A PROJECT REPORT ON

Personal Expense Tracker Application

Domain : Cloud App Development
TEAM ID : PNT2022TMID00377
COLLEGE NAME: St.JOSEPH"S COLLEGE OF ENGINEERING

B. Sushmitha
(312319106161)
Department of Electronics and Communication Engineering

J.Srimathi
(312319106154)
Department of Electronics and Communication Engineering

V.V Srinivedhini
(312319106155)
Department of Electronics and Communication Engineering

S.Sumitha
(312319106158)
Department of Electronics and Communication Engineering

TABLE OF CONTENTS

CHAPTER 1

1.INTRODUCTION

A Personal Expense Tracker Application is a particular form of digital diary that aids in keeping track of all of our cash transitions and moreover offers daily, weekly, monthly, and yearly reports on all financial activities.
Also known as expense manager and money manager, an expense tracker is a software or application that helps to keep an accurate record of your money inflow and outflow. Many people in India live on a fixed income, and they find that towards the end of the month they don't have sufficient money to meet their needs.

.

1.1. PROJECT OVERVIEW

An expense tracking app is an exclusive suite of services for people who seek to handle their earnings and plan their expenses and savings efficiently. It helps you track all transactions like bills, refunds, payrolls, receipts, taxes, etc., on a daily, weekly, and monthly basistracking app, you can automate the calculations for frequent travel expenses of your employees. For example, cloud-based expense tracking apps help in generating bills and reports for your on-field employees in real-time. Your employees can track their expenses on the go and send their bills to the accounting department for immediate reimbursement and settlement.

1.2. PURPOSE

Tracking your expenses involves identifying your expenditures throughout the month. It's an essential activity that you should ideally do every day throughout the month.It may seem like a lot of work to itemize your expenses when you first begin, but understanding why it's important to track expenses and how to do so with minimal effort can help you successfully commit to the activity and become more aware of your spending.

CHAPTER 2

2. LITERATURE SURVEY

A literature review is a piece of academic writing that places the academic literature on a particular topic in perspective to show knowledge of and understanding of it. This chapter shows the different techniques that have been implemented.

2.1. EXISTING PROBLEM

There can be many disadvantages of using a manual accounting system. Accounting, for any business, can be a complex undertaking. A manual accounting system requires you to understand the accounting process in a way that may be unnecessary with a computerized accounting system. This can be an advantage or a disadvantage, depending on the person doing the bookkeeping; often, a specially trained professional is needed to ensure that accounting is done
properly. Unraveling the complexity of your financial records by hand may be time consuming. Since it takes time to generate reports. Due to imperfect data maintenance, the current system is not user friendly. The sole negative where the rest are absent from this endeavor is that there will be no reminder to stay a human on a specified date. This project won't have any information because it doesn't remind people to do anything each month, which has some drawbacks. However, it can be used to calculate income and expenses, so we suggest a new project to solve this issue.

2.2 REFERENCE

[1] Expense Tracker ATIYA KAZI, PRAPHULLA S. KHERADE, RAJ S. VILANKAR, PARAG M. SAWANT May 2021
[2] Intelligent Online Budget Tracker Girish Bekaroo and Sameer Sunhaloo Proceedings of the 2007 Computer Science and 2007
[3] Online Income and Expense Tracker S. Chandini, T. Poojitha, D. Ranjith, V.J. Mohammed Akram, M.S. Vani, V. Rajyalakshmi Mar 2019
[4] Family Expense Manager Application Rajaprabha M N 2017
[5] A Novel Expense Tracker using Statistical Analysis Muskaan Sharma, Ayush Bansal, Dr. Raju Ranjan, Shivam Sethi June 2021
[6] Expense Tracker Hrithik Gupta, Anant Prakash Singh, Navneet Kumar and J. Angelin Blessy December 2020
[7] Expense Manager: An Expense Tracking Application using Image Processing Nupur Sawarkar, Pranay Yenagandula, Devang Shetye, Prof. Shruti Agrawal April 2022

[8] D2D Expense Tracker Application Anjali Kumar, Utkarsh Ra, Aman Kumar 2021[9] Daily Expense Tracker Mobile Application Nuura Najati Binti Mustafa 2021
[10] Daily Expense Tracker Shivam Mehra, Prabhat Parashar 2021

2.3 PROBLEM STATEMENT DEFINITION

In our daily life money is the most important portion and without it we cannot last one day on earth but if we keep on track all financial data then we can overcome this problem. Most of the people cannot track their expenses and income one way they face the money crisis and depression. Tracking the amount of money spent on the projects is important to invoice customers and determine the cost & profitability analysis when your company is providing services to another company. On the other hand, expense tracking or internal project is important for cost and ROI calculation. Understanding how this money is being utilized across the project is such a significant issue. The consequence for not properly tracking and reporting project expenses may lead to a budgetary issues.

CHAPTER 3

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

3.2. IDEATION & BRAINSTORMING

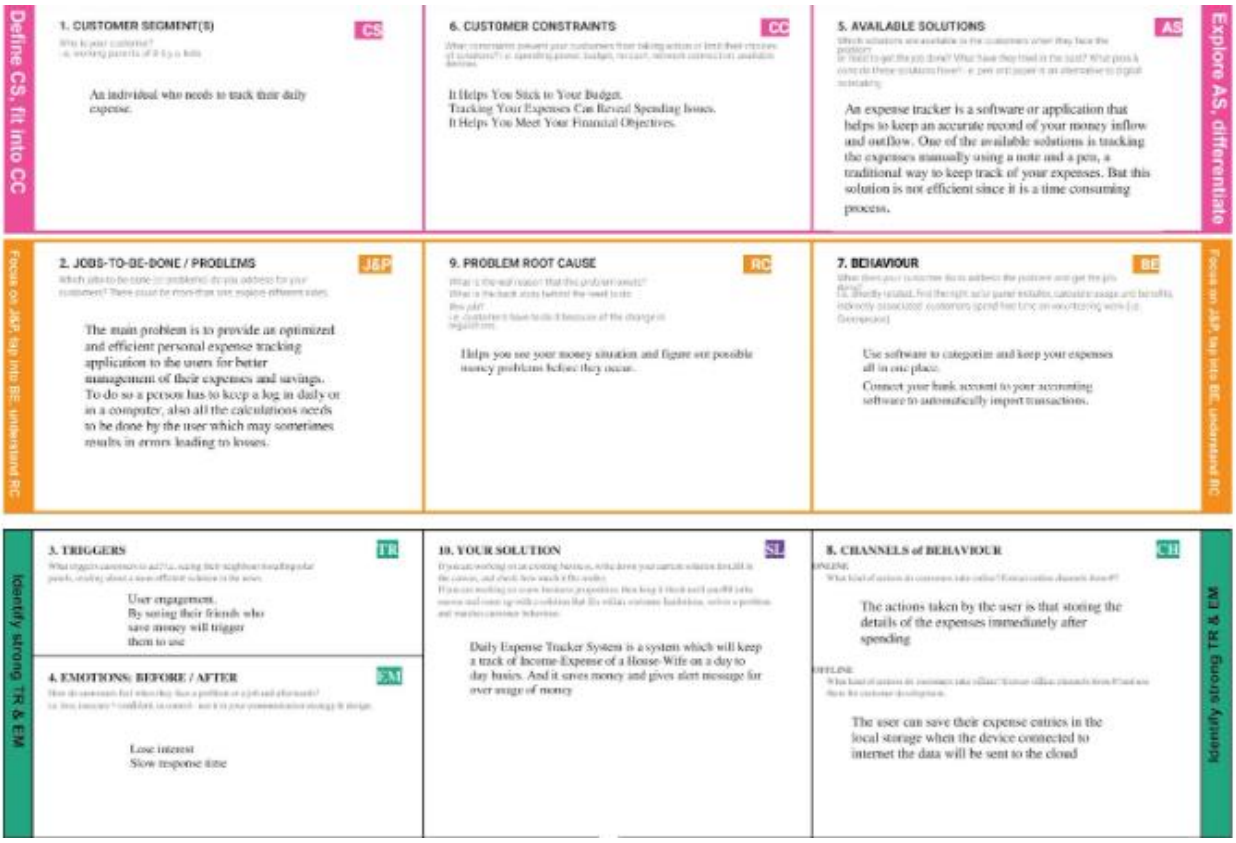
3.3. PROPOSED SOLUTION

S.NO.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Building a personal finance tracking application that will imbibe good spending habits into students..
2.	Idea / Solution description	To build a web application that is deployed in IBM cloud and leverage mailing service like send grid to implement the same

3.	Novelty / Uniqueness	The stats generated with visual graphs are more effective than log books. It also helps in using technology to gain better insights from patterns.
4.	Social Impact / Customer Satisfaction	Better financial knowledge is gained. Gamified approach can be used to give self satisfaction. Reduced chances of bad debt in future.

5.	Scalability of the Solution	As the application is containerized for deployment. It can be easily scaled in a cloud service provider like IBM.
6.	Business Model (Revenue Model)	Subscription can be incorporated to access premium tools within the app.

3.4. PROBLEM SOLUTION FIT



CHAPTER 4

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

Following are the functional requirements of the Proposed solution

FR NO.	Functional Requirement (Epic)	Sub Requirement (story/sub task)
FR-1	User Registration	Registration through email /registration through Gmail.
FR-2	User Confirmation	Confirmation via email confirmation via OTP
FR-3	Add Expenses	Enter the everyday expenses Split it into categories(example : food, petrol,movies)
FR-4	Remainder mail	Sending reminder mail on target (for ex : if user wants a reminder when his/her balance reaches some amount(5000)). Sending reminder mail to the user if he/she has not filled that day's expenses.

FR-5	Creating graph	Graphs showing everyday and weekly expenses.Categorical graphs on expenditure.
FR-6	Add salary	Users must enter the salary at the start of the month.
FR-7	Export CSV	User can export the raw data of their expenditure as CSV

4.2 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 satisfaction of the user
NFR-2	Security	Authentication,authorisation and encryption of the application
NFR-3	Reliability	Probability of failure free operations in a specified environment for a specified time

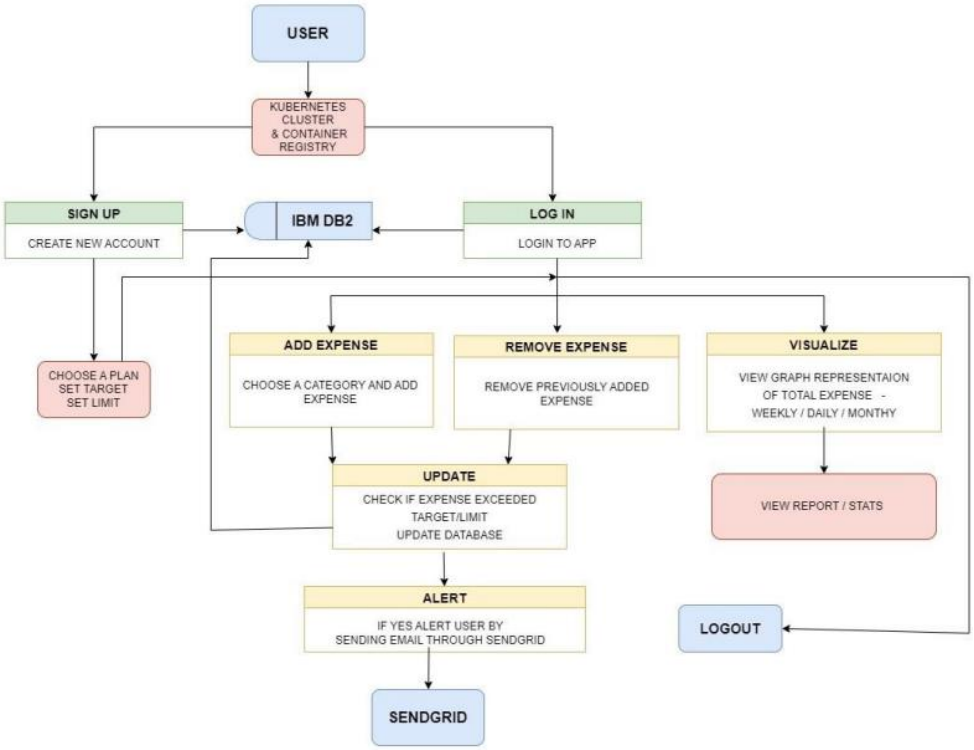
NFR-4	Performance	How the application is functioning and how responsive the application is to the end users
NFR-5	Availability	Without near 100% availability,application reliability and the user satisfaction will affect the solution
NFR-6	Scalability	Capacity of the application to handle growth,especially in handling more users

CHAPTER 5

5. PRODUCT DESIGN

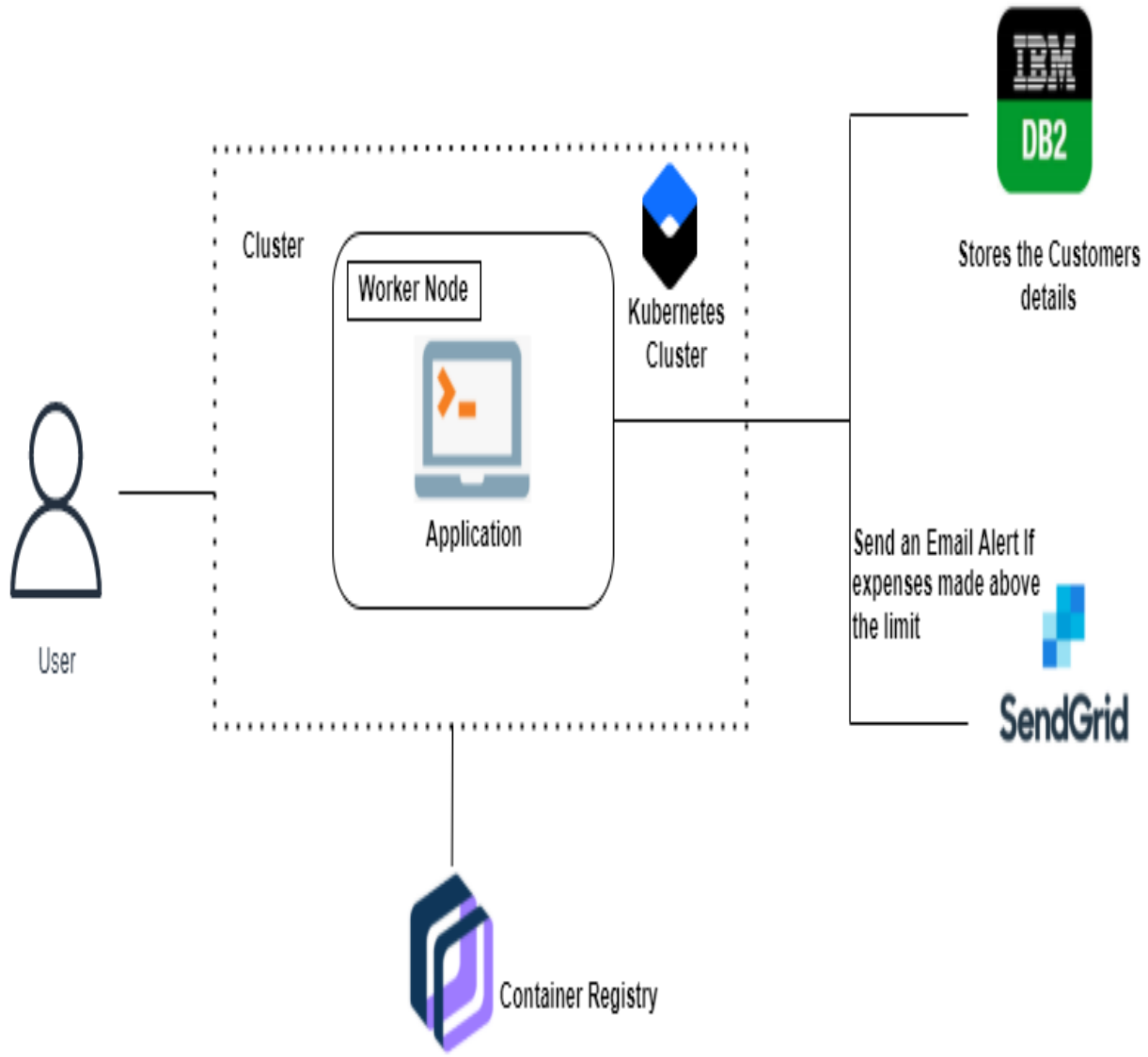
5.1 Data Flow Diagrams

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



5.2 Technical Architecture

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2



5.3 User Stories

Use the below template to list all the user stories of the product

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobileuser & 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	
		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	
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	
	Login	USN-4	As a user, I can log into the application by entering email & password	I can access the application	High	
	Dashboard	USN-5	As a user I can enter my income and expenditure details.	I can view my daily expenses	High	
Customer Care Executive		USN-6	As a customer care executive I can solve the log in issues and other issues of the application.	I can provide support or solution at any time 24*7	Medium	

Administrator	Application	USN-7	As an administrator I can upgrade or update the application.	I can fix the bug which arises for the customers and users of the application	Medium	

CHAPTER 6

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Creating basic login page	USN-1	As a user, I can create my account and get my login credentials	2	High	Srimathi J Srinivedhini v Sushmitha b Sumitha s
Sprint-1	IBM Cloud	USN-2	As a user, I will receive acknowledge for my account creation	1	High	Srimathi J Srinivedhini v Sushmitha b Sumitha s
Sprint-2	Downloading of required software	USN-3	As a user, I can get additional features along with the tracking of my monthly expenses	2	low	Srinivedhini v

Sprint-2	Creation of send grid account	USN-4	As a user, I can get details about expenses And income at any instant without any server issues.	2	medium	Sumitha S Sushmitha B
Sprint-3	Intergration of IBM Cloud and send grid	USN-5	As a user I can get my details ,transaction history lastest update about my expenses.	1	High	Srinivedhini v Sushmitha B Sumitha S Srimathi J
Sprint-3	Dashboard	USN-6	I can get all my to-do list and features at a instant in dashboard.	2	medium	Srinivedhini v Srimathi J
Sprint-4	Offline receipt	USN-7	I can get my transaction receipt in offline mode also.	2	high	Srinivedhini v Sushmitha B Sumitha S Srimathi J
Sprint-4	Over all	USN-8	Expense,trans-action, income, invest and many features can be benefited from this app.	2	High	Srinivedhini v Sushmitha B Sumitha S Srimathi J

6.2 Sprint Delivery Schedule

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	30 Sept 2022	2 Oct 2022	20	20 Oct 2022
Sprint-2	20	3 Days	3 Oct 2022	7 Oct 2022	20	20 Oct 2022
Sprint-3	20	5 Days	10 Oct2022	15 Oct 2022		20 Oct 2022
Sprint-4	20	4 Days	18 Oct 2022	22 Oct 2022		20 Oct 2022

CHAPTER 7

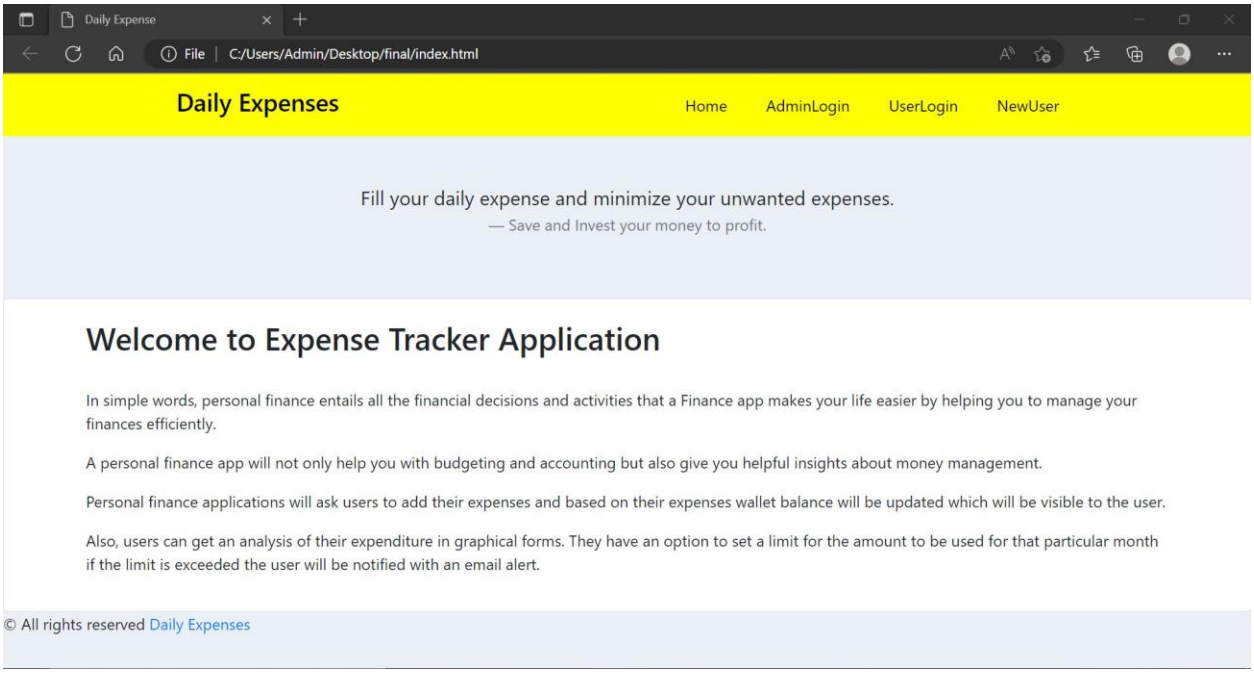
7.1 Feature 1

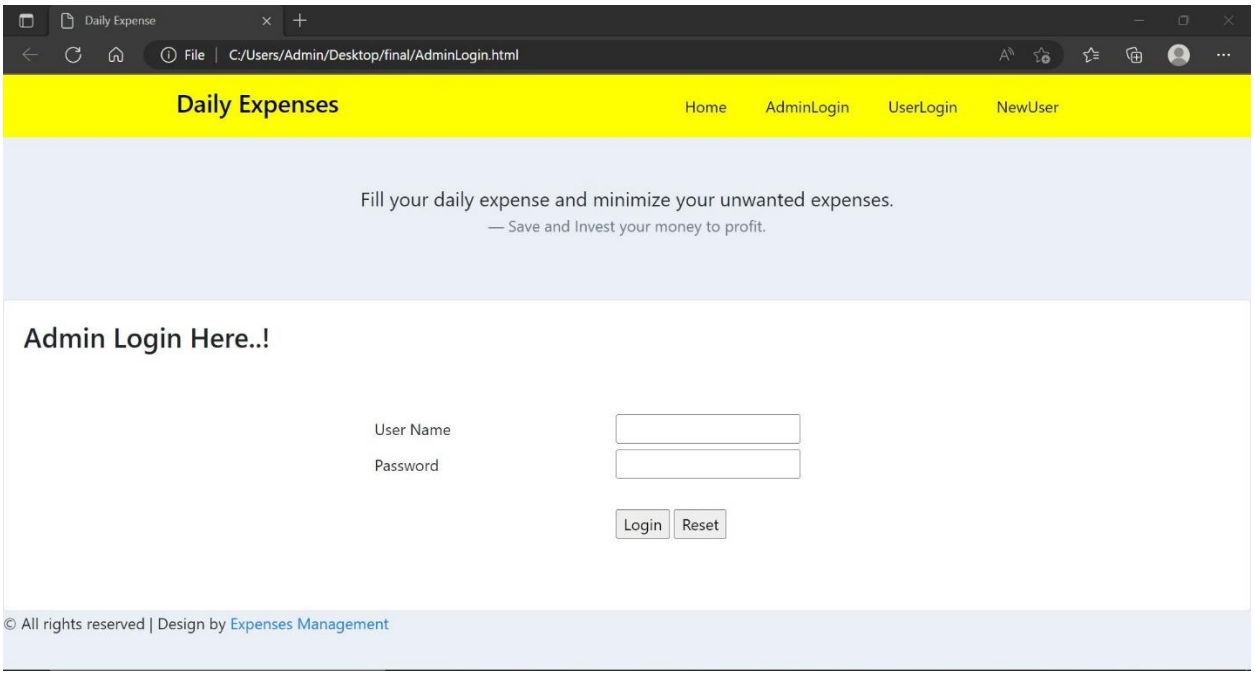
1. Track revenues & expenses.
2. Managing transaction receipts and records.
3. Record & arrange receipts
4. Paying taxes in time.
5. Processing payment and invoices.
6. Create in-depth reports

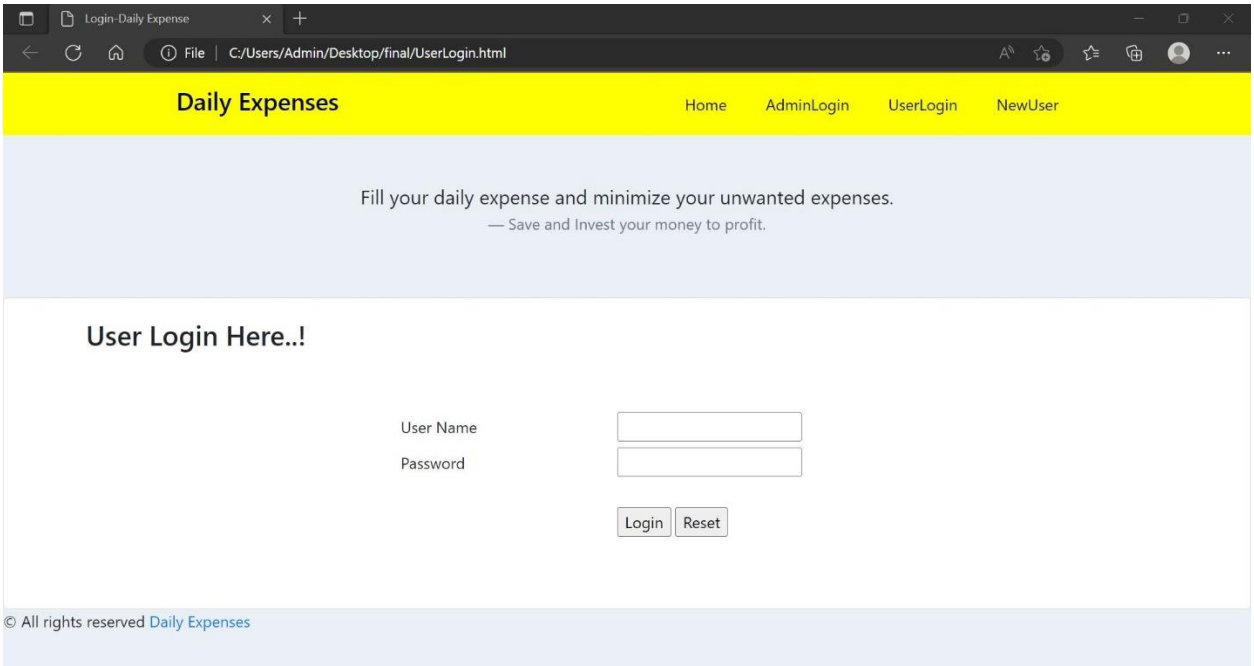
CHAPTER 8

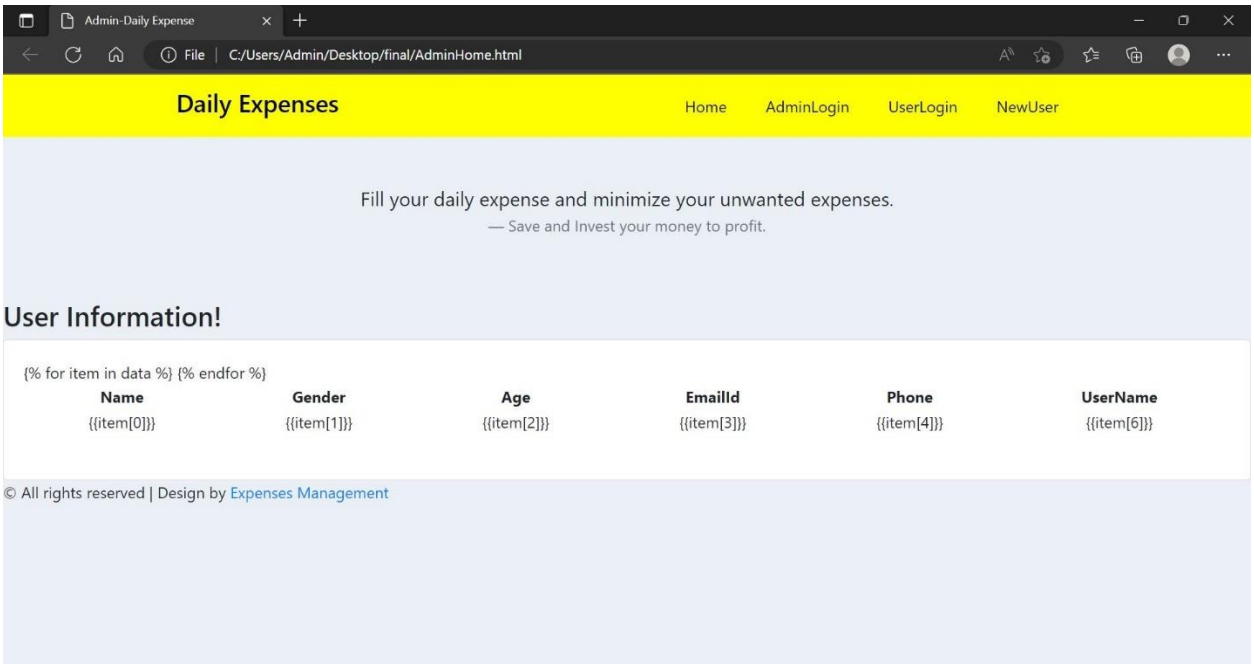
TESTING

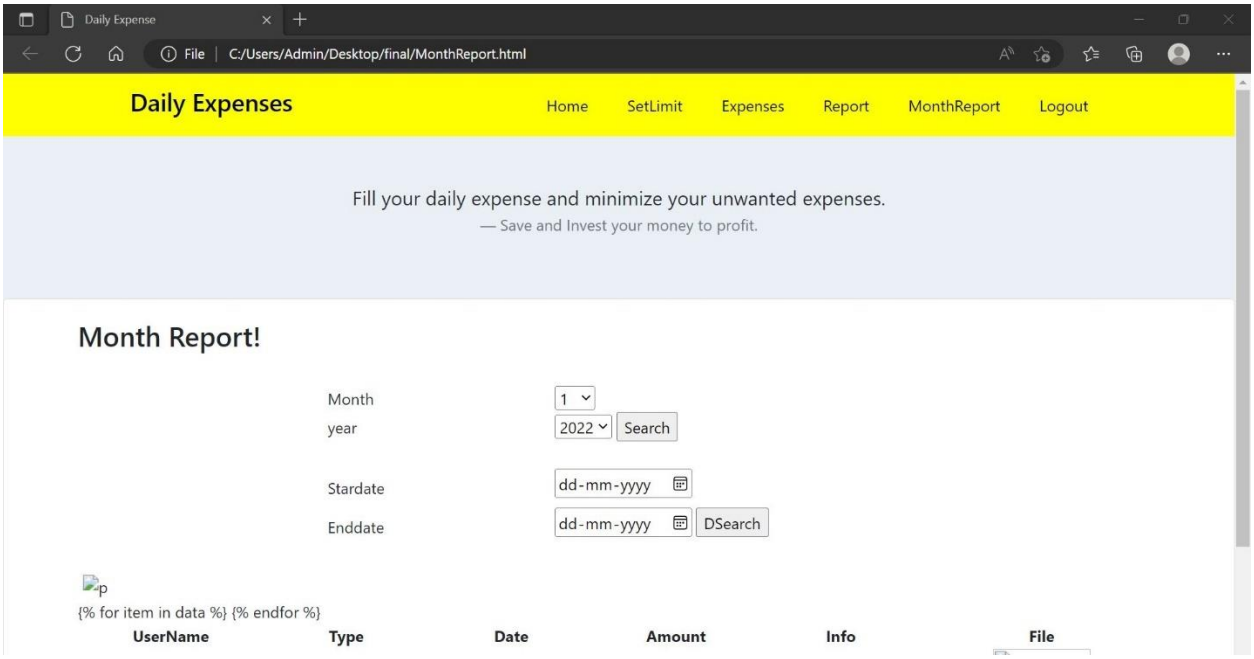
8.1 Test Cases

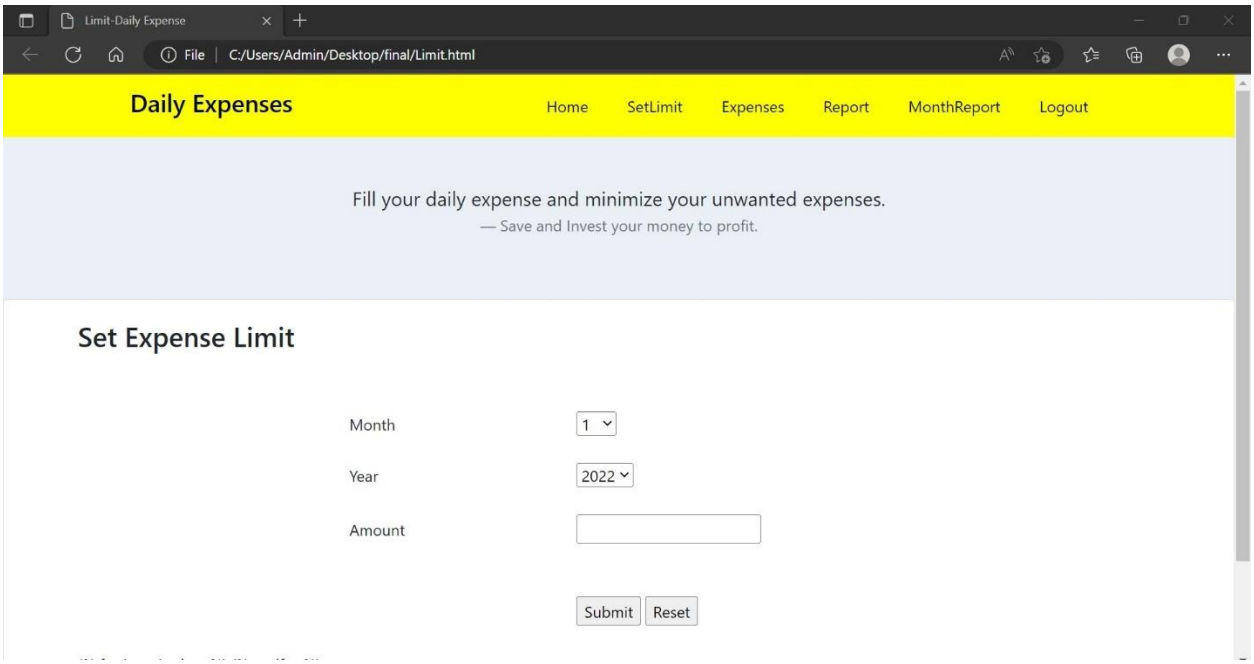


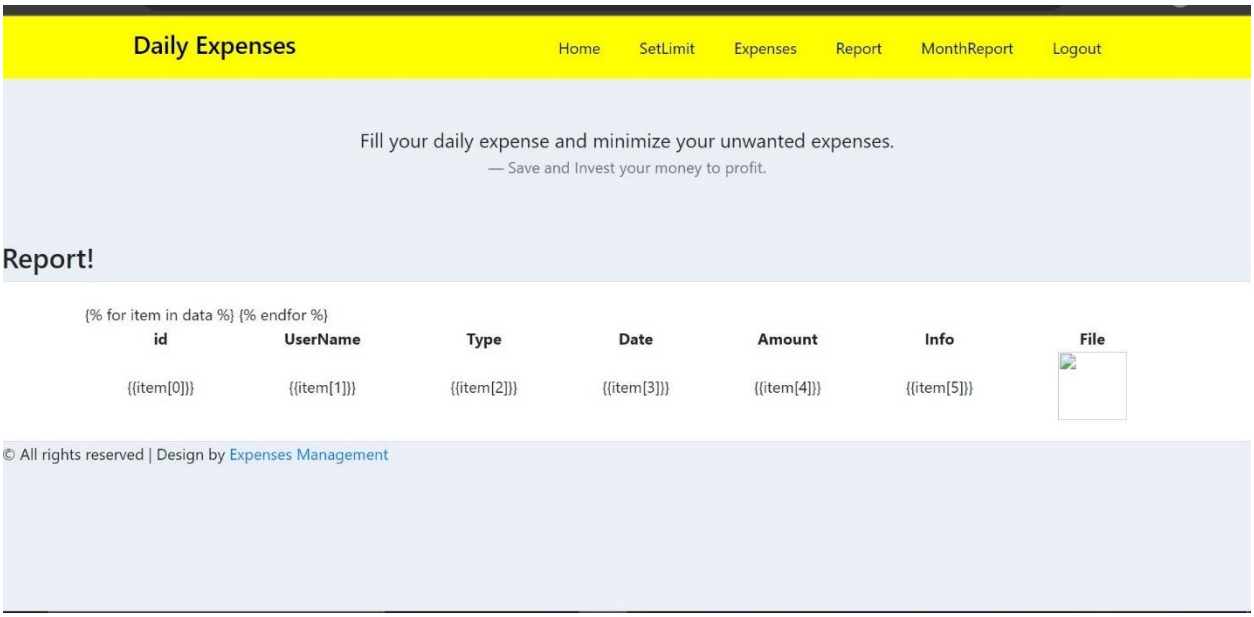

Daily Expense
File | C:/Users/Admin/Desktop/final/AdminLogin.html
Daily Expenses
Home
AdminLogin
UserLogin
NewUser
Fill your daily expense and minimize your unwanted expenses.
— Save and Invest your money to profit.
Admin Login Here..!
User Name
Password
Login
Reset
© All rights reserved | Design by Expenses Management

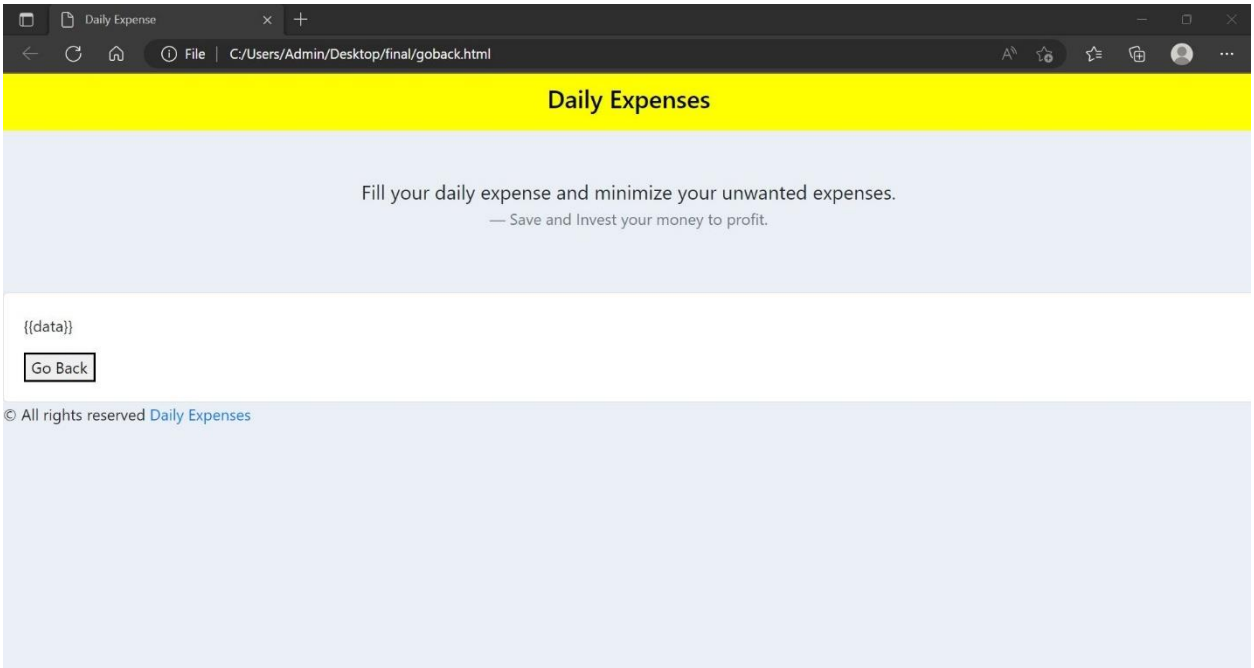

Login-Daily Expense
File | C:/Users/Admin/Desktop/final/UserLogin.html
Daily Expenses
Home
AdminLogin
UserLogin
NewUser
Fill your daily expense and minimize your unwanted expenses.
— Save and Invest your money to profit.
User Login Here..!
User Name
Password
Login
Reset
© All rights reserved Daily Expenses


Admin-Daily Expense
File | C:/Users/Admin/Desktop/final/AdminHome.html
Daily Expenses
Home
AdminLogin
UserLogin
NewUser
Fill your daily expense and minimize your unwanted expenses.
— Save and Invest your money to profit.
User Information!
{% for item in data %} {% endfor %}
Name
Gender
Age
EmailId
Phone
UserName
{{item[0]}}
{{item[1]}}
{{item[2]}}
{{item[3]}}
{{item[4]}}
{{item[6]}}
© All rights reserved | Design by Expenses Management


Daily Expense
File | C:/Users/Admin/Desktop/final/MonthReport.html
Daily Expenses
Home
SetLimit
Expenses
Report
MonthReport
Logout
Fill your daily expense and minimize your unwanted expenses.
— Save and Invest your money to profit.
Month Report!
Month
1
year
2022
Search
Stardate
dd-mm-yyyy
Enddate
dd-mm-yyyy
DSearch
{% for item in data %} {% endfor %}
UserName
Type
Date
Amount
Info
File


Limit-Daily Expense
File | C:/Users/Admin/Desktop/final/Limit.html
Daily Expenses
Home
SetLimit
Expenses
Report
MonthReport
Logout
Fill your daily expense and minimize your unwanted expenses.
— Save and Invest your money to profit.
Set Expense Limit
Month
1
Year
2022
Amount
Submit
Reset


Daily Expenses
Home
SetLimit
Expenses
Report
MonthReport
Logout
Fill your daily expense and minimize your unwanted expenses.
— Save and Invest your money to profit.
Report!
{% for item in data %} {% endfor %}
id
UserName
Type
Date
Amount
Info
File
{{item[0]}}
{{item[1]}}
{{item[2]}}
{{item[3]}}
{{item[4]}}
{{item[5]}}
© All rights reserved | Design by Expenses Management



CHAPTER 9

RESULTS

9.1Performance Metrics

An application can be a very powerful tool for businesses if once the app becomes a success.

However, the success of an app is measured through numbers, metrics, and analytics.

Developing an app takes quite a lot, so once you've dedicated much time, money, and effort to the process, it's mandatory to measure mobile app performance.

CHAPTER 10

ADVANTAGES & DISADVANTAGES

10.1 Advantages

- Make It Easier With an App
- Tracking Your Expenses Can Reveal Spending Issues
- Improved customer service
- It helps you to meet you expense tracker
- Cloud-based solution
- Order Fulfillment
- Harness Customer Loyalty and Retention

10.2 Disadvantages

- System Clash
- Reduced Physical Audits
- No solution to improve or eliminate bottlenecks in the service cycle

CHAPTER 11

11. CONCLUSION

Taking proper care of our record is crucial in every business, no matter how big or little, we must understand. The new system has overcome most of the limitations of the existing system and works according to the design specification given. The project what we have developed is work more efficient than the other income and expense tracker. The project successfully avoids the manual calculation for avoiding calculating the income and expense per month. The modules are developed with efficient and also in an attractive manner. The developed systems dispense the problem and meet the needs of by providing reliable and comprehensive information. All the requirements projected by the user have been met by the system. The newly developed system consumes less processing time and all the details are updated and processed immediately. Since the screen provides online help messages and is very use rfriendly, any user will get familiarized with its usage. Module s are designed to be highly flexible so that any failure requirements can be easily added to the modules without facing many problems. The best organizations have a way of tracking and handling these reimbursements. This ideal practice guarantees that the expenses tracked are accurately and in a timely We must educate ourselves about the idea of effective inventory management and its applications because we can see that managers do not fully grasp it. A company's inventory management system is one of the reasons for its failure. Many customs to combat failure are present, and we can start from this point. Modern technologies can support us in managing and keeping an eye on our inventory. We may learn, put new ideas into practice, and assess our company.

CHAPTER 12

12. 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 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.

CHAPTER 13

13. APPENDIX

13.1. SOURCE CODE

```
from flask import Flask, render_template, flash, request, session,send_file
from flask import render_template, redirect, url_for, request

import datetime

import sys



import ibm_db
import pandas
import ibm_db_dbi
from sqlalchemy import create_engine

engine = create_engine('sqlite://',
                       echo = False)

dsn_hostname = "b0aebb68-94fa-46ec-a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud"
dsn_uid = "lmf89137"
dsn_pwd = "veOooJR9cKcT0lB3"

dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "BLUDB"
dsn_port = "31249"
dsn_protocol = "TCPIP"
dsn_security = "SSL"

dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
    "PORT={3};"
    "PROTOCOL={4};"
    "UID={5};"
    "PWD={6};"
```

```
    "SECURITY={7};").format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol,
dsn_uid, dsn_pwd,dsn_security)



try:
    conn = ibm_db.connect(dsn, "", "")
    print ("Connected to database: ", dsn_database, "as user: ", dsn_uid, "on host: ",
dsn_hostname)

except:
    print ("Unable to connect: ", ibm_db.conn_errormsg() )




app = Flask(__name__)
app.config['DEBUG']
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'

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

    return render_template('index.html')

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

    return render_template('AdminLogin.html')




@app.route("/UserLogin")
def UserLogin():
    return render_template('UserLogin.html')

@app.route("/NewUser")
def NewUser():
    return render_template('NewUser.html')



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

```
def Search():
    return render_template('Search.html')

@app.route("/MonthReport")
def MonthReport():
    return render_template('MonthReport.html')


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

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery = "SELECT * from regtb "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data',
                     con=engine,
                     if_exists='append')

    # run a sql query
    data = engine.execute("SELECT * FROM Employee_Data").fetchall()
    return render_template('AdminHome.html',data=data)





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

    user = session['uname']

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery = "SELECT * FROM limtb where  username ='" + user + "' "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data',con=engine,if_exists='append')

    data = engine.execute("SELECT * FROM Employee_Data").fetchall()
```

```
    return render_template('Limit.html', data=data)





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

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery = "SELECT * FROM expensetb  "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data', con=engine, if_exists='append')

    data = engine.execute("SELECT * FROM Employee_Data").fetchall()


    return render_template('Report.html',data=data)




@app.route("/UserHome")
def UserHome():
    user = session['uname']



    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery = "SELECT * FROM regtb where username='" + user + "'"
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data', con=engine, if_exists='append')

    data = engine.execute("SELECT * FROM Employee_Data").fetchall()
    return render_template('UserHome.html',data=data)
```

```
@app.route("/adminlogin", methods=['GET', 'POST'])
def adminlogin():
    error = None
    if request.method == 'POST':
       if request.form['uname'] == 'admin' or request.form['password'] == 'admin':


           conn = ibm_db.connect(dsn, "", "")
           pd_conn = ibm_db_dbi.Connection(conn)

           selectQuery = "SELECT * FROM regtb "
           dataframe = pandas.read_sql(selectQuery, pd_conn)

           dataframe.to_sql('Employee_Data', con=engine, if_exists='append')

           data = engine.execute("SELECT * FROM Employee_Data").fetchall()

           return render_template('AdminHome.html' , data=data)

       else:
        return render_template('index.html', error=error)


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

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

```
    session['uname'] = request.form['uname']

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery = "SELECT * from regtb where UserName='" + username + "' and password='"
+ password + "'"
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    if dataframe.empty:
        data1 = 'Username or Password is wrong'
        return render_template('goback.html', data=data1)
    else:
        print("Login")
        selectQuery = "SELECT * from regtb where UserName='" + username + "' and
password='" + password + "'"
        dataframe = pandas.read_sql(selectQuery, pd_conn)

        dataframe.to_sql('Employee_Data',
                         con=engine,
                         if_exists='append')

        # run a sql query
        data1= engine.execute("SELECT * FROM Employee_Data").fetchall()

        for item in data1:
            session["mail"] = item[4]

        print(session["mail"])





    return render_template('UserHome.html', data=engine.execute("SELECT * FROM
Employee_Data").fetchall())
```

```
@app.route("/UReport")
def UReport():
    name1 = session['uname']

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery = "SELECT * FROM expensetb where username='"+ name1 +"' "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data', con=engine, if_exists='append')

    data = engine.execute("SELECT * FROM Employee_Data").fetchall()



    return render_template('UReport.html',data=data)

@app.route("/dsearch", methods=['GET', 'POST'])
def dsearch():
    if request.method == 'POST':

        import datetime


        file = request.files['fileupload']
        file.save('static/upload/'+file.filename)

        name1 = session['uname']
        type = request.form['c1']
        dat = request.form['t1']
        amt = request.form['t2']
        info = request.form['t3']

        date_object = datetime.datetime.strptime(dat, '%Y-%m-%d').date()

        mon = date_object.strftime("%m")
        yea = date_object.strftime("%Y")

        global lim1
        global lim2

        lim1 = 0
        lim2 = 0
```

```
    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery = "SELECT * from limtb where mon='" + mon + "' and yea='" + yea + "' and
Username='"+ name1 +"'"
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    if dataframe.empty:

        alert = 'Please Set Expense Limit'
        return render_template('goback.html', data=alert)
    else:

        dataframe.to_sql('limtb',con=engine,if_exists='append')

        data1 = engine.execute("SELECT * FROM limtb").fetchall()

        for item in data1:

            lim1 = item[4]
            print(lim1)




    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery =  "SELECT sum(Amount) as amt  from expensetb where mon='" + mon + "'
and yea='" + yea + "' and Username='" + name1 + "'"
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    if dataframe.empty:

        lim2 = float(0.00)
    else:

        dataframe.to_sql('expensetb', con=engine, if_exists='append')

        data1 = engine.execute("SELECT * FROM expensetb").fetchall()
```

```
        for item2 in data1:
            lim2 = item2[1]
            print(lim1)





    if lim2 is None:  # Checking if the variable is None

       lim2 = 0.00
    else:
        print("Not None")



    if (float(lim2) <= float(lim1)):





        conn = ibm_db.connect(dsn, "", "")

        insertQuery =  "INSERT INTO expensetb VALUES ('" + name1 + "','" + type + "','" + dat +
"','" + amt + "','" + info + "','" + file.filename + "','" + date_object.strftime("%m") + "','" +
date_object.strftime("%Y") + "')"
        insert_table = ibm_db.exec_immediate(conn, insertQuery)
        print(insert_table)





        alert = 'New Expense Info Saved'
        return render_template('goback.html', data=alert)
    else:
        alert = 'Limit Above  Expense'

        sendmsg(session["mail"],"Limit Above  Expense")
```

```
        return render_template('goback.html', data=alert)




@app.route("/setlimit", methods=['GET', 'POST'])
def setlimit():
    if request.method == 'POST':

        name1 = session['uname']
        mon = request.form['mon']
        yea = request.form['yea']
        amt = request.form['t2']

        conn = ibm_db.connect(dsn, "", "")
        pd_conn = ibm_db_dbi.Connection(conn)

        selectQuery = "SELECT * from limtb where username='" + name1 + "' and mon='" + mon +
"' and yea='"+ yea +"' "
        dataframe = pandas.read_sql(selectQuery, pd_conn)

        if dataframe.empty:

            insertQuery = "INSERT INTO limtb VALUES ('" + name1 + "','" + mon + "','" + yea + "','" +
amt + "')"
            insert_table = ibm_db.exec_immediate(conn, insertQuery)
            print(insert_table)

            conn = ibm_db.connect(dsn, "", "")
            pd_conn = ibm_db_dbi.Connection(conn)

            selectQuery = "SELECT * FROM limtb where  username ='" + name1 + "' "
            dataframe = pandas.read_sql(selectQuery, pd_conn)

            dataframe.to_sql('Employee_Data', con=engine, if_exists='append')

            data = engine.execute("SELECT * FROM Employee_Data").fetchall()
```

```
        return render_template('Limit.html', data=data)
    else:

        alert = 'Already Set  Expense limit Remove And Set New!'
        return render_template('goback.html', data=alert)



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



    uname =  request.args.get('uname')
    mon = request.args.get('mon')
    year = request.args.get('year')





    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    insertQuery = "delete from limtb  where UserName='"+ uname +"' and mon='"+ mon +"' and
Yea='"+ year +"' "
    insert_table = ibm_db.exec_immediate(conn, insertQuery)

    selectQuery = "SELECT * from limtb "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data',
                     con=engine,
                     if_exists='append')

    # run a sql query
    data = engine.execute("SELECT * FROM Employee_Data").fetchall()




    return render_template('Limit.html', data=data )
```

```
@app.route("/newuser", methods=['GET', 'POST'])
def newuser():
    if request.method == 'POST':

        name1 = request.form['name']
        gender1 = request.form['gender']
        Age = request.form['age']
        email = request.form['email']
        pnumber = request.form['phone']
        address = request.form['address']

        uname = request.form['uname']
        password = request.form['psw']




        conn = ibm_db.connect(dsn, "", "")

        insertQuery = "INSERT INTO regtb VALUES ('" + name1 + "','" + gender1 + "','" + Age +
"','" + email + "','" + pnumber + "','" + address + "','" + uname + "','" + password + "')"
        insert_table = ibm_db.exec_immediate(conn, insertQuery)
        print(insert_table)


        # return 'file register successfully'


    return render_template('UserLogin.html')




@app.route("/msearch", methods=['GET', 'POST'])
def msearch():
    if request.method == 'POST':
        if request.form["submit"] == "Search":


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

```
yea = request.form['yea']
uname = session['uname']

import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('Agg')



conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)



selectQuery = "select Type, sum(Amount) as MSales from expensetb where mon='" +
mon + "' and yea='"+ yea +"' and Username='"+ uname +"' group by Type "
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('expensetb',
                 con=engine,
                 if_exists='append')

# run a sql query
data = engine.execute("SELECT * FROM expensetb").fetchall()


Month = []
MSales = []
Month.clear()
MSales.clear()

for i in data:
    Month.append(i[1])
    MSales.append(i[2])

print("Month = ", Month)
print("Total Sales = ", MSales)

# Visulizing Data using Matplotlib
plt.bar(Month, MSales, color=['yellow', 'red', 'green', 'blue', 'cyan'])
# plt.ylim(0, 5)
plt.xlabel("Type")
plt.ylabel("Total Expenses")
plt.title("Monthly Expenses")
```

```
        import random

        n = random.randint(1111, 9999)

        plt.savefig('static/plott/' + str(n) + '.jpg')

        iimg = 'static/plott/' + str(n) + '.jpg'



        selectQuery = "SELECT * FROM expensetb where mon='" + mon + "' and yea='"+ yea
+"' and Username='"+ uname +"' "
        dataframe = pandas.read_sql(selectQuery, pd_conn)

        dataframe.to_sql('Employee_Data', con=engine, if_exists='append')

        data = engine.execute("SELECT * FROM Employee_Data").fetchall()



        return render_template('MonthReport.html', data=data, dataimg=iimg)

    elif request.form["submit"] == "DSearch":
        d1 = request.form['d1']
        d2 = request.form['d2']
        uname = session['uname']



        import matplotlib.pyplot as plt
        import matplotlib
        matplotlib.use('Agg')



        conn = ibm_db.connect(dsn, "", "")
        pd_conn = ibm_db_dbi.Connection(conn)



        selectQuery ="select Type, sum(Amount) as MSales,date from expensetb where date
between '" + d1 + "' and '" + d2 + "' and Username='" + uname + "' group by Type,date "
        dataframe = pandas.read_sql(selectQuery, pd_conn)
```

```
dataframe.to_sql('expensetb',
                 con=engine,
                 if_exists='append')

data = engine.execute("SELECT * FROM expensetb").fetchall()



Month = []
MSales = []
Month.clear()
MSales.clear()

for i in data:
    Month.append(i[1])
    MSales.append(i[2])

print("Month = ", Month)
print("Total Sales = ", MSales)

# Visulizing Data using Matplotlib
plt.bar(Month, MSales, color=['yellow', 'red', 'green', 'blue', 'cyan'])
# plt.ylim(0, 5)
plt.xlabel("Type")
plt.ylabel("Total Expenses")
plt.title("Date To Date  Expenses")
import random

n = random.randint(1111, 9999)

plt.savefig('static/plott/' + str(n) + '.jpg')

iimg = 'static/plott/' + str(n) + '.jpg'



selectQuery =  "SELECT * FROM expensetb where date between '" + d1 + "' and '" + d2
+ "' and Username='" + uname + "' "
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
```

```
        data = engine.execute("SELECT * FROM Employee_Data").fetchall()

        return render_template('MonthReport.html', data=data, dataimg=iimg)


def sendmsg(Mailid,message):
    import smtplib
    from email.mime.multipart import MIMEMultipart
    from email.mime.text import MIMEText
    from email.mime.base import MIMEBase
    from email import encoders

    fromaddr = "sampletest685@gmail.com"
    toaddr = Mailid

    # instance of MIMEMultipart
    msg = MIMEMultipart()

    # storing the senders email address
    msg['From'] = fromaddr

    # storing the receivers email address
    msg['To'] = toaddr

    # storing the subject
    msg['Subject'] = "Alert"

    # string to store the body of the mail
    body = message

    # attach the body with the msg instance
    msg.attach(MIMEText(body, 'plain'))

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

    # start TLS for security
    s.starttls()

    # Authentication
    s.login(fromaddr, "hneucvnontsuwgpj")

    # Converts the Multipart msg into a string
    text = msg.as_string()
```

```
    # sending the mail
    s.sendmail(fromaddr, toaddr, text)

    # terminating the session



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

13.2. GITHUB AND PROJECT DEMO LINK

GitHub Link : https://github.com/IBM-EPBL/IBM-Project-12336-1659447627

Project Demo Link :
https://drive.google.com/file/d/13uow9KKwQOZzbcELVbX0E5oOqBdBgYgB/view?usp=share_link

TEAM ID : PNT2022TMID00377