**PROJECT REPORT - PERSONAL EXPENSE TRACKER**

1. **INTRODUCTION:**
   1. **Project Overview:**

The main objective of this project is to design an application for tracking personal expenses and provide an analysis which can be understood easily by the user.

The project has been idealized and completed through eventual progress and brainstorming sessions of the team and its members:

Madhumitha V

M. L. Divija

S. Maheshwar

M. Aakaash Kumar

**1.2 Purpose:**

* To manage finances efficiently.
* To help with budgeting and accounting.
* To obtain insights about financial management.
* To reduce unnecessary expenses.
* To provide alerts when limits are exceeded.

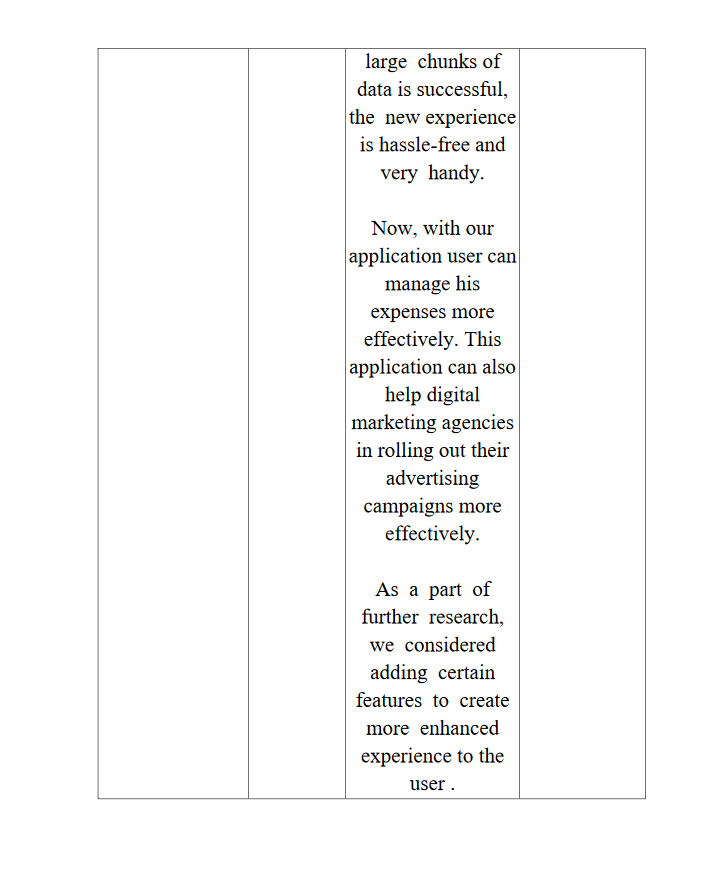
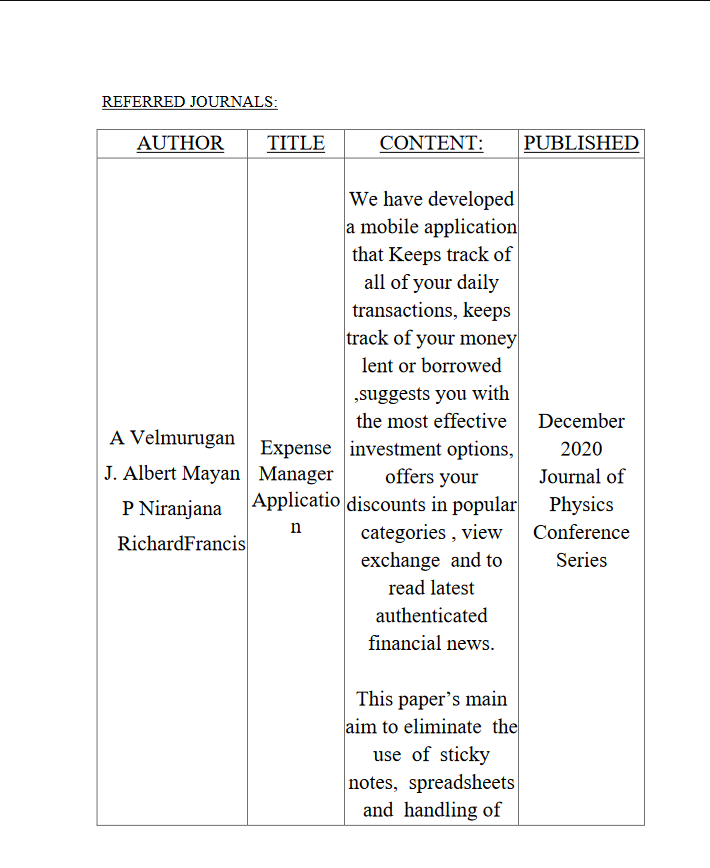
1. **LITERATURE SURVEY:**

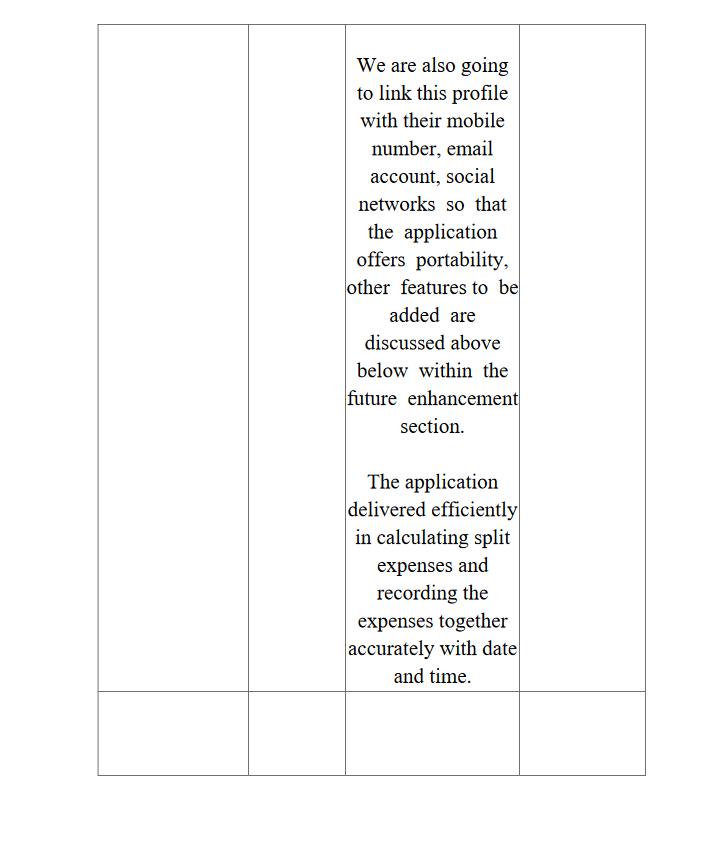
Personal Expense Tracker Application

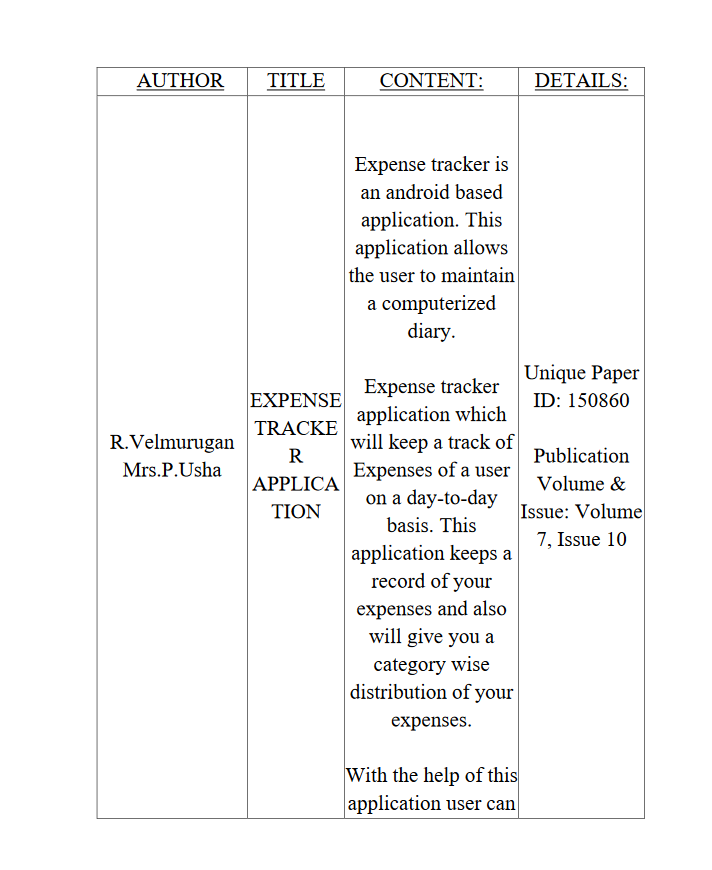
PROBLEM STATEMENT:

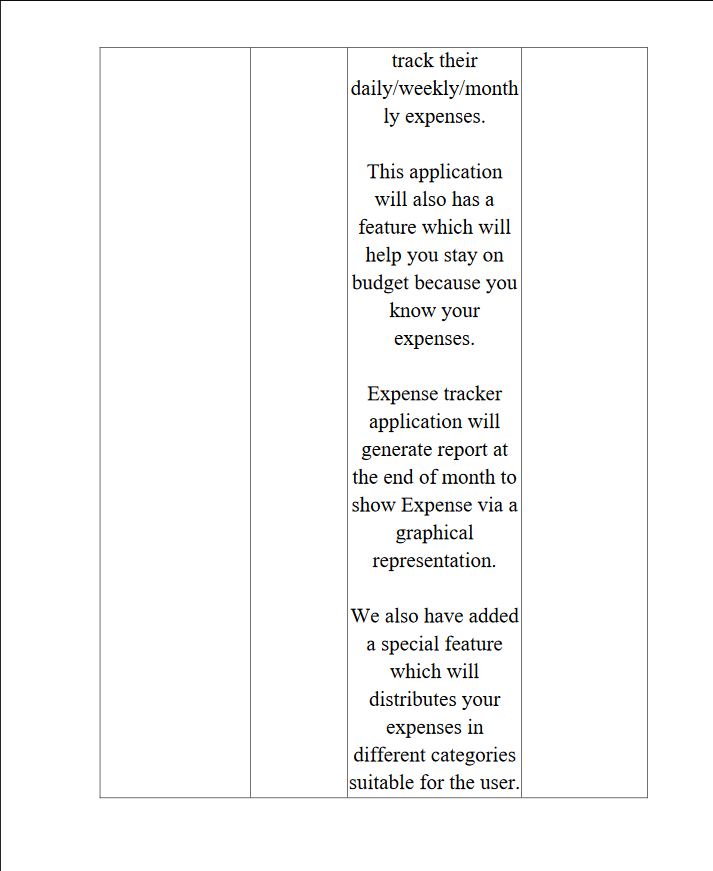
In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money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

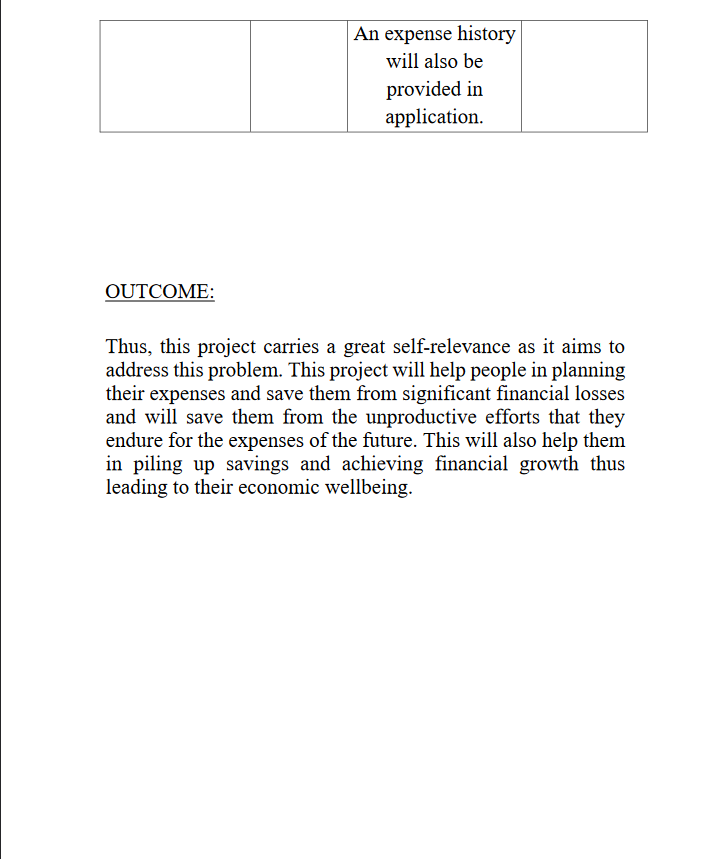
Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that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

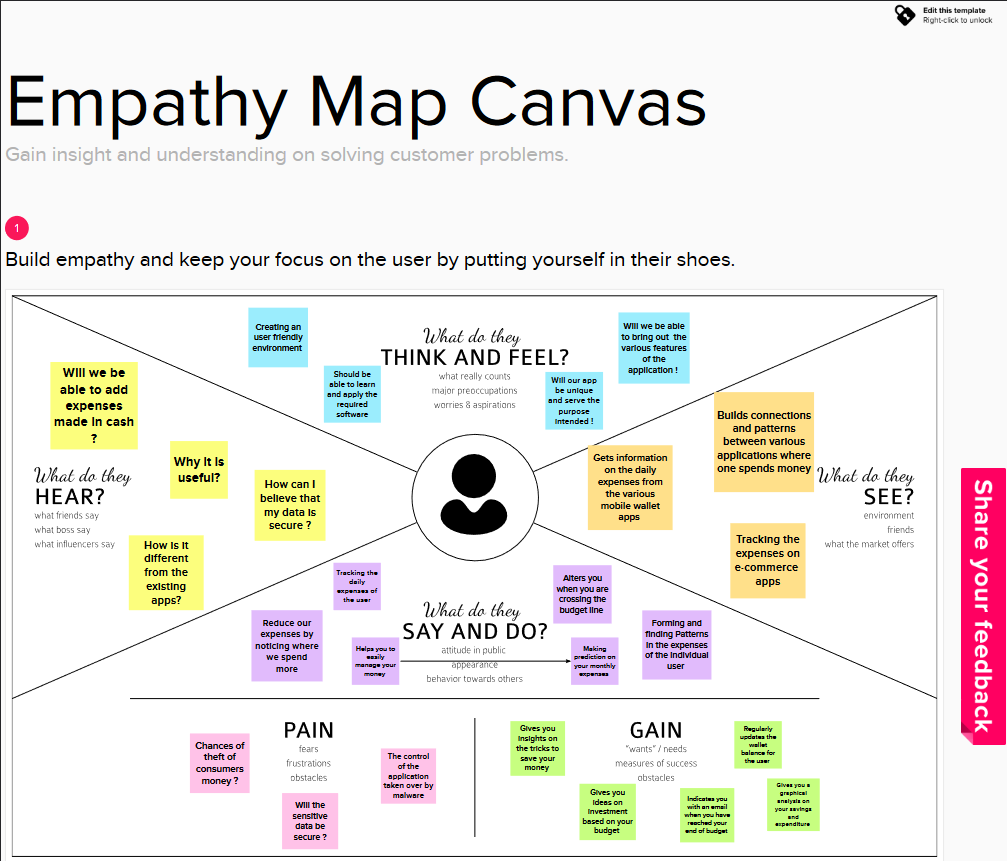


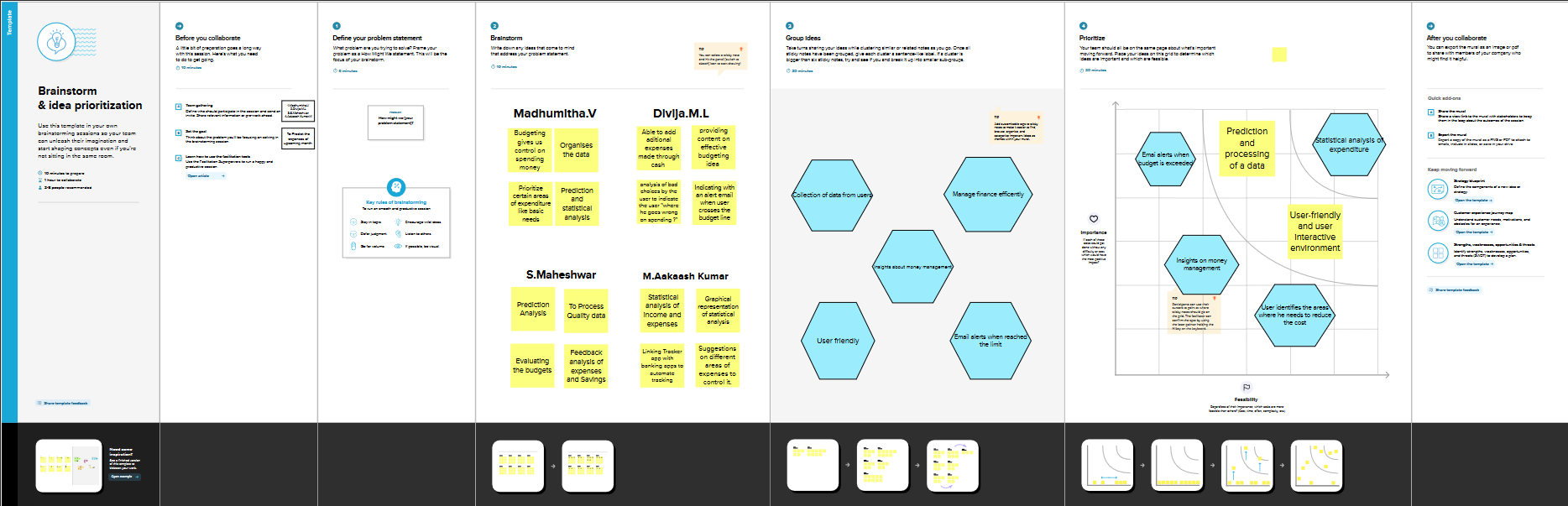


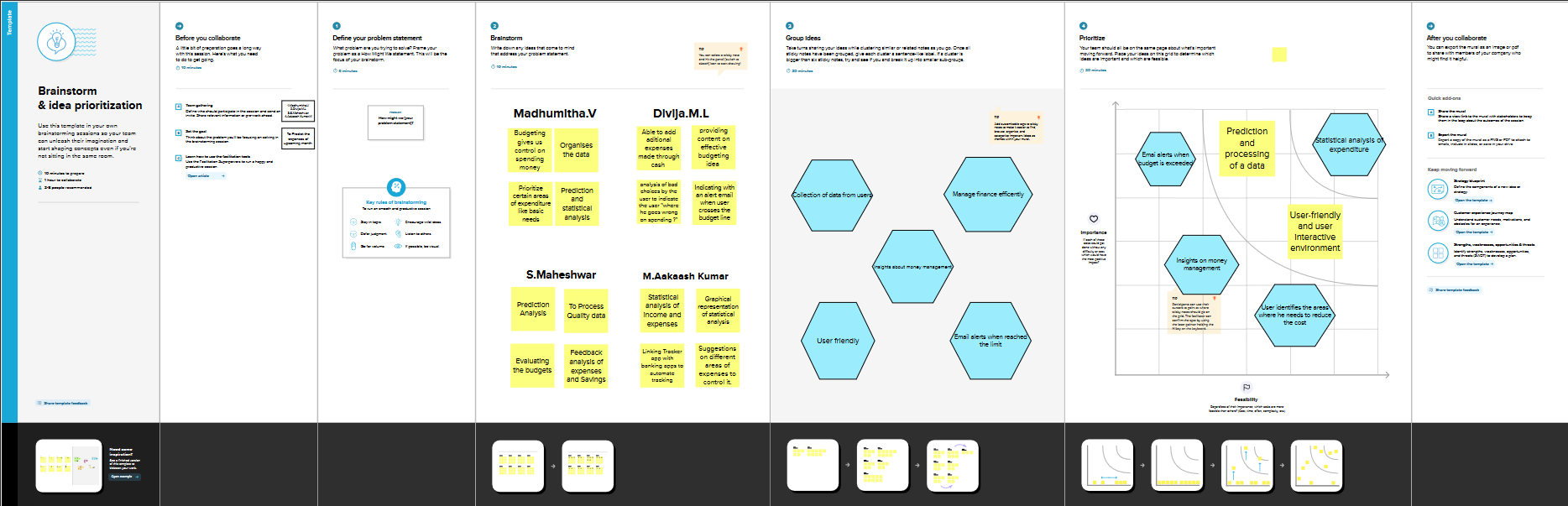






1. **IDEATION AND PROPOSED SOLUTION:\**
   1. **Empathy Map Canvas**
   2. **Ideation & Brainstorming**

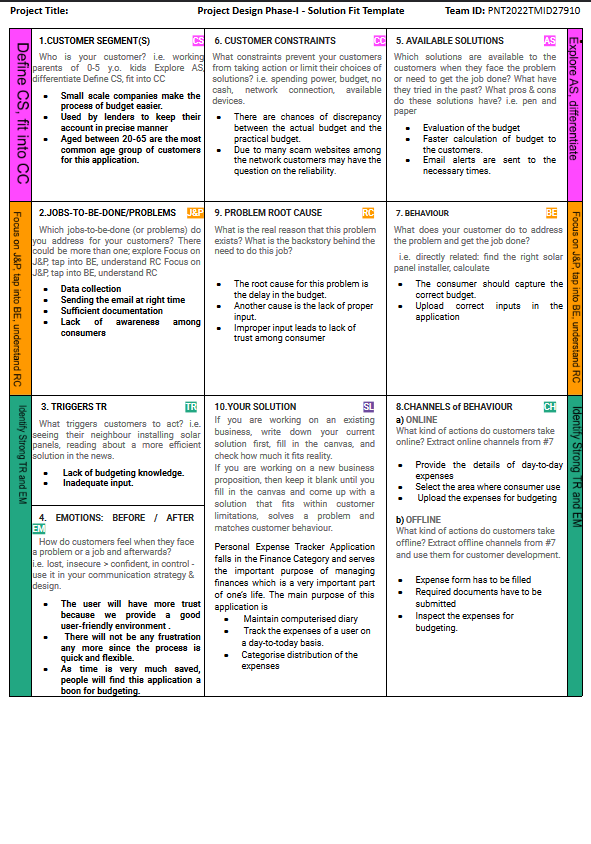
****

****

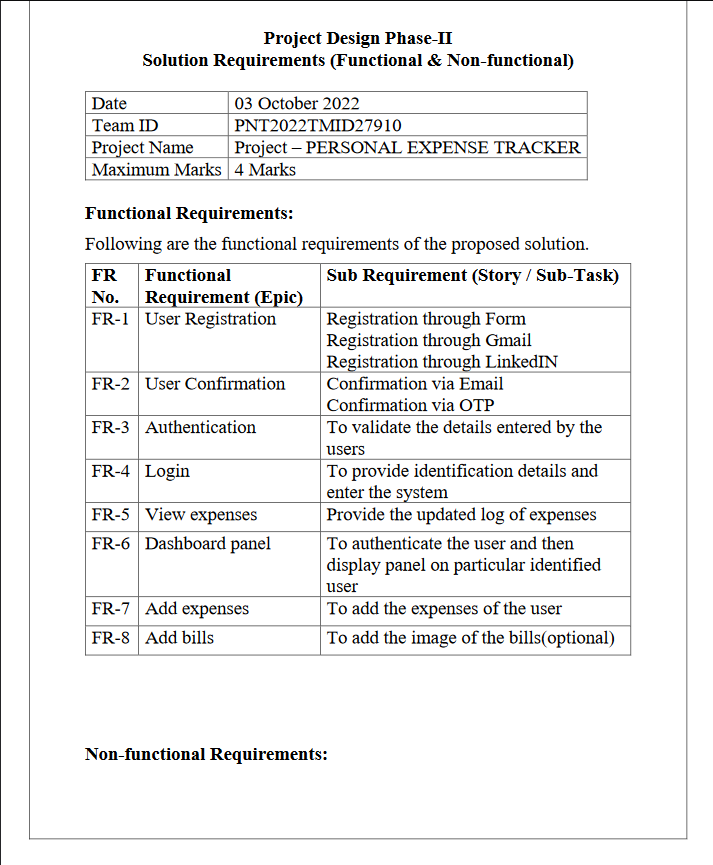
**3.3 Proposed Solution**

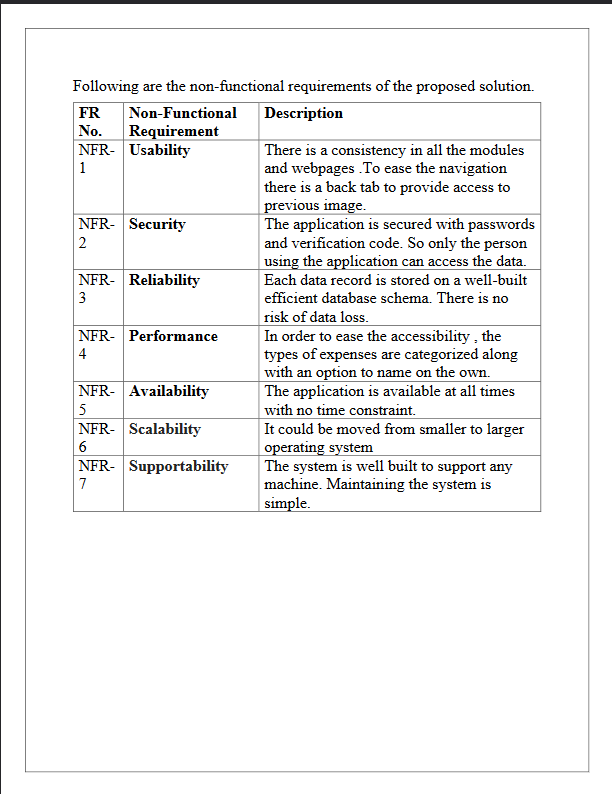
|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
|  | **Problem Statement (Problem to be solved)** | Keeping Proper track of our daily expenses is becoming challenging in today’s world. Without the proper money management knowledge people overspend on their wants instead of focusing on their needs. Especially when using online applications for purchasing their requirements consumers tend to over spend. This problem leads to improper distribution of their daily expenses. Without proper knowledge on managing money poor are becoming poorer and rich are becoming richer. |
|  | **Idea / Solution description** | An attempt to develop an app to manage our daily expenses  and give us insights on managing our money would be a good idea. This app will be able to track expenses on various online platforms and apps. The app can help with proper budgeting and give alerts when the user over spends or crosses the limit previously set by them. This will lead to proper spending habits and make them knowledgeable about money management. IBM cloud can be used to handle the data safely. |
|  | **Novelty / Uniqueness** | The speciality for the app will be the data security with IBM cloud being used for data storage  and this app genuinely helps with the money management. The proper and personalized budgeting of the user’s money leads them to trust the app and they wouldn’t have to worry about their expenditure on unnecessary things. |
|  | **Social Impact / Customer Satisfaction** | People using the app will be becoming better at their spending habits and will be able to save more than their peers who are not using the app. This application aims to improve the users' savings sustainably and steadily which leads them to trust the app without worrying about their money. |
|  | **Business Model (Revenue Model)** | This application leads to a business model, the user can be suggested the right products to buy based on their budget and this can lead to targeted business approaching the right consumers. The model leads to systematic and structured expenses of the user and also leads to predictive analysis of the future expenses of the consumer. This model makes the user more careful with expenses as they are provided with the money management insights. |
|  | **Scalability of the Solution** | This application can be created as a multi user model nationwide. The model can also be modified based on the country’s law on applications and data security which leads to international implementation of this application by maintaining proper gateway rules. This app when developed for multiple nations can be modified to their requirements. The app can also be modified for a particular group of people or organization. |

* 1. **Problem Solution fit**

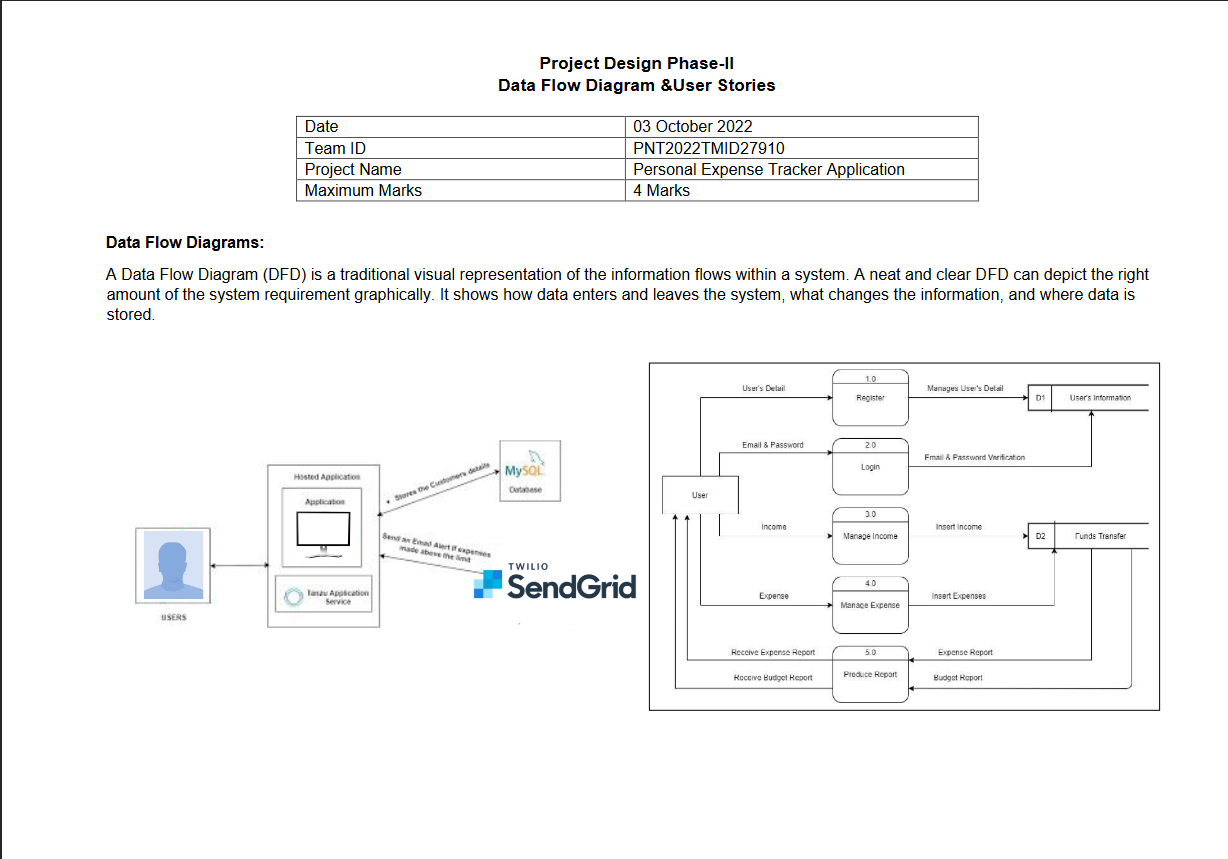
****

**4. REQUIREMENT ANALYSIS**

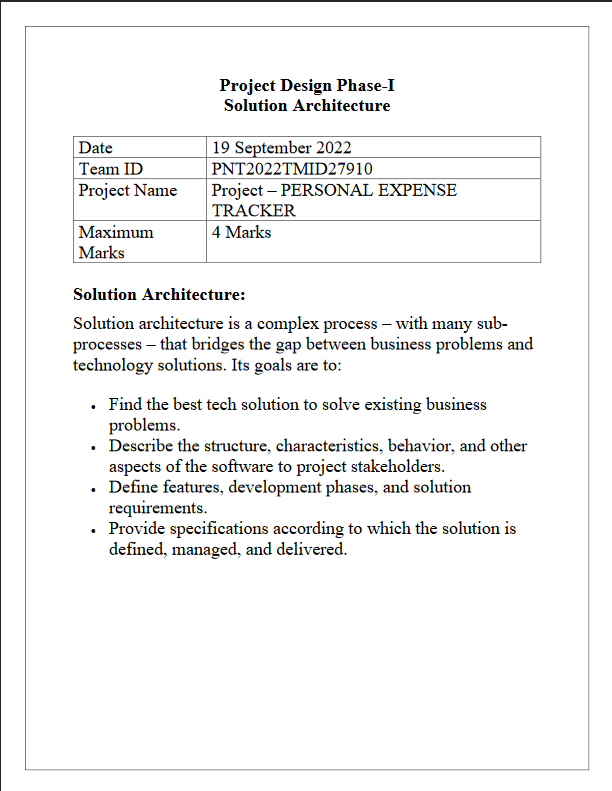
**4.1 Functional requirement**

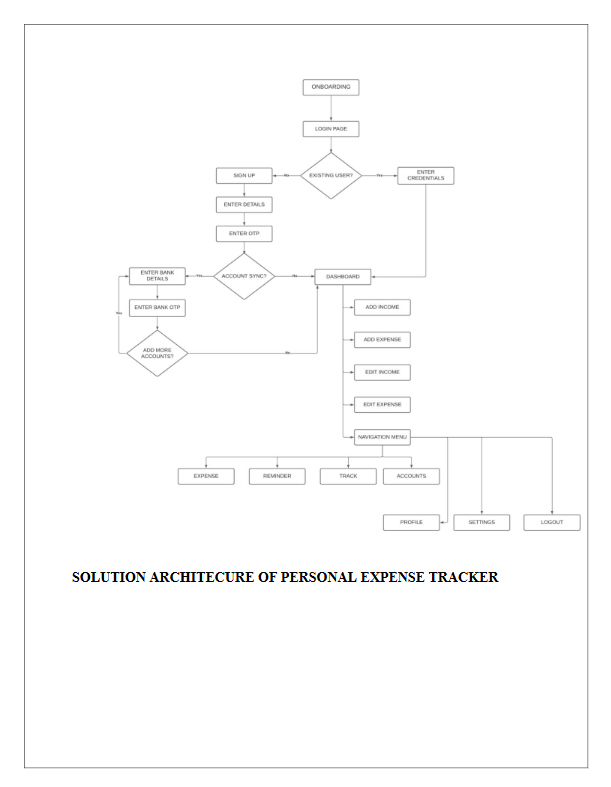
**4.2 Non-Functional requirements**

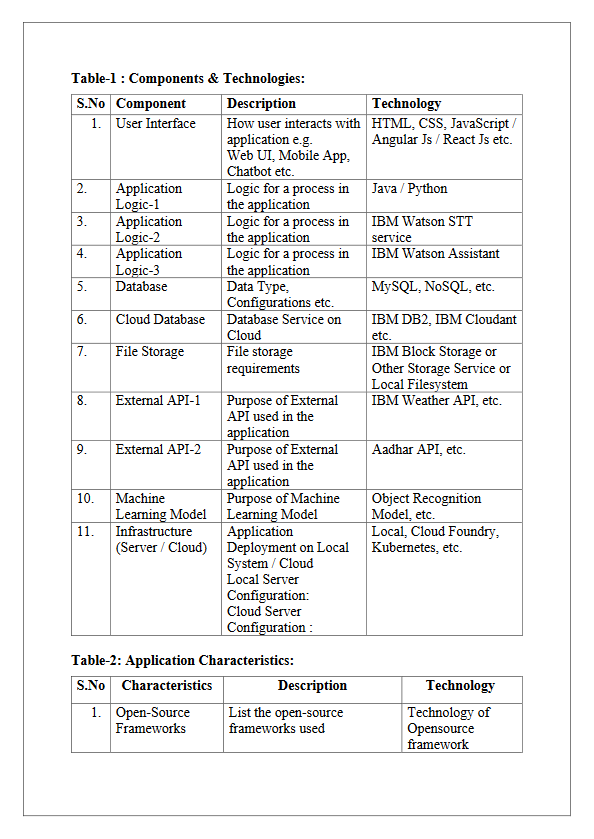
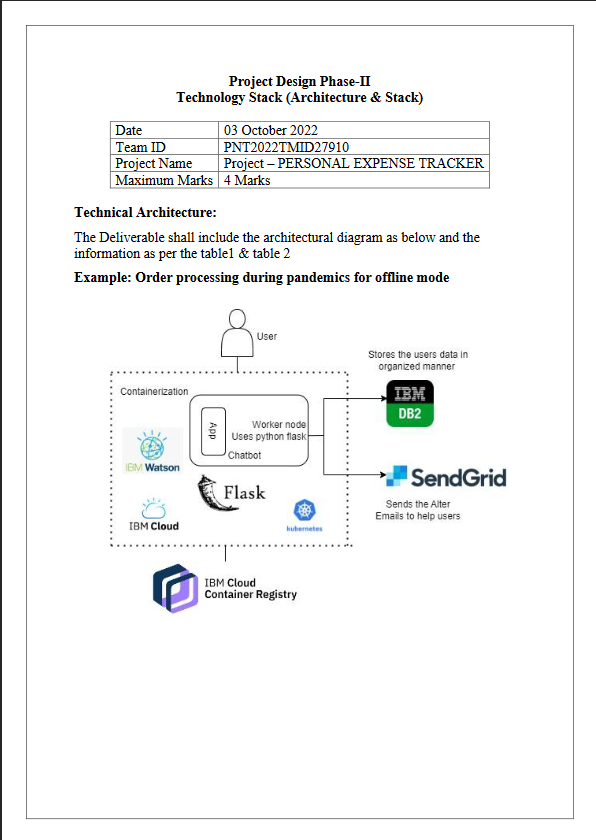
**5. PROJECT DESIGN**

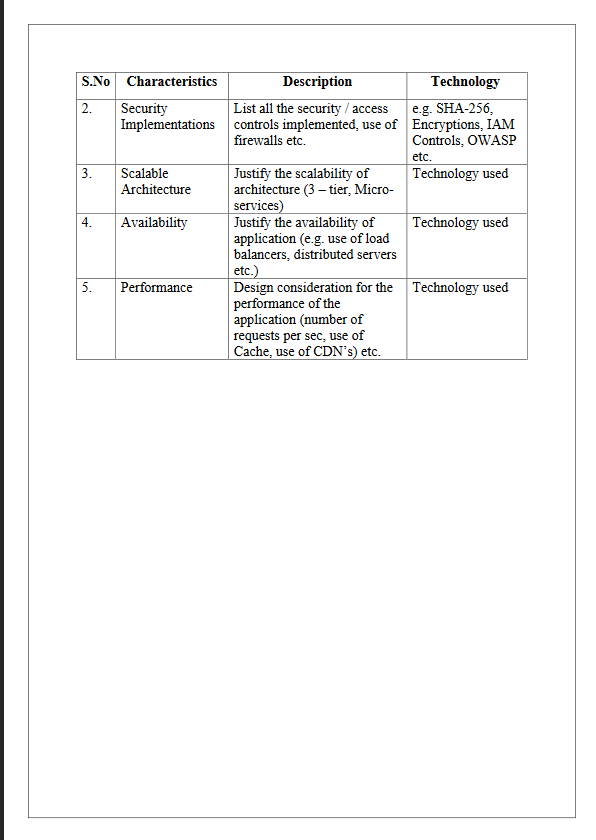
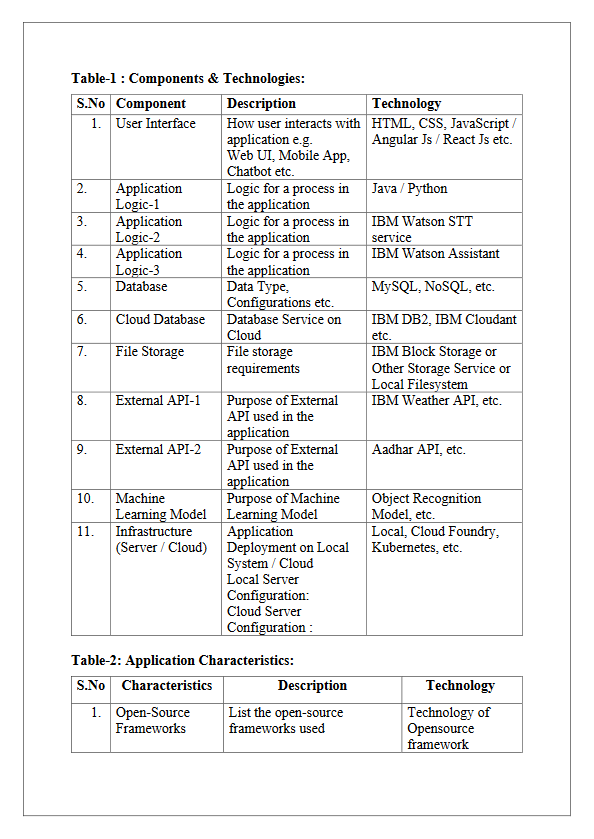
**5.1 Data Flow Diagrams**

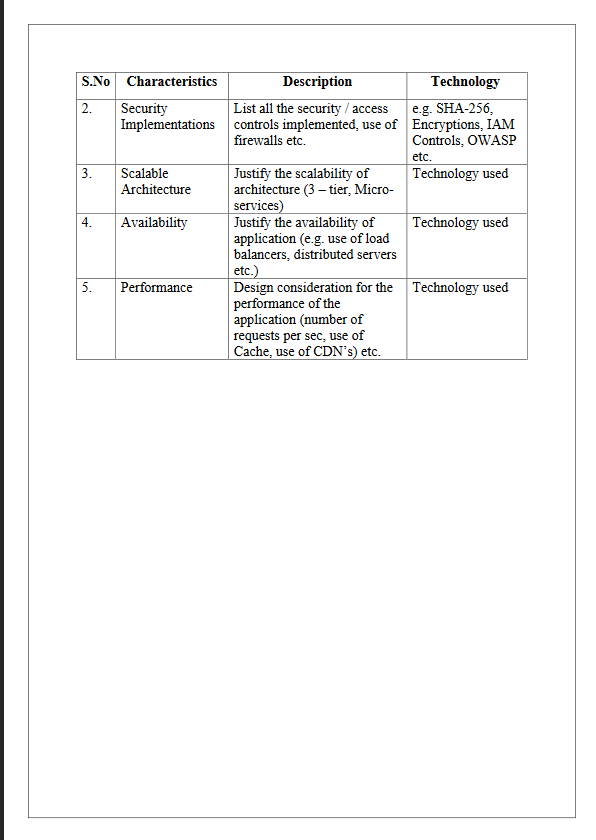
**5.2 Solution & Technical Architecture**

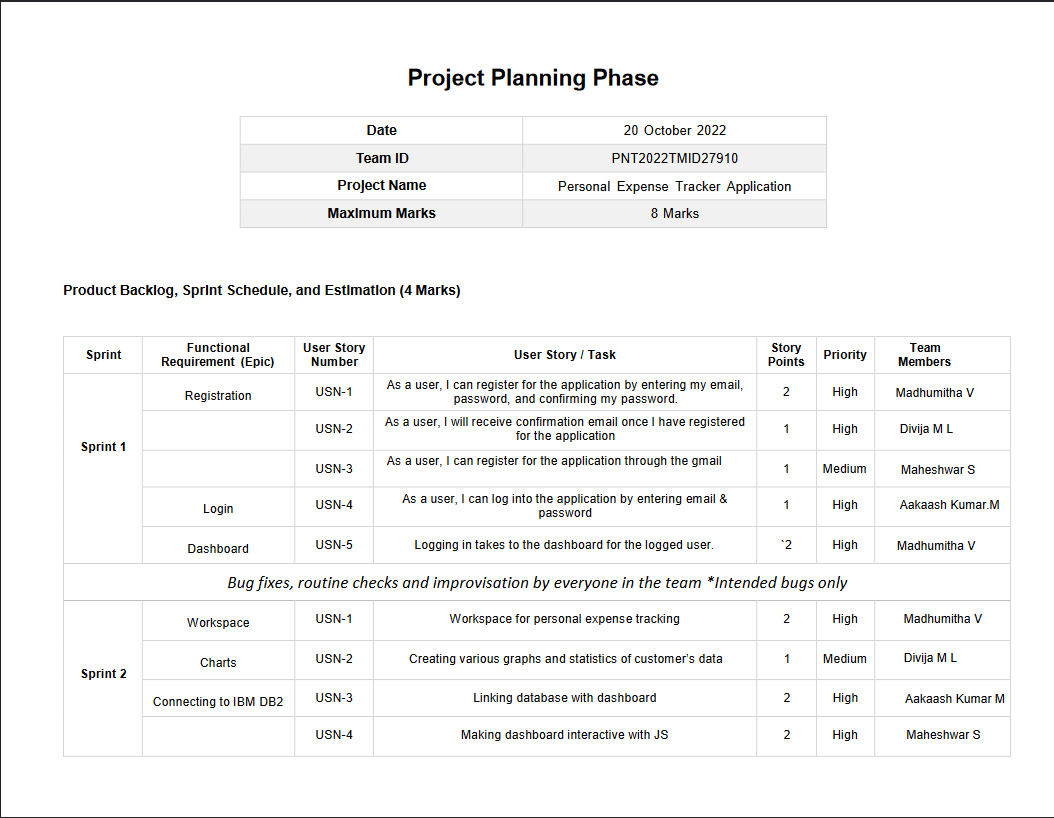
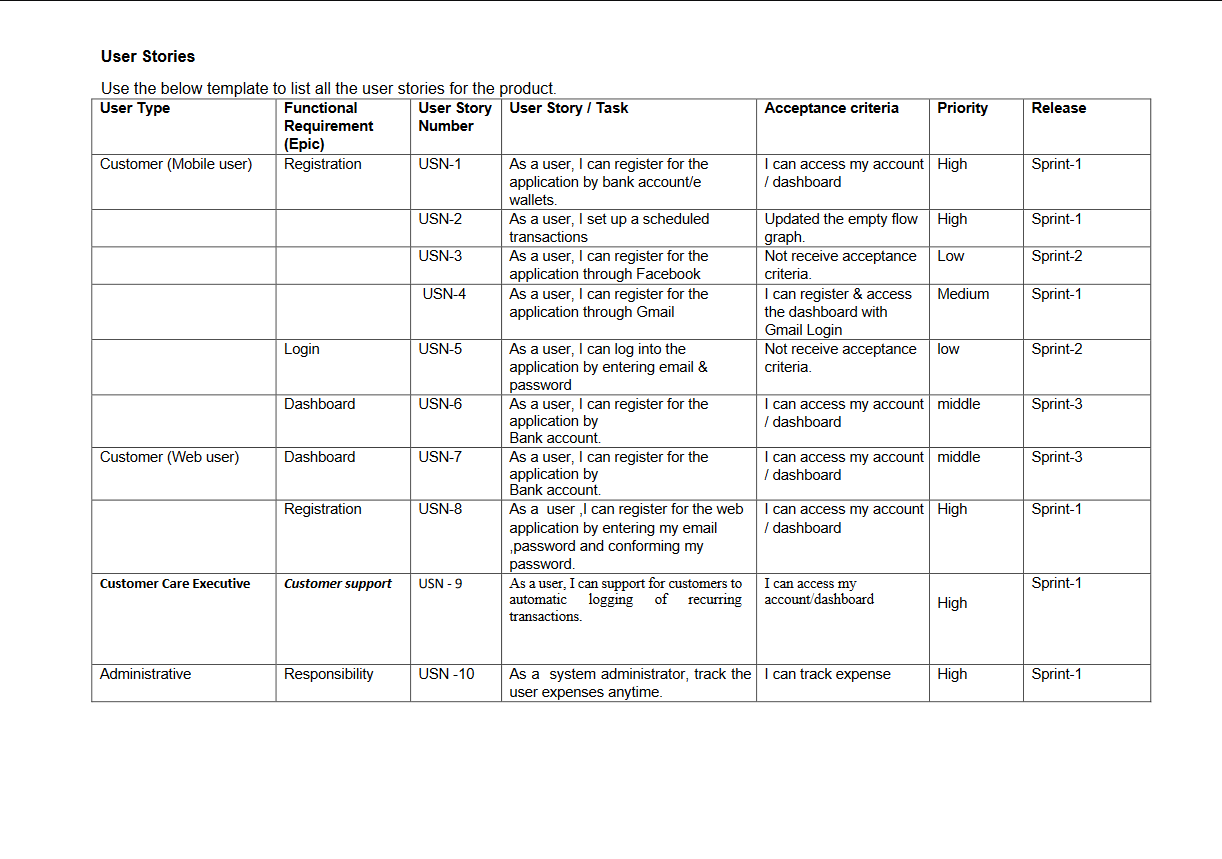
****

****

****

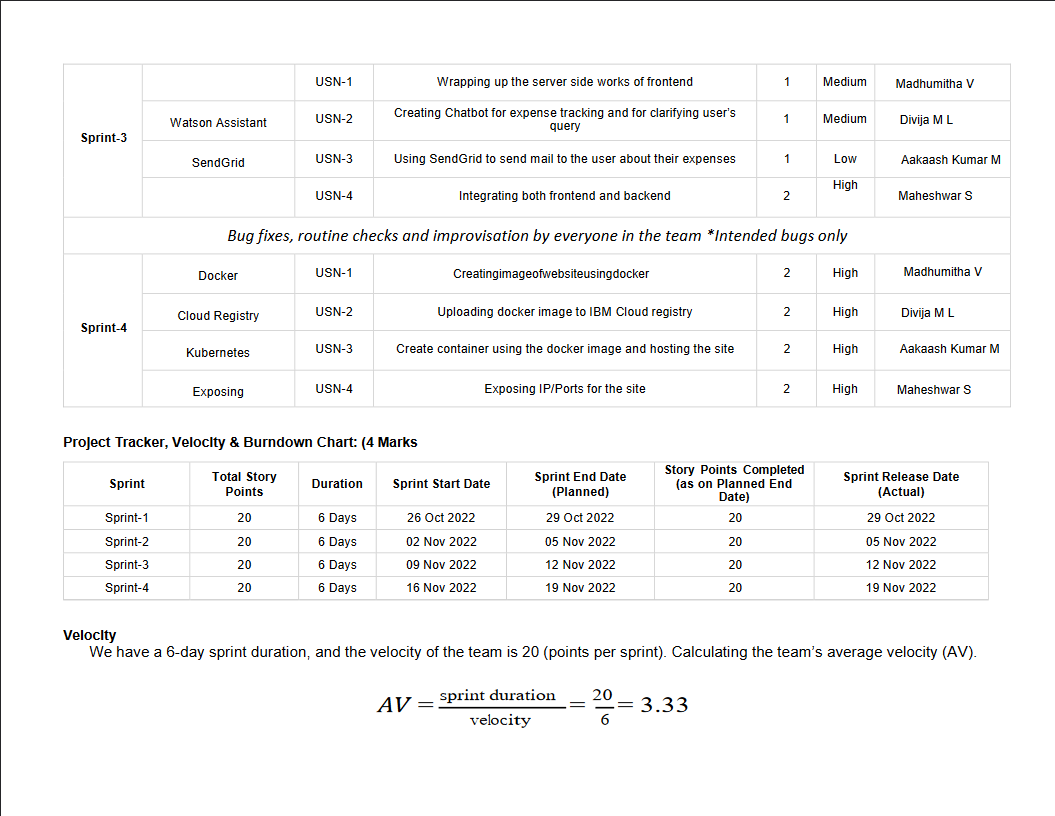
****

****

**5.3 User Stories**

**6. PROJECT PLANNING & SCHEDULING**

**6.1 Sprint Planning , Estimation and Delivery Schedule:**

**6.3 Reports from JIRA **

**7. CODING & SOLUTIONING 8. TESTING**

**#DISPLAY---graph**

**@app.route("/display")**

**def display():**

**query = "SELECT \* FROM expenses where id = ? ORDER BY 'dates' DESC"**

**stmt = ibm\_db.prepare(connection, query)**

**ibm\_db.bind\_param(stmt, 1, session['email'])**

**ibm\_db.execute(stmt)**

**dictionary=ibm\_db.fetch\_assoc(stmt)**

**rexpense=[]**

**while dictionary != False:**

**exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"])**

**rexpense.append(exp)**

**dictionary = ibm\_db.fetch\_assoc(stmt)**

**query = "SELECT MONTH(dates) as DATES, SUM(amount) as AMOUNT FROM expenses WHERE id=? AND YEAR(dates)= YEAR(now()) GROUP BY MONTH(dates);"**

**stmt = ibm\_db.prepare(connection, query)**

**ibm\_db.bind\_param(stmt, 1,session['email'])**

**ibm\_db.execute(stmt)**

**dictionary=ibm\_db.fetch\_assoc(stmt)**

**texpense=[]**

**while dictionary != False:**

**exp=(dictionary["DATES"],dictionary["AMOUNT"])**

**texpense.append(exp)**

**dictionary = ibm\_db.fetch\_assoc(stmt)**

**print(texpense)**

**query = "SELECT \* FROM expenses WHERE id = ? AND YEAR(dates)= YEAR(now());"**

**stmt = ibm\_db.prepare(connection, query)**

**ibm\_db.bind\_param(stmt, 1,session['email'])**

**ibm\_db.execute(stmt)**

**dictionary=ibm\_db.fetch\_assoc(stmt)**

**expense=[]**

**while dictionary != False:**

**exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"])**

**expense.append(exp)**

**dictionary = ibm\_db.fetch\_assoc(stmt)**

**total=0**

**t\_food=0**

**t\_entertainment=0**

**t\_business=0**

**t\_rent=0**

**t\_EMI=0**

**t\_career=0**

**t\_other=0**

**for x in expense:**

**total += x[3]**

**if x[5] == "food":**

**t\_food += x[3]**

**elif x[5] == "entertainment":**

**t\_entertainment += x[3]**

**elif x[5] == "business":**

**t\_business += x[3]**

**elif x[5] == "rent":**

**t\_rent += x[3]**

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

**t\_EMI += x[3]**

**elif x[5] == "Career":**

**t\_career += x[3]**

**elif x[5] == "other":**

**t\_other += x[3]**

**print(total)**

**print(expense)**

**print(t\_food)**

**print(t\_entertainment)**

**print(t\_business)**

**print(t\_rent)**

**print(t\_EMI)**

**print(t\_career)**

**print(t\_other)**

**qur = "SELECT \* FROM expenses WHERE id = ? AND MONTH(dates)= MONTH(now());"**

**stt = ibm\_db.prepare(connection, qur)**

**ibm\_db.bind\_param(stt, 1, session['email'])**

**ibm\_db.execute(stt)**

**dictionary=ibm\_db.fetch\_assoc(stt)**

**lexpense=[]**

**while dictionary != False:**

**exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"])**

**lexpense.append(exp)**

**dictionary = ibm\_db.fetch\_assoc(stt)**

**ttotal=0**

**to\_food=0**

**to\_entertainment=0**

**to\_business=0**

**to\_rent=0**

**to\_EMI=0**

**to\_other=0**

**for x in lexpense:**

**ttotal += x[3]**

**if x[5] == "food":**

**to\_food += x[3]**

**elif x[5] == "entertainment":**

**to\_entertainment += x[3]**

**elif x[5] == "business":**

**to\_business += x[3]**

**elif x[5] == "rent":**

**to\_rent += x[3]**

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

**to\_EMI += x[3]**

**elif x[5] == "other":**

**to\_other += x[3]**

**print(ttotal)**

**qy = "SELECT max(IDX) as IDX FROM limits where id=?;"**

**smt = ibm\_db.prepare(connection, qy)**

**ibm\_db.bind\_param(smt, 1, session['email'])**

**ibm\_db.execute(smt)**

**dictionary = ibm\_db.fetch\_assoc(smt)**

**uexpense=[]**

**while dictionary != False:**

**exp=(dictionary["IDX"])**

**uexpense.append(exp)**

**dictionary = ibm\_db.fetch\_assoc(smt)**

**k=uexpense[0]**

**qu = "SELECT NUMBER FROM limits where id=? and idx=?"**

**sm = ibm\_db.prepare(connection, qu)**

**ibm\_db.bind\_param(sm, 1, session['email'])**

**ibm\_db.bind\_param(sm, 2, k)**

**ibm\_db.execute(sm)**

**dictionary = ibm\_db.fetch\_assoc(sm)**

**fexpense=[]**

**while dictionary != False:**

**exp=(dictionary["NUMBER"])**

**fexpense.append(exp)**

**dictionary = ibm\_db.fetch\_assoc(stmt)**

**if len(fexpense) <= 0:**

**print("Enter the limit First")**

**else:**

**if ttotal > fexpense[0]:**

**m=sendemail.sendgridmail(session["email"])**

**print(m)**

**else: print("Error")**

**return render\_template("display.html",rexpense=rexpense, texpense = texpense, expense = expense, total = total ,**

**t\_food = t\_food,t\_entertainment = t\_entertainment,**

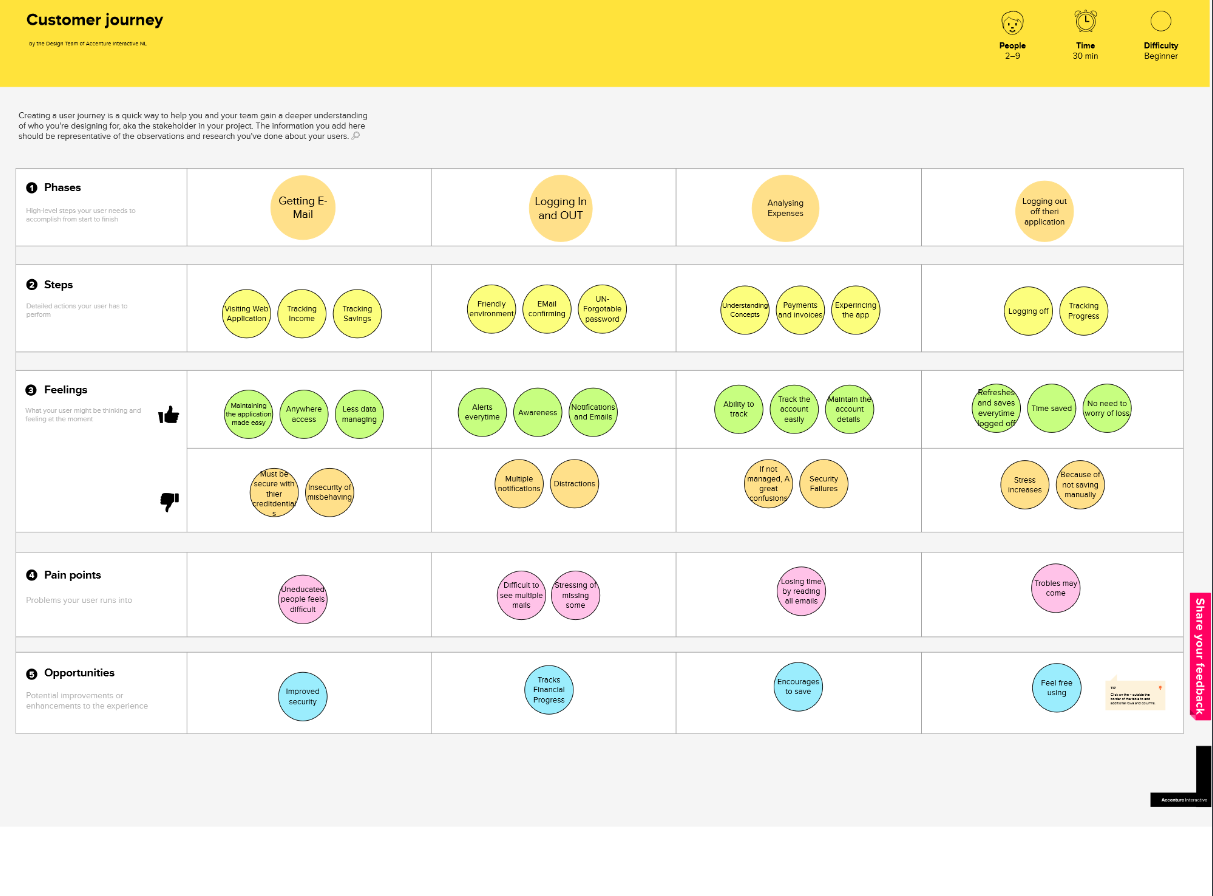
**t\_business = t\_business, t\_rent = t\_rent,**

**t\_EMI = t\_EMI, t\_career = t\_career , t\_other = t\_other )**

Features:

* Analytical Processing of expenses and data.
* Rendering of the recorded data into user readable graphs
* Manual editing of expenses tracked by the app which enables manual corrections if required.

**8.1 Test Cases**



**9. RESULTS**

**9.1 Performance Metrics**

CPU AND MEMORY USAGE:

A major part of the project has been processed with the help of python. This ensures that a very minimal segment of the Processing Unit and the Memory is used. This application can be run as in both foreground and background environments without affecting the realtime performance of the system.

**10. ADVANTAGES:**

* On the go expense submission.
* Access anywhere, anytime.
* No more data loss.
* Real time approval and notifications.
* Better spending awareness.
* Visual representation of data enables easier understanding.

**DISADVANTAGES**:

* Consistency is the key and without it users may eventually lose track of their expenses.
* Information may be less secure with the involvement of external factors.
* Automating everything to do with your finances can make you financially lazy.

**11. CONCLUSION:**

Monitoring your everyday expenses can set aside you cash, yet it can likewise help you set your monetary objectives for what’s to come. On the off chance that you know precisely where your sum is going much of a stretch see where a few reductions and bargains can be made. Expense Tracker project is for keeping our day-to-day expenditures will helps us to keep record of our money daily. The project what we have created is work more proficient than the other income and expense tracker. The project effectively keeps away from the manual figuring for trying not to ascertain the pay and cost each month. It’s a user-friendly application.

**12. FUTURE SCOPE:**

1) It will have various options to keep record (for example Food, Travelling Fuel, Salary etc.).

2) Automatically it will keep on sending notifications for our daily expenditure.

3) In today’s busy and expensive life, we are in a great rush to make moneys, but at the end of the month we broke off. As we are unknowingly spending money on title and unwanted things. So, we have come over with the plan to follow our profit.

4) Here user can define their own categories for expense type like food, clothing, rent and bills where they have to enter the money that has been spend and likewise can add some data in extra data to indicate the expense.

**13. APPENDIX**

**Source Code:**

|  |
| --- |
| from flask import Flask, render\_template, request, redirect, session ,url\_for |
|  |

|  |
| --- |
| import ibm\_db |
|  |

|  |
| --- |
| import re |
|  |

|  |
| --- |
| import sendemail |
|  |

|  |
| --- |
|  |
|  |

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

|  |
| --- |
|  |
|  |

|  |
| --- |
| app.secret\_key = 'a' |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| connection = ibm\_db.connect("DATABASE=bludb;HOSTNAME=9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32459;SECURITY=SSL;SSLServiceCertificate=DigiCertGlobalRootCA.crt;UID=njl64817;PWD=gaHRlHcT8nyConim",'','') |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #HOME--PAGE |
|  |

|  |
| --- |
| @app.route("/home") |
|  |

|  |
| --- |
| def home(): |
|  |

|  |
| --- |
| return render\_template("homepage.html") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route("/") |
|  |

|  |
| --- |
| def add(): |
|  |

|  |
| --- |
| return render\_template("home.html") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #SIGN--UP--OR--REGISTER |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route("/signup") |
|  |

|  |
| --- |
| def signup(): |
|  |

|  |
| --- |
| return render\_template("signup.html") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

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

|  |
| --- |
| def register(): |
|  |

|  |
| --- |
| global user\_email |
|  |

|  |
| --- |
| msg = '' |
|  |

|  |
| --- |
| if request.method == 'POST' : |
|  |

|  |
| --- |
| username = request.form['username'] |
|  |

|  |
| --- |
| email = request.form['email'] |
|  |

|  |
| --- |
| password = request.form['password'] |
|  |

|  |
| --- |
| query = "SELECT \* FROM register WHERE email=?;" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, email) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| account = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| print(account) |
|  |

|  |
| --- |
| if account: |
|  |

|  |
| --- |
| msg = 'Account already exists !' |
|  |

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

|  |
| --- |
| msg = 'Invalid email address !' |
|  |

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

|  |
| --- |
| msg = 'name must contain only characters and numbers !' |
|  |

|  |
| --- |
| else: |
|  |

|  |
| --- |
| query = "INSERT INTO register values(?,?,?);" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

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

|  |
| --- |
| ibm\_db.bind\_param(stmt, 2, email) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 3, password) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| session['loggedin'] = True |
|  |

|  |
| --- |
| session['id'] = email |
|  |

|  |
| --- |
| user\_email = email |
|  |

|  |
| --- |
| session['email'] = email |
|  |

|  |
| --- |
| session['username'] = username |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| msg = 'You have successfully registered ! Proceed Login Process' |
|  |

|  |
| --- |
| return render\_template('login.html', msg = msg) |
|  |

|  |
| --- |
| else: |
|  |

|  |
| --- |
| msg = 'PLEASE FILL OUT OF THE FORM' |
|  |

|  |
| --- |
| return render\_template('register.html', msg=msg) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #LOGIN--PAGE |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route("/signin") |
|  |

|  |
| --- |
| def signin(): |
|  |

|  |
| --- |
| return render\_template('login.html') |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route('/login',methods =['GET', 'POST']) |
|  |

|  |
| --- |
| def login(): |
|  |

|  |
| --- |
| global user\_email |
|  |

|  |
| --- |
| msg = '' |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if request.method == 'POST' : |
|  |

|  |
| --- |
| email = request.form['email'] |
|  |

|  |
| --- |
| password = request.form['password'] |
|  |

|  |
| --- |
| sql = "SELECT \* FROM register WHERE email =? AND password=?;" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, sql) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt,1,email) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt,2,password) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| account = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| print (account) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if account: |
|  |

|  |
| --- |
| session['loggedin'] = True |
|  |

|  |
| --- |
| session['id'] = account['EMAIL'] |
|  |

|  |
| --- |
| user\_email= account['EMAIL'] |
|  |

|  |
| --- |
| session['email']=account['EMAIL'] |
|  |

|  |
| --- |
| session['username'] = account['USERNAME'] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| return redirect('/home') |
|  |

|  |
| --- |
| else: |
|  |

|  |
| --- |
| msg = 'Incorrect username / password !' |
|  |

|  |
| --- |
| return render\_template('login.html', msg = msg) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #CHANGE FORGOT PASSWORD |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route("/forgot") |
|  |

|  |
| --- |
| def forgot(): |
|  |

|  |
| --- |
| return render\_template('forgot.html') |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route("/forgotpw", methods =['GET', 'POST']) |
|  |

|  |
| --- |
| def forgotpw(): |
|  |

|  |
| --- |
| msg = '' |
|  |

|  |
| --- |
| if request.method == 'POST' : |
|  |

|  |
| --- |
| email = request.form['email'] |
|  |

|  |
| --- |
| password = request.form['password'] |
|  |

|  |
| --- |
| query = "SELECT \* FROM register WHERE email=?;" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, email) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| account = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| print(account) |
|  |

|  |
| --- |
| if account: |
|  |

|  |
| --- |
| query = "UPDATE register SET password = ? WHERE email = ?;" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, password) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 2, email) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| msg = 'Successfully changed your password ! Proceed Login Process' |
|  |

|  |
| --- |
| return render\_template('login.html', msg = msg) |
|  |

|  |
| --- |
| else: |
|  |

|  |
| --- |
| msg = 'PLEASE FILL OUT THE CORRECT DETAILS' |
|  |

|  |
| --- |
| return render\_template('forgot.html', msg=msg) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #ADDING----DATA |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route("/add") |
|  |

|  |
| --- |
| def adding(): |
|  |

|  |
| --- |
| return render\_template('add.html') |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route('/addexpense',methods=['GET', 'POST']) |
|  |

|  |
| --- |
| def addexpense(): |
|  |

|  |
| --- |
| global user\_email |
|  |

|  |
| --- |
| global i |
|  |

|  |
| --- |
| que = "SELECT \* FROM expenses where id = ? ORDER BY 'dates' DESC" |
|  |

|  |
| --- |
| stm = ibm\_db.prepare(connection, que) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stm, 1, session['email']) |
|  |

|  |
| --- |
| ibm\_db.execute(stm) |
|  |

|  |
| --- |
| dictionary=ibm\_db.fetch\_assoc(stm) |
|  |

|  |
| --- |
| expense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"]) |
|  |

|  |
| --- |
| expense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stm) |
|  |

|  |
| --- |
| i=len(expense)+1 |
|  |

|  |
| --- |
| indexes=str(i) |
|  |

|  |
| --- |
| dates = request.form['date'] |
|  |

|  |
| --- |
| expensename = request.form['expensename'] |
|  |

|  |
| --- |
| amount = request.form['amount'] |
|  |

|  |
| --- |
| paymode = request.form['paymode'] |
|  |

|  |
| --- |
| category = request.form['category'] |
|  |

|  |
| --- |
| query = "INSERT INTO expenses VALUES (?,?,?,?,?,?,?);" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, session['email']) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 2, dates) |
|  |

|  |
| --- |
| 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.bind\_param(stmt, 7, indexes) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| print(dates + " " + expensename + " " + amount + " " + paymode + " " + category) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| return redirect("/display") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #DISPLAY---graph |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route("/display") |
|  |

|  |
| --- |
| def display(): |
|  |

|  |
| --- |
| query = "SELECT \* FROM expenses where id = ? ORDER BY 'dates' DESC" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, session['email']) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| dictionary=ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| rexpense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"]) |
|  |

|  |
| --- |
| rexpense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| query = "SELECT MONTH(dates) as DATES, SUM(amount) as AMOUNT FROM expenses WHERE id=? AND YEAR(dates)= YEAR(now()) GROUP BY MONTH(dates);" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1,session['email']) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| dictionary=ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| texpense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["DATES"],dictionary["AMOUNT"]) |
|  |

|  |
| --- |
| texpense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| print(texpense) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| query = "SELECT \* FROM expenses WHERE id = ? AND YEAR(dates)= YEAR(now());" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1,session['email']) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| dictionary=ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| expense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"]) |
|  |

|  |
| --- |
| expense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| total=0 |
|  |

|  |
| --- |
| t\_food=0 |
|  |

|  |
| --- |
| t\_entertainment=0 |
|  |

|  |
| --- |
| t\_business=0 |
|  |

|  |
| --- |
| t\_rent=0 |
|  |

|  |
| --- |
| t\_EMI=0 |
|  |

|  |
| --- |
| t\_career=0 |
|  |

|  |
| --- |
| t\_other=0 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| for x in expense: |
|  |

|  |
| --- |
| total += x[3] |
|  |

|  |
| --- |
| if x[5] == "food": |
|  |

|  |
| --- |
| t\_food += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "entertainment": |
|  |

|  |
| --- |
| t\_entertainment += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "business": |
|  |

|  |
| --- |
| t\_business += x[3] |
|  |

|  |
| --- |
| elif x[5] == "rent": |
|  |

|  |
| --- |
| t\_rent += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "EMI": |
|  |

|  |
| --- |
| t\_EMI += x[3] |
|  |

|  |
| --- |
| elif x[5] == "Career": |
|  |

|  |
| --- |
| t\_career += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "other": |
|  |

|  |
| --- |
| t\_other += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| print(total) |
|  |

|  |
| --- |
| print(expense) |
|  |

|  |
| --- |
| print(t\_food) |
|  |

|  |
| --- |
| print(t\_entertainment) |
|  |

|  |
| --- |
| print(t\_business) |
|  |

|  |
| --- |
| print(t\_rent) |
|  |

|  |
| --- |
| print(t\_EMI) |
|  |

|  |
| --- |
| print(t\_career) |
|  |

|  |
| --- |
| print(t\_other) |
|  |

|  |
| --- |
| qur = "SELECT \* FROM expenses WHERE id = ? AND MONTH(dates)= MONTH(now());" |
|  |

|  |
| --- |
| stt = ibm\_db.prepare(connection, qur) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stt, 1, session['email']) |
|  |

|  |
| --- |
| ibm\_db.execute(stt) |
|  |

|  |
| --- |
| dictionary=ibm\_db.fetch\_assoc(stt) |
|  |

|  |
| --- |
| lexpense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"]) |
|  |

|  |
| --- |
| lexpense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stt) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ttotal=0 |
|  |

|  |
| --- |
| to\_food=0 |
|  |

|  |
| --- |
| to\_entertainment=0 |
|  |

|  |
| --- |
| to\_business=0 |
|  |

|  |
| --- |
| to\_rent=0 |
|  |

|  |
| --- |
| to\_EMI=0 |
|  |

|  |
| --- |
| to\_other=0 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| for x in lexpense: |
|  |

|  |
| --- |
| ttotal += x[3] |
|  |

|  |
| --- |
| if x[5] == "food": |
|  |

|  |
| --- |
| to\_food += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "entertainment": |
|  |

|  |
| --- |
| to\_entertainment += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "business": |
|  |

|  |
| --- |
| to\_business += x[3] |
|  |

|  |
| --- |
| elif x[5] == "rent": |
|  |

|  |
| --- |
| to\_rent += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "EMI": |
|  |

|  |
| --- |
| to\_EMI += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "other": |
|  |

|  |
| --- |
| to\_other += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| print(ttotal) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| qy = "SELECT max(IDX) as IDX FROM limits where id=?;" |
|  |

|  |
| --- |
| smt = ibm\_db.prepare(connection, qy) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(smt, 1, session['email']) |
|  |

|  |
| --- |
| ibm\_db.execute(smt) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(smt) |
|  |

|  |
| --- |
| uexpense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["IDX"]) |
|  |

|  |
| --- |
| uexpense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(smt) |
|  |

|  |
| --- |
| k=uexpense[0] |
|  |

|  |
| --- |
| qu = "SELECT NUMBER FROM limits where id=? and idx=?" |
|  |

|  |
| --- |
| sm = ibm\_db.prepare(connection, qu) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(sm, 1, session['email']) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(sm, 2, k) |
|  |

|  |
| --- |
| ibm\_db.execute(sm) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(sm) |
|  |

|  |
| --- |
| fexpense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["NUMBER"]) |
|  |

|  |
| --- |
| fexpense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if len(fexpense) <= 0: |
|  |

|  |
| --- |
| print("Enter the limit First") |
|  |

|  |
| --- |
| else: |
|  |

|  |
| --- |
| if ttotal > fexpense[0]: |
|  |

|  |
| --- |
| m=sendemail.sendgridmail(session["email"]) |
|  |

|  |
| --- |
| print(m) |
|  |

|  |
| --- |
| else: print("Error") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| return render\_template("display.html",rexpense=rexpense, 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\_career = t\_career , t\_other = t\_other ) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #delete---the--data |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route('/delete/<idx>', methods = ['POST', 'GET' ]) |
|  |

|  |
| --- |
| def delete(idx): |
|  |

|  |
| --- |
| query = "DELETE FROM expenses WHERE id=? and idx=?;" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, session["email"]) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 2, idx) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| print('deleted successfully') |
|  |

|  |
| --- |
| return render\_template("display.html") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #UPDATE---DATA |
|  |

|  |
| --- |
|  |
|  |

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

|  |
| --- |
| def edit(id): |
|  |

|  |
| --- |
| query = "SELECT \* FROM expenses WHERE id=? and idx=?;" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, session['email']) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 2, id) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| dictionary=ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| expense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"]) |
|  |

|  |
| --- |
| expense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| print(expense) |
|  |

|  |
| --- |
| return render\_template('edit.html', expenses = expense[0]) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

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

|  |
| --- |
| def update(id): |
|  |

|  |
| --- |
| if request.method == 'POST' : |
|  |

|  |
| --- |
| dates = request.form['date'] |
|  |

|  |
| --- |
| expensename = request.form['expensename'] |
|  |

|  |
| --- |
| amount = request.form['amount'] |
|  |

|  |
| --- |
| paymode = request.form['paymode'] |
|  |

|  |
| --- |
| category = request.form['category'] |
|  |

|  |
| --- |
| query = "UPDATE expenses SET dates = ? , expensename = ? , amount = ?, paymode = ?, category = ? WHERE id = ? and idx=?;" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, dates) |
|  |

|  |
| --- |
| 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, session['email']) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 7, id) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| print('successfully updated') |
|  |

|  |
| --- |
| return redirect("/display") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route("/limit" ) |
|  |

|  |
| --- |
| def limit(): |
|  |

|  |
| --- |
| return render\_template('limit.html') |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route("/limitnum" , methods = ['GET','POST' ]) |
|  |

|  |
| --- |
| def limitnum(): |
|  |

|  |
| --- |
| que = "SELECT \* FROM limits where id = ? ;" |
|  |

|  |
| --- |
| stm = ibm\_db.prepare(connection, que) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stm, 1, session['email']) |
|  |

|  |
| --- |
| ibm\_db.execute(stm) |
|  |

|  |
| --- |
| if request.method == "POST": |
|  |

|  |
| --- |
| dictionary=ibm\_db.fetch\_assoc(stm) |
|  |

|  |
| --- |
| expense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary['ID'],dictionary['NUMBER'],dictionary['IDX']) |
|  |

|  |
| --- |
| expense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stm) |
|  |

|  |
| --- |
| i=len(expense)+1 |
|  |

|  |
| --- |
| idx=str(i) |
|  |

|  |
| --- |
| number= request.form['number'] |
|  |

|  |
| --- |
| query = "INSERT INTO limits VALUES(?,?,?)" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, session['email']) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 2, number) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 3, idx) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| return redirect('/limitn') |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route("/limitn") |
|  |

|  |
| --- |
| def limitn(): |
|  |

|  |
| --- |
| query = "SELECT max(IDX) as IDX FROM limits where id=?;" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, session['email']) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| expense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["IDX"]) |
|  |

|  |
| --- |
| expense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| k=expense[0] |
|  |

|  |
| --- |
| que = "SELECT NUMBER FROM limits where id=? and idx=?" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, que) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, session['email']) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 2, k) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| texpense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["NUMBER"]) |
|  |

|  |
| --- |
| texpense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| s=texpense[0] |
|  |

|  |
| --- |
| return render\_template("limit.html" , y= s) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #REPORT |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route("/today") |
|  |

|  |
| --- |
| def today(): |
|  |

|  |
| --- |
| query = "SELECT dates, amount FROM expenses WHERE id = ? AND DATE(dates) = DATE(NOW()); " |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, str(session['email'])) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| dictionary=ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| texpense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["DATES"],dictionary["AMOUNT"]) |
|  |

|  |
| --- |
| texpense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| print(texpense) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| query = "SELECT \* FROM expenses WHERE id = ? AND DATE(dates) = DATE(NOW())" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, session['email']) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| dictionary=ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| expense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"]) |
|  |

|  |
| --- |
| expense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| total=0 |
|  |

|  |
| --- |
| t\_food=0 |
|  |

|  |
| --- |
| t\_entertainment=0 |
|  |

|  |
| --- |
| t\_business=0 |
|  |

|  |
| --- |
| t\_rent=0 |
|  |

|  |
| --- |
| t\_EMI=0 |
|  |

|  |
| --- |
| t\_career=0 |
|  |

|  |
| --- |
| t\_other=0 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| for x in expense: |
|  |

|  |
| --- |
| total += x[0] |
|  |

|  |
| --- |
| if x[2] == "food": |
|  |

|  |
| --- |
| t\_food += x[0] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[2] == "entertainment": |
|  |

|  |
| --- |
| t\_entertainment += x[0] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[2] == "business": |
|  |

|  |
| --- |
| t\_business += x[0] |
|  |

|  |
| --- |
| elif x[2] == "rent": |
|  |

|  |
| --- |
| t\_rent += x[0] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[2] == "EMI": |
|  |

|  |
| --- |
| t\_EMI += x[0] |
|  |

|  |
| --- |
| elif x[2] == "Career": |
|  |

|  |
| --- |
| t\_career += x[0] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[2] == "other": |
|  |

|  |
| --- |
| t\_other += x[0] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| print(total) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| print(t\_food) |
|  |

|  |
| --- |
| print(t\_entertainment) |
|  |

|  |
| --- |
| print(t\_business) |
|  |

|  |
| --- |
| print(t\_rent) |
|  |

|  |
| --- |
| print(t\_EMI) |
|  |

|  |
| --- |
| print(t\_career) |
|  |

|  |
| --- |
| 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\_career = t\_career , t\_other = t\_other ) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route("/month") |
|  |

|  |
| --- |
| def month(): |
|  |

|  |
| --- |
| query = "SELECT dates, SUM(amount) as AMOUNT FROM expenses WHERE id= ? AND MONTH(dates)= MONTH(now()) GROUP BY dates ORDER BY dates;" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, str(session['email'])) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| dictionary=ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| texpense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["DATES"],dictionary["AMOUNT"]) |
|  |

|  |
| --- |
| texpense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| print(texpense) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| query = "SELECT \* FROM expenses WHERE id = ? AND MONTH(dates)= MONTH(now());" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1, session['email']) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| dictionary=ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| expense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"]) |
|  |

|  |
| --- |
| expense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| total=0 |
|  |

|  |
| --- |
| t\_food=0 |
|  |

|  |
| --- |
| t\_entertainment=0 |
|  |

|  |
| --- |
| t\_business=0 |
|  |

|  |
| --- |
| t\_rent=0 |
|  |

|  |
| --- |
| t\_EMI=0 |
|  |

|  |
| --- |
| t\_career=0 |
|  |

|  |
| --- |
| t\_other=0 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| for x in expense: |
|  |

|  |
| --- |
| total += x[3] |
|  |

|  |
| --- |
| if x[5] == "food": |
|  |

|  |
| --- |
| t\_food += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "entertainment": |
|  |

|  |
| --- |
| t\_entertainment += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "business": |
|  |

|  |
| --- |
| t\_business += x[3] |
|  |

|  |
| --- |
| elif x[5] == "rent": |
|  |

|  |
| --- |
| t\_rent += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "EMI": |
|  |

|  |
| --- |
| t\_EMI += x[3] |
|  |

|  |
| --- |
| elif x[5] == "Career": |
|  |

|  |
| --- |
| t\_career += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "other": |
|  |

|  |
| --- |
| t\_other += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| print(total) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| print(t\_food) |
|  |

|  |
| --- |
| print(t\_entertainment) |
|  |

|  |
| --- |
| print(t\_business) |
|  |

|  |
| --- |
| print(t\_rent) |
|  |

|  |
| --- |
| print(t\_EMI) |
|  |

|  |
| --- |
| print(t\_career) |
|  |

|  |
| --- |
| print(t\_other) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| return render\_template("month.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\_career = t\_career , t\_other = t\_other ) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route("/year") |
|  |

|  |
| --- |
| def year(): |
|  |

|  |
| --- |
| query = "SELECT MONTH(dates) as DATES, SUM(amount) as AMOUNT FROM expenses WHERE id=? AND YEAR(dates)= YEAR(now()) GROUP BY MONTH(dates);" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1,session['email']) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| dictionary=ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| texpense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["DATES"],dictionary["AMOUNT"]) |
|  |

|  |
| --- |
| texpense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| print(texpense) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| query = "SELECT \* FROM expenses WHERE id = ? AND YEAR(dates)= YEAR(now());" |
|  |

|  |
| --- |
| stmt = ibm\_db.prepare(connection, query) |
|  |

|  |
| --- |
| ibm\_db.bind\_param(stmt, 1,session['email']) |
|  |

|  |
| --- |
| ibm\_db.execute(stmt) |
|  |

|  |
| --- |
| dictionary=ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
| expense=[] |
|  |

|  |
| --- |
| while dictionary != False: |
|  |

|  |
| --- |
| exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"]) |
|  |

|  |
| --- |
| expense.append(exp) |
|  |

|  |
| --- |
| dictionary = ibm\_db.fetch\_assoc(stmt) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| total=0 |
|  |

|  |
| --- |
| t\_food=0 |
|  |

|  |
| --- |
| t\_entertainment=0 |
|  |

|  |
| --- |
| t\_business=0 |
|  |

|  |
| --- |
| t\_rent=0 |
|  |

|  |
| --- |
| t\_EMI=0 |
|  |

|  |
| --- |
| t\_career=0 |
|  |

|  |
| --- |
| t\_other=0 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| for x in expense: |
|  |

|  |
| --- |
| total += x[3] |
|  |

|  |
| --- |
| if x[5] == "food": |
|  |

|  |
| --- |
| t\_food += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "entertainment": |
|  |

|  |
| --- |
| t\_entertainment += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "business": |
|  |

|  |
| --- |
| t\_business += x[3] |
|  |

|  |
| --- |
| elif x[5] == "rent": |
|  |

|  |
| --- |
| t\_rent += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "EMI": |
|  |

|  |
| --- |
| t\_EMI += x[3] |
|  |

|  |
| --- |
| elif x[5] == "Career": |
|  |

|  |
| --- |
| t\_career += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| elif x[5] == "other": |
|  |

|  |
| --- |
| t\_other += x[3] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| print(total) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| print(t\_food) |
|  |

|  |
| --- |
| print(t\_entertainment) |
|  |

|  |
| --- |
| print(t\_business) |
|  |

|  |
| --- |
| print(t\_rent) |
|  |

|  |
| --- |
| print(t\_EMI) |
|  |

|  |
| --- |
| print(t\_career) |
|  |

|  |
| --- |
| print(t\_other) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| return render\_template("year.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\_career=t\_career , t\_other = t\_other ) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| #log-out |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| @app.route('/logout') |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| def logout(): |
|  |

|  |
| --- |
| session.pop('loggedin', None) |
|  |

|  |
| --- |
| session.pop('id', None) |
|  |

|  |
| --- |
| session.pop('username', None) |
|  |

|  |
| --- |
| return render\_template('home.html') |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if \_\_name\_\_ == "\_\_main\_\_": |
|  |

app.run(debug=True)

**GITHUB:** <https://github.com/IBM-EPBL/IBM-Project-20438-1659719451.git>

**DEMO VIDEO LINK (GOOGLE DRIVE):**

https://drive.google.com/drive/folders/1\_NX0z9x3seWmZBgNY2nxU2O4q1NMv99t?usp=sharing