Project Report Format

Personal Expense Tracker Application

PNT2022TMID26057

Prithivi Krishna.PK

Yeshwanth

Chandra Mouli.M

Vikraman. A

1. **INTRODUCTION**

   a. Project Overview

The Daily Expense Tracker System separates inputs into daily expense allocable amounts. If you go over that day's allocable amount, it deducts it from your income and gives you a new daily expense allocable amount; if that day's allocable amount is less, it adds it to your savings. At the end of the month, a report generating system for daily expenses will produce the income-expenditure curve. You can use it to add any savings you have set aside for special occasions like birthdays and anniversaries.

   b. Purpose

The "Expense Tracker" programme was created to help manage daily spending in a more effective and practical manner. utilising this programme. We can lessen the manual tracking of expenditures and daily expense calculations.

2. **LITERATURE SURVEY**

   a. Existing problem

Using a manual accounting system may have a number of drawbacks. Any business' accounting process can be challenging. A computerised accounting system may not require you to comprehend the accounting process in the same way that a manual accounting system does. Depending on who is conducting the bookkeeping, this could be a benefit or a drawback. To ensure that accounting is

done correctly, a skilled professional is frequently required. By hand, it could take a while to sort through your financial records' intricacy. Report generation takes

b. References

[1]. Palestinian Ministry of Education and Higher Education. Palestinian Higher Education Statistics.

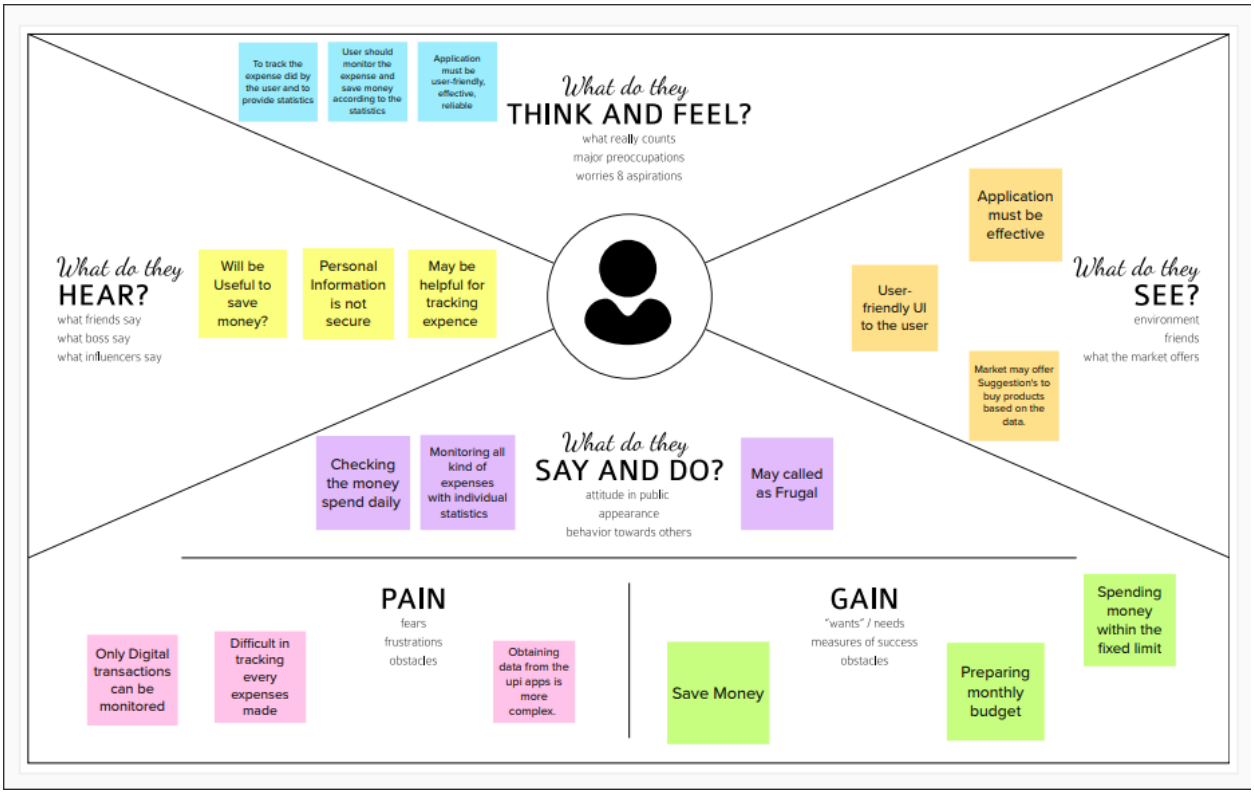
[2]. Accreditation and Quality Assurance Committee (AQAC) in Palestine. General Report of Information Technology and Engineering Higher Education in Palestine. Accreditation and Quality Assurance Commission (AQAC). Ramallah, Palestine: Palestinian Ministry of Education and Higher Education; 2007 Apr.

[3]. Engineering Association of Palestine. Current Engineering Statistics Book. Ramallah; 2005.

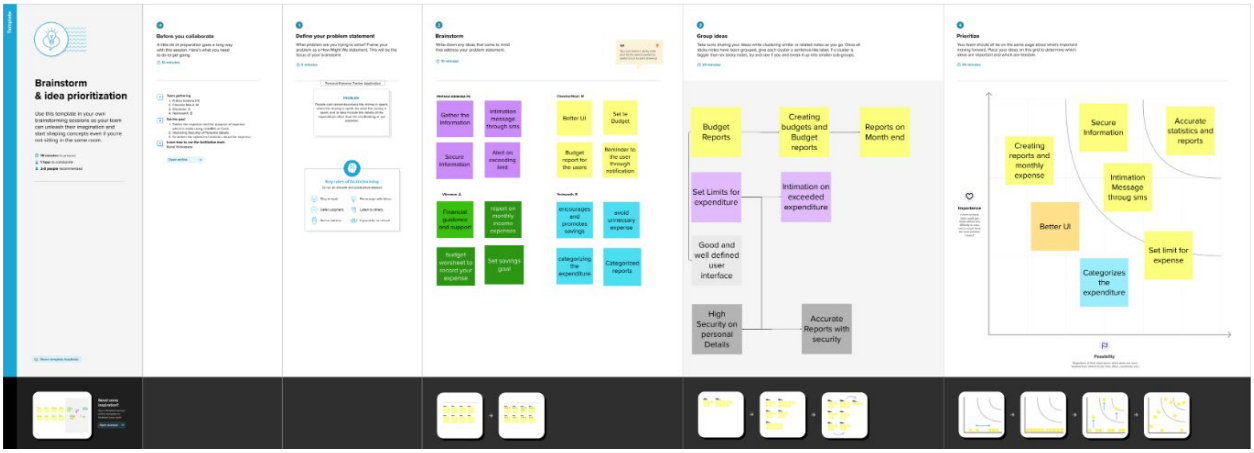
[4]. Prados J, Peterson G, Lattuca L. Quality Assurance of Engineering Education Through Accreditation: The Impact of Engineering Criteria 2000 and Its Global Influence. Journal of Engineering Education. 2005 Jan; 94(1):165–84.

3. IDEATION & PROPOSED SOLUTION

a. Empathy Map Canvas



b. Ideation & Brainstorming



c. Proposed Solution

Project Design Phase-I Proposed Solution Template

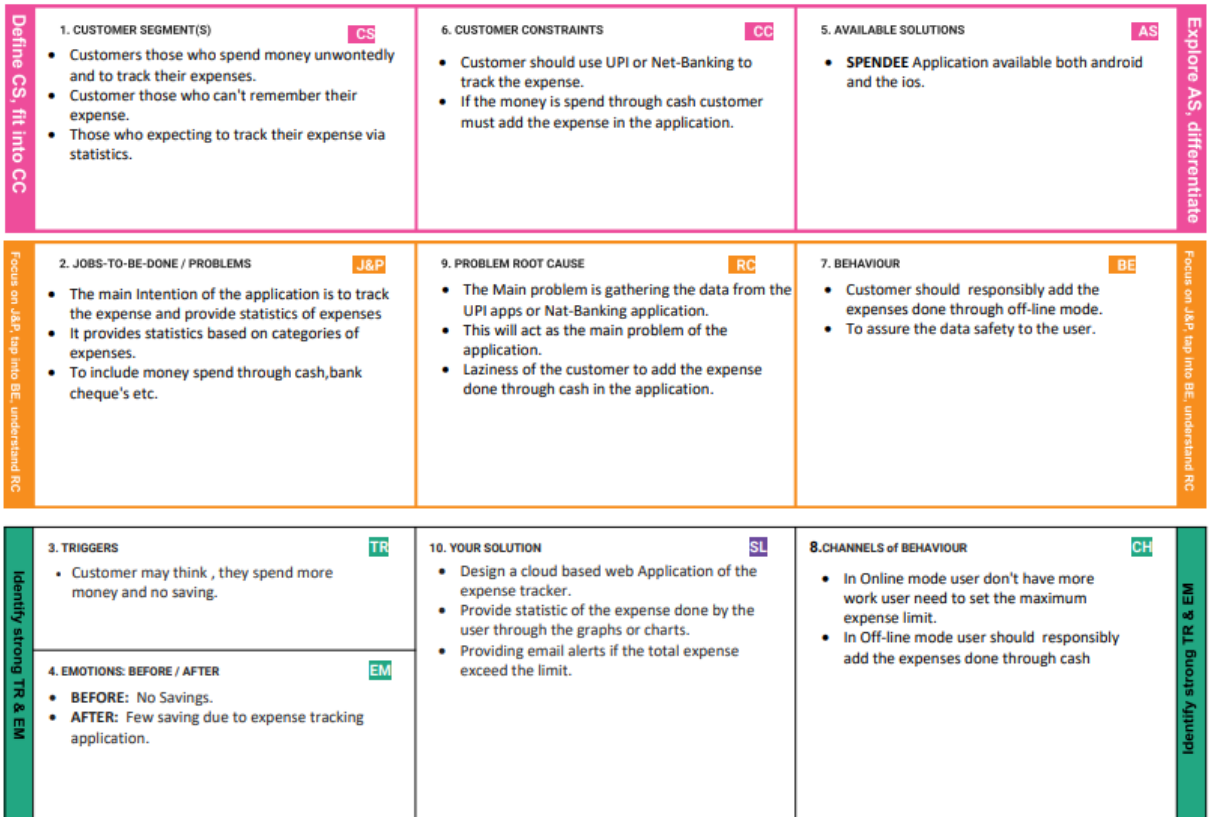
Date	06 October 2022
Team ID	PNT2022TMID26057
Project Name	Project - Personal expense tracker Application
Maximum Marks	2 Marks

Proposed Solution Template:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	There are many budgeting tools online, but not all of them are effective in assisting users in actually creating and adhering to a budget. The ongoing maintenance, the consolidation of all user financial accounts and activity into a single dashboard are some of the negatives. However, a lot of this current software includes convoluted features that are difficult to use. The major problem is to take count paper receipts and calculate the expense statistics.
2.	Idea / Solution description	People tend to neglect their budgets due to their busy and chaotic lifestyles, which results in them spending more than they planned. Future costs cannot be foreseen by the user. Their carelessness with money management will be a concern even though they can record their expense.
3.	Novelty / Uniqueness	Including all the expense including money spend using cash, Bank-cheques, etc. This application keeps track of all of your spending. To enter your expense, simply click. To decrease human error, prevent data loss, and expedite settlements. In this program to display the pie chart or graph lines.
4.	Social Impact / Customer Satisfaction	One can keep track of their own costs and create a monthly or annual budget with this tool. The application will display the statistics and sent alert message, if your spending exceeds the specified limit.
5.	Business Model (Revenue Model)	The subscription/premium to access extra features of this application can be used by business people, and also adding advertisement to generate revenue.
6.	Scalability of the Solution	Using IBM-cloud statistics can be shown to the user with high scalability, and with high accuracy.

d. Problem Solution fit

Project Title: Personal Expense Tracker Application **Project Design Phase-I - Solution Fit Template** **Team ID:** PNT2022TMID26057



4. REQUIREMENT ANALYSIS

a. Functional requirement

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Data	Data gathered in the application server is saved in the high security cloud server.
FR-4	Alert Notification	Alert messages through the Email or SMS.
FR-5	User Monthly Budget Plan	Setting Monthly budget to manage their expenses.
FR-6	Cloud Data Storage	To save the user valuable data high security cloud storage are used (AWS, IBM, GOOGLE, etc)

b. Non-Functional requirements

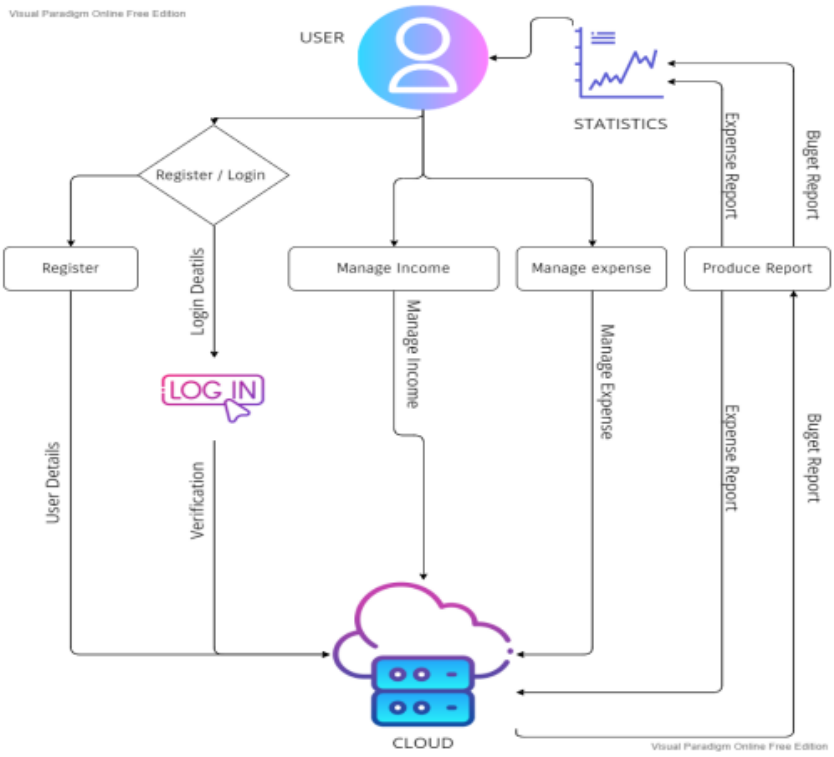
Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Effectiveness, efficiency, and overall experience of the user interacting with our application should be maximised.
NFR-2	**Security**	Authentication, authorization, encryption of the data must be done in the application.
NFR-3	**Reliability**	Probability of error in the operations in a specified environment for a specified time should be minimised.
NFR-4	**Performance**	How the application is functioning accurately and effectively the application is to the end-users.
NFR-5	**Availability**	Using Cloud Storage and database, application reliability and the user satisfaction will affect the solution
NFR-6	**Scalability**	Capacity of the application to handle growth, especially in handling more users.

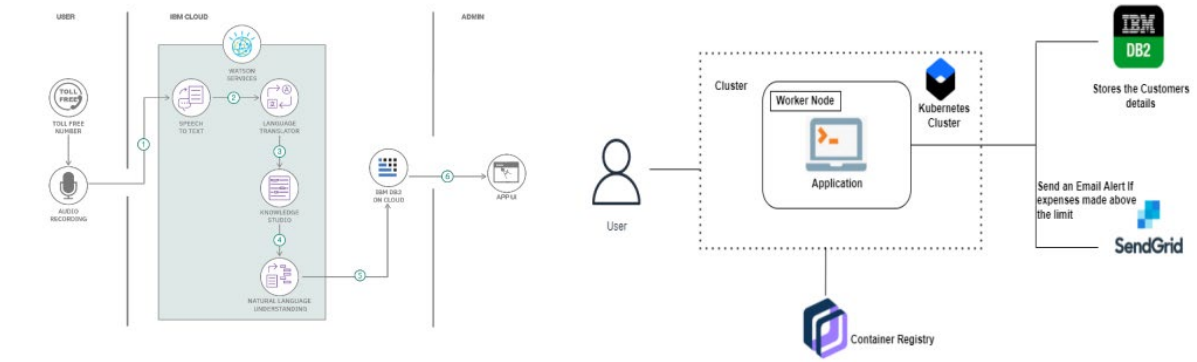
5. PROJECT DESIGN

a. Data Flow Diagrams



6. Solution & Technical Architecture

Technical Architecture:



7. User Stories

User Stories

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

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-2	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard		As a user, I can access my detail, manage the expense, add budget, expense report from the app etc..		High	Sprint-1
Customer (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	Login	USN-2	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-3	As a user, I can access my detail, manage the expense, add budget, expense report from the app etc..		High	Sprint-1
Customer Care Executive	Email or Customer Care no		As a user, I can contact the service administration for the support.	I can solve the Issue.	High	Sprint-3
Administrator	Email or Customer Care no		As a user, I can contact the service administration for the support.	I can solve the Issue.	High	Sprint-1

8. PROJECT PLANNING & SCHEDULING

a. Sprint Planning & Estimation

Project Planning Phase

Project Planning Template (Product Backlog, Sprint Planning, Stories, Story points)

Date	20 October 2022
Team ID	PNT2022TMID26057
Project Name	Project – Personal Expense Tracker Application
Maximum Marks	8 Marks

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

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Chandra Mouli . M
Sprint-2		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Vikraman A
Sprint-3		USN-3	As a user, I can register for the application through Facebook	2	Low	Yeshwanth B
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium	Prithivi Krishna PK
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	1	High	Vikraman A
	Dashboard	USN-6	As a user, I access my detail, manage my expense, add budget, expense report from the app etc..	2	High	Prithivi Krishna PK

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	18	06 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	15	14 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	19	21 Nov 2022

b. Sprint Delivery Schedule

Project Planning Phase-II
Sprint Delivery Plan

Date	20 October 2022
Team ID	PNT2022TMID26057
Project Name	Personal Expense Tracker Application
Maximum Marks	4 Marks

Sprints	Task	Commencing Date	Finishing Date
Sprint-1	UI Task	24 Oct 2022	29 Oct 2022
Sprint-2	Connecting IBM-DB2	31 Oct 2022	06 Nov 2022
Sprint-3	Integrating with Send Grid.	07 Nov 2022	14 Nov 2022
Sprint-4	Submission of project	14 Nov 2022	19 Nov 2022

+

9. **CODING & SOLUTIONING (Explain the features added in the project along with code)**

a. Feature 1

We have added the data visualiza on methods for expenditure. The pie chart have been used to represent the monthly expenses. The pie chart is a pictorialrepresentation of data that makes it possible to visualize the relationshipsbetween the parts and the whole of a variable. For example, it is possible tounderstand the industry count or percentage of a variable level from the division by areas or sectors. The recommended use for pie charts is twodimensional, as three-dimensional use can be confusing

code

```python
import re
import ibm_db
from flask import Flask, render_template, request, session

global table
global userid


def insertTableData(conn, username, email, password, age, profession, table):
    table = 'no'
    sql = "INSERT INTO usersdetails(username,email,password,age,profession,table) VALUES ('{}','{}','{}','{}','{}','{}')".format(
        username, email,
        password, age, profession, table)
    out = ibm_db.exec_immediate(conn, sql)
    print('Number of affected rows : ', ibm_db.num_rows(out), "\n")


def conditionCheak():
    username = session.get('username', None)
    sql = "SELECT table FROM usersdetails WHERE username=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.execute(stmt)
    out = ibm_db.fetch_assoc(stmt)
    print("out cheak condition ->", out)
    value = out['TABLE']
    session['tablequery'] = value


```

```python

def createNewTableForUser(userid):
    sqlc = "CREATE TABLE userid=? (date DATE , expensename VARCHAR(22),expenseamount INTEGER,paymode VARCHAR(24),category VARCHAR(22))"
    stmt = ibm_db.prepare(conn, sqlc)
    ibm_db.bind_param(stmt,0, userid)
    createtable = ibm_db.execute(stmt)
    session['createtable']=createtable


def updateTabletable(table):
    username = session.get('username', None)
    sql1 = "UPDATE usersdetails SET TABLE=? WHERE USERNAME=?"
    stmt = ibm_db.prepare(conn, sql1)
    ibm_db.bind_param(stmt, 1, table)
    ibm_db.bind_param(stmt, 2, username)
    ibm_db.execute(stmt)


def updateTableData(username, password, email, profession):
    sql = "SELECT * FROM usersdetails  WHERE username =? AND password=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.bind_param(stmt, 2, password)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    print(account)
    if account:
        username = account['USERNAME']
        sql1 = "UPDATE usersdetails SET EMAIL=?,PROFESSION=? WHERE USERNAME=?"
        stmt = ibm_db.prepare(conn, sql1)
```

```
        stmt = ibm_db.prepare(conn, sql1)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, profession)
        ibm_db.bind_param(stmt, 3, username)
        ibm_db.execute(stmt)


def displayDetails(userid):
    sql1 = "SELECT username, email, age, profession FROM usersdetails WHERE username=?"
    stmt_db = ibm_db.prepare(conn, sql1)
    ibm_db.bind_param(stmt_db, 1, userid)
    ibm_db.execute(stmt_db)
    accounts = ibm_db.fetch_assoc(stmt_db)
    return accounts


try:
    conn = ibm_db.connect(
        "DATABASE=bludb;HOSTNAME=98538591-7217-4024-b027-8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=30875;SECURITY=SSL;SSLServerCertificate=DigiCertGloba
        "", "")
    print("Db connected")
except:
    print("Error")

app = Flask(__name__)
app.secret_key = 'aa'


```

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

    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        sql = "SELECT * FROM usersdetails  WHERE username =? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)

        table = account['TABLE']
        userid = account['USERNAME']

        session['username'] = userid

        sql1 = "SELECT username, email, age, profession FROM usersdetails WHERE username=?"
        stmt_db = ibm_db.prepare(conn, sql1)
        ibm_db.bind_param(stmt_db, 1, userid)
        ibm_db.execute(stmt_db)
        accounts = ibm_db.fetch_assoc(stmt_db)

        if account:
            session['id'] = account['USERNAME']
```

```
            session['username'] = account['USERNAME']
            msg = 'Logged in successfully !'
            return render_template('dashboard.html', accounts=accounts)
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg=msg)


@app.route("/register", methods=['POST', 'GET'])
def register():
    if request.method == "POST":
        email = request.form['email']
        password = request.form['password']
        age = request.form['age']
        profession = request.form['profession']
        username = request.form['username']
        sql = "SELECT * FROM usersdetails WHERE username =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            insertTableData(conn, username, email, password, age, profession, table)
```

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


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

    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        date = request.form['date']
        expensename = request.form['expensename']
        expenseamount = request.form['expenseamount']
        paymode = request.form['paymode']
        category = request.form['category']

        sql = "SELECT * FROM usersdetails  WHERE username =? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            userids = account['USERNAME']
            sqli = "INSERT INTO expenses(username,date,expensename,expenseamount,paymode,category) VALUES ('{}','{}','{}','{}','{}','{}')".format(
                username,
                date, expensename, expenseamount, paymode, category)
            out = ibm_db.exec_immediate(conn, sqli)
```

```
            accounts = displayDetails(userids)
            return render_template('dashboard.html', accounts=accounts)
    return render_template('add.html')

    if request.method == 'GET':
        return render_template('dashboard.html', accounts=accounts)


@app.route("/changedetails", methods=['POST', 'GET'])
def changedetails():
    if request.method == "POST":
        username = request.form['username']
        password = request.form['password']
        email = request.form['email']
        profession = request.form['profession']
        updateTableData(username, password, email, profession)
        return render_template('login.html')

    return render_template('changedetails.html')


@app.route("/dashboard",methods=['POST','GET'])
def dashboard():
    username = session.get('username',None)
    accounts = displayDetails(username)
    return render_template('dashboard.html',accounts=accounts)


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

```
     if request.method == 'GET':
        user = session.get('username', None)
        print(user)
        expensedetails = []
        sql = "SELECT CHAR(DATE(date),USA) as date, expensename, expenseamount, paymode, category FROM expenses WHERE username=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, user)
        ibm_db.execute(stmt)
        details = ibm_db.fetch_assoc(stmt)
        while details != False:
            expensedetails.append(details)
            details = ibm_db.fetch_assoc(stmt)

        print(expensedetails)

        sql2 = "SELECT SUM(expenseamount) AS TOTALVAL FROM expenses WHERE username = ?"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2, 1, user)
        ibm_db.execute(stmt2)
        totalexpense = ibm_db.fetch_assoc(stmt2)
        print(totalexpense)
        return render_template('dispexpense.html', expensedetails=expensedetails, totalexpense=totalexpense['TOTALVAL'])


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

```
        totalexpense = ibm_db.fetch_assoc(stmt2)
        print(totalexpense)
        return render_template('dispexpense.html', expensedetails=expensedetails, totalexpense=totalexpense['TOTALVAL'])


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


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

b. Feature 2

Email notifications will be sent to the users once they cross the expenditure limit through send grid mail system. Most notifications are transactional, meaning a recipient's action or account activity triggers them. But some notifications are marketing related, encouraging the recipient to take a specific action. Ecommerce product notifications inform recipients about new products or discounts. Plus, unlike general marketing emails, these are highly personalized and focus on a single product. For example, if a customer views an item on your website and that item goes on sale, you can send the customer a notification to let them know this is the best time to buy. Users can also opt into receiving notifications when an out-of-stock item is back in stock. Notification emails tend to perform well because the content is highly relevant to the recipient. But the only way for the recipient to know this is if you state the content clearly in the subject line

10. **TESTING**

a. Test Cases

				Date	19 November 2022							
				Team ID	PNT2022TMID26057							
				Project Name	Personal Expense Tracker Application							
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comment	TC for Atom	BUG ID	Executed By
LoginPage_TC_OO1	Functional	Login Page	Verify user is able to see t e Login/Signup popup when user clicked on My account button	None	1. Go to website 2.Login page appears	Username: Prithivi password: qwerty	Login/Signup popup should display	Workingas expected	Pass	-			Prithivi Krishna
LoginPage_TC_OO2	UI	Login Page	Verify the UI elements in Login/Signup	Home	1. Go to website 2. Enter details and click login	Username: Prithivi password: qwerty	Application should show below UIelements: a. username text box b. password text box c. Login button d. New customer? register	Workingas expected	Pass	-			Vikranam
LoginPage_TC_OO3	Functional	Login page	Verify user is able to log into application with Valid credentials	Username & password	1. Go to website 2. Enter details and click login	Username: Prithivi password: qwerty	User should navigate to user accounthomepage	Workingas expected	Pass	-			Yeshwanth
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with Invalid credentials.	Username & password	1. Go to website 2. Enter details and click login	Username: Prithivi password: 123456	Application should show 'Incorrect username or password ' validation message.	Workingas expected	Pass	-			Chandra Mouli
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with Invalid credentials.	Login first	1. Go to website 2. Enter details and click login	Username: Prithivi password: qwerty	Application should show 'Incorrect username or password ' validation message.	Workingas expected	Pass	-			Prithivi & Yeshwanth
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with Invalid credentials.	Login first	1. Go to website 2. Enter details and click login	Username: Prithivi password: qwerty	Application should show 'Incorrect username or password ' validation message.	Workingas expected	Pass	-			Chandra Mouli
Add Expense Page	Functional	Add Expense page	Verify whether user is able to add expense or not	Have some expense to add	1. Add date, expense name and other details . 2. Check if the expense gets added	add expense = 550	Application adds expenses	Workingas expected	Pass	-			Vikraman

b. User Acceptance Testing

Acceptance Testing
UAT Execution & Report Submission

Date	19 November 2022
Team ID	PNT2022TMID26057
Project Name	Personal Expense Tracker Application

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Personal Expense Tracker Application project at the time of the release to User AcceptanceTesting (UAT).

2. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	6	0	2	4	12
Duplicate	1	0	0	0	1
External	2	0	0	1	3
Fixed	11	2	2	10	25
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	5	2	1	8
Totals	20	7	6	16	49

3.Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	15	0	0	15
Security	2	0	0	2
Outsource Shipping	0	0	0	6
Exception Reporting	3	0	0	3
Final Report Output	5	0	0	5
Version Control	1	0	0	1

11. **RESULTS**

a. Performance Metrics

Tracking income and expenses: Monitoring the income and tracking all expenditures (through bank accounts, mobile wallets, and credit & debit cards).

12. **ADVANTAGES & DISADVANTAGES**

*Advantages

It is easy to set up and use. It keeps track of everything for you in real time using an automatic app. It has a wealth of information, so any data you believe is crucial to track is there and at your disposal; you only need to look to see it. Everything has a simple user interface as well. There is a tab or an option available for you whether you want to create a budget, keep track of a specific sort of spending, or review your financial history .It happens instantly. Your data will be

tracked for you by the application. In contrast to what you might manually perform, it doesn't do it once a week or once a month.

*Disadvantage

Your data is likely being exploited and sold, and it is less secure. If the product is a free service, then you are the product. Like other financial apps, Mint.com offers its services for nothing. No matter what their privacy policy may or may not say, just expect that someone, somewhere is going to record and analyse your spending patterns because they need to pay their expenses. Now, you shouldn't have to worry about identity theft or credit card fraud since these firms are big enough and safe enough to prevent those things from happening.

13. CONCLUSION

You can save money by keeping track of your daily costs, but it can also help you set future financial goals. If possible, look for areas where you may cut costs and negotiate better terms if you know exactly where your money is going. The Expense Tracker project will help us keep track of our everyday spending and make a record of it. Compared to other income and expense trackers, the project we designed is more effective. The project successfully avoids manual calculation by attempting to avoid determining the salary and expense each month. It's an intuitive application.

14. FUTURE SCOPE

1) It will have a variety of record-keeping choices (such as food, travel expenses, salary, etc.).
2) It will continue to give updates about our daily spending automatically.
3. Despite being in a haste to make money in today's hectic and expensive world, we eventually gave up. As we naively waste money on unnecessary items and titles. We so came over with the intention of following our profit.
4) The user can specify their own expense categories here, such as those for food, clothing, rent, and bills, where they must input the money that has been spent and may also add additional information to denote the expense.

15. APPENDIX

•Source Code & GitHub

https://github.com/IBM-EPBL/IBM-Project-1647-1658407723

IBM-Project-1647-1658407723

- Project Demo Link & Live Link

159.122.177.70:31327

- Youtube Link : https://youtu.be/NPK3pOJ7sHU