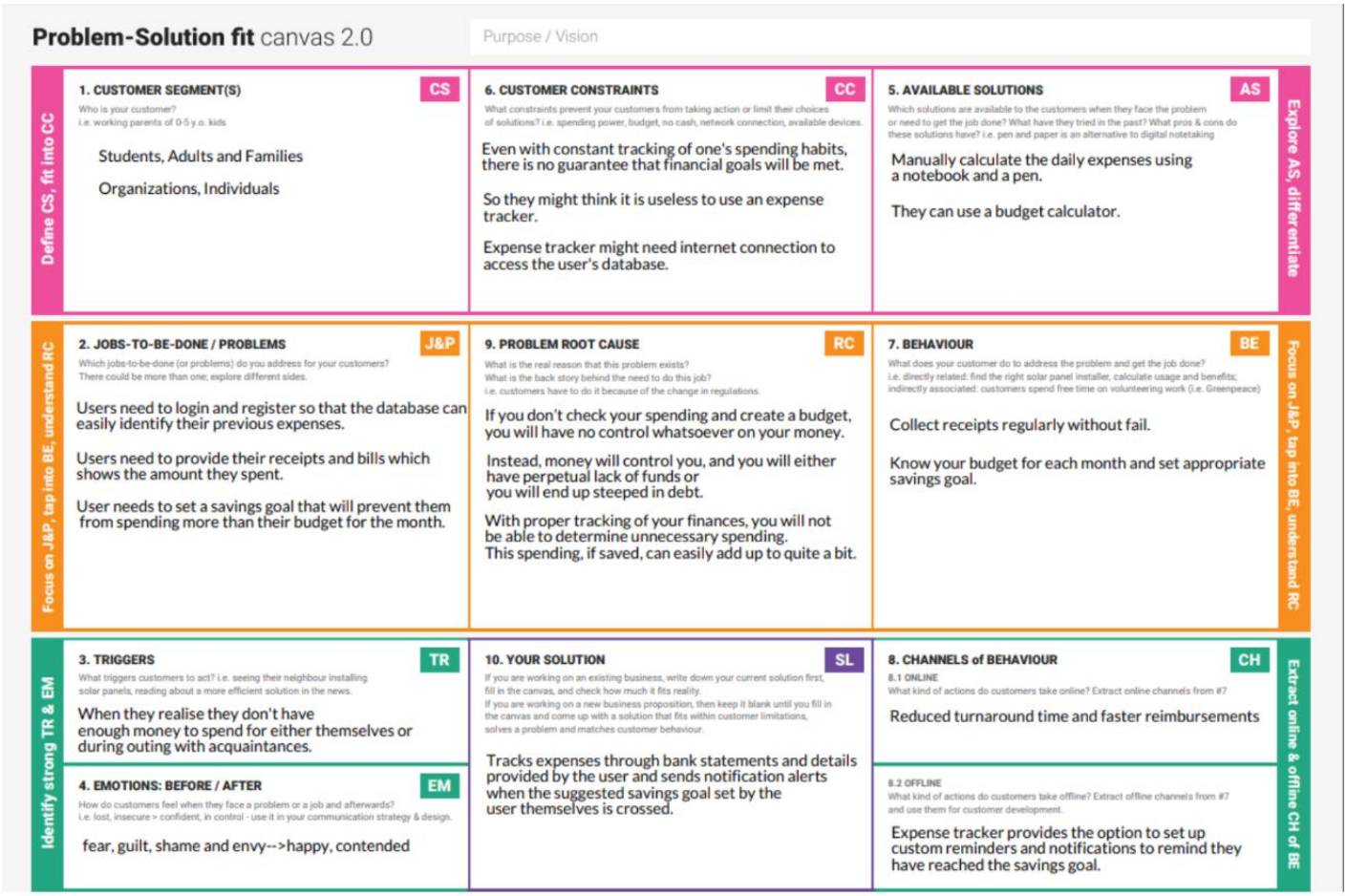
3.3 Proposed Solution

S.No	Parameter	Description
1.	Problem Statement	Many Organizations have their own system to record their income and expenses. Itis good for a person to record daily expenses and tracking the expenses throughout the month is essential. Thus, personal expense tracker application has made tracking and managing expenses a breeze.
2.	Novelty	This app effectively works in providing financial management and helps in maintaining healthy and happier financial life fulfilling all needs and requirements as the user's comfort. This app provides a higher range of accuracy regarding real-time effectively and security.

3.	Feasibility of Idea	User can easily maintain their untracked expenses and the app helps the user to record their expenditure. This app can achieve economic feasibility and security feasibility with at most care and support to the user.
4.	Business Model	The application can be provided based onuser required feature and the cost depends onthe usage.
5.	Social Impact	This application can create awareness among common people about finance and stuffs. This application also helps user to be financially responsible.
6.	Scalability of the Solution	This application can handle large number of users and data with high performance and security at any given point of time.

3.4 Problem Solution fit

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why.



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

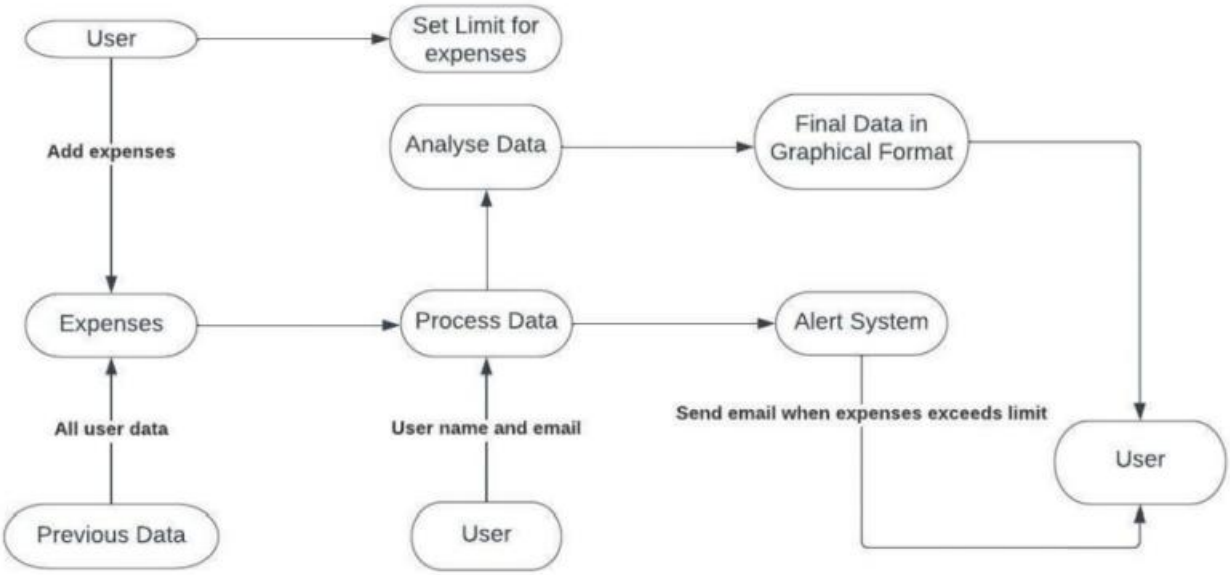
Following are the non-functional requirements of the proposed solution.

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

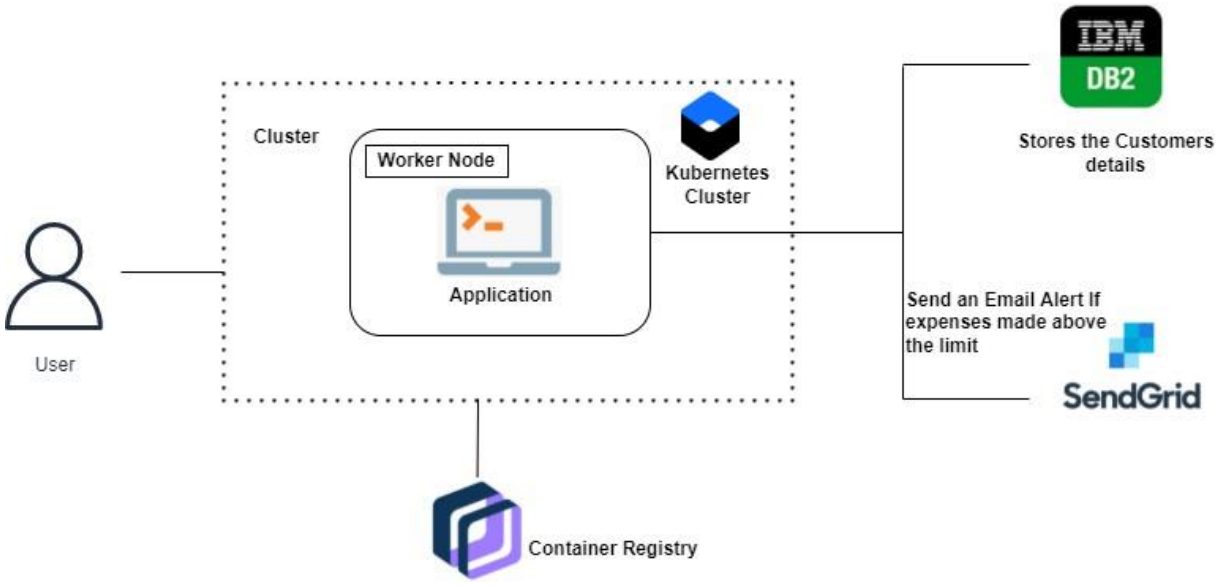
A Data flow diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



5.2 Solution & Technical Architecture

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

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.



5.3 User Stories

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
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email& click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Google account.	I can register & access the dashboard with Google login.	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register through Gmail.	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access the application.	High	Sprint-1
	Dashboard	USN-6	As a user, I can enter my incomeand expenditure details (Set Budget).	I can view my daily expenses	High	Sprint-2
Customer (Web user)	Registration And Login	USN-7	As a user, I can register for the application by entering my emailand will receive a confirmation email. Then I can log into the application by entering email & password	I can register & access the dashboard with Google and login.	Medium	Sprint-1
	Dashboard	USN-8	As a user, I can enter my incomeand expenditure details (Set Budget).	I can access my account / dashboard	High	Sprint-2
Customer Care Executive		USN-9	As a customer care executive Ican solve the medium log in issues and other issues of the application.	I can provide support or solution at any time 24x7	Medium	Sprint-2
Administrator		USN-10	As an administrator I can upgradeor update the application.	I can fix the bug which arises for the customers	Medium	Sprint-2

6.PROJECT PLANNING AND SCHEDULING:

6.1SPRINT PLANNING AND ESTIMATION:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	8	High	Nehasri Melbina
Sprint-1	Login	USN-2	As a user, I can log into the application by entering email & password.	7	High	Nehasri Neeraja
Sprint-1	Dashboard	USN-3	As a user, I can see the accounts that I registered and can login from.	5	Medium	Vandana
Sprint-2	User panel	USN-4	As a user, I can upload my expenses and set a savings goal.I can get an analysis of my expenditure in graphical forms	20	High	Neeraja Nehasri Melbina Vandana
Sprint-3	Email alerts and backend connection	USN-5	As a user, I will be notified throughemail alerts when I cross the savings goals I set. And getting data from users, Storing data in database.	20	High	Neeraja Nehasri Melbina Vandana
Sprint -4	Final deployment of the application and delivery	USN-6	Run trials to understand traffic and deploy the application.	20	High	Neeraja Nehasri Melbina Vandana

6.2 SPRINT DELIVERY SCHEDULE:

Sprint	Total Story Points	Duration	Sprint StartDate	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 Oct 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

Velocity:

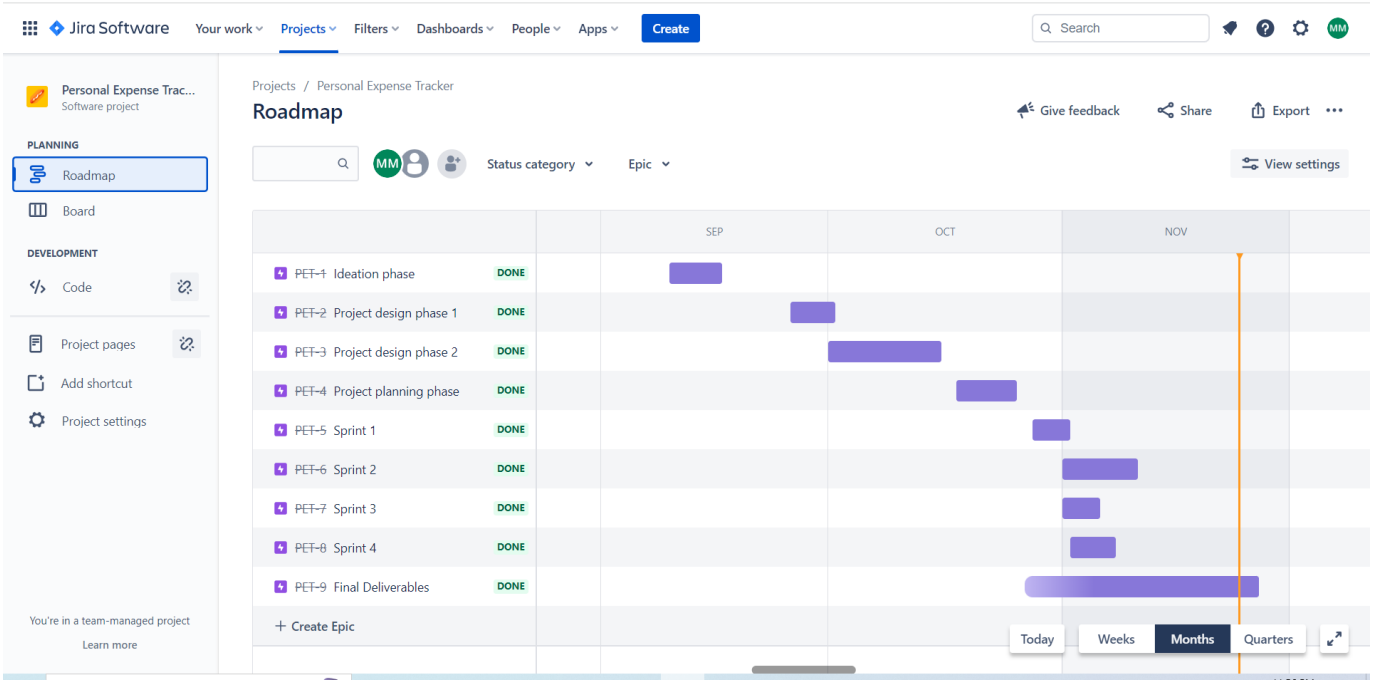
Velocity is a metric that predicts how much work an Agile software development team can successfully complete within a two-week sprint (or similar time-boxed period). Velocity is a useful planning tool for estimating how fast work can be completed and how long it will take to complete a project

Average velocity = Total story points/ No. of iterations = 80/4 = 20

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

6.3 REPORTS FROM JIRA :



7. CODING AND SOLUTIONING

7.1 FEATURE 1

#DISPLAY---graph

```
@app.route("/display")
def display():
    print(session["username"])

    sql = "SELECT * FROM EXPENSES1 "
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    list1=[]
    row = ibm_db.fetch_tuple(stmt)
    while(row):
        list1.append(row)
        row = ibm_db.fetch_tuple(stmt)
    print(list1)

    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0


 for x in list1:
        total =total+int(x[3])
        if x[4] == "food":
            t_food =t_food+ int( x[3] )
        elif x[4] == "entertainment":
            t_entertainment =t_entertainment+ int( x[3] )

         elif x[4] == "business":
            t_business =t_business+ int( x[3] )
        elif x[4] == "rent":
            t_rent  =t_rent+ int( x[3] )
        elif x[4] == "EMI":
            t_EMI  =t_EMI+ int( x[3] )
        elif x[4] == "other":
            t_other  =t_other+ int( x[3] )
```

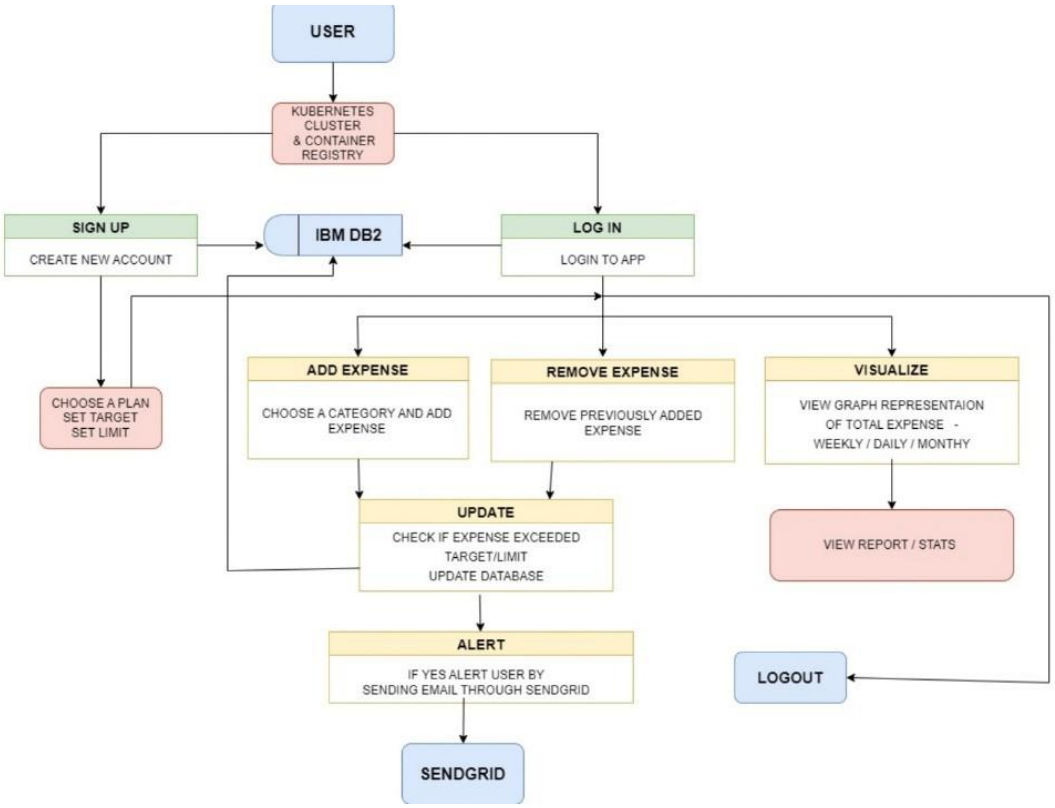
```
    return render_template('display.html' ,expense = list1, total = total ,
                        t_food = t_food, t_entertainment =  t_entertainment,
                        t_business = t_business, t_rent =  t_rent, t_EMI =  t_EMI,  t_other =  t_other)
```

7.2 FEATURE 2

```
#ADDING----DATA
@app.route("/add")
def adding():
    return render_template('add.html')

@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():
    id=request.form['id']
    date = request.form['date']
    title = request.form['title']
    amount = request.form['amount']
    category = request.form['category']

    sql = "INSERT INTO EXPENSES1(USERID,DATE,TITLE,AMOUNT,CATEGORY)
VALUES(?,?,?,?,?)"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,id)
    ibm_db.bind_param(stmt,2,date)
    ibm_db.bind_param(stmt,3,title)
    ibm_db.bind_param(stmt,4,amount)
    ibm_db.bind_param(stmt,5,category)
    ibm_db.execute(stmt)


 print(date + " " + title + " " + amount + " " + category)
sql1 = "SELECT * FROM EXPENSES1 WHERE MONTH(date)=MONTH(DATE(NOW()))"

stmt1 = ibm_db.prepare(conn, sql1)
    ibm_db.execute(stmt1)
    list2=[]
    expense1 = ibm_db.fetch_tuple(stmt1)
while(expense1):
        list2.append(expense1)
        expense1 = ibm_db.fetch_tuple(stmt1)
    total = 0
    for x in list2:
        total=total+int(x[3])

    sql2 = "SELECT explimit FROM LIMITS  DESC LIMIT 1"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.execute(stmt2)
```

```
    limit=ibm_db.fetch_tuple(stmt2)

    return redirect("/display")

 #limit
@app.route("/limit" )
def limit():
       return redirect('/limitn')

@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
     if request.method == "POST":
         userid=request.form['id']
         number= request.form['number']

         sql = "INSERT INTO LIMITS(USERID,EXPLIMIT) VALUES(?,?)"
         stmt = ibm_db.prepare(conn, sql)
         ibm_db.bind_param(stmt,1,userid)
         ibm_db.bind_param(stmt,2,number)
         ibm_db.execute(stmt)
         return redirect('/limitn')

@app.route("/limitn")
def limitn():

    sql = "SELECT EXPLIMIT FROM LIMITS DESC LIMIT 1"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    row=ibm_db.fetch_tuple(stmt)

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


#REPORT

#log-out

@app.route('/logout')

   def logout():
   session.pop('loggedin', None)
   session.pop('username', None)
   session.pop('email',None)
   return render_template('home.html')

if __name__ == "__main__":
    app.run(host='0.0.0.0',port=5000,debug=True)
```

7.3 DATABASE SCHEMA



8. TESTING

8.1. TEST CASES

S.NO	Test case id	Feature type	Component	Test Description	Input Test Data	Actual Output	Expected Output	Remarks
1	LoginPage_TC_001	Functional	Login Page	Verify user is able to see the Login/Sign up popup when user clicked on My account button	Username: xxxx Password: Test	Login/Signup popup should display	Login/Signup popup should display	Pass
2	RegisterPage_TC_001	Functional	Register page	New user will register their details	Username:xxxx Email: yyy Password: test	Working as expected	Register page should navigate to login page	Pass
3	HomePage_TC_001	Functional	Home page	User adds income and expense	1.User ID 2.Title 3.Category 4.Click Add	Expense Addes	Successfully adds the expense	Pass
4	History of expense_TC_001	Functional	Display Page	Display total expenses, report and overall expense in bar chart	No test data	Report is shown	Expense total, report and bar chart must be displayed	Pass
5	Saving Goal_TC_001	UI	Saving page	Set expense limit	1.Enter userid 2.Set limit amount	Limit is set	Limit is set	Pass

8.2. User Acceptance Testing

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the “Personnel Expense Tracker in Cloud” project at the time of the release to User AcceptanceTesting (UAT).

2. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	3	2	5	15
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	1	0
Won't Fix	0	5	2	1	7
Totals	19	13	11	25	69

2. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	51	0	0	51
Security	0	0	0	0
Outsource Shipping	0	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9. RESULTS

9.1 Performance metrics

10. ADVANTAGES AND DISADVANTAGES

10.1. ADVANTAGES:

Keeping accurate records of personal expenses is very important for many reasons,

- 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

In short, knowing where each dollar is going to go before you spend it will help it spend more time in your bank account.

10.2. DISADVANTAGES:

A con with any system used to track spending is that one may start doing it then taper off until it's forgotten about all together. Yet, this is a risk for any new goal such as trying to lose weight or quit smoking. If a person first makes a budget plan, then places money in savings before spending any new pay period or month, the tracking goal can help. In this way, tracking spending and making sure all receipts are accounted for only needs to be done once or twice a month. Even with constant tracking of one's spending habits, there is no guarantee that financial goals will be met. Although this can be considered to be a con of tracking spending, it could be changed into apro if one makes up his or her mind to keep trying to properly manage all finances. Another con that may occur when spending is being tracked is anerror, but this may also be able to be changed into a pro if the person does regular tracking.

11. CONCLUSION

In this paper, after making this application we assure that this application will help its users to manage the cost of their daily expenditure. It will guide them and make them aware about their daily expenses. It will prove to be helpful for the people who are frustrated with their daily budget management, irritated because of the amount of expenses and wish to manage money and to preserve the record of their daily cost which may be useful to change their way of spending money. In short, this application will help its users to overcome the wastage of money.

12. FUTURE SCOPE

1. Achieve your business goals with a tailored mobile app that perfectlyfits your business.
2. Scale-up at the pace your business is growing.
3. Deliver an outstanding customer experience through additional controlover the app.
4. Control the security of your business and customer data.
5. Open direct marketing channels with no extra costs with methods suchas push notifications.
6. Boost the productivity of all the processes within the organization.
7. Increase efficiency and customer satisfaction with an app aligned totheir needs.
8. Prediction: With the help of AI, your mobile app can predict your next purchase, according to your spending behavior. Moreover, it can recommend products and provide unique insights on saving money. Itbrings out the factors causing fluctuations in your expenses.
9. Employee Travel Budgeting: Most businesses save money with a travel budgeting app as It helps prepare a budget for an employee's entire business trip. The feature will predict the expenses and allocateresources according to the prediction.

13. APPENDIX

13.1 SOURCE CODE

app.py

```
from flask import Flask, render_template, request, redirect, session
import ibm_db
import re
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase

app = Flask(__name__)

app.secret_key = 'team23362'

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=6667d8e9-9d4d-4ccb-ba32-
21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30376;
SECURITY=SSL;
SSLServerCertificate=DigiCertGlobalRootCA.crt;
UID=ptc81939;PWD=F4Kdma3oL5ddCVAk",",")

#HOME--PAGE
@app.route("/home")
def home():
    return render_template("homepage.html")

@app.route("/")

def add():
    return render_template("home.html")
#SIGN--UP--OR--REGISTER

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

def register():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
```

```
        password = request.form['password']

        sql = "SELECT * FROM REGISTERUSER WHERE USERNAME =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        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:
            sql1="INSERT  INTO REGISTERUSER(USERNAME,PASSWORD,EMAIL) VALUES(?,?,?)"
            stmt1 = ibm_db.prepare(conn, sql1)

            ibm_db.bind_param(stmt1,1,username)
            ibm_db.bind_param(stmt1,2,password)
            ibm_db.bind_param(stmt1,3,email)
            ibm_db.execute(stmt1)
            msg = 'You have successfully registered !'
            return render_template('signup.html', msg = msg)

 #LOGIN--PAGE
@app.route("/signin")
def signin():
    return render_template("login.html")

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

 if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        sql = "SELECT * FROM REGISTERUSER WHERE USERNAME =? AND PASSWORD =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)


        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
```

```
        if account:
            session['loggedin'] = True
            session['username'] = account["USERNAME"]
            session['email']=account["EMAIL"]

            return redirect('/home')
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)

#ADDING----DATA
@app.route("/add")
def adding():
    return render_template('add.html')

@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():
    id=request.form['id']
    date = request.form['date']
    title = request.form['title']
    amount = request.form['amount']
    category = request.form['category']

    sql = "INSERT INTO EXPENSES1(USERID,DATE,TITLE,AMOUNT,CATEGORY)
VALUES(?,?,?,?,?)"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,id)
    ibm_db.bind_param(stmt,2,date)
    ibm_db.bind_param(stmt,3,title)
    ibm_db.bind_param(stmt,4,amount)
    ibm_db.bind_param(stmt,5,category)
    ibm_db.execute(stmt)

 print(date + " " + title + " " + amount + " " + category)
sql1 = "SELECT * FROM EXPENSES1 WHERE MONTH(date)=MONTH(DATE(NOW()))"

stmt1 = ibm_db.prepare(conn, sql1)
    ibm_db.execute(stmt1)
    list2=[]
    expense1 = ibm_db.fetch_tuple(stmt1)

while(expense1):
        list2.append(expense1)
        expense1 = ibm_db.fetch_tuple(stmt1)
    total = 0
    for x in list2:
```

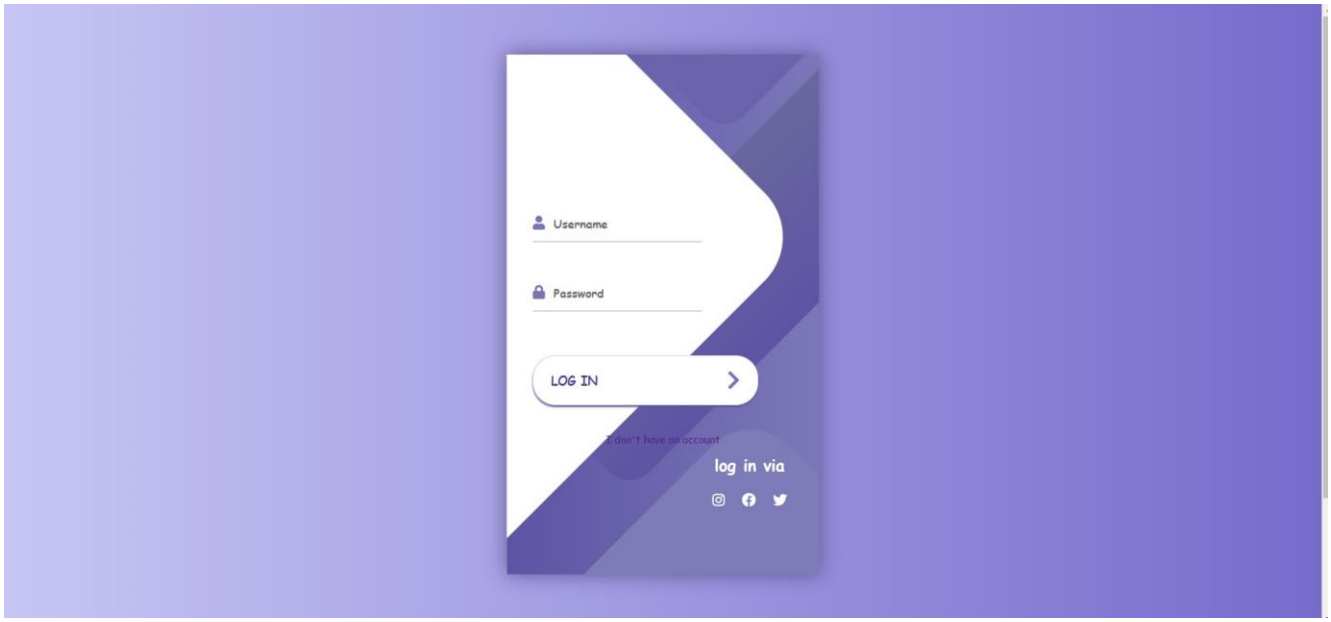
```
        total=total+int(x[3])

    sql2 = "SELECT explimit FROM LIMITS  DESC LIMIT 1"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.execute(stmt2)
    limit=ibm_db.fetch_tuple(stmt2)

    return redirect("/display")

#DISPLAY---graph

@app.route("/display")
def display():
    print(session["username"])

    sql = "SELECT * FROM EXPENSES1 "
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    list1=[]
    row = ibm_db.fetch_tuple(stmt)
    while(row):
        list1.append(row)
        row = ibm_db.fetch_tuple(stmt)
    print(list1)

    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0


 for x in list1:
        total =total+int(x[3])
        if x[4] == "food":
            t_food =t_food+ int( x[3] )
        elif x[4] == "entertainment":
            t_entertainment =t_entertainment+ int( x[3] )

         elif x[4] == "business":
            t_business =t_business+ int( x[3] )
        elif x[4] == "rent":
            t_rent  =t_rent+ int( x[3] )
        elif x[4] == "EMI":
            t_EMI  =t_EMI+ int( x[3] )
```

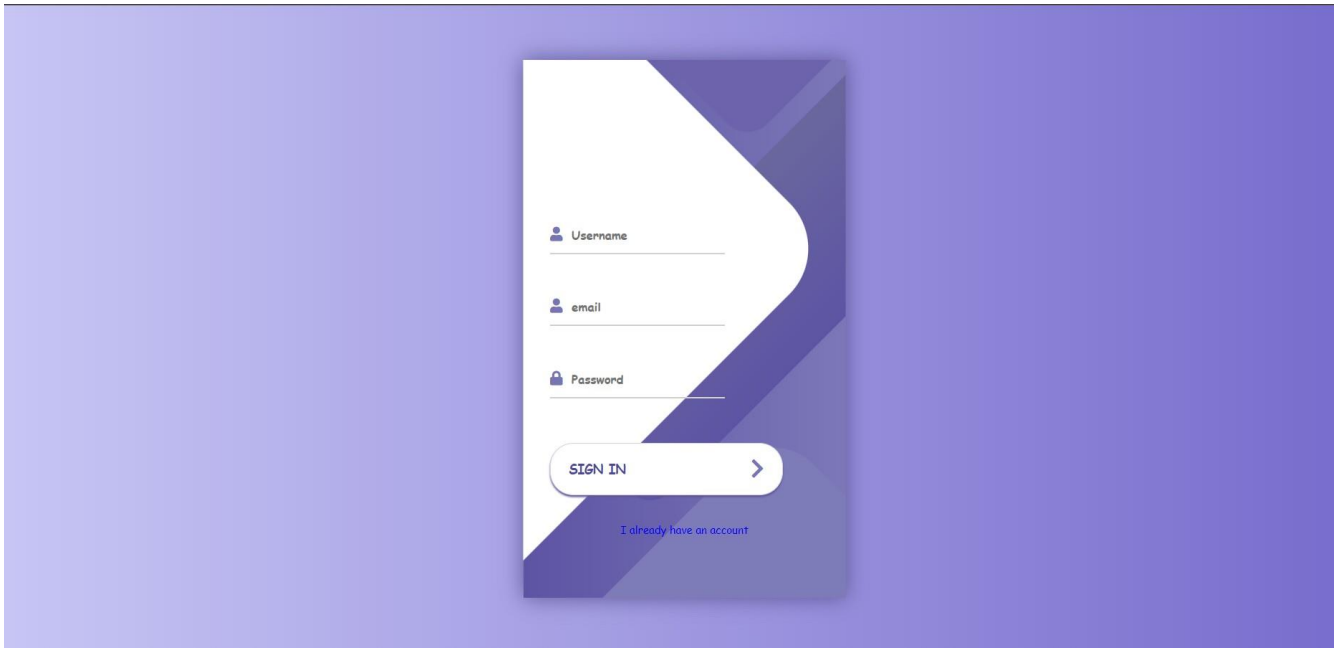
```
        elif x[4] == "other":
            t_other  =t_other+ int( x[3] )

    return render_template('display.html' ,expense = list1, total = total ,
                           t_food = t_food, t_entertainment =  t_entertainment,
                           t_business = t_business, t_rent =  t_rent, t_EMI =  t_EMI,  t_other =  t_other)

 #limit
@app.route("/limit" )
def limit():
       return redirect('/limitn')

@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
     if request.method == "POST":
         userid=request.form['id']
         number= request.form['number']

         sql = "INSERT INTO LIMITS(USERID,EXPLIMIT) VALUES(?,?)"
         stmt = ibm_db.prepare(conn, sql)
         ibm_db.bind_param(stmt,1,userid)
         ibm_db.bind_param(stmt,2,number)
         ibm_db.execute(stmt)
         return redirect('/limitn')

@app.route("/limitn")
def limitn():

    sql = "SELECT EXPLIMIT FROM LIMITS DESC LIMIT 1"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    row=ibm_db.fetch_tuple(stmt)

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

#REPORT

#log-out

@app.route('/logout')

   def logout():
   session.pop('loggedin', None)
   session.pop('username', None)
   session.pop('email',None)
   return render_template('home.html')
```

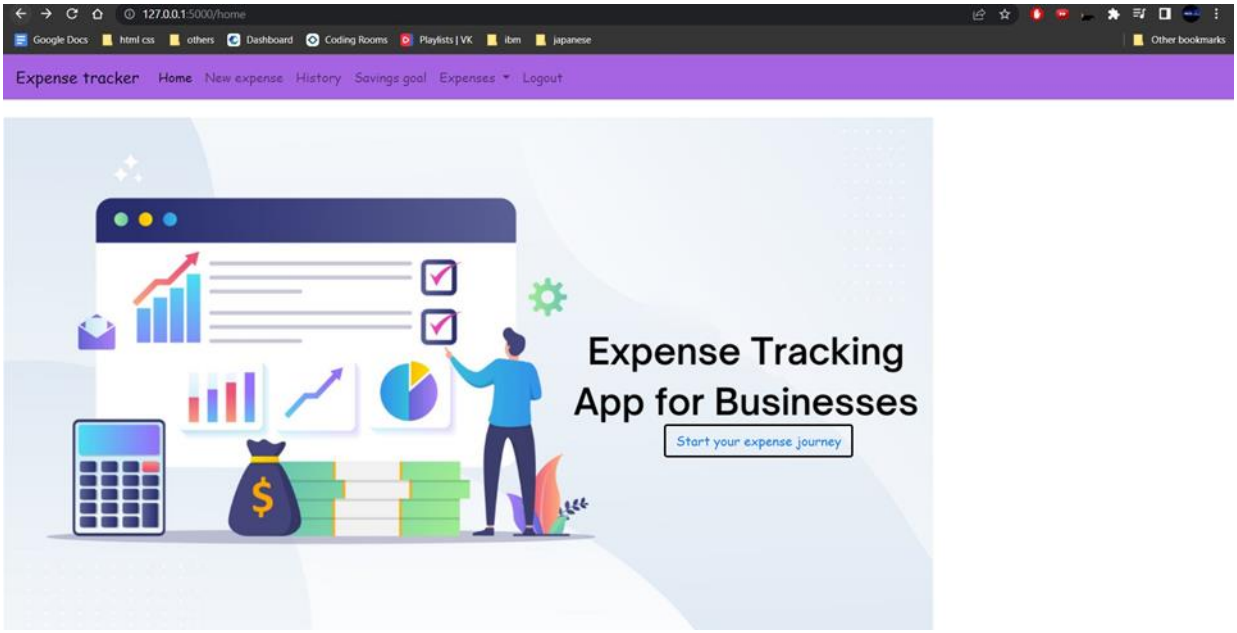
```
if __name__ == "__main__":
    app.run(host='0.0.0.0',port=5000,debug=True)
```

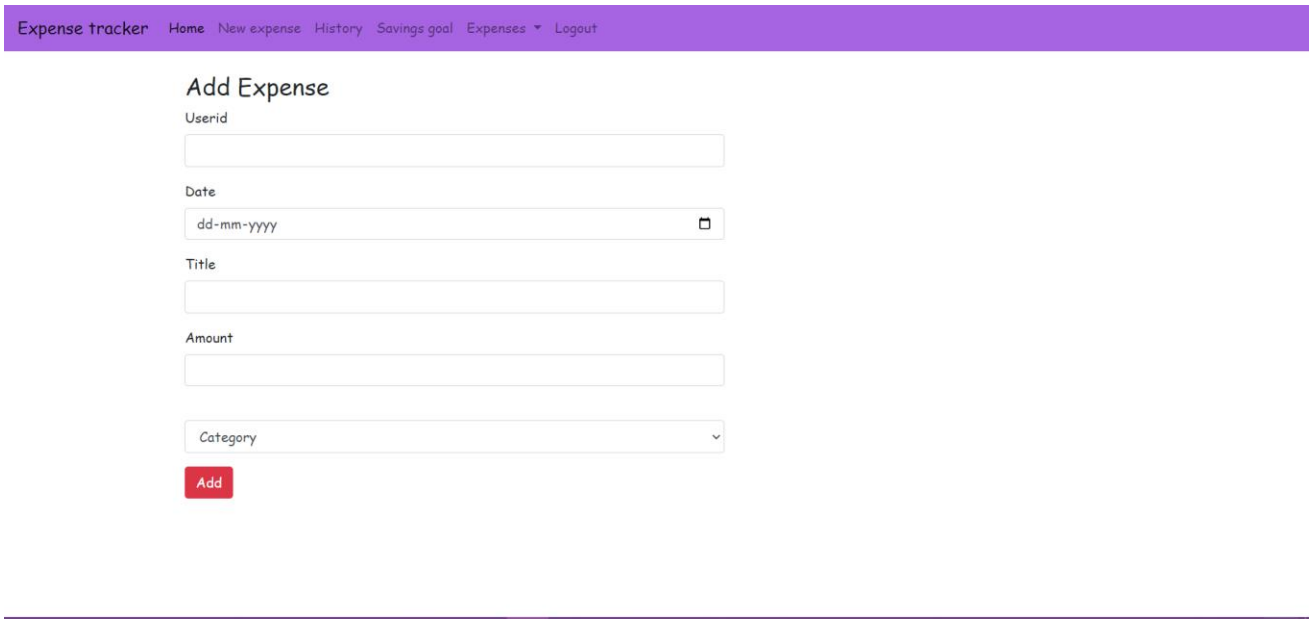
Complete code available at : https://github.com/IBM-EPBL/IBM-Project-22062-1659802763

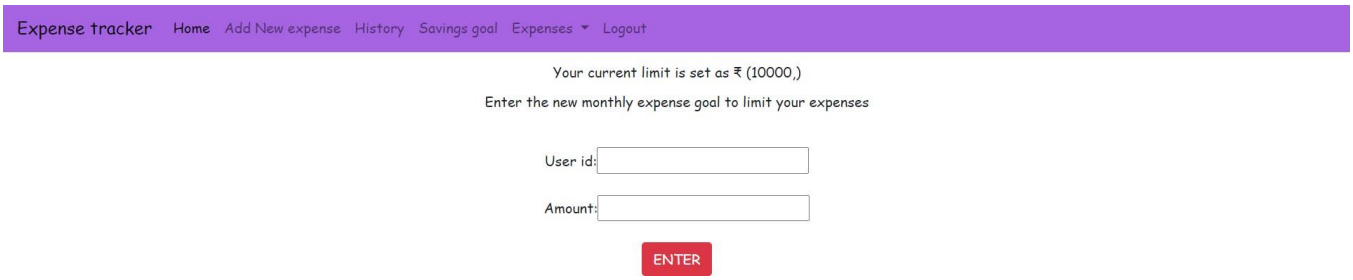
13.2 OUTPUT

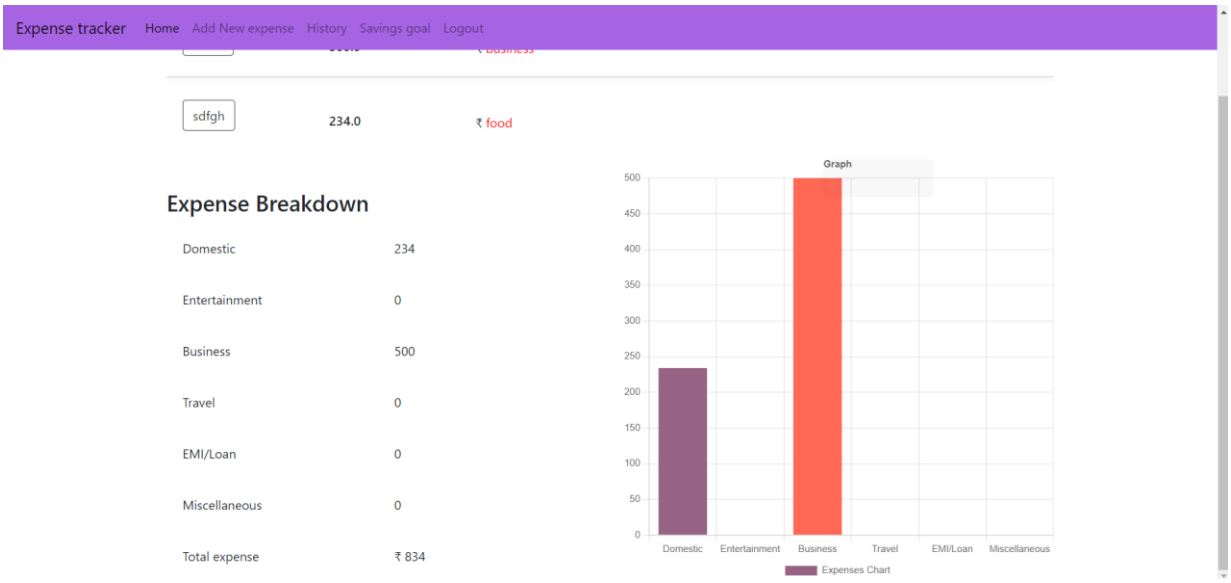



Username
email
Password
SIGN IN
I already have an account


127.0.0.1:5000/home
Google Docs
html css
others
Dashboard
Coding Rooms
Playlists | VK
ibm
japanese
Other bookmarks
Expense tracker
Home
New expense
History
Savings goal
Expenses
Logout
Expense Tracking App for Businesses
Start your expense journey


Expense tracker
Home
New expense
History
Savings goal
Expenses
Logout
Add Expense
Userid
Date
dd-mm-yyyy
Title
Amount
Category
Add


Expense tracker
Home
Add New expense
History
Savings goal
Expenses
Logout
Your current limit is set as ₹ (10000,)
Enter the new monthly expense goal to limit your expenses
User id:
Amount:
ENTER


Expense tracker
Home
Add New expense
History
Savings goal
Logout
sdfgh
234.0
₹ food
Expense Breakdown
Domestic 234
Entertainment 0
Business 500
Travel 0
EMI/Loan 0
Miscellaneous 0
Total expense ₹ 834
Graph
Domestic
Entertainment
Business
Travel
EMI/Loan
Miscellaneous
Expenses Chart

13.3 LINKS

- The Code is in the following GitHub link: Code
- **Demo link:** Video