 **PERSONAL EXPENSES TRACKER** 

## A PROJECT REPORT

***Submitted by***

**MADHUSUDHAN NS(19EUEC076)**

**MANISHKUMAR V (19EUEC078)**

**MANANITHYANANDHAM T M (19EUEC077)**

**MASSODH M (19EUEC080)**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

***in***

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(An Autonomous Institution, Affiliated to Anna University Chennai - 600 025)**

### NOVEMBER 2022

**BONAFIDE CERTIFICATE**

Certified that this project report titled **“PERSONAL EXPENSES TRACKER”** is the bonafide work of **MADHUSUDHAN NS (19EUEC076),MANISHKUMAR V (19EUEC078),MANANITHYANANDHAM (19EUEC077), MASOODH (19EUEC080)** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**Dr.S. SASI PRIYA D.DEVI**

### HEAD OF THE DEPARTMENT SUPERVISOR

Department of Electronics and Communication Engineering

Sri Krishna College of Engineering and Technology Kuniamuthur,Coimbatore

**Submitted for the Project viva-voce examination held on\_\_\_\_\_\_\_\_\_\_\_**

**INTERNAL EXAMINER EXTERNALEXAMINER**

**ABSTRACT**

Tracking regular expense is a key factor to maintain a budget. People often track expense using pen and paper method or take notes in a mobile phone or a computer. These processes of storing expense require further computations and processing for these data to be used as a trackable record. In this work, we are proposing an automated system to store and calculate these data. It is an application that runs on Android smartphones. By using this application, users can save their expense by simply scanning the bills or receipt copies. This application extracts the textual information from the receipts and saves the amount and description for further processing. It also monitors user's income by tracking the received SMS's from the user's saving accounts. By calculating income and expense it produces the user's balance in monthly and yearly basis. Overall, this is a smart automated solution for tracking expense.

**1 INTRODUCTION**

**PROJECT OVERVIEW :**

When it comes to tracking expenses, you can make your system as simple as collecting receipts and organizing them once a month.

You might get a little more information from other expense tracking systems (listing them in a spreadsheet, using money management software or even choosing an online application), but all methods have one thing in common: you have to get in the habit of thinking about your expenses.

It’s very easy to misplace a receipt or forget about any cash you spent. You may even think that a cup of coffee or a trip to the vending machine isn’t worth tracking — although those little expenses can add up amazingly fast.

There are all sorts of opportunities to throw a kick into your plan to track expenses. You have to get in the habit of doing so, to reduce those lapses, and make sure that the data you’re basing financial decisions on is solid.

This project will request the clients to add their expenses and in view of their costs, wallet status will be refreshed which will be noticeable to the client.

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

* Setting up Application Environment
* Create Flask project
* Work with IBM Cloud CLI, Docker CLI, Sendgrid
* Implementation of Web Application
* Create UI to Interact with the application
* Connect IBM DB2 with Python
* Integration of Sendgrid Service with Python
* Deployment of Cloud Application
* Containerize the application
* Upload Image in IBM container directory
* Deploy on Kubernetes Cluster

**1.1 PURPOSE :**

* Help the people to track their expenses.
* Alert users when they exceed the limit of their budget.
* A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about financial management

**2 LITERATURE SURVEY :**

**2.1 EXISTING PROBLEM :**

* Lack of visual analytics for visual data.
* Lack of support for splitting up group expenses.
* Most of the applications are used only for personal use.
* Most of the applications does not incorporate shared group expenses.
* Efforts has to be made to include each and every transactions into the input field.

**2.2 REFERENCES :**

**SPENDING TRACKER :** A Smart Approach to Track Daily Expenses

**Authors:** UP Singh, AK Gupta, Dr. B. Balamurugan

**Description:**

In this paper, a Java GUI based application was proposed to assure that it will Help its users to manage the cost of their daily expenditure. It will guide them and aware them of their daily expenses. The proposed design contained the basic modules for Adding and viewing expenses, managing expense categories. Supports CRUD Operations on expense data.

**Year: 2021**

**Technologies:** Java

**EXPENSE TRACKER APPLICATION**

**Authors:** Velmurugan.R, Mrs.P.Usha

**Description:**

This is an android based application that allows users to maintain a computerized Diary to track expenses on a day-to-day basis to stay on budget and know expenses That are represented via a graphical representation with special features of distributing Expenses in different categories suitable for the user.

**Year:** 2021

**Technologies:** Java, XML, and MySQL

**STUDENT EXPENSE TRACKING APPLICATION**

**Authors:** Saumya Dubey , Pragya Dubey , Rigved Rishabh Kumar , Aaisha Khatoon

**Description:**

This is an android application which is used to track the daily expenses of a Student. It is like a digital diary that keeps a record of expenses done by a student. The Application keeps track of money spent and the earnings of both of the students on a Day-to-day basis. It also has the feature that it gives warning messages if we are Exceeding our expenses and hence, we can limit our expenses and avoid Overspending. If you spend less money than the daily expense allowed amount, the Money left after spending is added into the user’s savings.

**Year:** 2022

**Technologies:** Java

**EXPENSE TRACKER USING STATISTICAL ANALYSIS**

**Authors:** Muskaan Sharma, Ayush Bansal, Dr. Raju Ranjan, Shivam Sethi

**Description:**

In this paper, an approach has been proposed on how to efficiently manage house-old budget. This application will allow users to keep track of their expenses. This novel expense tracker uses statistical analysis which is going to keep a track of your expenses and would even give you results accordingly.

**Year:** 2021

**Technologies:** Java

**BUDGET ESTIMATOR ANDROID APPLICATION**

**Authors:** Namita Jagtap, Priyanka Joshi, Aditya Kamble

**Description:**

The system known as Budget Estimator is designed to manage the application user ‘s daily expenses in a more efficient and manageable way. This project is about mobile application Expenses system with geo-location tracking, based on the location of the user, it using Google Places, to check, the available store in the area, provides a notification for offers purpose, In term of security design, this system may implement a login authentication such as OTP message to your mobile device, this function may bring more security confidence to user. To reduce manual calculations, we propose an application which is developed by android. This application allows users to maintain a digital automated diary.

**Year:** 2021

**Technologies:** Java

**SAWANT-EXPENSE TRACKER**

**Authors:** Atiya Kazi, Praphulla S. Kherade , Raj S. Vilankar , Parag M

**Description:**

In this approach, the application keeps track of the Income and Expenses of both users on a day-to-day basis. This application takes the income of a user and manages its daily expenses so that the user can save money. If you exceed the daily expense allowed amount it will give you a warning, so that you don’t spend much and that specific day. If you spend less money than the daily expense allowed amount, the money left after spending is added into the user's savings. The application generates report of the expenses of each end of the month.

**Year:** 2021

**Technologies:** Java

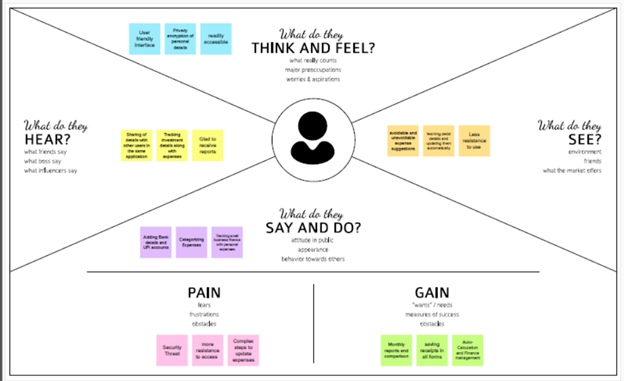
**2.3 PROMBLEM STATEMENT :**

|  |  |
| --- | --- |
| Who doesthe problem affect? | Investors, savers, big spenders, debtors  ,shoppers, budgetconscious consumers. |
| What are the boundaries of the problem? | Expense tracker for working individuals,  students, commonpeople. |
| What is the issue? | To be vigilant aboutthe expense spent,increases financial stress.  Being indecisive about the finances may result in less financial security and exceedthe  budget. |
| When does this issueoccur? | When usingwrong budgeting techniques.  When not tracking the expenses doesn’t help you to know the amount that is actually spent. |
| Where is the issueoccurring? | Working individuals who find it difficult to  track theirexpenses |
| Why is it important thatwe fix theproblem? | Fixing this issue, brings accountability and helps to be intentional with the income by assign it to spending, saving and giving. This  leads to financial stability. |

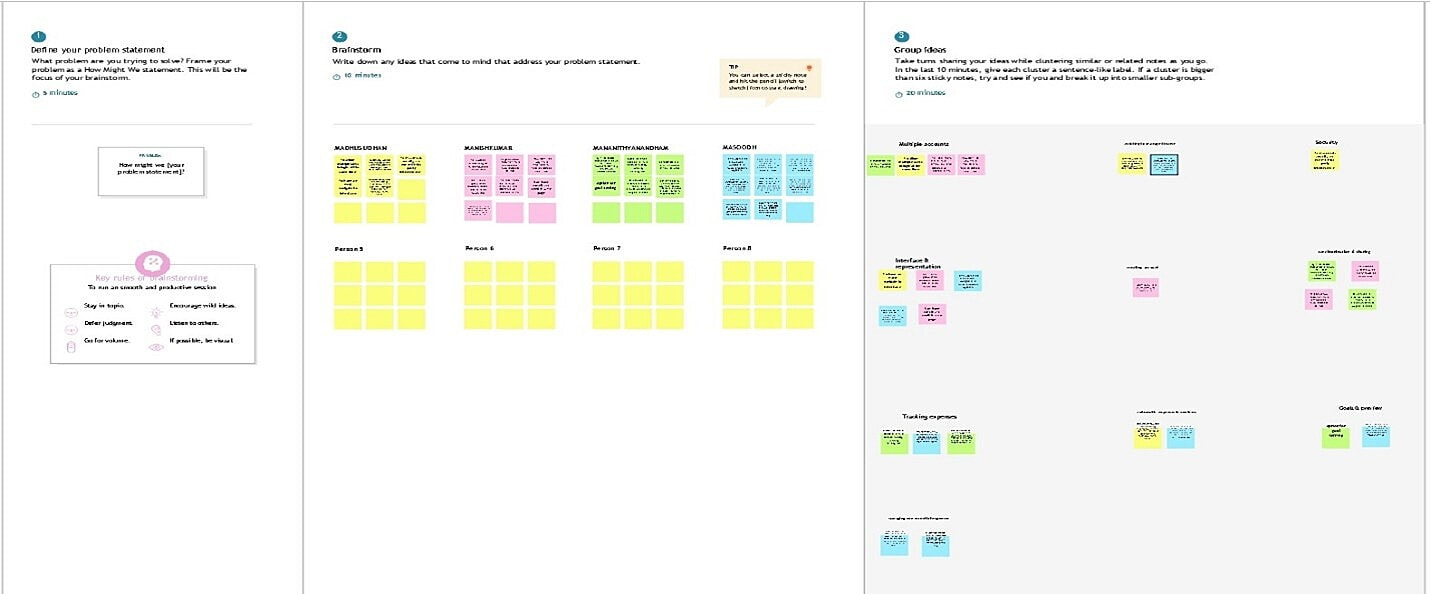
* Abella, who is a shopholic, finds it hard to control her desire to shop. To stop her from overindulging in impulsive purchases, she needs to track her expenses and hold herselfaccountable.
* John, who is interested to invest in stocks, finds itdifficult to figureout the expensethat he can spend on investing stocks. With the help of expense tracking, he can easily plan out the expensesfor investing in an efficient way.
* Akshay, is a high schoolstudent, who usuallygets a limited allowance from his parents. So tracking his expenses and good budgeting technique allows him to spend on his regularexpenses as well as on himself.
* Udhay, who is a novice budgeter, finds it tedious to track and manage the expenses amongst his busy schedule. Prioritizing his expenses will help him to curtail his unnecessary expenditures.

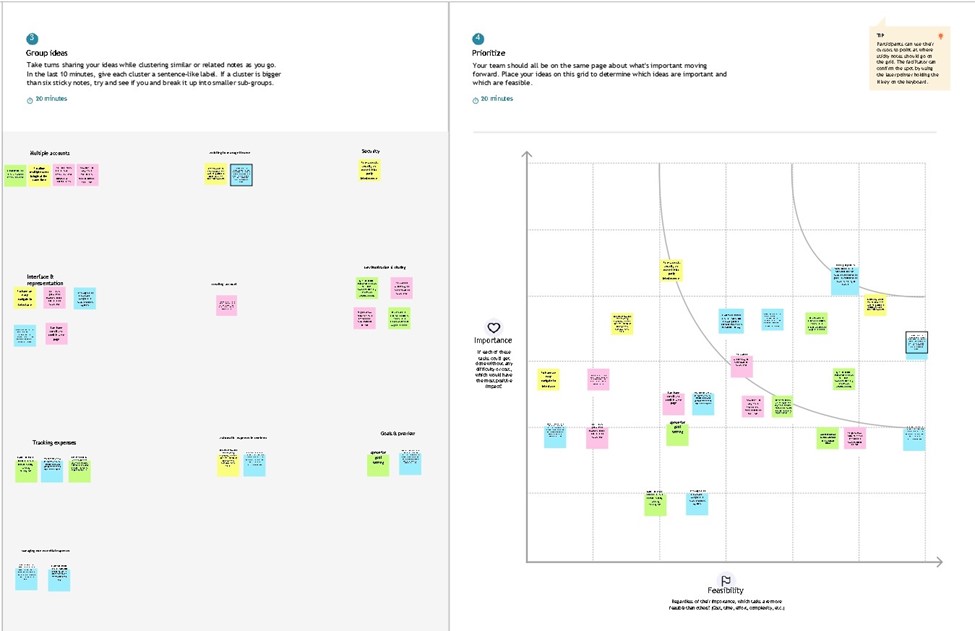
1. **IDEATION & PROPOSED SOLUTION :**

**3.1 EMPATHY MAP CANVAS :**

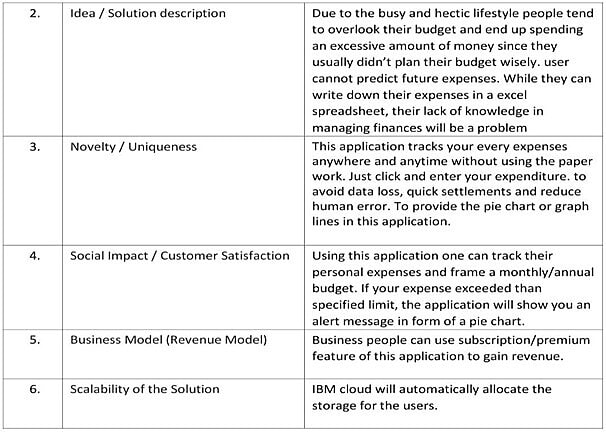


**3.2 IDEATION & BRAINSTORMING:**





**3.3 PROPOSED SOLUTION:**



**3.4 PROBLEM SOLUTION FIT:**



|  |  |
| --- | --- |
|  |  |
|  |  |

**3.3 REQUIREMENT ANALYSIS**

1. **REQUIREMENT ANALYSIS :**
   1. **FUNCTIONAL REQUIREMENT :**

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story/ Sub-Task)** |
| FR-1 | User Registration | Registration through Form Registration throughGmail  Registration through LinkedIN |
| FR-2 | User Confirmation | Confirmation via Email Confirmation via OTP |
| FR-3 | Multiple login | Many userscan log in by usingseparate mail identities.  Also,using the mail identity ,theuser can log into anydevice. |
| FR-4 | Alerting the user | A limitmust be set on the amount of money to be  spent.Whenever the user exceeds thel imit, he will benotified through mailor tex message. |
| FR-5 | Reporting | An analysis on the expenses should be done.Based on theanalysis ,a detailed report (in any graphical form)  must be generated to help the user in accounting and budgeting. |

* 1. **NON FUNCTIONAL REQUIREMENT :**

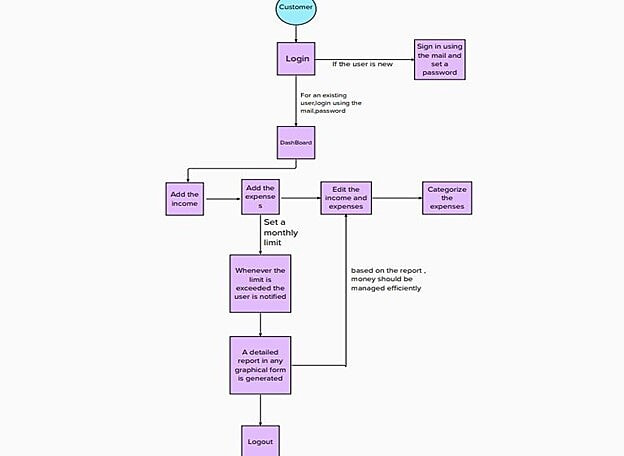


|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | The interface must be user friendly that makes it easyto use for all types of users.The basicfeatures  must be available free of cost to users. |
| NFR-2 | **Security** | The application should have multi-factor  authentication when logging in.Also, banking datamust be secured by some encryption technology. |
| NFR-3 | **Reliability** | The transaction must rollback if there is any technical or network issue .The data must be savedwhen updation of data failsin between the process. Even if there is a failure,it should be restored within  a few minutes. |
| NFR-4 | **Performance** | The application must not take more than 30 seconds  to load. The response time should be quick even when there is heavy traffic. |
| NFR-5 | **Availability** | When the app is being updated,except for the module that is being updated,the rest can be used. |
| NFR-6 | **Scalability** | The app must be designed to work efficiently even when there is heavy traffic. |

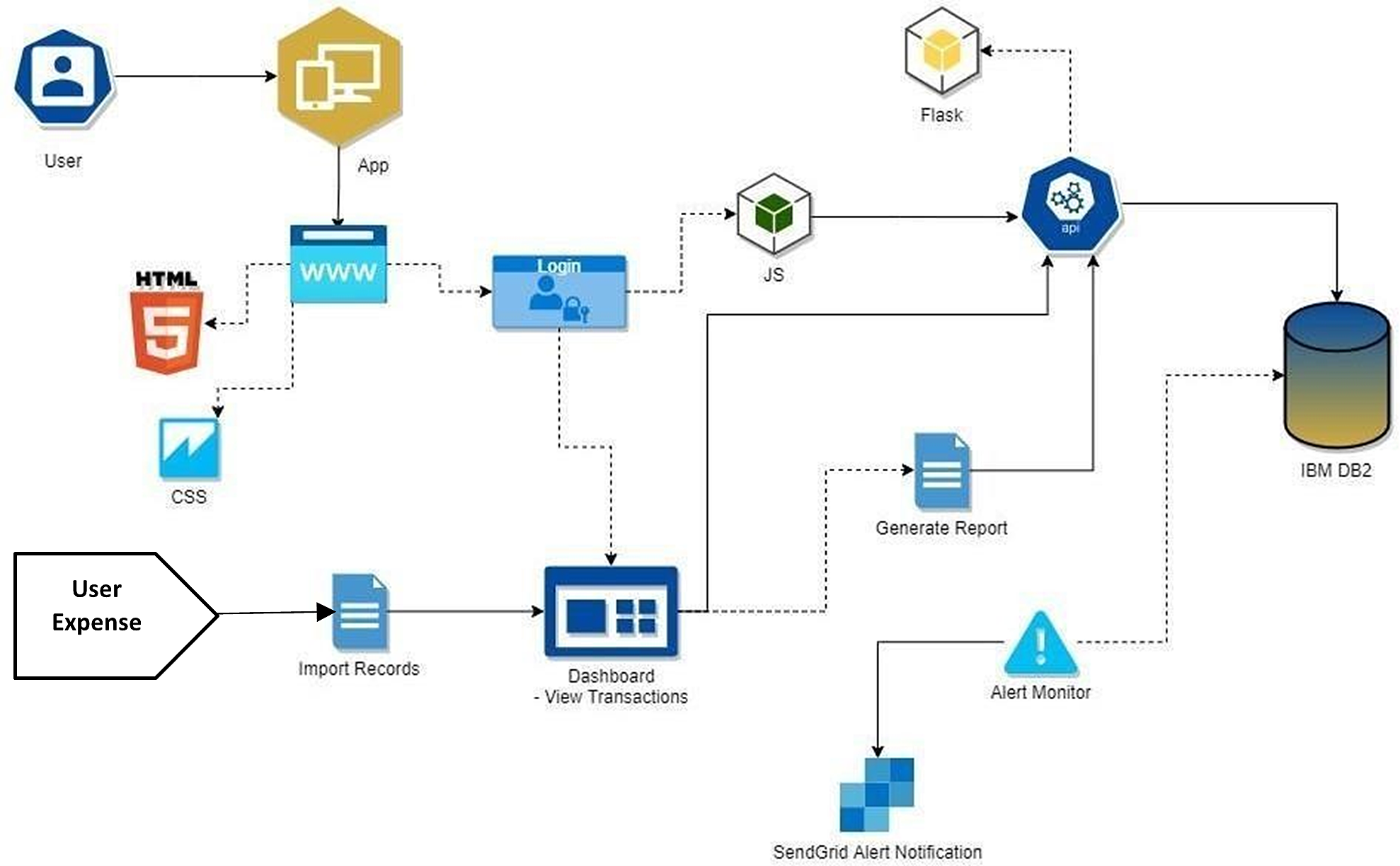
**5.PROJECT DESIGN :**

**5.1 DATAFLOW DIAGRAM :**

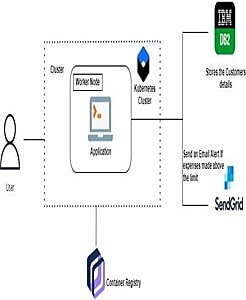
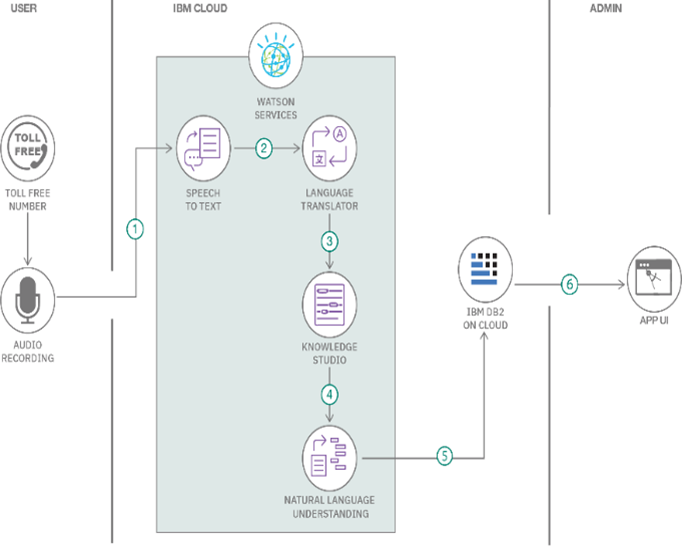
A Data Flow Diagram(DFD) is a traditional visualrepresentation of the information flows within a system. A neat and clear DFD can depictthe right amount of the system requirement graphically. It shows how data enter and leaves the system, what changes the information, and where data is stored.



**5.2 SOLUTION AND TECHNICAL ARCHITECTURE :**  
 **SOLUTION ARCHITECTURE:**



**TECHNICAL ARCHITECTURE :**



**Table-1 : Components & Technologies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.**  **No** | **Component** | **Description** | **Technology** |
| 1. | User Interface | The usercan Interact withthe application with use of IBM WatsonChatbot. | HTML, CSS, JavaScript  / Angular-js / React-js etc. |
| 2. | Application Logic-1 | The application contains the sign in/sign up where the userwill login intothe main dashboard. | Java / Python |
| 3. | Application Logic-2 | Dashboard contains the fields like Add income, AddExpenses, Save Money, Add budget, Profileetc… | IBM Watson STT service |
| 4. | Application Logic-3 | The user will get the expense report in the Statistics formand get alertsif the expense limit exceeds. | IBM WatsonAssistant |
| 5. | Database | TheIncome and Expensedata are stored in the IBM Cloud database. | MySQL, NoSQL,etc. |
| 6. | Cloud Database | Database Service on Cloud | IBM DB2, IBM-Cloudant etc. |
| 7. | File Storage | IBM CloudStorage used to store the financial dataof the user | IBM Block Storage or Other Storage  Service  or Local Filesystem |
| 8. | External API-1 | Purpose of External APIused in the application | IBM Weather API, etc. |
| 9. | External API-2 | Purpose of External API usedin the application | Aadhar API, etc. |
| 10. | Machine Learning Model | Purpose of MachineLearning Model | Object Recognition Model, etc. |
| 11. | Infrastructure (Server / Cloud) | Application Deployment on Local System/ Cloud Local Server Configuration:  Cloud Server Configuration : | Local, CloudFoundry, Kubernetes, etc. |

**Table-2: Application Characteristics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.**  **No** | **Characteristics** | **Description** | **Technology** |
| 1. | Open-Source Frameworks | Flask Framework in Python is used to implement thisapplication to connect the UI and the Backend. | Flask |
| 2. | Security Implementations | This Application Provides high security to the user financial data.It can be done by using the Container Registry in IBM cloud | SHA-256, Encryptions, IAM Controls,  OWASP etc. |
| 3. | Scalable Architecture | Expense Tracker is a lifetime access webblication. Its demand willincrease when the user’s increases. | Container Registry,  Kubernetes Cluster. |
| 4. | Availability | This application will be available to the user at any part of timeusing the Internet. | Container Registry,  KubernetesCluster |
| 5. | Performance | The performance willbe high because there will be no network traffics in the application. | Container Registry,  KubernetesCluster. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional Requirement**  **(Epic)** | **User Story Number** | **UserStory / Task** | **Acceptance criteria** | **Priority** | **Release** |
| Customer (Webuser) | 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 Gmail | I can accessmy account | Medium | Sprint-1 |
|  | Login | USN-4 | As a user, I can log into the application by  entering email& password | I can access the  dashboard | High | Sprint-1 |
|  | Dashboard | USN-5 | As a user, I can add incomeand expenses in theapplication | I can keep track of the expenses | high | Sprint-1 |
|  |  | USN-6 | As a user, I can changethe expenses as I spendand can evencategorise them | I can keep track, account  and budget for the expenses | High | Sprint- |
|  | Alerting | USN-7 | As a user, I canset a limit on the amountof money thatcan be spent. | Whenever the limit is exceeded,  the user gets notified through mail or  text messages. | High | Sprint-1 |
|  | Reporting | USN-8 | As a user,the expense that is spent can be  categorised and a report (in any graphical form) can be generated. | I can manage money efficiently from the  report | High | Sprint-1 |
| Customer(Mobileuser) | Accounting | USN-9 | As a user, the income and expenses canbe added and categorised | From the report generated, money  management could be done | Medium | Sprint-2 |
| Administrator | Supervising and updating | USN-10 | As an administrator, I supervise and update from the user feedback | Updating the app makes it more user  friendly | Medium | Sprint-1 |

**5.3 USER STORIES :**

Use the belowtemplate to list all the user stories for the product.

1. **PROJECT PLANNING AND SCHEDULING :**

**6.1 SPRINT PLANNING & ESTIMATION**

Use the below template to create productbacklog and sprintschedule

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
| Sprint-1 | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | 2 | High | Madhusudhan NS |
| Sprint-1 |  | USN-2 | As a user,I will receive confirmation email onceI have registered for the application | 1 | High | Manish Kumar |
| Sprint-1 | Login | USN-3 | As a user, I can register for the application through Gmail | 1 | High | Mananithyanantham |
| Sprint-1 | Dashboard | USN-4 | As a user,I can log into the application by entering email& password | 2 | High | Masoodh |
| Sprint-2 | Workspace | USN-1 | Workspace for personal expensetracking | 2 | High | Madhusudhan NS |
| Sprint-2 | Charts | USN-2 | Creating various graphsand statistics of customer’s data | 1 | Medium | Manish Kumar |
| Sprint-2 | Connecting to IBM DB2 | USN-3 | Linking database withdashboard | 2 | High | Mananithyanantham |
| Sprint-2 |  | USN-4 | Making dashboard interactive with JS | 2 | High | Masoodh |
| Sprint-3 |  | USN-1 | Wrapping up the server side works of frontend | 1 | Medium | Madhusudhan NS |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-3 | Watson Assistant | USN-2 | Creating Chatbot for expense tracking and for calrifying user’s query | 1 | Medium | Manish Kumar |
| Sprint-3 | SendGrid | USN-3 | Using SendGrid to send mail to the user about | 1 | Low | Mananithyanantham |
| Sprint-3 |  | USN-4 | Integrating both frontend and backend | 2 | High | Masoodh |
| Sprint-4 | Docker | USN-1 | Creating image of website using docker | 2 | High | Madhusudhan NS |
| Sprint-4 | Cloud Registry | USN-2 | Uploading docker imageto IBM Cloud registry | 2 | High | Manish Kumar |
| Sprint-4 | kubernetes | USN-3 | Create container usingthe docker imageand hosting thesite | 2 | High | Mananithyanantham |
| Sprint-4 | Exposing | USN-4 | Exposing IP/Ports for the site | 2 | High | Masoodh |

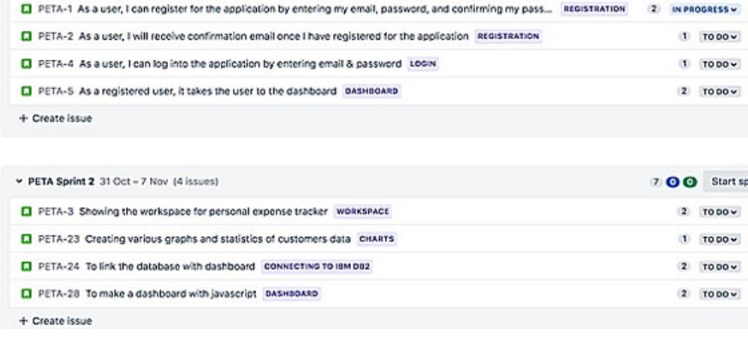
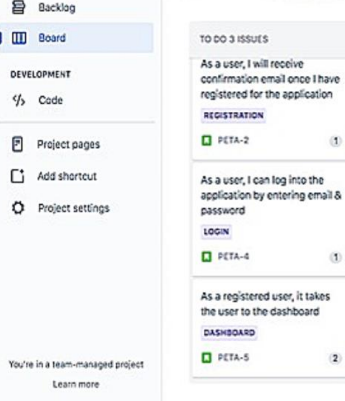
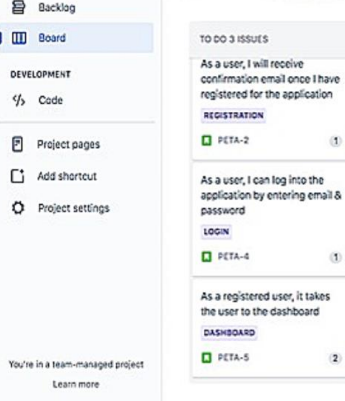
* 1. **SPRINT DELIVERY SCHEDULE :**

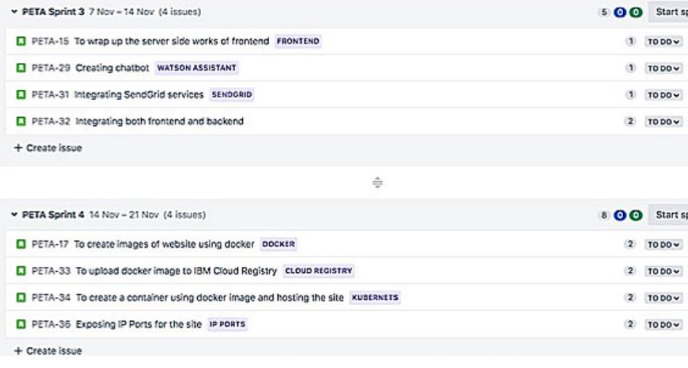
**Project Tracker, Velocity & Burndown Chart:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint | Total StoryPoints | Duration | Sprint StartDate | Sprint End Date (Planned) | Story PointsCompleted (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 | 2 Days | 12 Nov 2022 | 14 Nov 2022 | 20 | 1. Nov 2022 |

**6.3 REPORT FROM JIRA:**

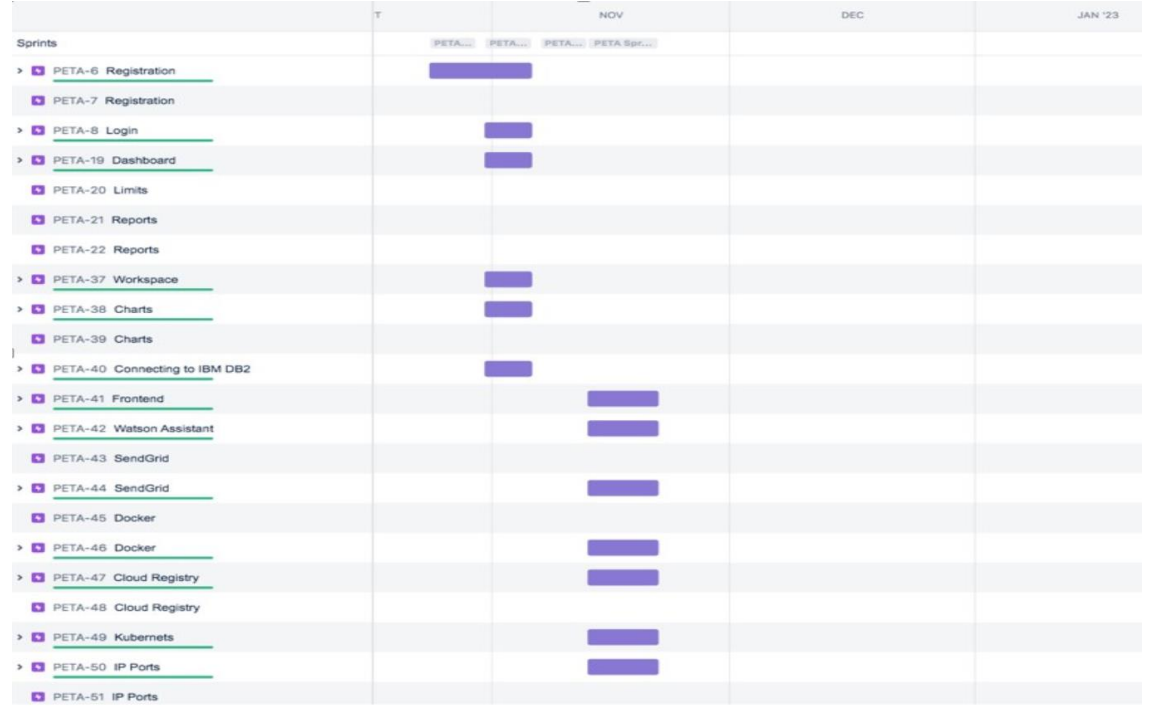
**BACKLOG:**





**BOARD:**

**ROAD MAP:**



### *7. CODING & SOLUTIONING*

Features

Feature 1: Add Expense

Feature 2: Update expense

Feature 3: Delete Expense

Feature 4: Set Limit

Feature 5: Send Alert Emails to users

Other Features: Track your expenses anywhere, anytime. Seamlessly manage your money and budget without any financial paperwork. Just click and submit your invoices and expenditures. Access, submit, and approve invoices irrespective of time and location. Avoid data loss by scanning your tickets and bills and saving in the app. Approval of bills and expenditures in real-time and get notified instantly. Quick settlement of claims and reduced human errors with an automated and streamlined billing process.

**Codes:**

**app.py:**

# -\*- coding: utf-8 -\*-

"""

Spyder Editor

This is a temporary script file.

"""

from flask import Flask, render\_template, request, redirect, session

# from flask\_mysqldb import MySQL # import MySQLdb.cursorsimport re

from flask\_db2 import DB2 import ibm\_dbimport ibm\_db\_dbi from sendemail import sendgridmail,sendmail

# from gevent.pywsgi import WSGIServer import os

app = Flask(\_\_name\_\_)

app.secret\_key = 'a'

# app.config['MYSQL\_HOST'] = 'remotemysql.com'

# app.config['MYSQL\_USER'] = 'D2DxDUPBii'

# app.config['MYSQL\_PASSWORD'] = 'r8XBO4GsMz'

# app.config['MYSQL\_DB'] = 'D2DxDUPBii'

"""

dsn\_hostname = "3883e7e4-18f5-4afe-be8c-

fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"dsn\_uid = "sbb93800" dsn\_pwd = "wobsVLm6ccFxcNLe" dsn\_driver = "{IBM DB2 ODBC DRIVER}"dsn\_database = "bludb"dsn\_port = "31498" dsn\_protocol = "tcpip"

dsn = (

"DRIVER={0};"

"DATABASE={1};"

"HOSTNAME={2};"

"PORT={3};"

"PROTOCOL={4};"

"UID={5};"

"PWD={6};" ).format(dsn\_driver, dsn\_database, dsn\_hostname, dsn\_port, dsn\_protocol, dsn\_uid, dsn\_pwd)

"""

# app.config['DB2\_DRIVER'] = '{IBM DB2 ODBC DRIVER}' app.config['database'] = 'bludb'

app.config['hostname'] = '3883e7e4-18f5-4afe-be8cfa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'app.config['port'] = '31498' app.config['protocol'] = 'tcpip' app.config['uid'] = 'sbb93800' app.config['pwd'] = 'wobsVLm6ccFxcNLe' app.config['security'] = 'SSL' try:

mysql = DB2(app)

conn\_str='database=bludb;hostname=3883e7e4-18f5-4afe-be8c-

fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=31498;protocol=tcpi p;\

uid=sbb93800;pwd=wobsVLm6ccFxcNLe;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())

# app.config['']

# mysql = MySQL(app)

#HOME--PAGE

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

return render\_template("homepage.html")

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

return render\_template("home.html")

#SIGN--UP--OR--REGISTER

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

return render\_template("signup.html")

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

def register():

msg = ''

print("Break point1")

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

print("Break point2" + "name: " + username + "------" + email + "------" + password)

try:

print("Break point3")

connectionID = ibm\_db\_dbi.connect(conn\_str, '', '') cursor = connectionID.cursor()

print("Break point4")

except:

print("No connection Established")

# cursor = mysql.connection.cursor() # with app.app\_context():

# print("Break point3")

# cursor = ibm\_db\_conn.cursor()

# print("Break point4")

print("Break point5")

sql = "SELECT \* FROM register WHERE username = ?" stmt = ibm\_db.prepare(ibm\_db\_conn, sql) ibm\_db.bind\_param(stmt, 1, username) ibm\_db.execute(stmt)

result = ibm\_db.execute(stmt)

print(result)

account = ibm\_db.fetch\_row(stmt) print(account)

param = "SELECT \* FROM register WHERE username = " + "\'" + username + "\'" res = ibm\_db.exec\_immediate(ibm\_db\_conn, param) print("---- ")

dictionary = ibm\_db.fetch\_assoc(res) while dictionary != False:

print("The ID is : ", dictionary["USERNAME"]) dictionary = ibm\_db.fetch\_assoc(res)

# dictionary = ibm\_db.fetch\_assoc(result)

# cursor.execute(stmt)

# account = cursor.fetchone()

# print(account)

# while ibm\_db.fetch\_row(result) != False:

# # account = ibm\_db.result(stmt)

# print(ibm\_db.result(result, "username"))

# print(dictionary["username"]) print("break point 6") if account:

msg = 'Username already exists !'

elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

msg = 'Invalid email address !'

elif not re.match(r'[A-Za-z0-9]+', username):

msg = 'name must contain only characters and numbers !' else:

sql2 = "INSERT INTO register (username, email,password) VALUES (?, ?, ?)" stmt2 = ibm\_db.prepare(ibm\_db\_conn, sql2) ibm\_db.bind\_param(stmt2, 1, username) ibm\_db.bind\_param(stmt2, 2, email) ibm\_db.bind\_param(stmt2, 3, password)

ibm\_db.execute(stmt2)

# cursor.execute('INSERT INTO register VALUES (NULL, % s, % s, % s)', (username, email,password))

# mysql.connection.commit()

msg = 'You have successfully registered !'

return render\_template('signup.html', msg = msg)

#LOGIN--PAGE

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

return render\_template("login.html")

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

if request.method == 'POST' :

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

# cursor = mysql.connection.cursor()

# cursor.execute('SELECT \* FROM register WHERE username = % s AND password = % s', (username, password ),)

# account = cursor.fetchone()

# print (account)

sql = "SELECT \* FROM register WHERE username = ? and password = ?" stmt = ibm\_db.prepare(ibm\_db\_conn, sql)

ibm\_db.bind\_param(stmt, 1, username)

ibm\_db.bind\_param(stmt, 2, password) result = ibm\_db.execute(stmt) print(result)

account = ibm\_db.fetch\_row(stmt) print(account)

param = "SELECT \* FROM register WHERE username = " + "\'" + username + "\'" + " and password = " + "\'" + password + "\'" res = ibm\_db.exec\_immediate(ibm\_db\_conn, param)

dictionary = ibm\_db.fetch\_assoc(res)

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

if account:

session['loggedin'] = True session['id'] = dictionary["ID"] 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)

#ADDING----DATA

@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)

# cursor = mysql.connection.cursor()

# cursor.execute('INSERT INTO expenses VALUES (NULL, % s, % s, % s, % s, % s, % s)', (session['id'] ,date, expensename, amount, paymode, category))

# mysql.connection.commit()

# print(date + " " + expensename + " " + amount + " " + paymode + " " + category)

sql = "INSERT INTO expenses (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 expenses 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 for x in expense: total += x[4] param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER BY id DESC LIMIT 1"

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["LIMITSS"]) row.append(temp) dictionary = ibm\_db.fetch\_assoc(res)

s = temp[0]

if total > int(s):

msg = "Hello " + session['username'] + " , " + "you have crossed the monthly limit of Rs.

" + s + "/- !!!" + "\n" + "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'])

# cursor = mysql.connection.cursor()

# cursor.execute('SELECT \* FROM expenses WHERE userid = % s AND date ORDER

BY `expenses`.`date` DESC',(str(session['id'])))

# expense = cursor.fetchall()

param = "SELECT \* FROM expenses 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):

# cursor = mysql.connection.cursor()

# cursor.execute('DELETE FROM expenses WHERE id = {0}'.format(id))

# mysql.connection.commit()

param = "DELETE FROM expenses WHERE id = " + 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):

# cursor = mysql.connection.cursor()

# cursor.execute('SELECT \* FROM expenses WHERE id = %s', (id,))

# row = cursor.fetchall()

param = "SELECT \* FROM expenses WHERE id = " + 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']

# cursor = mysql.connection.cursor()

# cursor.execute("UPDATE `expenses` SET `date` = % s , `expensename` = % s , `amount` = % s, `paymode` = % s, `category` = % s WHERE `expenses`.`id` = % s ",(date, expensename, amount, str(paymode), str(category),id))

# mysql.connection.commit()

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 id = ?"

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 limits (userid, limitss) 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 id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER BY id DESC LIMIT 1"

res = ibm\_db.exec\_immediate(ibm\_db\_conn, param) dictionary = ibm\_db.fetch\_assoc(res) row = [] s = " /-" while dictionary != False:

temp = [] temp.append(dictionary["LIMITSS"]) row.append(temp) dictionary = ibm\_db.fetch\_assoc(res) s = temp[0]

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 expenses 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 expenses 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 += x[4] if x[6] == "food": t\_food += x[4]

elif x[6] == "entertainment": t\_entertainment += x[4]

elif x[6] == "business": t\_business += x[4] elif x[6] == "rent": t\_rent += x[4]

elif x[6] == "EMI": t\_EMI += x[4]

elif x[6] == "other": t\_other += x[4]

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 expenses 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 expenses 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 += x[4]

if x[6] == "food": t\_food += x[4]

elif x[6] == "entertainment": t\_entertainment += x[4]

elif x[6] == "business":

t\_business += x[4]

elif x[6] == "rent":

t\_rent += x[4]

elif x[6] == "EMI": t\_EMI += x[4]

elif x[6] == "other": t\_other += x[4]

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 expenses 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 expenses 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 += x[4]

if x[6] == "food": t\_food += x[4]

elif x[6] == "entertainment":

t\_entertainment += x[4]

elif x[6] == "business": t\_business += x[4]

elif x[6] == "rent": t\_rent += x[4]

elif x[6] == "EMI": t\_EMI += x[4]

elif x[6] == "other": t\_other += x[4]

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')

port = os.getenv('VCAP\_APP\_PORT', '8080') if \_\_name\_\_ == "\_\_main\_\_":

app.secret\_key = os.urandom(12) app.run(debug=True, host='0.0.0.0', port=port)

**deployment.yaml:**apiVersion: apps/v1 kind: Deployment metadata:

name: sakthi-flask-node-deploymentspec: replicas: 1

selector:

matchLabels:

app: flasknode

template: metadata: labels:

app: flasknode

spec:

containers: - name: flasknode

image: icr.io/sakthi\_expense\_tracker2/flask-template2 imagePullPolicy: Always

ports:

1. containerPort: 5000

**flask-service.yaml:** apiVersion: v1 kind: Servicemetadata:

name: flask-app-servicespec: selector:

app: flask-app ports: - name: http protocol: TCP

port: 80 targetPort: 5000 type: LoadBalancer**manifest.yml:**applications:

1. name: Python Flask App IBCMR 2022-10-19 random-route: true memory: 512M disk\_quota: 1.5G

**sendemail.py:** import smtplib import sendgrid as sgimport os

from sendgrid.helpers.mail

import Mail, Email, To, Content

SUBJECT = "expense tracker" s = smtplib.SMTP('smtp.gmail.com', 587)

def sendmail(TEXT,email):

print("sorry we cant process your candidature")

s = smtplib.SMTP('smtp.gmail.com', 587)

s.starttls()

# s.login("il.tproduct8080@gmail.com", "oms@1Ram")

s.login("tproduct8080@gmail.com", "lxixbmpnexbkiemh") message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT) # s.sendmail("il.tproduct8080@gmail.com", email, message) s.sendmail("il.tproduct8080@gmail.com", email, message)

s.quit()

def sendgridmail(user,TEXT):

# from\_email = Email("abcd@gmail.com") from\_email = Email("tproduct8080@gmail.com") to\_email = To(user)

subject = "Sending with SendGrid is Fun" content = Content("text/plain",TEXT)

mail = Mail(from\_email, to\_email, subject, content)

# Get a JSON-ready representation of the Mail object mail\_json = mail.get()

# Send an HTTP POST request to /mail/send response = sg.client.mail.send.post(request\_body=mail\_json) print(response.status\_code) print(response.headers)

**Database Schema**

Tables :

1.Admin:

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,username VARCHAR(32) NOT NULL, email

VARCHAR(32) NOT NULL,password VARCHAR(32)

NOT NULL

2.Expense:

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,

userid INT NOT NULL, date TIMESTAMP(12) NOT NULL,expensename VARCHAR(32) NOT NULL, amount

VARCHAR(32) NOT NULL, paymode VARCHAR(32) NOT NULL,

category VARCHAR(32) NOT NULL

3.LIMIT

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,userid VARCHAR(32) NOT NULL, limit

VARCHAR(32) NOT NULL

**8. TESTING :**

**8.1 TEST CASES :**

|  |  |  |
| --- | --- | --- |
| S.NO | **Test Cases** | **Result** |
| 1 | Verify the user is able to see the Sign up page when the user clicks the signup button in navigation bar | Positive |
| 2 | Verify the UI elements in the Sign up page | Positive |
| 3 | Verify the user is able to register into the application by providing valid details | Positive |
| 4 | Verify the user is able to see the sign in page when the user clicks the sign in button in navigation bar | Positive |
| 5 | Verify the UI elements in the Sign in page | Positive |
| 6 | Verify the user is able to login into the application by providing valid details | Positive |
| 7 | Verify the user is able to see the add expenses page when the user clicks the add expenses navigation bar | Positive |
| 8 | Verify the UI elements in the add expenses page | Positive |
| 9 | Verify the user is able to add expenses by providing valid details | Positive |
| 10 | Verify the user is able to see the home button in the add expenses page | Positive |
| 11 | Verify whether the expenses added can be deleted from the cloud | Positive |
| 12 | Verify whether the expenses added can be edited | Positive |
| 13 | Verify whether the date, month and year are valid while user entering the expenses | Positive |
| 14 | Verify whether the data types entered by the user in the add expenses page are float or integers | Positive |
| 15 | Verify whether expenses added previously still exist | Positive |
| 16 | Verify whether the expenses amount entered fits between in the range of integers or float datatypes | Positive |

**8.2 USER ACCEPTANCE TESTING :**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | Feature Type | Component | Test Scenario | Steps To Execute | Test Data | Expected Result | Actual Result |
| LoginPage\_TC\_OO1 | Functional | Home Page | Verify user is able to see the Login/Signup popup when user clicked on My account button | 1. Click the my account button.  2. Check whether login/signup pop up appears | - | Login/Signup popup should display | Working as expected |
| LoginPage\_TC\_OO2 | UI | Home Page | Verify the UI elements in Login/Signup popup | 1. Visit the Signup/Login page. 2. Verify the UI elements. | - | Application should show below UI elements:  a.email text box  b.password text box  c.Login button with orange colour  d.New customer? Create account link  e.Last password? Recovery password link | Working as expected |
| LoginPage\_TC\_OO4 | Functional | Login page | Verify user is able to log into application with Valid credentials | 1. Visit the login page. 2. Enter right credientials. | User id: madhusudhans  @gmail.com  Password: Keerthu | Application should not show 'inorrect email or password ' validation message. | Working as expected |
| LoginPage\_TC\_OO5 | Functional | Login page | Verify user is able to log into application with InValid credentials | 1. Visit the login page. 2. Enter wrong credientials | User id: madhusudhans  @gmail.com  Password: madhu | Application should show 'Incorrect email or password ' validation message. | Working as expected |
| Add\_expense\_page\_TC\_001 | Functional | Add expense page | Verify the user could add the expenses & delete them | 1. Visit the login page.   2.Enter right credientials.  3. Enter input data and delete them. | Input data: 25 | Application should store the value on entering and deletes it on clicking delete button. | Working as expected |
| Add\_expense\_page\_TC\_002 | Functional | Add expense page | Verify the user enters only numericals in add expenses input. | 1. Visit the login page.   2.Enter right credientials.  3.Enter numerical values in input  And not strings. | Input data 1: 50  Input data 2: m | Application should store numerical values and not strings. | Working as expected |
| Add\_expense\_page\_TC\_003 | Functional | Add expense page | Verify the date, month and year are valid | 1. Visit the login page.   2.Enter right credientials.  3. Check whether date, month and year are valid. | June 5, 1999 | Application should display only valid date,month and year | Working as expected |

1. **ADVANTAGES & DISADVANTAGES :**

**ADVANTAGES :**

* With proper tracking of your finances, you will not be able to determine unnecessary spending. This spending, if saved, can easily add up to quite a bit.
* In this day and age, when expenses are going through the roof, it has become crucial that you learn to make your money work for you so that you can create a nest egg for the future.
* Has various components of updating and viewing users expenditure
* User can track his expenses by choosing a day and using various filtering Options to study expenses
* Visualization using pie chart with percentage view shows graphical Representation.
* This approach effectively keeps away from the manual figuring for trying
* Not to ascertain the pay and cost each month.
* With a daily expense manager, you will be able to allocate money to different priorities and this will also help you cut down on unnecessary spending. As a result, you will be able to save and be able to keep worry at bay.
* A daily money tracker helps you budget your money so that you use it wisely. If you find that every month your expenses are more than what you earn, it is time to put your house in order and get a money manager app that keeps track of your money without any problem.

**DISADVANTAGES :**

* Lack of visual analytics for expense data
* Lack of support for splitting group expenses
* Suitable for only Personal use.
* Errors are another common problem expense reports drafted with Excel. As is the case with all tasks performed manually and based on paper, it is extremely likely that business travellers who report expenses make involuntary mistakes: numbers entered wrongly, duplicated expenses or expenses relating to a previous settlement period, misapplication of the expense policy, etc.
* Frequent tracking of cash spending can allow one to catch and correct errors so that the budget plan is still able to be adhered to despite the mistake.

**APPENDIX**

**SOURCE CODE:**

### *7. CODING & SOLUTIONING*

**app.py:**

# -\*- coding: utf-8 -\*-

"""

Spyder Editor

This is a temporary script file.

"""

from flask import Flask, render\_template, request, redirect, session

# from flask\_mysqldb import MySQL # import MySQLdb.cursorsimport re

from flask\_db2 import DB2 import ibm\_dbimport ibm\_db\_dbi from sendemail import sendgridmail,sendmail

# from gevent.pywsgi import WSGIServer import os

app = Flask(\_\_name\_\_)

app.secret\_key = 'a'

# app.config['MYSQL\_HOST'] = 'remotemysql.com'

# app.config['MYSQL\_USER'] = 'D2DxDUPBii'

# app.config['MYSQL\_PASSWORD'] = 'r8XBO4GsMz'

# app.config['MYSQL\_DB'] = 'D2DxDUPBii'

"""

dsn\_hostname = "3883e7e4-18f5-4afe-be8c-

fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"dsn\_uid = "sbb93800" dsn\_pwd = "wobsVLm6ccFxcNLe" dsn\_driver = "{IBM DB2 ODBC DRIVER}"dsn\_database = "bludb"dsn\_port = "31498" dsn\_protocol = "tcpip"

dsn = (

"DRIVER={0};"

"DATABASE={1};"

"HOSTNAME={2};"

"PORT={3};"

"PROTOCOL={4};"

"UID={5};"

"PWD={6};"

).format(dsn\_driver, dsn\_database, dsn\_hostname, dsn\_port, dsn\_protocol, dsn\_uid, dsn\_pwd)

"""

# app.config['DB2\_DRIVER'] = '{IBM DB2 ODBC DRIVER}' app.config['database'] = 'bludb'

app.config['hostname'] = '3883e7e4-18f5-4afe-be8cfa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'app.config['port'] = '31498' app.config['protocol'] = 'tcpip' app.config['uid'] = 'sbb93800' app.config['pwd'] = 'wobsVLm6ccFxcNLe' app.config['security'] = 'SSL' try:

mysql = DB2(app)

conn\_str='database=bludb;hostname=3883e7e4-18f5-4afe-be8c-

fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=31498;protocol=tcpi p;\

uid=sbb93800;pwd=wobsVLm6ccFxcNLe;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())

# app.config['']

# mysql = MySQL(app)

#HOME--PAGE

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

return render\_template("homepage.html")

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

return render\_template("home.html")

#SIGN--UP--OR--REGISTER

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

return render\_template("signup.html")

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

def register():

msg = ''

print("Break point1")

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

print("Break point2" + "name: " + username + "------" + email + "------" + password)

try:

print("Break point3")

connectionID = ibm\_db\_dbi.connect(conn\_str, '', '') cursor = connectionID.cursor()

print("Break point4")

except:

print("No connection Established")

# cursor = mysql.connection.cursor() # with app.app\_context():

# print("Break point3")

# cursor = ibm\_db\_conn.cursor()

# print("Break point4")

print("Break point5")

sql = "SELECT \* FROM register WHERE username = ?" stmt = ibm\_db.prepare(ibm\_db\_conn, sql) ibm\_db.bind\_param(stmt, 1, username) ibm\_db.execute(stmt)

result = ibm\_db.execute(stmt)

print(result)

account = ibm\_db.fetch\_row(stmt) print(account)

param = "SELECT \* FROM register WHERE username = " + "\'" + username + "\'" res = ibm\_db.exec\_immediate(ibm\_db\_conn, param) print("---- ")

dictionary = ibm\_db.fetch\_assoc(res) while dictionary != False:

print("The ID is : ", dictionary["USERNAME"]) dictionary = ibm\_db.fetch\_assoc(res)

# dictionary = ibm\_db.fetch\_assoc(result)

# cursor.execute(stmt)

# account = cursor.fetchone()

# print(account)

# while ibm\_db.fetch\_row(result) != False:

# # account = ibm\_db.result(stmt)

# print(ibm\_db.result(result, "username"))

# print(dictionary["username"]) print("break point 6") if account:

msg = 'Username already exists !'

elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

msg = 'Invalid email address !'

elif not re.match(r'[A-Za-z0-9]+', username):

msg = 'name must contain only characters and numbers !' else:

sql2 = "INSERT INTO register (username, email,password) VALUES (?, ?, ?)" stmt2 = ibm\_db.prepare(ibm\_db\_conn, sql2) ibm\_db.bind\_param(stmt2, 1, username) ibm\_db.bind\_param(stmt2, 2, email) ibm\_db.bind\_param(stmt2, 3, password)

ibm\_db.execute(stmt2)

# cursor.execute('INSERT INTO register VALUES (NULL, % s, % s, % s)', (username, email,password))

# mysql.connection.commit()

msg = 'You have successfully registered !'

return render\_template('signup.html', msg = msg)

#LOGIN--PAGE

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

return render\_template("login.html")

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

if request.method == 'POST' :

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

# cursor = mysql.connection.cursor()

# cursor.execute('SELECT \* FROM register WHERE username = % s AND password = % s', (username, password ),)

# account = cursor.fetchone()

# print (account)

sql = "SELECT \* FROM register WHERE username = ? and password = ?" stmt = ibm\_db.prepare(ibm\_db\_conn, sql)

ibm\_db.bind\_param(stmt, 1, username)

ibm\_db.bind\_param(stmt, 2, password) result = ibm\_db.execute(stmt) print(result)

account = ibm\_db.fetch\_row(stmt) print(account)

param = "SELECT \* FROM register WHERE username = " + "\'" + username + "\'" + " and password = " + "\'" + password + "\'" res = ibm\_db.exec\_immediate(ibm\_db\_conn, param)

dictionary = ibm\_db.fetch\_assoc(res)

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

if account:

session['loggedin'] = True session['id'] = dictionary["ID"] 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)

#ADDING----DATA

@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)

# cursor = mysql.connection.cursor()

# cursor.execute('INSERT INTO expenses VALUES (NULL, % s, % s, % s, % s, % s, % s)', (session['id'] ,date, expensename, amount, paymode, category))

# mysql.connection.commit()

# print(date + " " + expensename + " " + amount + " " + paymode + " " + category)

sql = "INSERT INTO expenses (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 expenses 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 for x in expense: total += x[4] param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER BY id DESC LIMIT 1"

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["LIMITSS"]) row.append(temp) dictionary = ibm\_db.fetch\_assoc(res)

s = temp[0]

if total > int(s):

msg = "Hello " + session['username'] + " , " + "you have crossed the monthly limit of Rs.

" + s + "/- !!!" + "\n" + "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'])

# cursor = mysql.connection.cursor()

# cursor.execute('SELECT \* FROM expenses WHERE userid = % s AND date ORDER

BY `expenses`.`date` DESC',(str(session['id'])))

# expense = cursor.fetchall()

param = "SELECT \* FROM expenses 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):

# cursor = mysql.connection.cursor()

# cursor.execute('DELETE FROM expenses WHERE id = {0}'.format(id))

# mysql.connection.commit()

param = "DELETE FROM expenses WHERE id = " + 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):

# cursor = mysql.connection.cursor()

# cursor.execute('SELECT \* FROM expenses WHERE id = %s', (id,))

# row = cursor.fetchall()

param = "SELECT \* FROM expenses WHERE id = " + 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']

# cursor = mysql.connection.cursor()

# cursor.execute("UPDATE `expenses` SET `date` = % s , `expensename` = % s , `amount` = % s, `paymode` = % s, `category` = % s WHERE `expenses`.`id` = % s ",(date, expensename, amount, str(paymode), str(category),id))

# mysql.connection.commit()

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 id = ?"

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 limits (userid, limitss) 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 id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER BY id DESC LIMIT 1"

res = ibm\_db.exec\_immediate(ibm\_db\_conn, param) dictionary = ibm\_db.fetch\_assoc(res) row = [] s = " /-" while dictionary != False:

temp = [] temp.append(dictionary["LIMITSS"]) row.append(temp) dictionary = ibm\_db.fetch\_assoc(res) s = temp[0]

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 expenses 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 expenses 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 += x[4] if x[6] == "food": t\_food += x[4]

elif x[6] == "entertainment": t\_entertainment += x[4]

elif x[6] == "business": t\_business += x[4] elif x[6] == "rent": t\_rent += x[4]

elif x[6] == "EMI": t\_EMI += x[4]

elif x[6] == "other": t\_other += x[4]

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 expenses 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 expenses 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 += x[4]

if x[6] == "food": t\_food += x[4]

elif x[6] == "entertainment": t\_entertainment += x[4]

elif x[6] == "business":

t\_business += x[4]

elif x[6] == "rent":

t\_rent += x[4]

elif x[6] == "EMI": t\_EMI += x[4]

elif x[6] == "other": t\_other += x[4]

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 expenses 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 expenses 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 += x[4]

if x[6] == "food": t\_food += x[4]

elif x[6] == "entertainment":

t\_entertainment += x[4]

elif x[6] == "business": t\_business += x[4]

elif x[6] == "rent": t\_rent += x[4]

elif x[6] == "EMI": t\_EMI += x[4]

elif x[6] == "other": t\_other += x[4]

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')

port = os.getenv('VCAP\_APP\_PORT', '8080') if \_\_name\_\_ == "\_\_main\_\_":

app.secret\_key = os.urandom(12) app.run(debug=True, host='0.0.0.0', port=port)

**deployment.yaml:**apiVersion: apps/v1 kind: Deployment metadata:

name: sakthi-flask-node-deploymentspec: replicas: 1

selector:

matchLabels:

app: flasknode

template: metadata: labels:

app: flasknode

spec:

containers: - name: flasknode

image: icr.io/sakthi\_expense\_tracker2/flask-template2 imagePullPolicy: Always

ports:

1. containerPort: 5000

**flask-service.yaml:** apiVersion: v1 kind: Servicemetadata:

name: flask-app-servicespec: selector:

app: flask-app ports: - name: http protocol: TCP

port: 80 targetPort: 5000 type: LoadBalancer**manifest.yml:**applications:

1. name: Python Flask App IBCMR 2022-10-19 random-route: true memory: 512M disk\_quota: 1.5G

**sendemail.py:** import smtplib import sendgrid as sgimport os

from sendgrid.helpers.mail

import Mail, Email, To, Content

SUBJECT = "expense tracker" s = smtplib.SMTP('smtp.gmail.com', 587)

def sendmail(TEXT,email):

print("sorry we cant process your candidature")

s = smtplib.SMTP('smtp.gmail.com', 587)

s.starttls()

# s.login("il.tproduct8080@gmail.com", "oms@1Ram")

s.login("tproduct8080@gmail.com", "lxixbmpnexbkiemh") message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT) # s.sendmail("il.tproduct8080@gmail.com", email, message) s.sendmail("il.tproduct8080@gmail.com", email, message)

s.quit()

def sendgridmail(user,TEXT):

# from\_email = Email("abcd@gmail.com") from\_email = Email("tproduct8080@gmail.com") to\_email = To(user)

subject = "Sending with SendGrid is Fun" content = Content("text/plain",TEXT)

mail = Mail(from\_email, to\_email, subject, content)

# Get a JSON-ready representation of the Mail object mail\_json = mail.get()

# Send an HTTP POST request to /mail/send response = sg.client.mail.send.post(request\_body=mail\_json) print(response.status\_code) print(response.headers)

**Database Schema**

Tables :

1.Admin:

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,username VARCHAR(32) NOT NULL, email

VARCHAR(32) NOT NULL,password VARCHAR(32)

NOT NULL

2.Expense:

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,

userid INT NOT NULL, date TIMESTAMP(12) NOT NULL,expensename VARCHAR(32) NOT NULL, amount

VARCHAR(32) NOT NULL, paymode VARCHAR(32) NOT NULL,

category VARCHAR(32) NOT NULL

3.LIMIT

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,userid VARCHAR(32) NOT NULL, limit

VARCHAR(32) NOT NULL

**FUTURE SCOPE:**

1. **Mobility**

Businesses are becoming increasingly global and employees are more mobile than ever. According to the Certify Expense Management Trends Report 2018, 47 percent of organizations felt that mobile applications and accessibility were a critical capability of expense reporting software.

Expense management software will begin to respond to this change in the manner that people work by facilitating mobility. Employees will be able to submit reports and managers can approve the claims from a smartphone. Also, mobile applications will become more intuitive and responsive, encouraging greater adoption.

**2. Travel booking**

According to a 2018 study by Tripactions, 90 percent of respondents believe that travel is important for growth. However, 50 percent of employees don’t use the corporate travel solution offered by their organization owing to difficulties and the time taken to book a trip. They prefer using consumer channels.

Using multiple external vendors and disjointed channels to book travel can become tedious and lead to a lot of last-minute chaos. Also, the expenses have to be recorded accurately in the expense management system. It’s imperative to integrate the same into expense management software.

**3. Integrated system**

In 2019, expense management software will begin to consolidate currently disparate actions into a seamless system with credit cards, bank accounts, accounting, payroll, CRM, and more in one place. All these systems share a lot of common information, and it makes perfect sense to connect all these to ensure data integrity.

When integrated with travel applications and travel management company (TMC) solutions, transactions can be entered directly into the system without the need for manual input. Corporate cards can also be linked to the software, which makes it easier to reconcile credit card statements with expense reports.

**4. Optical character recognition**

According to 66 percent of respondents from the Certify Expense Management Trends Report 2018, ease of use was the top-most feature needed. Traveling employees spend a lot of time filling out expense reports. By using expense management software, businesses brought down the time and cost of expense reporting. This will further go down with advancements in OCR technology.

Optical character recognition (OCR) will be one of the significant ways in which the ease-of-use of expense management tool will be enhanced. It eliminates the need to manually input data from receipts into forms. It will also facilitate mobility.

Employees could just snap a picture of a receipt and have the software extract all the necessary information from it to fill a report. OCR technology is accurate up to 95 percent and doesn’t need too much human intervention to verify the information that’s been uploaded.

**5. Artificial intelligence**

As machine learning and artificial intelligence (AI) evolve, they will add to the sophistication of expense management software. There will be improved ability to assign expense general ledger codes based on submissions and smarter categorization of expense types. Rudimentary versions of this exist in a few tools that use rule-based categorization.

AI-powered automation is also expected to minimize fraudulent expense reports. It can monitor, track, and approve expenses, and compliance-related issues will immediately be flagged and eliminated.

**GITHUB LINK:**

**<https://github.com/IBM-EPBL/IBM-Project-18040-1659678602>**