IBM - NALAIYA THIRAN PROJECT

PERSONAL EXPENSE TRACKER APPLICATION

TEAM ID : PNT2022TMID18300

TEAM SIZE : 4

TEAM LEADER : GANESH KUMAR K

TEAM MEMBER : ASHOK SANDEEP N B S

TEAM MEMBER : ALAGURAJA K

TEAM MEMBER : SANTHOSH V

INDUSTRY MENTOR : KUSBOO

FACULTY MENTOR : KIRUBAKARAN G

TABLE OF CONTENTS

1.INTRODUCTION

1.1 Project overview

With the increase in sales of smartphones over the last few years, people are using mobile applications to get their work done, which makes their lives easier. Mobile applications include different categories such as Entertainment, Sports, Lifestyle, Education, Games, Food and Drink, Health and Fitness, Finance, etc.

This Expense Tracker application falls in the Finance Category and serves the important purpose of managing finances which is a very important part of one's life.

The software product went through the design, development, and the testing phase as a part of the Software Development Lifecycle. The application's interface is designed using custom art elements, the functionality is implemented using iOS SDK, and the phase of testing the product was accomplished successfully. The application is not much user intensive but just comprises of having them enter the expense amount, date, category, merchant, and other optional attributes (taking picture of the receipts, entering notes about the expense, adding subcategories to the categories). With this entered information, the user can see the expense details for daily, weekly, monthly, and yearly in figures, graphs, PDF format, and can print them as well if a printer is detected or scanned nearby. All these topics have been explained in detail in their respective chapters.

The aim of this project is to provide a solution for users on how to manage finances in any circumstance by keeping track of their expenses every day. Ultimately, this contributes to societal well-being as well.

1.2 Purpose

The motivation to work in this project is our real-life experience. As a user We face many difficulties in our daily file. In our dailylife money is the most important portion and without it we cannot last one day on earth but if we keep on track all financial data then we can overcome this problem. Most of the people cannot track their expenses and income one way they face the money crisis and depression. This situation motivates us to make an android app to track all financial activities. Using the Daily Expense Tracker user can be tracking expenses day to day and making life tension free.

A comprehensive money management strategy requires clarity and conviction for decision-making. You will need a defined goal and a clear vision for grasping the business and personal finances. That's when an expense tracking app comes into the picture.

An expense tracking app is an exclusive suite of services for people who seek to handle their earnings and plan their expenses and savings efficiently. It helps you track all transactions like bills, refunds, payrolls, receipts, taxes, etc., on a daily, weekly, and monthly basis.

2. LITERATURE SURVEY

2.1 Existing Problem

Parents are worried that their children lack financial literacy unliketheir global counterparts. They are looking for applications where theirchildren can learn about spending money. This financial illiteracy is prevalent even among elders.

Existing Solutions :

1. Application Name - Spendee.

Spendee is a free money tracker app for budget planning and money management. Everyone is not good in budget handling and expense tracking. It is useful to someone who just wants to track daily expenses, instead of being confused by the complicated expenses bookkeeping. It is useful in both Personal financial management and organization's financial management.

Pros

- Free to use: Spendee has a free plan that provides limited functionality for users. The most useful tools, however, are reserved for the paid subscription plans.

- Easy-to-use design: The Spendee app sports a simple design that optimizes the user experience. The beautiful interface allows for a smooth signup process, easy navigation and generally attractive displays and charts. It is available in both light and black themes.

- Global availability: Spendee is available in Canada and countries in North America, South America, Asia, Europe, and Africa. Whichever country you are in, you can set up a Spendee account, and gain access to more than 2,500 banks globally.

- You can create your account with which currency you desire. You are also free to switch currencies depending on your immediate need.

- Bank-level security: Spendee deploys tight security measures to ensure that customers' data is securely protected. All transactions and information exchange are encrypted such that only parties authorized by you have access to it. Spendee's servers are currently hosted on Google Cloud, a trusted and tested security-oriented platform.

- One-glance overview of your money: The Spendee app provides you with an opportunity to link all your financial institutions with your Spendee account. You can synchronize different banks, online financial platforms like PayPal, as well as cryptocurrency trading platforms such as Finance and Coinbase. This enables you to see all your important financial details in one place.

- Monitor and regulate expenditure: Seeing all your money in one place gives you a feel of the bigger picture, and you can make more informed and well- rounded financial decisions.

- With your financial information neatly displayed with insightful analytics, you can take steps to optimize your spendings and savings to reach your desired financial goal.

Cons

- Bank services are limited to paid plans: Spendee operates on a three-tiered basis, each with its own cost. Bank linking facilities are available only paid plans. However, the most advanced tools are restricted to the Spendee Premium, which is the highest of all three tiers.

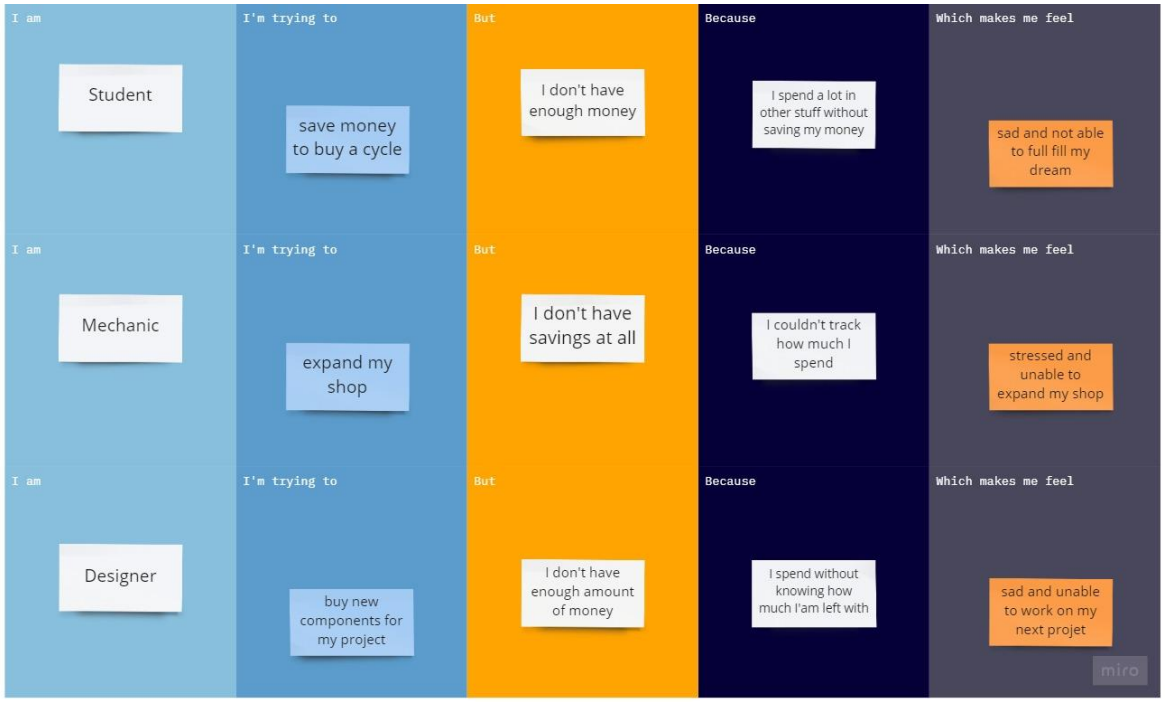
- Problems with app updates: Android and iOS users of the Spendee app complain of bugs that come with new updates. On many occasions, currencies fail to display, automatic synchronization breaks down and error messages interrupt transactions.

- Does not support some banks: Despite being available in many countries of the world, Spendee does not support some Canadian banks such as HSBC, the Bank of Montreal, the Equitable Bank, Indian banks such as Indian Bank, Indian Overseas Bank, Standard Chartered Bank among others.

2.2 References:

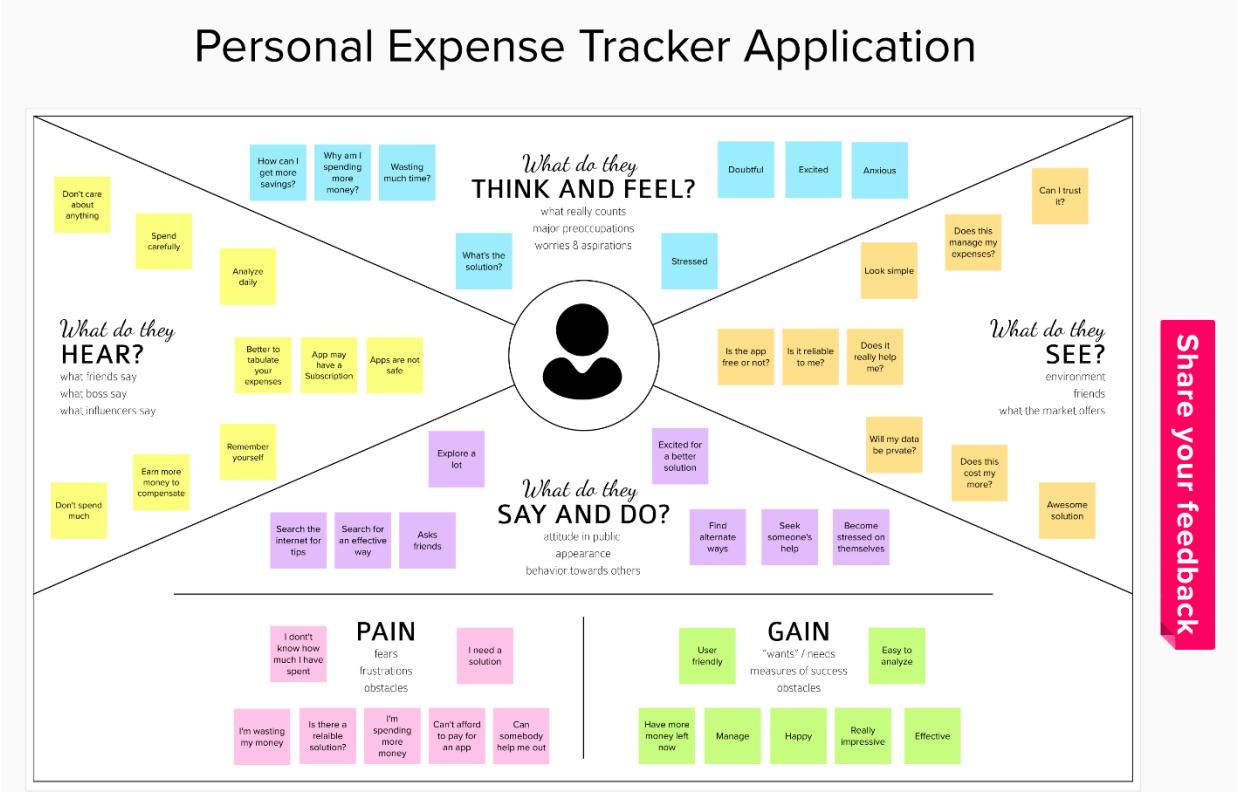
1. https://cloud.google.com/customers/spendee
2. https://www.wealthrocket.com/budgeting/spendee-revie

2.3 Problem Statement Definition

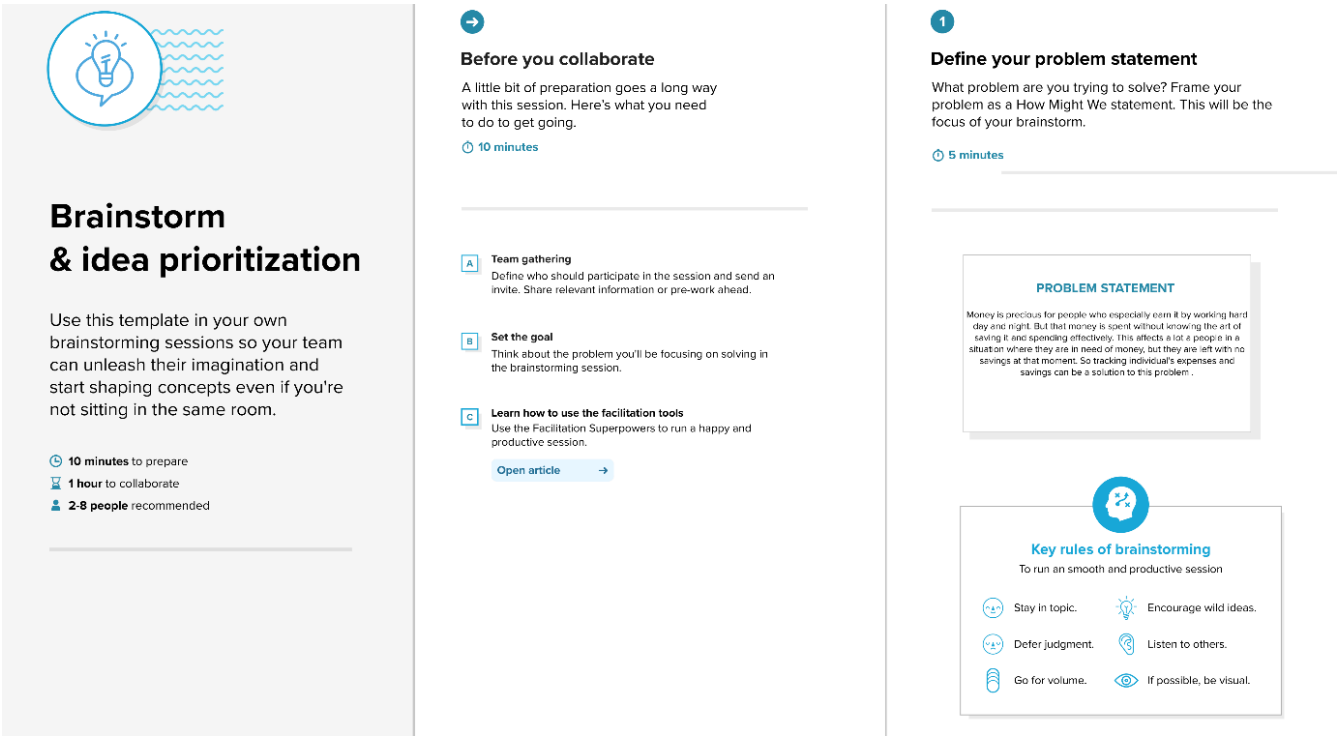


3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

Ganesh Kumar K

Track data	Analyze data	Visualize data
Make suggestions	User-friendly	Attractive interface
Notifications	Spend efficiently	Have more savings

Ashok Sandeep N B S

Perform calculations quickly	lagfree interface	interface should be catchy
handle large data	App servers should be active	Privacy must be maintained
Notify user at the right time	Improve savings	Spend for necessity

Alaguraja K

Track my expenses	Must alert me	Improve my savings
Notify me regularly	Keep track of my spendings	Alert me when I overspend
Analyze the data	Display my spendings visually	UI must be smooth

Santhosh V

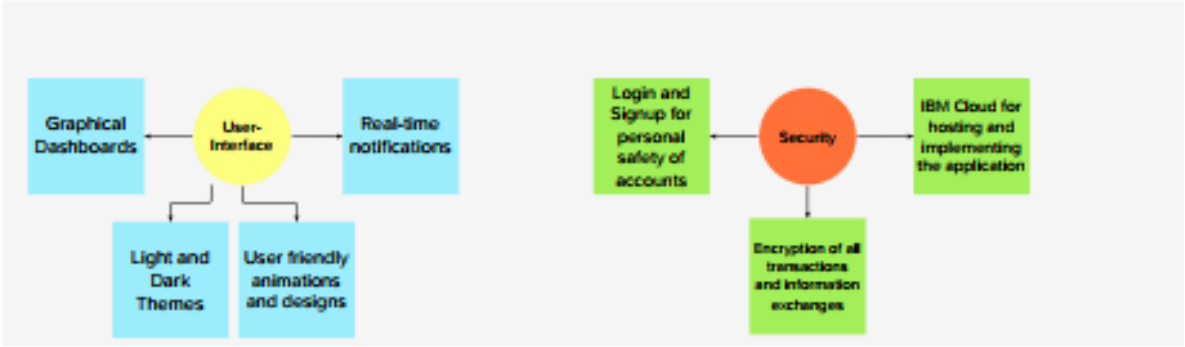
Track spendings and savings	Suggest ways to spend effectively	App with smooth UI
Alerts at the right time	Regularly notify	My data is private
App must be active	Send me mail alerts	Daily see my spendings

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

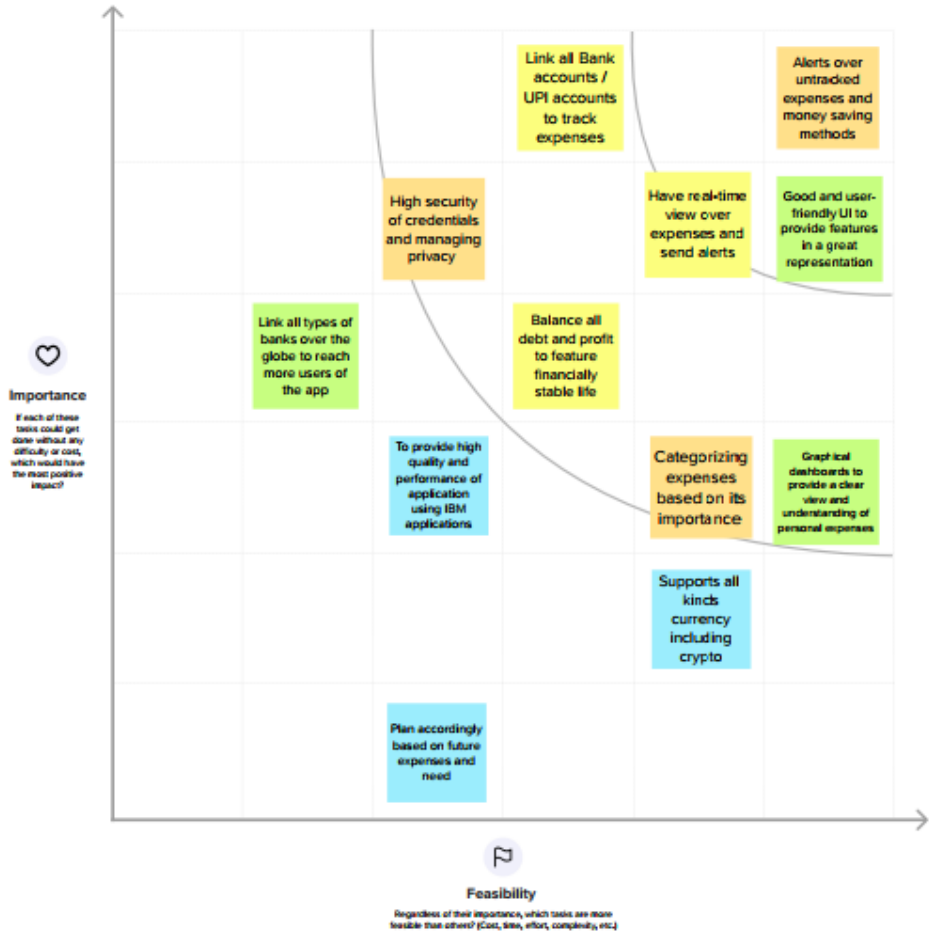


4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To simplify the expense tracking process
2.	Idea / Solution description	A web application for tracking expenses backed with IBM Cloud which sends notifications and insights using SendGrid
3.	Novelty / Uniqueness	Using 2FA which protects the expense data of the user and clean UI for easy navigation
4.	Social Impact / Customer Satisfaction	It improves the quality of spending of the user which results in better economic growth for our users
5.	Business Model (Revenue Model)	Ads on the platform and premium features that remove the ads for the premium user and more
6.	Scalability of the Solution	Using Kubernetes to manage the docker containers and create new pods whenever the traffic increases

3.4 Problem Solution fit

Project title : Personal Expense Tracker Application

Project Design Phase 1 - Solution Fit

Team ID : PNT2022TMID18300

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

Daily spenders who are working and making money.

A Person who is busy, doesn't care about spending on a regular basis.

College Students who wants to track their expense.

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

This solution offers visual insights on their expenditures.

Monitor the daily spending.

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem
or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

For keeping tabs on their spending, customers have used notes or paper.Developed in this project is a substitute is a personal expense tracker.

Manual calculations with no improving outcomes of the overall effects of spending patterns for review and improvement.

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

Method to alert them as soon as their budget limit is exceeded so they can be informed of their expenses.

Both the user's monthly income and monthly expenses must be added.

The amount that can be spent in a given month needs to be restricted.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

Because there are so many ways and places to spend money in the modern world, people need to be aware of when and how much they spend.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; Indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

The user chooses manual expense tracking over virtual application tracking.
So, they decide how much you want to save by the end of the month.Every time a purchase is made, an update is made.

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

Identify strong TR & EM

3. TRIGGERS TR

Increased debt
More unnecessary expenses
Unable to begin saving.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

Before:
More Stressed and anxious about unnoticable expenses.

After:
Manage expenses effectively and stress free.

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

With the use of this app, users may effectively manage and keep tabs on their spending and income.

Users have control over their finances due to periodic notifications and updates as well as the ability to access and analyse their records.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

Online:
Customer can learn more strategies to manage expenses and savings

Offline:
Think twice before taking important financial decisions.

Identify strong TR & EM

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

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

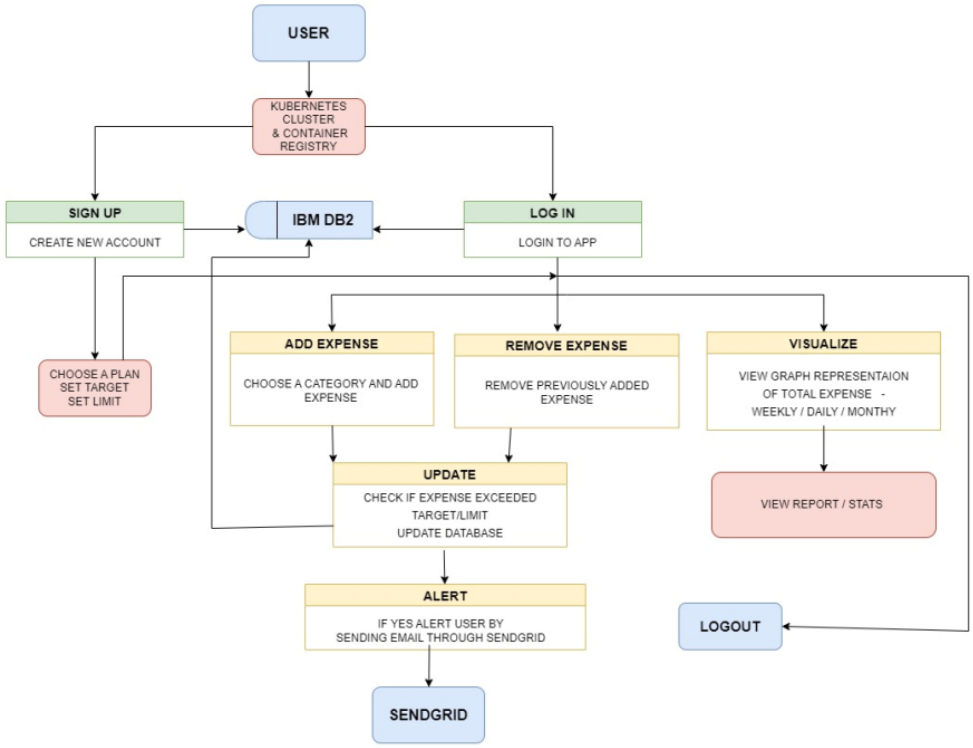
Non-functional Requirements:

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

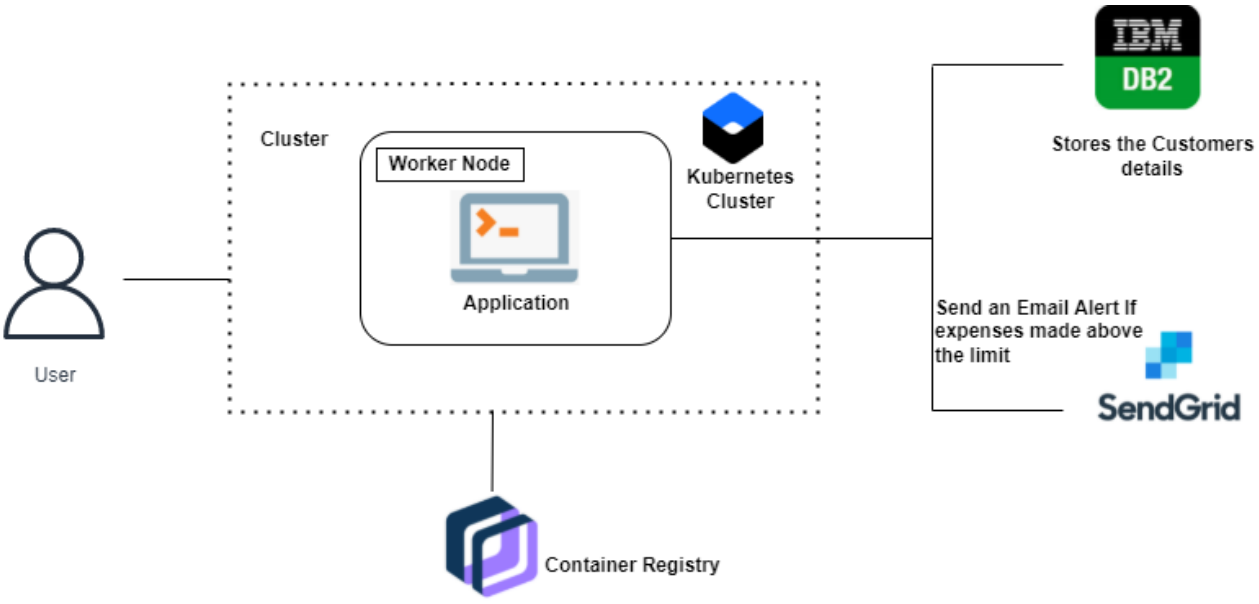
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	By using this application, the user can keep track of their expenses and can ensure that user's money is used wisely.
NFR-2	**Security**	Maintain user personal details in a encrypted manner by using data security algorithms .
NFR-3	**Reliability**	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



5.2 Solution & Technical Architecture



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.
	Login	USN-2	As a user, I can log into the application by entering email & password
	Add	USN -3	As a user , I can add in new expenses.
	Remove	USN – 4	As a user , I can remove previously added expenses.
	View	USN - 5	As a user , I can view my expenses in the form of graphs and get insights.
	Get alert message	USN - 6	As a user , I will get alert messages if I exceed my target amount.
Administrator	Add / remove user	USN – 7	As admin , I can add or remove user details on db2 manually.
		USN - 8	As admin , I can add or remove user details on sendgrid.

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

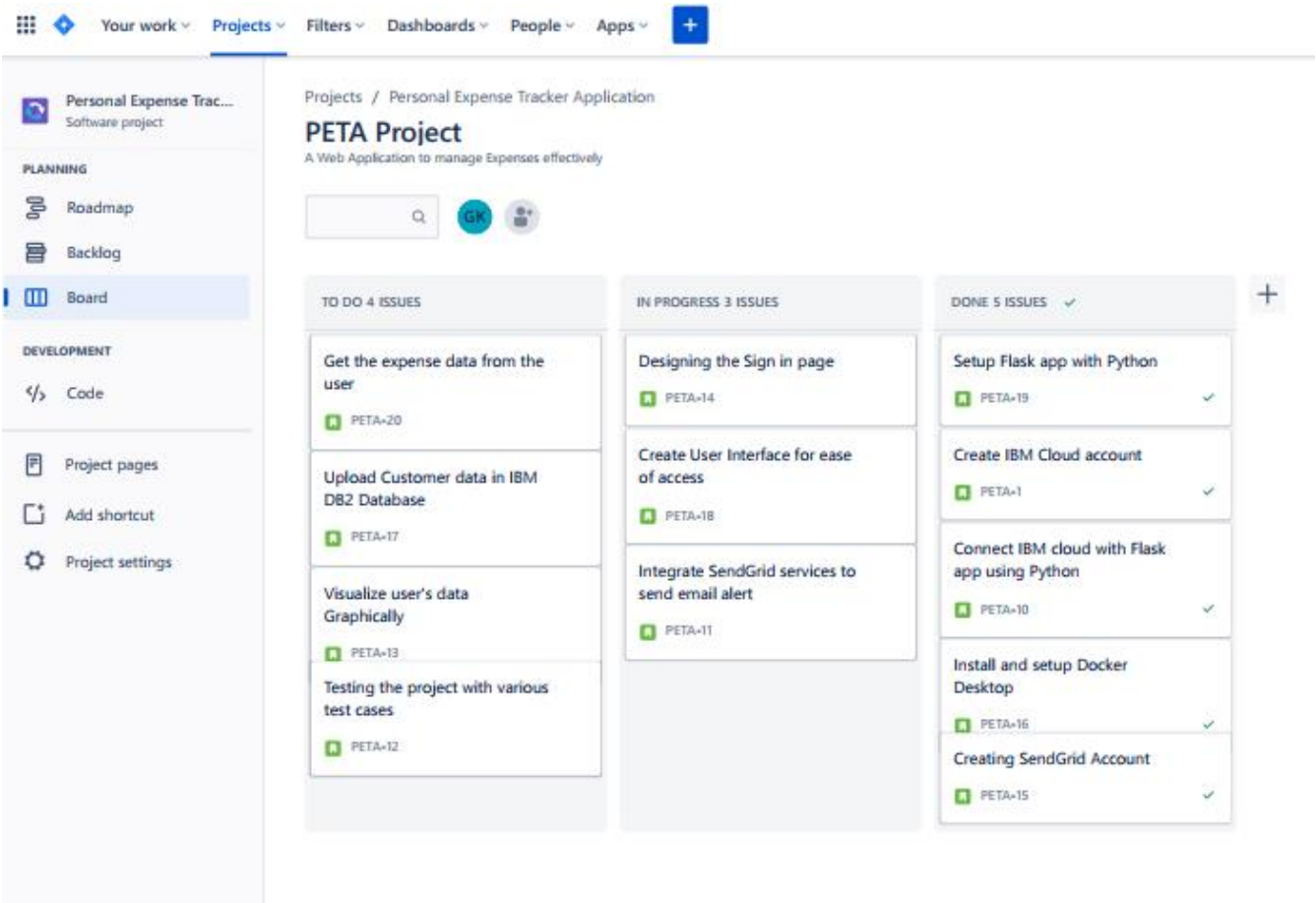
Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	I can sign up for the application as a user by providing my email address with password, and a password for confirmation.	2	High	2
Sprint-1		USN-2	When I register for the application as a user, I will get a confirmation email.	1	High	1
Sprint-1		USN-3	I can sign up for the application as a user through Gmail.	1	High	1
Sprint-1	Login	USN-4	I may access the program as a user by providing my email address and password.	3	High	3

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint -2	Login	USN-5	After logging in, I as a user, must be able to update my profile by adding all thenecessary information.	4	High	4
Sprint-3	Income update	USN-6	I can add wallet balance and add income as a user.	4	Medium	4
Sprint-3	Income update	USN-7	I can update all income in the wallet as a user.	2	Medium	2
Sprint-4	Expense update	USN-8	I can add wallet balance, add or remove expenses as a user.	3	High	3
Sprint-1	Expense update	USN-9	I can add Categories for expenses.	2	High	2
Sprint-2	Graphical Representation	USN-10	I can get my expenses in a graphical representation.	4	High	4
Sprint-3	improvisation	USN-11	Application is tested for improvisation and bugs to provide Quality of service to the user.	3	High	3
Sprint-4	Output	USN-12	I can protect my privacy as a user with the aid of a username and password.	3	High	4

6.2 Reports from JIRA



7. CODING & SOLUTIONING

7.1 FEATURE 1

Python Code

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

app = Flask(__name__)

app.secret_key = 'a'

app.config['database'] = 'bludb'
app.config['hostname'] = '19af6446-6171-4641-8aba-
9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud'
app.config['port'] = '30699'
app.config['protocol'] = 'tcpip'
app.config['uid'] = 'wdc34377'
app.config['pwd'] = 'GSX4f44l31U8e7sb'
app.config['security'] = 'SSL'
try:
    mysql = DB2(app)
    conn_str='database=bludb;hostname=19af6446-6171-4641-8aba-
9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;po
rt=30699;protocol=tcpip;\
        uid=wdc34377;pwd=GSX4f44l31U8e7sb;security=SSL'
    ibm_db_conn = ibm_db.connect(conn_str,'','')
```

```
    print("Database connected without any error !!")
except:
    print("IBM DB Connection error   :     " + DB2.conn_errormsg())

#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():
    if request.method == "POST":
        user_name = request.form['username']
        email = request.form['email']
        pass_word = request.form['password']
        query = "INSERT INTO Admin (username,email,password) values
(?,?,?)"
        insert_stmt = ibm_db.prepare(ibm_db_conn, query)
        ibm_db.bind_param(insert_stmt, 1, user_name)
        ibm_db.bind_param(insert_stmt, 2, email)
        ibm_db.bind_param(insert_stmt, 3, pass_word)
        ibm_db.execute(insert_stmt)
        msg = 'Account Created Successfully'
```

```
        return render_template("signup.html", msg=msg)

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

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



    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']

        sql = "SELECT * FROM Admin WHERE username = ? and
password = ?"
        stmt = ibm_db.prepare(ibm_db_conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        result = ibm_db.execute(stmt)
        print(result)
        account = ibm_db.fetch_row(stmt)
        print(account)

        param = "SELECT * FROM Admin WHERE username = " + "\'" +
username + "\'" + " and password = " + "\'" + password + "\'"
        res = ibm_db.exec_immediate(ibm_db_conn, param)
        dictionary = ibm_db.fetch_assoc(res)
```

```
        # sendmail("hello sakthi","sivasakthisairam@gmail.com")

        if account:
            session['loggedin'] = True
            session['id'] = dictionary["ID"]
            userid = dictionary["ID"]
            session['username'] = dictionary["USERNAME"]
            session['email'] = dictionary["EMAIL"]

            return redirect('/home')
        else:
            msg = 'Incorrect username / password !'

    return render_template('login.html', msg = msg)

@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)
    sql = "INSERT INTO Expense (userid, date, expensename, amount,
paymode, category) VALUES (?,?, ?, ?, ?, ?)"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, session['id'])
    ibm_db.bind_param(stmt, 2, p4)
    ibm_db.bind_param(stmt, 3, expensename)
    ibm_db.bind_param(stmt, 4, amount)
    ibm_db.bind_param(stmt, 5, paymode)
    ibm_db.bind_param(stmt, 6, category)
    ibm_db.execute(stmt)

    print("Expenses added")

    # email part

    param = "SELECT * FROM Expense WHERE MONTH(date) =
MONTH(current timestamp) AND YEAR(date) = YEAR(current
timestamp) ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        # temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
```

```
        dictionary = ibm_db.fetch_assoc(res)

    total=0
    for x in expense:
        total += int(x[3])



    param = "SELECT userid, limit FROM limit WHERE userid = " +
str(session['id'])
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = 0
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMIT"])
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[len(temp)-1]

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

    return redirect("/display")

    #DISPLAY---graph

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

```
    param = "SELECT * FROM Expense WHERE userid = " +
str(session['id']) + " ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        # temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

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

#delete---the--data

@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
def delete(id):
    param = "DELETE FROM Expense WHERE  userid = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)

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

#UPDATE---DATA

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

```
def edit(id):
    param = "SELECT * FROM Expense WHERE  userid = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    while dictionary != False:
        temp = []
        # temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        row.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

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

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

      date = request.form['date']
      expensename = request.form['expensename']
      amount = request.form['amount']
      paymode = request.form['paymode']
      category = request.form['category']
      p1 = date[0:10]
      p2 = date[11:13]
      p3 = date[14:]
```

```
    p4 = p1 + "-" + p2 + "." + p3 + ".00"

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

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


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

@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
   if request.method == "POST":
       number= request.form['number']
      #  cursor = mysql.connection.cursor()
      #  cursor.execute('INSERT INTO limits VALUES (NULL, % s, % s)
',(session['id'], number))
      #  mysql.connection.commit()

       sql = "INSERT INTO limit (userid, limit) VALUES (?, ?)"
       stmt = ibm_db.prepare(ibm_db_conn, sql)
```

```
        ibm_db.bind_param(stmt, 1, session['id'])
        ibm_db.bind_param(stmt, 2, number)
        ibm_db.execute(stmt)
        return redirect('/limitn')

@app.route("/limitn")
def limitn():
    # cursor = mysql.connection.cursor()
    # cursor.execute('SELECT limitss FROM `limits` ORDER BY
`limits`.`id` DESC LIMIT 1')
    # x= cursor.fetchone()
    # s = x[0]

    param = "SELECT userid,limit FROM limit WHERE userid = " +
str(session['id'])
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    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)

#REPORT

@app.route("/today")
```

```
def today():
    #   cursor = mysql.connection.cursor()
    #   cursor.execute('SELECT TIME(date)   , amount FROM expenses
WHERE userid = %s AND DATE(date) = DATE(NOW())
',(str(session['id'])))
    #   texpense = cursor.fetchall()
    #   print(texpense)

     param1 = "SELECT TIME(date) as tn, amount FROM Expense
WHERE userid = " + str(session['id']) + " AND DATE(date) =
DATE(current timestamp) ORDER BY date DESC"
     res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
     dictionary1 = ibm_db.fetch_assoc(res1)
     texpense = []

     while dictionary1 != False:
        temp = []
        temp.append(dictionary1["TN"])
        temp.append(dictionary1["AMOUNT"])
        texpense.append(temp)
        print(temp)
        dictionary1 = ibm_db.fetch_assoc(res1)

    #   cursor = mysql.connection.cursor()
    #   cursor.execute('SELECT * FROM expenses WHERE userid = % s
AND DATE(date) = DATE(NOW()) AND date ORDER BY
`expenses`.`date` DESC',(str(session['id'])))
    #   expense = cursor.fetchall()

     param = "SELECT * FROM Expense WHERE userid = " +
str(session['id']) + " AND DATE(date) = DATE(current timestamp)
ORDER BY date DESC"
     res = ibm_db.exec_immediate(ibm_db_conn, param)
```

```
dictionary = ibm_db.fetch_assoc(res)
expense = []
while dictionary != False:
    temp = []
  #   temp.append(dictionary["ID"])
    temp.append(dictionary["USERID"])
    temp.append(dictionary["DATE"])
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    expense.append(temp)
    print(temp)
    dictionary = ibm_db.fetch_assoc(res)

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

for x in expense:
    total += int(x[3])
    if x[5] == "food":
        t_food += int(x[3])

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

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

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

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

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

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)
    return render_template("today.html", texpense = texpense,
expense = expense,  total = total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )


@app.route("/month")
def month():
   #   cursor = mysql.connection.cursor()
   #   cursor.execute('SELECT DATE(date), SUM(amount) FROM
expenses WHERE userid= %s AND MONTH(DATE(date))=
MONTH(now()) GROUP BY DATE(date) ORDER BY DATE(date)
',(str(session['id'])))
   #   texpense = cursor.fetchall()
   #   print(texpense)
```

```
    param1 = "SELECT DATE(date) as dt, SUM(amount) as tot FROM
Expense WHERE userid = " + str(session['id']) + " AND MONTH(date)
= MONTH(current timestamp) AND YEAR(date) = YEAR(current
timestamp) GROUP BY DATE(date) ORDER BY DATE(date)"
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpense = []

    while dictionary1 != False:
        temp = []
        temp.append(dictionary1["DT"])
        temp.append(dictionary1["TOT"])
        texpense.append(temp)
        print(temp)
        dictionary1 = ibm_db.fetch_assoc(res1)



  #  cursor = mysql.connection.cursor()
  #  cursor.execute('SELECT * FROM expenses WHERE userid = % s
AND MONTH(DATE(date))= MONTH(now()) AND date ORDER BY
`expenses`.`date` DESC',(str(session['id'])))
  #  expense = cursor.fetchall()

    param = "SELECT * FROM Expense WHERE userid = " +
str(session['id']) + " AND MONTH(date) = MONTH(current
timestamp) AND YEAR(date) = YEAR(current timestamp) ORDER BY
date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
```

```
  #   temp.append(dictionary["ID"])
    temp.append(dictionary["USERID"])
    temp.append(dictionary["DATE"])
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    expense.append(temp)
    print(temp)
    dictionary = ibm_db.fetch_assoc(res)


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


for x in expense:
    total += int(x[3])
    if x[5] == "food":
        t_food += int(x[3])

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

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

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

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

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)



    return render_template("today.html", texpense = texpense,
expense = expense,  total = total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )

@app.route("/year")
def year():
   #   cursor = mysql.connection.cursor()
   #   cursor.execute('SELECT MONTH(date), SUM(amount) FROM
expenses WHERE userid= %s AND YEAR(DATE(date))= YEAR(now())
GROUP BY MONTH(date) ORDER BY MONTH(date)
',(str(session['id'])))
   #   texpense = cursor.fetchall()
   #   print(texpense)
```

```
    param1 = "SELECT MONTH(date) as mn, SUM(amount) as tot
FROM Expense WHERE userid = " + str(session['id']) + " AND
YEAR(date) = YEAR(current timestamp) GROUP BY MONTH(date)
ORDER BY MONTH(date)"
    res1 = ibm_db.exec_immediate(ibm_db_conn, param1)
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpense = []

    while dictionary1 != False:
       temp = []
       temp.append(dictionary1["MN"])
       temp.append(dictionary1["TOT"])
       texpense.append(temp)
       print(temp)
       dictionary1 = ibm_db.fetch_assoc(res1)



  #   cursor = mysql.connection.cursor()
  #   cursor.execute('SELECT * FROM expenses WHERE userid = % s
AND YEAR(DATE(date))= YEAR(now()) AND date ORDER BY
`expenses`.`date` DESC',(str(session['id'])))
  #   expense = cursor.fetchall()

    param = "SELECT * FROM Expense WHERE userid = " +
str(session['id']) + " AND YEAR(date) = YEAR(current timestamp)
ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
       temp = []
     #   temp.append(dictionary["ID"])
```

```
    temp.append(dictionary["USERID"])
    temp.append(dictionary["DATE"])
    temp.append(dictionary["EXPENSENAME"])
    temp.append(dictionary["AMOUNT"])
    temp.append(dictionary["PAYMODE"])
    temp.append(dictionary["CATEGORY"])
    expense.append(temp)
    print(temp)
    dictionary = ibm_db.fetch_assoc(res)

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


for x in expense:
    total += int(x[3])
    if x[5] == "food":
        t_food += int(x[3])

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

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

    elif x[5] == "EMI":
```

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

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

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)
    return render_template("today.html", texpense = texpense,
expense = expense,  total = total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )

#log-out

@app.route('/logout')
def logout():
  session.pop('loggedin', None)
  session.pop('id', None)
  session.pop('username', None)
  session.pop('email', None)
  return render_template('home.html')

app.run(debug=True)
```

7.2 FEATURE 1

HTML Code (Home.html)

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

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">


  <link rel="stylesheet" href="..\static\css\home.css">
  <title>My Website</title>
</head>

<body>
  <!-- Header -->
  <section id="header">
    <div class="header container">
      <div class="nav-bar">
        <div class="brand">
          <a href="#hero">
            <h1><span>B</span>udget <span>T</span>racker</h1>
          </a>
        </div>
        <div class="nav-list">
          <div class="hamburger">
            <div class="bar"></div>
          </div>
          <ul>
```

```
        <li><a href="#hero" data-after="Home">Home</a></li>
        <li><a href="#services" data-after="Service">Services</a></li>

        <li><a href="#about" data-after="About">About</a></li>
        <li><a href="#contact" data-after="Contact">Contact</a></li>
        <LI><a href="/signin" data-after="Login">-Login-</a></LI>
       </ul>
     </div>
    </div>
  </div>
</section>
<!-- End Header -->


<!-- Hero Section  -->
<section id="hero">
  <div class="hero container">
    <div>
      <h1>Hello, <span></span></h1>
      <h1>Welcome To <span></span></h1>
      <h1>Personal Expense Tracker App <span></span></h1>
      <a href="/signup" type="button" class="cta">Sign-up</a>
    </div>
  </div>
</section>
<!-- End Hero Section  -->

<!-- Service Section -->
<section id="services">
  <div class="services container">
    <div class="service-top">
      <h1 class="section-title">Serv<span>i</span>ces</h1>
      <p>Budget Tracker provides a many services  to the customer and industries.
```

```
Financial solutions to meet your needs whatever your money goals,there is a
Budget solution to help you reach them </p>
    </div>
    <div class="service-bottom">
     <div class="service-item">
      <div class="icon"><img
src="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
      <h2>Personal Expenses</h2>
      <p>Budgeting is more than paying bills and setting aside savings.it's about
creating a money plan for the life you want</p>
     </div>
     <div class="service-item">
      <div class="icon"><img
src="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
      <h2>Investments</h2>
      <p>Follow your investments and bring your portfolio into focus with support
for stocks,bonds,CDs,mutual funds and more</p>
     </div>
     <div class="service-item">
      <div class="icon"><img
src="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
      <h2>Online Banking</h2>
      <p>Budget Tracker application can automatically download transactions and
send payments online from many financial institutions</p>
     </div>
     <div class="service-item">
      <div class="icon"><img
src="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
      <h2>Financial Life</h2>
      <p>Get your Complete financial picture at a glance. With Budget Tracker
application you can view your all the financial activities
      </p>
     </div>
    </div>
```

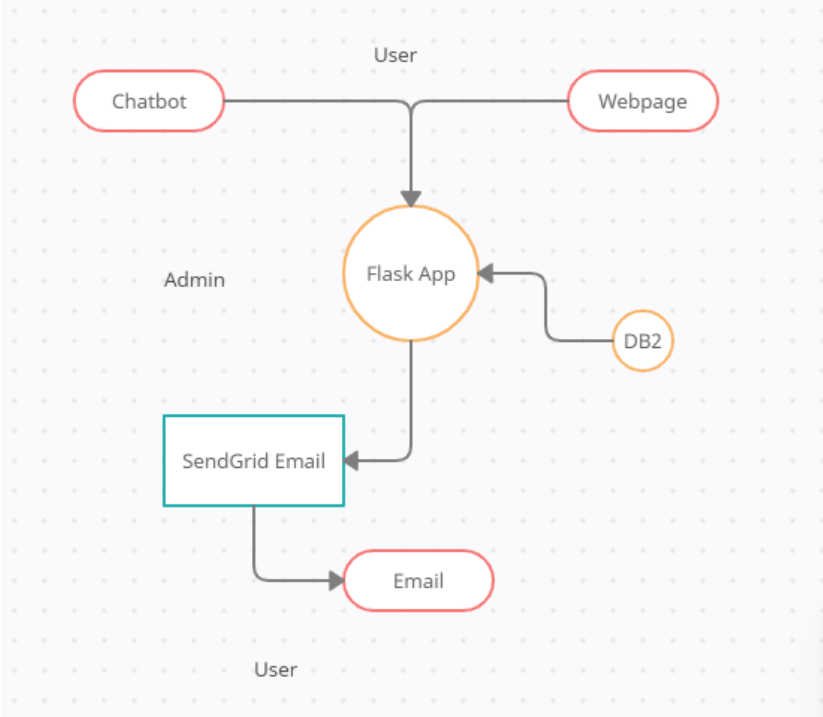
```
  </div>
 </section>
 <!-- End Service Section -->

 <!-- About Section -->
 <section id="about">
  <div class="about container">
   <div class="col-left">
    <div class="about-img">
     <img src="..\static\images\marillyn-hewson.jpg" alt="img">
     <div><h2>Expense Tracker</h2></div>
    </div>
   </div>

   <div class="col-right">
    <h1 class="section-title">About <span>Us</span></h1>
    <h2>Financial Solution</h2>
    <p>Budget Tracker financial solution  is one among Leading financial company from many years.Budget Tracker provides a many services  to the customer and industries. Financial solutions to meet your needs whatever your money goals,there is a MyBudget solution to help you reach them.u can Contact our service center for further information and also follow our social media for update on new services </p>
     <a href="#footer" class="cta">Follow Us</a>

   </div>
  </div>
 </section>
 <!-- End About Section -->

 <!-- Contact Section -->
 <section id="contact">
  <div class="contact container">
   <div>
```

```
      <h1 class="section-title">Contact <span>info</span></h1>
     </div>
     <div class="contact-items">
      <div class="contact-item">
       <div class="icon"><img
src="https://img.icons8.com/bubbles/100/000000/phone.png" /></div>
       <div class="contact-info">
        <h1>Phone</h1>
        <h2>+91 123456789</h2>
       </div>
      </div>
      <div class="contact-item">
       <div class="icon"><img
src="https://img.icons8.com/bubbles/100/000000/new-post.png" /></div>
       <div class="contact-info">
        <h1>Email</h1>
        <h2>sample@gmail.com</h2>
       </div>
      </div>
      <div class="contact-item">
       <div class="icon"><img
src="https://img.icons8.com/bubbles/100/000000/map-marker.png" /></div>
       <div class="contact-info">
        <h1>Address</h1>
        <h2>Tamil Nadui, India</h2>
       </div>
      </div>
     </div>
    </div>
  </section>
  <!-- End Contact Section -->

  <!-- Footer -->
```

```
  <section id="footer">
    <div class="footer container">
      <div class="brand">
        <h1><span>B</span>udget <span>T</span>racker</h1>
      </div>
      <h2>Your Complete Financial Solution</h2>
          <p>Team ID: PNT2022TMID18300</p>
      <p>All rights reserved</p>
      <p>Copyright © 2019-2023 </p>
    </div>
  </section>
  <!-- End Footer -->
  <script src="..\static\js\home.js"></script>
</body>

</html>
```

8. TESTING

8.1 Test Cases

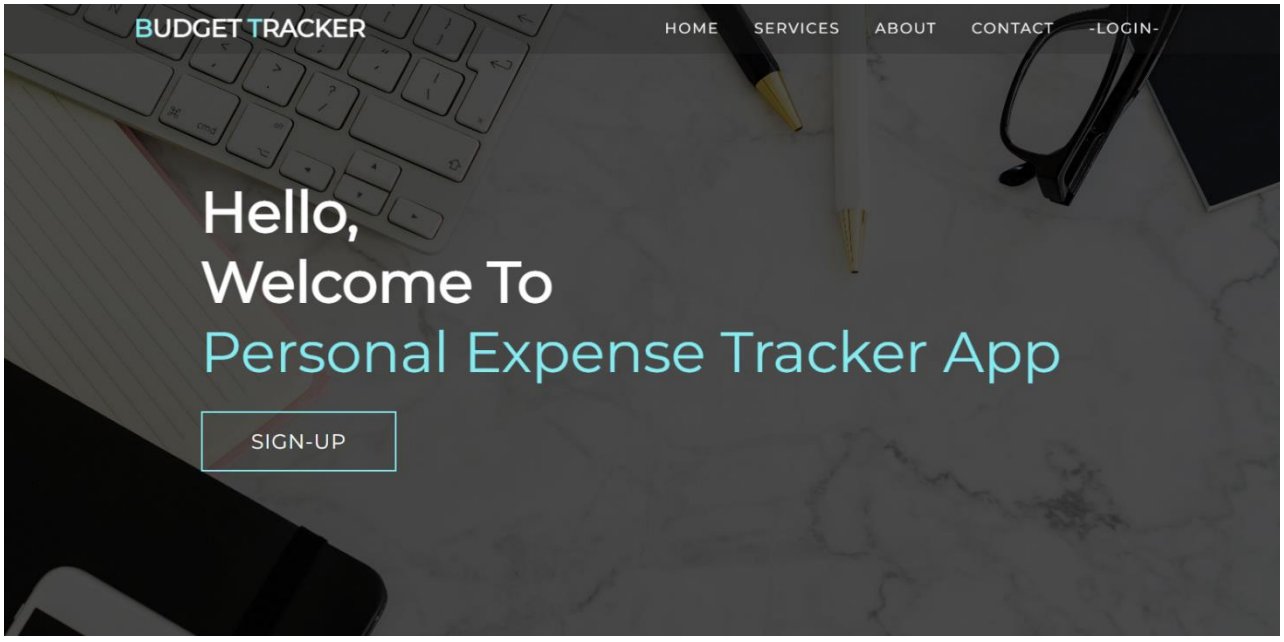


Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	5	0	0	5
Client Application	12	0	0	12
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	1	0	0	1
Final Report Output	2	0	0	2
Version Control	2	0	0	2

9. RESULTS



SERVICES

Budget Tracker provides a many services to the customer and industries. Financial solutions to meet your needs whatever your money goals,there is a Budget solution to help you reach them

PERSONAL EXPENSES

Budgeting is more than paying bills and setting aside savings.it's about creating a money plan for the life you want

INVESTMENTS

Follow your investments and bring your portfolio into focus with support for stocks,bonds,CDs,mutual funds and more

ONLINE BANKING

Budget Tracker application can automatically download transactions and send payments online from many financial institutions

FINANCIAL LIFE

Get your Complete financial picture at a glance. With Budget Tracker application you can view your all the financial activities

Expense Tracker

ABOUT US

Financial Solution

Budget Tracker financial solution is one among Leading financial company from many years.Budget Tracker provides a many services to the customer and industries. Financial solutions to meet your needs whatever your money goals,there is a MyBudget solution to help you reach them.u can Contact our service center for further information and also follow our social media for update on new services

FOLLOW US

CONTACT INFO

Phone
+91 123456789

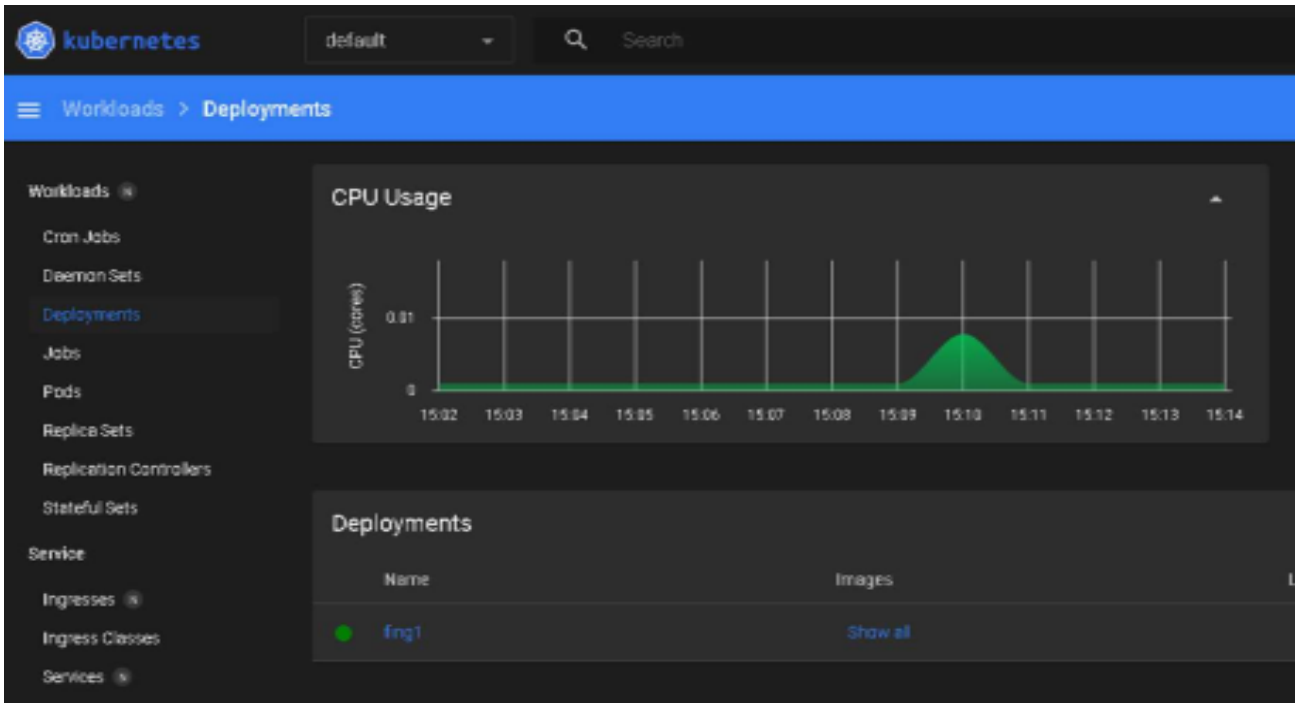
Email
sample@gmail.com

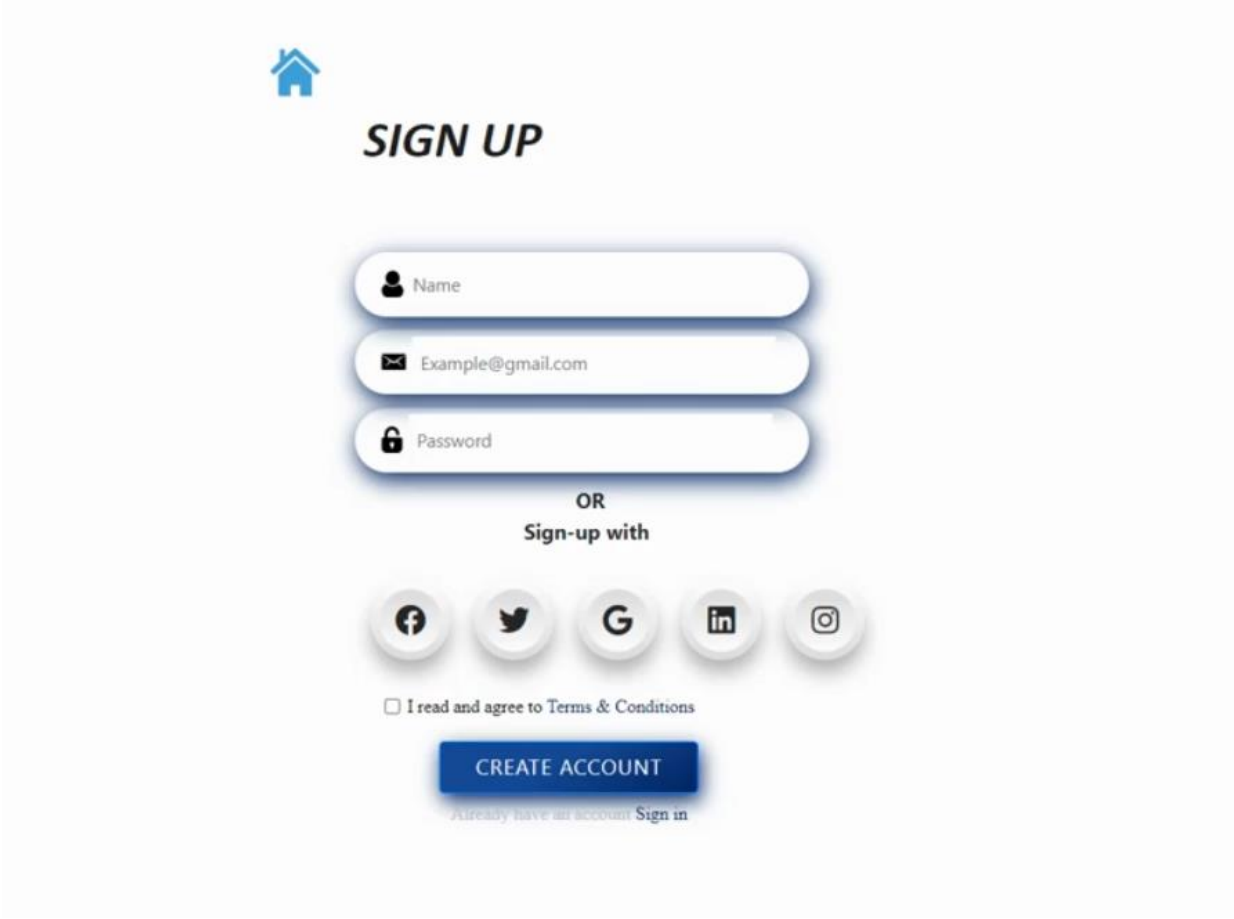
Address
Tamil Nadui, India

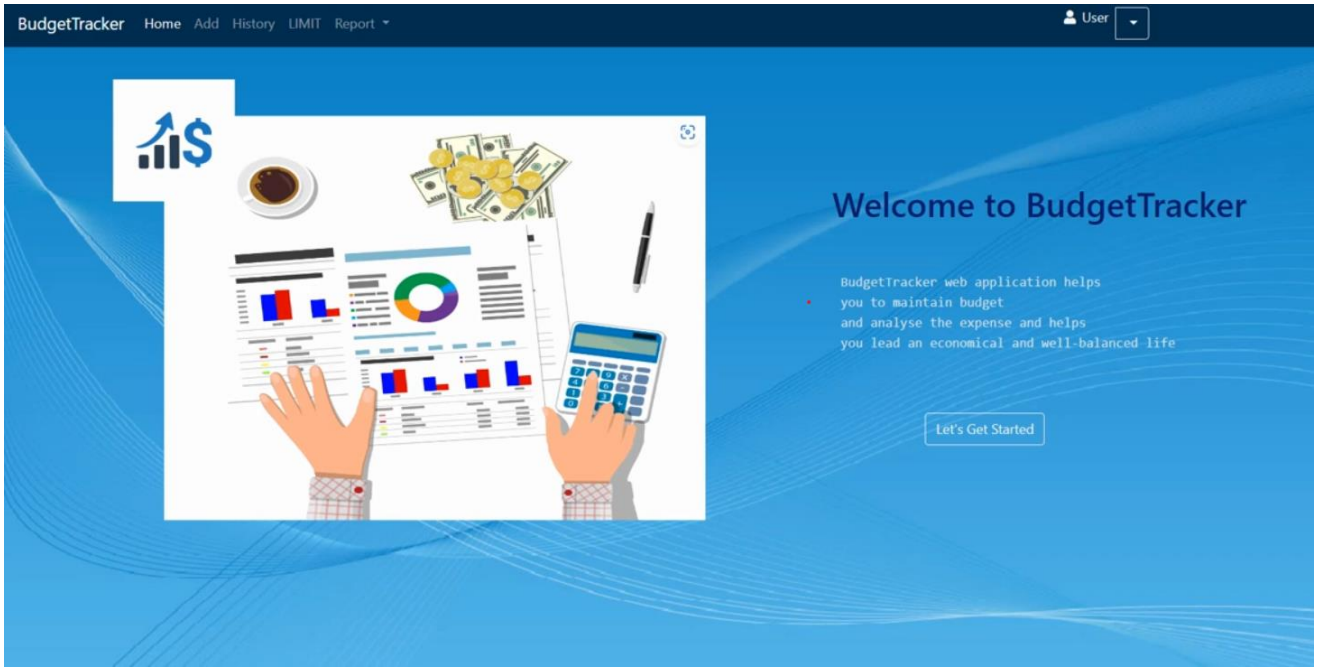
BUDGET TRACKER

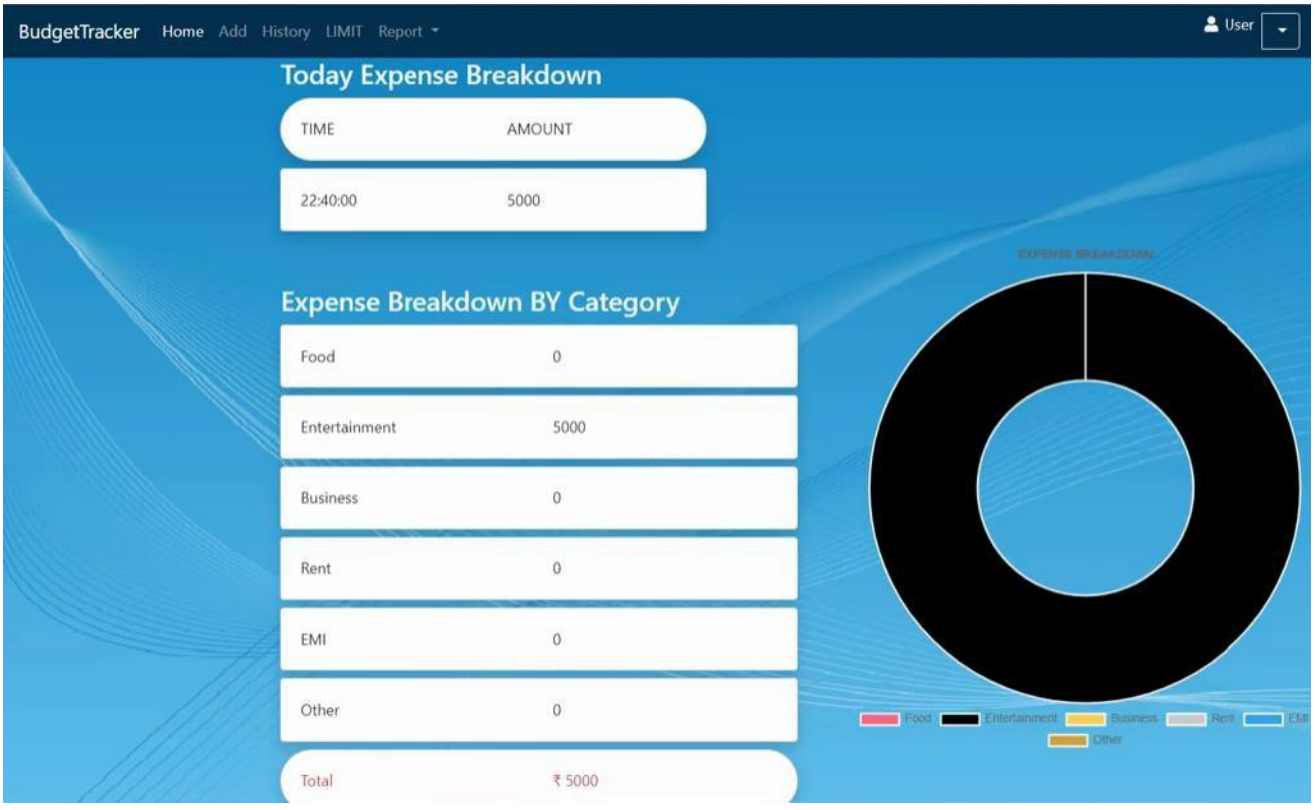
Your Complete Financial Solution

Team ID: PNT2022TMID18300


kubernetes
default
Search
Workloads > Deployments
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Service
Ingresses
Ingress Classes
Services
CPU Usage
CPU (cores)
0.01
0
15:02
15:03
15:04
15:05
15:06
15:07
15:08
15:09
15:10
15:11
15:12
15:13
15:14
Deployments
Name
Images
fng1
Show all


SIGN UP
Name
Example@gmail.com
Password
OR
Sign-up with
I read and agree to Terms & Conditions
CREATE ACCOUNT
Already have an account Sign in


BudgetTracker
Home
Add
History
LIMIT
Report
User
Welcome to BudgetTracker
BudgetTracker web application helps
you to maintain budget
and analyse the expense and helps
you lead an economical and well-balanced life
Let's Get Started


BudgetTracker
Home
Add
History
LIMIT
Report
User
Today Expense Breakdown
TIME
AMOUNT
22:40:00
5000
Expense Breakdown BY Category
Food
0
Entertainment
5000
Business
0
Rent
0
EMI
0
Other
0
Total
₹ 5000
Food
Entertainment
Business
Rent
EMI
Other

10. ADVANTAGES & DISADVANTAGES

Advantages

1. Helps anticipate the costs of similar projects When you formally track and report expenses, you have a permanent documentation which helps you correctly anticipate expenses for similar projects in the future. This is even more significant when it comes to budget-making process.

2. The best organizations have a way of tracking and handling these reimbursements. This ideal practice guarantees that the expenses tracked are accurately and in a timely manner. From a company perspective, timely settlements of these expenses when tracked well will certainly boost employees' morale.

3. Tracking the amount of money spent on the projects is important to invoice customers and determine the cost & profitability analysis when your company is providing services to another company. On the other hand, expense tracking or internal project is important for cost and ROI calculation.

Disadvantages

1. 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.

2. Requires active internet connection to add, remove or view expense data.

3. Can be affected by server issues or webpage issues occasionally.

4. Unraveling the complexity of your financial records by hand may be time consuming. Since it takes time to generate reports.

11. CONCLUSION & FUTURE SCOPE

CONCLUSION

In conclusion, developing a personal budget and tracking all expenses and spending is a crucial aspect of personal finances. It can help ordinary people as well as Businesspeople to manage their spendings and savings in an efficient way. There can be a significant difference in the amount of savings made after using the personal expense tracker application.

FUTURE SCOPE

The main objective of this project is to help the user to avoid unexpected expenses and bad financial situations. In future this can be developed to perform the following operations for better results.

- Enable the notification system for the user to get notification daily at a specific time that can help the user insert expenses and income.
- Backup and restore all information.
- Generate report in PDF format either category wise or by time period.

12. APPENDIX

GitHub Repository Link - TEAM ID: PNT2022TMID18300

https://github.com/IBM-EPBL/IBM-Project-1321-1658384094

GitHub Final Deliverables Link:

https://github.com/IBM-EPBL/IBM-Project-1321-1658384094/tree/main/Final%20Deliverables

Project Demo Video Link:

https://drive.google.com/file/d/1ybuIUCH1gbDbui2oGxdsqwe166pBv5_k/view?usp=share_link