IBM – NALAIYA THIRAN PROJECT

PERSONAL EXPENSE TRACKER APPLICATION

PROJECT REPORT

INDUSTRY MENTOR : KUSBOO

FACULTY MENTOR : GOWTHAMANI R

TEAM ID : PNT2022TMID02725

TEAM LEAD : KARTHIC R S (19EUCS060)

TEAM MEMBER : AJAI BHALAJI S (19EUCS005)

TEAM MEMBER : BRINDHA K K (19EUCS025)

TEAM MEMBER : HARESHVAR A R (19EUCS037)

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution, Affiliated to Anna University Chennai - 600 025)

NOVEMBER 2022

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution. Affiliated to Anna University, Chennai)
Kuniamuthur, Coimbatore - 641 008

BONAFIDE CERTIFICATE

Certified that this project report titled **"PERSONAL EXPENSE TRACKER APPLICATION"** is the bonafide work of **Mr.KARTHIC R S(19EUCS060) , Mr.AJAI BHALAJI S (19EUCS005) , Miss.BRINDHA K K (19EUCS025) , Mr.HARESHVAR A R (19EUCS037)** who carried out the project work under my supervision.

SIGNATURE
Dr.K. SASI KALA RANI, M.E., Ph.D.,
HEAD OF THE DEPARTMENT

SIGNATURE
Dr.R.GOWTHAMANI,M.E.,
SUPERVISOR

Department of Computer Science and Engineering
Sri Krishna College of Engineering and Technology
Kuniamuthur,Coimbatore

Submitted for the Project viva-voce examination held on_____

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

1.1OVERVIEW

Today many people fall into bad financial practices and sometimes even crushing debt owing to lack of awareness about good spending habits and reliable tracking/management resources. An application tailored to young adults to start tracking their expenses, splitting their bills, learning to budget/save and review their spending practices frequently and bring about any necessary changes in order to get a grip on their finances. More portable than traditional systems and equipped to help users between the younger ages/adults of to efficiently manage and track their expenses and non-invasive parent involvement and supervision. Sense of financial freedom, instill good expense management practices in young adults early on, effective parent/guardian participation in teaching financial responsibilities Data Monetization can be employed. Scalability is ensured using micro-service architecture.

1.1 PURPOSE

If you don't check your spending and create a budget, you will have no control whatsoever on your money. Instead, money will control you, and you will either have perpetual lack of funds or you will end up steeped in debt. A money manager app helps you decide between short-term and long-term spending. If you are spending money frivolously, you will not have money to set financial goals. However, when you have a daily expense manager, you will be able to work with limited resources and use your money in a wise manner so that you can create financial goals and ensure you meet them. If you are clueless about how much is your inflow and how much you are

spending, you will not know at the end of the month what happened to your money. An expense tracker helps you figure out what is happening to your money, and whether you can afford something you want. If you don't have great financial management skills, you will not know how to categorize your expenses. However, tracking your expenses and budgeting them will help you become aware of how much you have to allocate to each expense category, and if you are short, you will be able to make adjustments with ease.

LITERATURE SURVEY

2.1 EXISTING PROBLEM

Today many people fall into bad financial practices and sometimes even crushing debt owing to lack of awareness about good spending habits and reliable tracking/management resources. An application tailored to young adults to start tracking their expenses, splitting their bills, learning to budget/save and review their spending practices frequently and bring about any necessary changes in order to get a grip on their finances. More portable than traditional systems and equipped to help users between the younger ages/adults of to efficiently manage and track their expenses and non-invasive parent involvement and supervision.

2.2 REFERENCES

[1] Intelligent online budget that manages the expenses. Archived from the original on 2007. Published by Girish Bekaroo.

[2] Managing finances is a practice carried out daily in homes across the world. Archived from the original on December 28, 2015. Published by Stephan snow and Dhayal Vyas.

[3] Online Income and Expense Tracker. Archived from the original on 2019. Published by S. Chandini, T. Poojitha, D. Ranjith, V.J. Mohammed Akram, M.S. Vani, V. Rajyalakshmi.

[4] Family Expense Manager Application. Archived from the original on 2017. Published by Rajaprabha M N.

[5] Personalized Expense Managing Assistant Using Android. Archived from the original on 2016. Published by N.ZahiraJahan MCA.,M.Phil. , K.I.Vinodhini.
[6] Student Expense Tracking Application. Archived from the original on 2022. Published by Saumya Dubey,Pragya Dubey,Rigved Rishabh Kumar,Aaisha Khatoon.
[7] A research at university on Tennessee on expense tracker . Archived from the original on 2011. Published by Dan Underwood.

2.3 PROBLEM STATEMENT DEFINITION

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.

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

- An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.
- It is a useful tool to helps teams better understand their users.
- Creating an effective solution requires understanding the true problem and the person who is experiencing it.
- The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

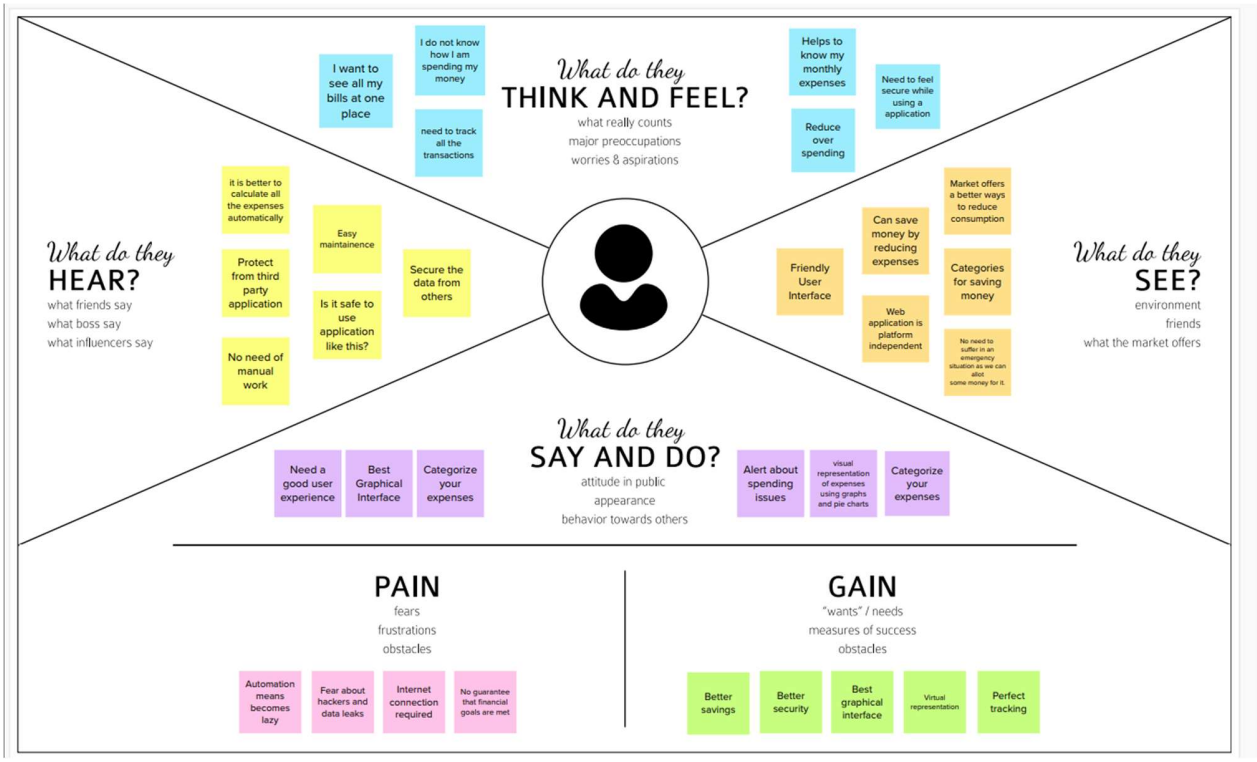

Fig 1.1.Empathy Map

3.2 IDEATION & BRAINSTORMING

Fig 1.2.Problem Statement & Brainstorm

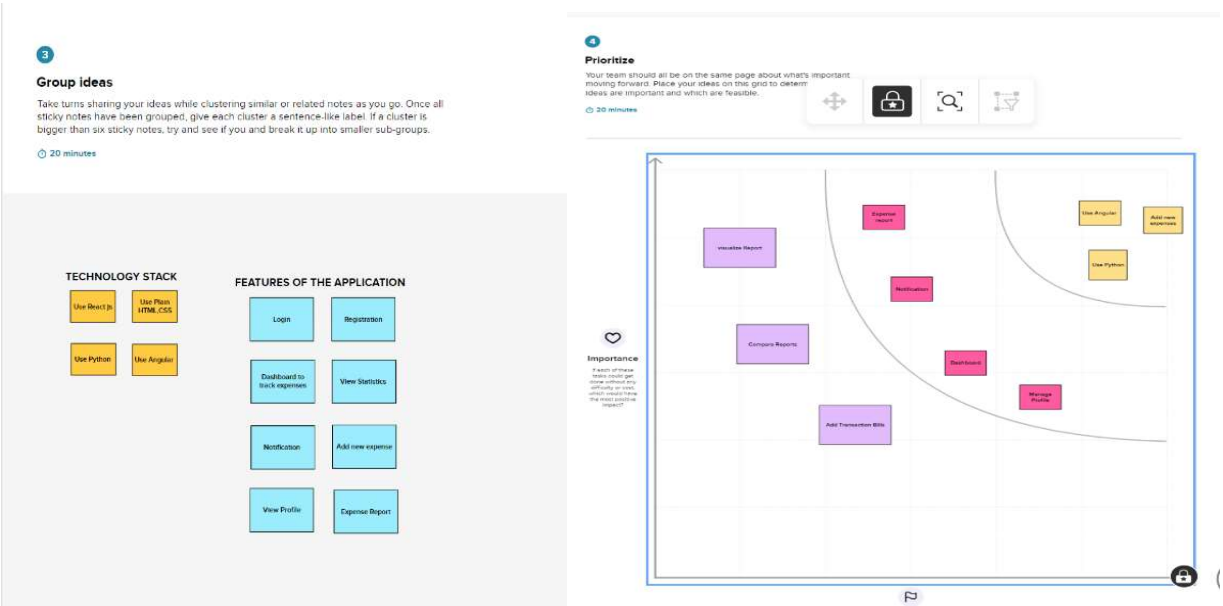

Fig 1.3.Group Ideas & Prioritize

3.3 PROPOSED SOLUTION

An application tailored to young adults to start tracking their expenses, splitting their bills, learning to budget/save and review their spending practices frequently and bring about any necessary changes in order to get a grip on their finances. More portable than traditional systems and equipped to help users between the younger ages/adults of to efficiently manage and track their expenses and non-invasive parent involvement and supervision. Sense of financial freedom, instill good expense management practices in young adults early on, effective parent/guardian participation in teaching financial responsibilities Data Monetization can be employed. Scalability is ensured using micro-service architecture. Our solution helps user to maintain their expense and view the summary. User want to enter the expense amount, category, description and the date. It automates the expense tracking and eliminate the manual calculations. User able to see the graphical representation of the expenses.

3.4 Problem Solution fit

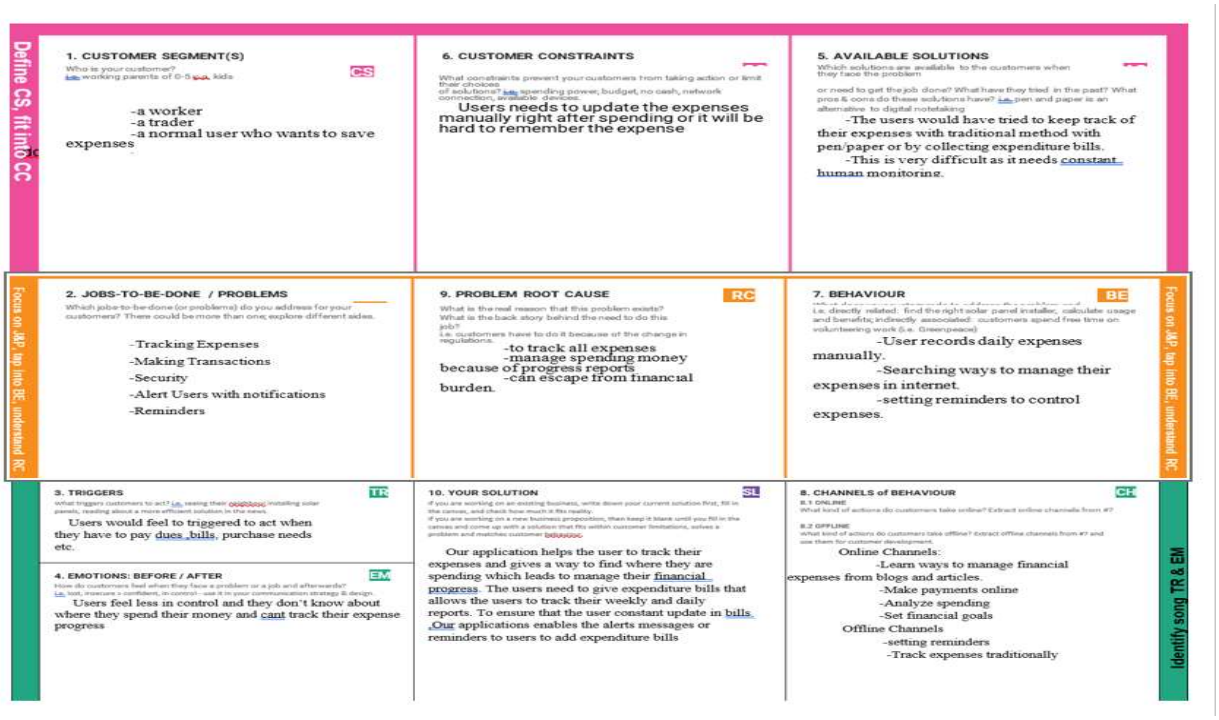

Fig 1.4.Problem Solution fit

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

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 Website
FR-2	User Confirmation	Confirmation via Email
FR-3	User Login	Login through registered Username
FR-4	Add Expense	User should add their expense for tracking their expenditure.
FR-5	Add Wallet	User should add money for comparing with expenses.

.

4.2 NON-FUNCTIONAL REQUIREMENTS

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

NFR No.	Non Functional Requirement (Epic)	Description
NFR-1	Usability	The plasma Donor application is user friendly and does not involve any complex process
NFR-2	Security	The donor/recipient details are stored in a secured cloud based database.
NFR-3	Reliability	The application will have no down time so that you can always rely on and the information provided by it are so reliable
NFR-4	Performance	The application will work efficiently in emergency situations with an instant notification system.
NFR-5	Availability	The application will be available online 24x7
NFR-6	Scalability	The application can be accessed by multiple users at the same time and it has the ability to increase or decrease the IT resources as needed.

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

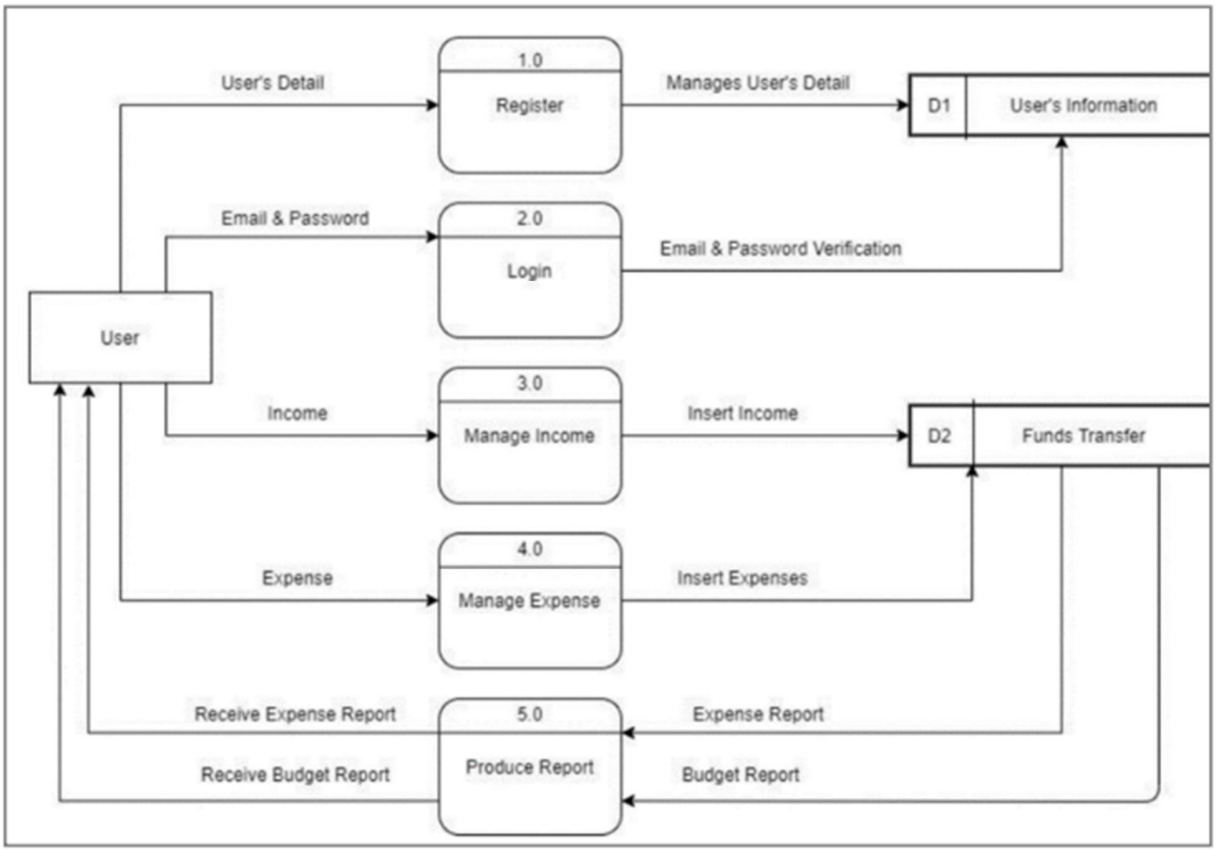

Fig 2.1.Data Flow Diagram

5.2 SOLUTION & TECHNICAL ARCHITECTURE

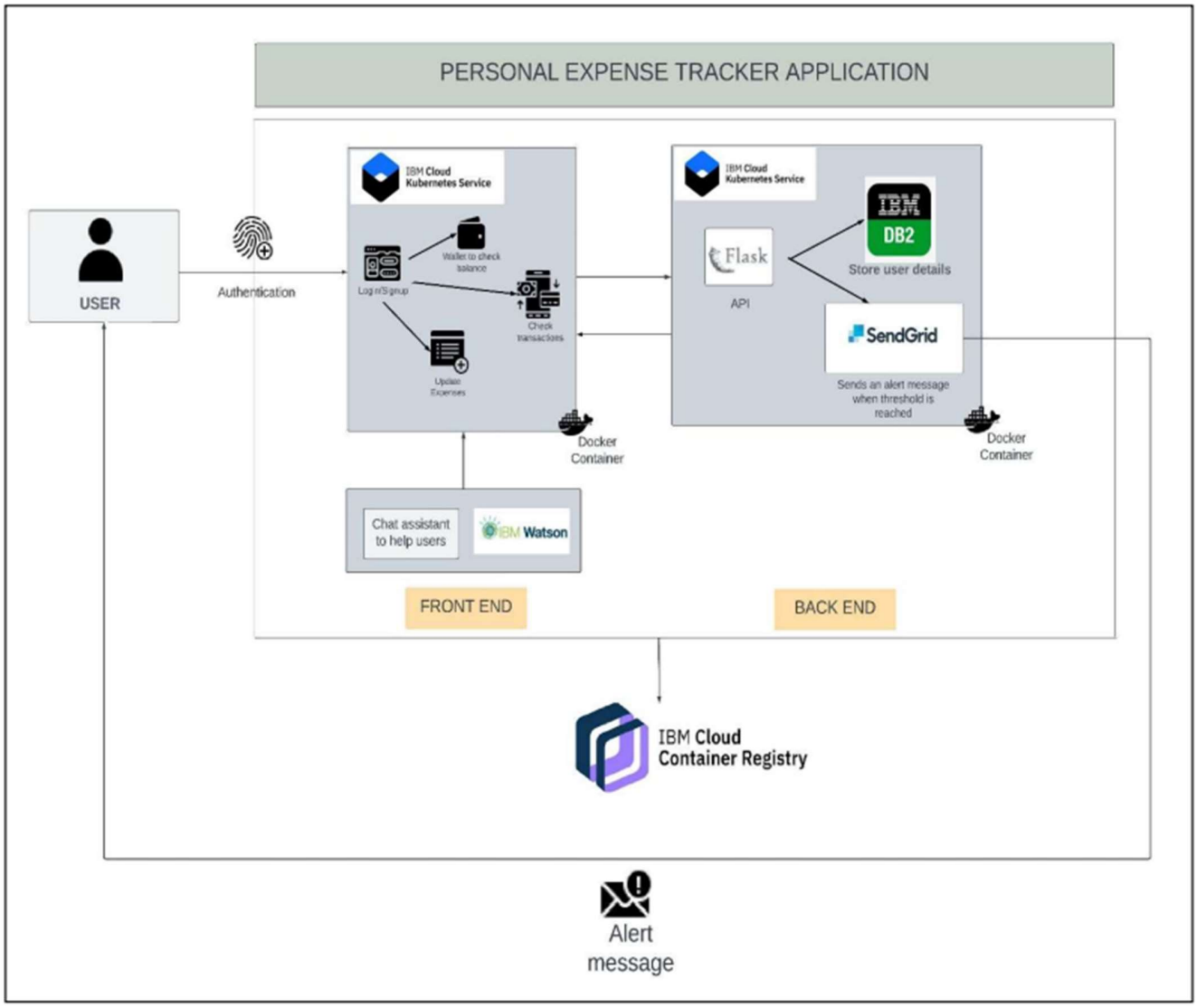

Fig 2.2. Architecture diagram

5.3 USER STORIES

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-3	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	I can register & access the dashboard with Gmail Login	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can login into application if correct mail id and password are entered	High	Sprint-1
	Dashboard	USN-6	As a user, I can see my transaction history	I can see if login is successful	Medium	Sprint-1
		USN-7	As a user, I can check balance	I can see if login is successful	Medium	Sprint-1
		USN-8	As a user, I can update my salary and expenses	I can see if login is successful	High	Sprint-2
Customer Care Executive (IBM Watson)	Application	USN-9	As a customer care executive, IBM Watson chat assistant can solve the problem and help the users facing issue	Chat assistant can provide support at any time	Medium	Sprint-2
Administrator	Application	USN-10	As an administrator, I can update the application based on user reviews	I can upgrade the application	Medium	Sprint-3
		USN-11	As an administrator, I can fix the bugs	I can fix bugs in case of issues	Medium	Sprint-3

PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

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	Karthic
Sprint-1		USN-2	As a user, I will receive confirmation emailonce I have registered for the application	1	High	Ajai Bhalaji
Sprint-1	Login	USN-3	As a user, I can log into the application byentering email & password	1	High	Hareshvar
Sprint- 1	Dashboard	USN-4	Logging in takes to the dashboard for the logged user.	2	High	Brindha
Sprint- 2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Hareshvar
Sprint- 2	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Ajai Bhalaji
Sprint- 2	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Karthic
Sprint- 2		USN-4	Making dashboard interactive with JS chatbot	2	High	Brindha
Sprint- 2		USN-1	Wrapping up the server side works of	1	Medium	Ajai Bhalaji

			frontend			
Sprint-3	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user's query	1	Medium	Karthic
Sprint-3	SendGrid	USN-3	Using SendGrid to send mail to the user about their expenses	1	Low	Brindha
Sprint-3		USN-4	Integrating both frontend and backend	2	High	Hareshvar
Sprint-4	Docker	USN-1	Creating image of website using docker	2	High	Karthic
Sprint-4	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Brindha
Sprint-4	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Hareshvar
Sprint-4	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Ajai Bhalaji

6.2 SPRINT DELIVERY SCHEDULE

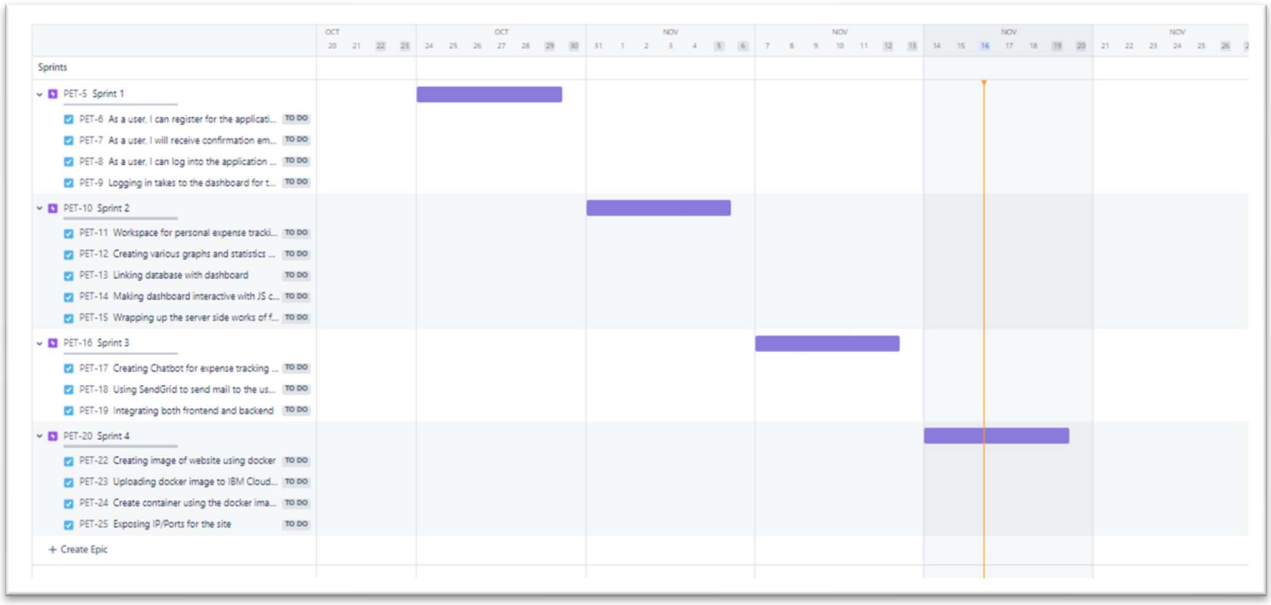

Fig 3.1. Sprint Delivery Schedule

6.3 REPORTS FROM JIRA

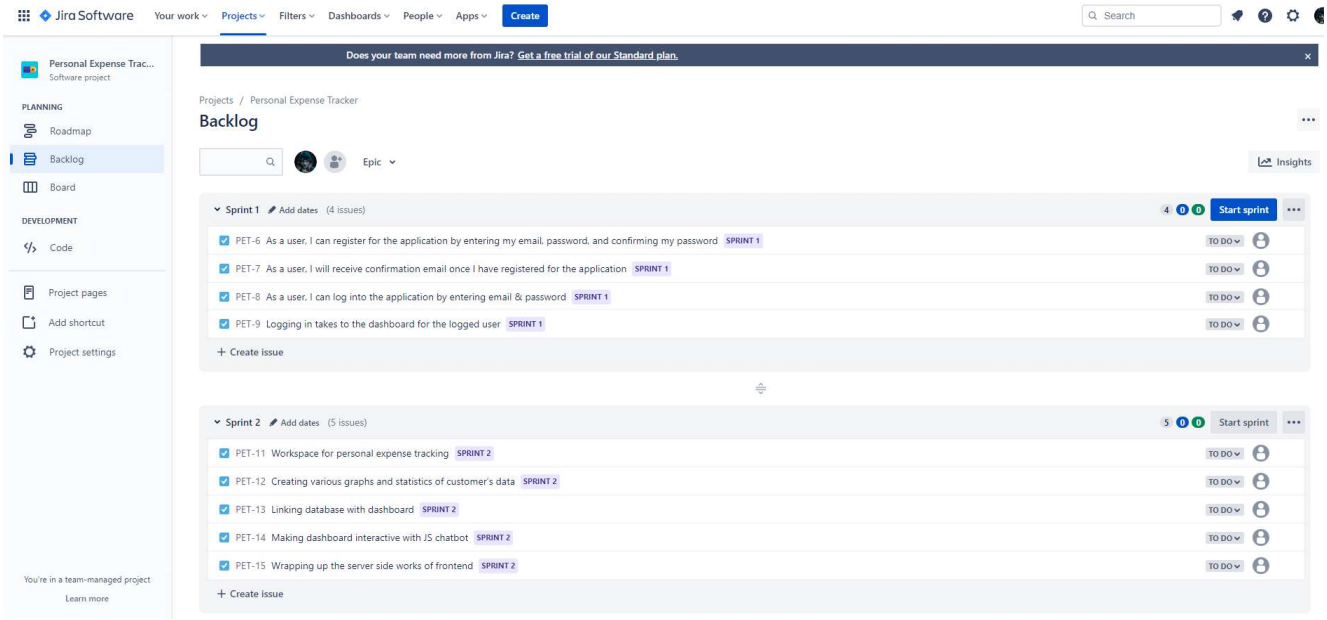

Fig 3.2. Sprint report

CODING & SOLUTIONING

7.1 FEATURE 1 – USER REGISTRATION

PYTHON SNIPPET :

```
@app.route('/register/', methods=['GET','POST'])
def register_page():
    try:
        form = RegistrationForm(request.form)
        if request.method == 'POST' and form.validate():
            _username = form.username.data
            _name = form.name.data
            _email = form.email.data
            _password = sha256_crypt.encrypt(str(form.password.data))
            sql = 'select * from profiles where username='+'\''+_username+'\''
            stmt = ibm_db.exec_immediate(conn, sql)
            dictionary = ibm_db.fetch_assoc(stmt)
            if dictionary == False:
                try:
                    insertValueQuery='insert into profiles values(?,?,?,?)'
                    print(_username+' '+_name+' '+_password+' '+_email)
                    param=_name,_username,_email,_password
                    stmt = ibm_db.prepare(conn, insertValueQuery)
                    ibm_db.execute(stmt, param)
                    flash("Thank you for registering!", "success")
                    gc.collect()
                    session['logged_in'] = True
                    session['username'] = _username
                    session.modified = True
                    return redirect(url_for('dashboard'))
                except Exception as e:
```

```
                print(e)
                return render_template('error.html',e=e)
            else:
                flash('User Already registered with  username {}'.format(_username), "warning")
                return render_template('register.html', form=form)
        return render_template('register.html', form=form)
    except Exception as e:
        return render_template('error.html',e=e)
```

7.2 FEATURE 2 – ADD EXPENSE

PYTHON SNIPPET :

```
@app.route('/addexpense/<balance>', methods=['POST'])
@login_required
def addexpense(balance):
    amount = request.form['amount']
    detail = request.form['details']
    if (int(amount) == 0):
        flash("Please enter some amount", "danger")
        return redirect(url_for('dashboard'))
    else:
        sql = "INSERT INTO USERDATA(USERID,AMOUNT,DETAILS) VALUES(?,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1,session["username"])
        ibm_db.bind_param(stmt, 2, amount)
        ibm_db.bind_param(stmt, 3, detail)
        ibm_db.execute(stmt)
        print("Balance")
        print(balance)
        flash("Expense added successfully", "success")
        return redirect(url_for('dashboard'))
```

TESTING

8.1 TEST CASES

1. Login button click with wrong credentials entered.
2. Signup with already registered mail ID.
3. Signup with wrong form data entered.
4. Entering home page with logged out session.
5. Clicking home page buttons with logged out session.
6. Invalid data entered in change password page and requested for change in password

8.2 USER ACCEPTANCE TESTING

S.NO	TEST CASE	REQUIRED OUTPUT	RESULT OUTPUT	STATUS
1	**Login button click with wrong credentials**	**Wrong credentials entered notification**	**Wrong credentials entered notification**	**ACCEPTED**
2	**Signup with already registered mail ID.**	**Email already registered notification**	**Email already registered notification**	**ACCEPTED**
3	**Signup with wrong form data entered.**	**Wrong credentials entered notification**	**Wrong credentials entered notification**	**ACCEPTED**
4	**Entering home page with logged out session.**	**Take user to login page**	**Take user to login page**	**ACCEPTED**
5	**Clicking home page buttons with logged out session.**	**Take user to login page**	**Take user to login page**	**ACCEPTED**
6	**Invalid data entered in change password page and requested for change in password.**	**Wrong form data entered notification**	**Wrong form data entered notification**	**ACCEPTED**

RESULTS

9.1 PERFORMANCE METRICS

Web application performance metrics help determine certain aspects that impact the performance of an application. There are eight key metrics, including: User Satisfaction—also known as Apdex Scores, uses a mathematical formula in order to determine user satisfaction.

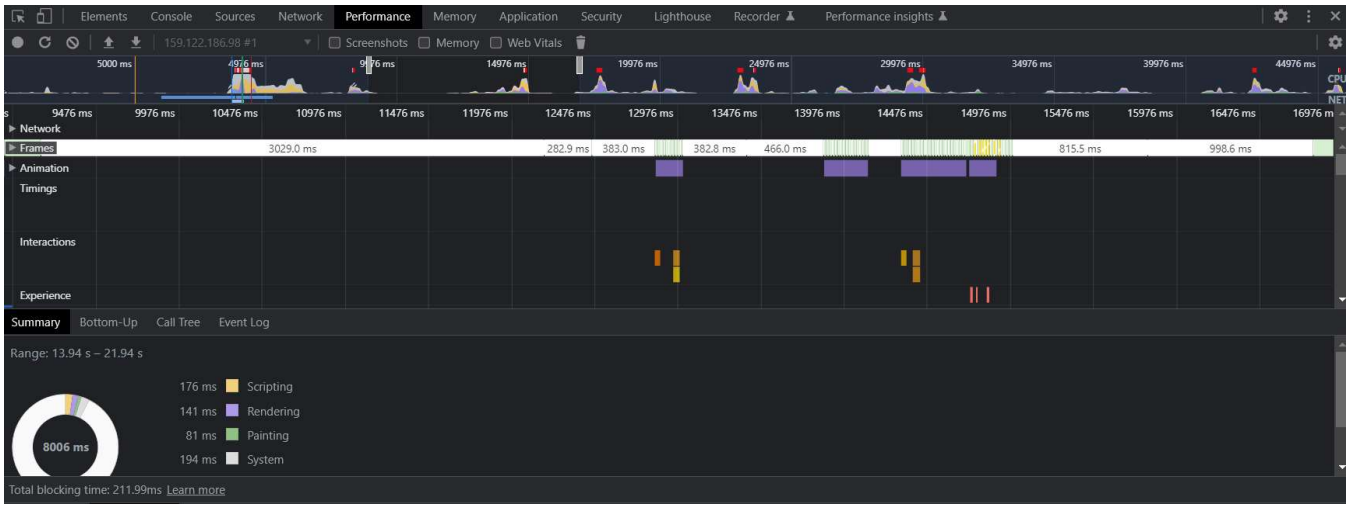

Fig 4.1. Performance Metrics

ADVANTAGES & DISADVANTAGES

ADVANTAGES

1. Which allows users to track their expenses daily, weekly, monthly, and yearly in terms of summary, bar graphs, and pie-charts.
2. Separate view for credit and debit transactions
3. no burden of manual calculations
4. generate and save reports.
5. You can insert, delete records
6. You can track expenses by categories like food, automobile, entertainment, education etc.
7. You can track expenses by time, weekly, month, year etc.
8. Setting monthly limits and we can update it later
9. Customized email alerts when limit exceeds.

DISADVANTAGES

1. User have entry every records manually
2. The category divided may be blunder or messy
3. Can’t able to customized user defined categories

CONCLUSION

In this paper, After making this application we assure that this application will help its users to manage the cost of their daily expenditure. It will guide them and make them aware about their daily expenses. It will prove to be helpful for the people who are frustrated with their daily budget management, irritated because of the amount of expenses and wish to manage money and to preserve the record of their daily cost which may be useful to change their way of spending money. In short, this application will help its users to overcome the wastage of money.

FUTURE SCOPE

In further days, there will be mails and payment embedded with the app. Also, backup details will be recorded on cloud.

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 .

Alerts for paying dues and remainders to record input at particular user defined time.

APPENDIX

SOURCE CODE

Dashboard.html :

```
{% extends 'header.html' %}
{% block body %}

{% if success%}

        <div class="container-fluid position-fixed z">
        <div class="mt-4 float-end">
        <div class="alert alert-success alert-dismissible fade show" role="alert">
                <i class="fas fa-check-circle"></i>  {{ success }}
                <button type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button>
        </div>
        </div>
        </div>
{% endif%}

{% if danger%}
        <div class="container-fluid position-fixed z">
        <div class="mt-4 float-end">
        <div class="alert alert-danger alert-dismissible fade show" role="alert">
        <i class="fas fa-exclamation-triangle"></i>   {{ danger }}
        <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
        </div>
        </div>
        </div>
{% endif %}
```

```
    <div class="d-flex justify-content-center mt-5">
       <h1 class="h2 text-black"> Welcome Board {{ name }} !!!</h1>
    </div><br>


 <div class="container mt-5">
        <button type="button" class="btn btn-danger col-auto" data-toggle="modal" data-
target="#walletpop">
                <i class="fas fa-wallet"></i>  Your Wallet  
           </button><br><br>
    <div class="row">
      <div class="col-12">
        <div class="card">
          <div class="card-body text-center">
            <h5 class="card-title m-b-0 text-white">Your Expenses</h5>
          </div>
            <div class="table-responsive">
              <table class="table text-center">
                <thead class="thead-light">
                  <tr>
                    <th scope="col">Amount</th>
                    <th scope="col">Expense Details</th>
                    <th scope="col">Date & Time</th>
                    <th scope="col">Action</th>
                  </tr>
                </thead>
                <tbody class="customtable">
                  {% for row in expensedetails %}
                        <tr>
                          <td>{{ row['AMOUNT'] }}</td>
                          <td>{{ row['DETAILS'] }}</td>
                          <td>{{ row['DATEADDED'] }}   {{ row['TIMEADDED'] }}</td>
```

```
                        <td><a href="{{url_for('deleteexpense', val=row['TIMEADDED'],
amount=row['AMOUNT'])}}" class="blackhvr" title="Delete"><i class="fas fa-trash
del"></i></a></td>
                    </tr>
                {% endfor %}
                </tbody>
            </table>
        </div>
    </div>
  </div>

    <br><br>
    <h3 class="h5 float-end col-auto mb-2"> Your total expense : ₹{{ totalexpense }}</h3>

    <br><br>
   <button type="button" class="btn btn-primary float-end col-auto" data-toggle="modal" data-
target="#expensemodal">
        <i class="fas fa-plus"></i> Add Expense  
   </button>

   <br><br><br><br><br>

  <div class="col-7 d-flex mt-5">
    <canvas id="linechart" height="125">
    </canvas>

    <div class="float-end" style="height: 57%; width:53%; margin-left: 25%;">
      <canvas id="linechart2" class="float-end">
      </canvas>
    </div>
  </div>
```

```
  </div>



  <!-- Expense Modal -->

<div class="modal fade" id="expensemodal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Expense</h5>
      </div>
      <div class="modal-body">
        <form action="{{ url_for('addexpense',balance=walletbalance )}}" method="post"
id="addexpense">
          <div class="form-group">
            <label for="amount" class="col-form-label">Amount:</label>
            <input type="number" class="form-control" name="amount" id="amount" required>
          </div>
          <div class="form-group">
            <label for="detail" class="col-form-label">Detail:</label>
            <textarea class="form-control" name="details" id="detail" required></textarea>
          </div>
        </form>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
        <button type="submit" form="addexpense" class="btn btn-primary">Add</button>
      </div>
    </div>
  </div>
</div>
```

```
<!-- Expense Modal End-->

<div class="modal fade" id="walletmodal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel" aria-hidden="true">
        <div class="modal-dialog" role="document">
          <div class="modal-content">
            <div class="modal-header">
              <h5 class="modal-title" id="exampleModalLabel">Add Money To Your Wallet</h5>
            </div>
            <div class="modal-body">
              <form id="addmoney" action="{{ url_for('addmoney')}}" method="post">
                <div class="form-group">
                  <label for="amount" class="col-form-label">Amount:</label>
                  <input type="number" class="form-control" name="walletamount" id="amount" required>
                </div>
                <div class="modal-footer">
                  <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
                  <button type="submit" form="addmoney" class="btn btn-primary" data-
target="#walletpop">Add</button>
                </div>
                </form>
            </div>

          </div>
        </div>
</div>

<div class="modal fade" id="walletpop" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel"
     aria-hidden="true" data-backdrop="true">
     <div class="modal-dialog modal-side modal-bottom-right modal-notify modal-danger"
role="document">
```

```
<!--Content-->
<div class="modal-content">
 <!--Header-->
 <div class="modal-header bg-danger">
  <p class="heading text-white h5">
   <strong> Your wallet </strong>
  </p>
 </div>

 <!--Body-->
 <div class="modal-body">
  <div class="row">
   <div class="col-3">
    <p></p>
    <p class="text-center">
     <i class="fas fa-wallet fa-5x"></i>
    </p>
   </div>

   <div class="col-9">
    <p>
     <strong>Your wallet Balance </u>:</strong>
    </p>
    <h2 class="h2 text-success">₹{{ walletbalance }}</h2>



   </div>
  </div>
 </div>

 <!--Footer-->
 <div class="modal-footer flex-center">
```

```
          <button type="button" class="btn btn-danger" data-toggle="modal" data-
target="#walletmodal" data-dismiss="modal">
            <i class="fas fa-plus"></i> Add Money</button>

          <a type="button" class="btn btn-outline-danger" data-dismiss="modal">Close</a>
        </div>
      </div>
    </div>
  </div>

 <!-- wallet Modal End-->
<script>


 // Line Graph

   var ctx = document.getElementById('linechart').getContext('2d');
   var linechart = new Chart(ctx,{
    type:'line',
    data:{
     labels:{{ label | safe }},
     datasets:[
       {
        label:"Expense Amount",
        data:{{ amountlabel | safe }},
        backgroundColor: 'rgba(75, 192, 192, 0.363)',
        borderColor: 'rgb(255, 99, 132)',
        fill:true,
        tension:0.2
       }
     ]
    },
    options:{
```

```
        resposive:true,
        hitRadius:25,
        hoverRadius:10,
        animation:{
          delay:700
        }
      }
    })


//     // Polar Graph

    var ctx2 = document.getElementById('linechart2').getContext('2d');
    var linechart2 = new Chart(ctx2,{
      type:'polarArea',
      data:{
        labels:{{ label | safe }},
        datasets:[
          {
            label: 'Spent',
            data: {{ amountlabel | safe }},
            backgroundColor: [
                'rgb(255, 99, 132)',
                'rgb(75, 192, 192)',
                'rgb(255, 205, 86)',
                'rgb(201, 203, 207)',
                'rgb(54, 162, 235)'
              ]
        }
        ]
      },
      options:{
        resposive:true,
```

```
        animation:{
          delay:1000
        }
       }
      })

</script>

<footer style="position: fixed; bottom: 0px; width: 100%; height: 30px; background-color: green; text-align: center;">
        <p class="credits" style="color: white; "><strong>Site created by <a href="https://github.com/IBM-EPBL/IBM-Project-24260-1659940730" style="color: black">CODE CRACKERS</a></strong></p>
</footer>
{% endblock %}
```

MyApp.py :

```
from flask import Flask, render_template, flash, request, url_for, redirect, session,  get_flashed_messages
from flask_mail import Mail, Message
from Forms.forms import RegistrationForm, LoginForm
from passlib.hash import sha256_crypt
from functools import wraps
import random
import gc, os
import ibm_db

app = Flask(__name__)
app.config['SECRET_KEY'] = os.urandom(24)

# Setting up mailing config!
app.config.update(
```

```
    MAIL_SERVER='smtp.gmail.com',
    MAIL_PORT=465,
    MAIL_USE_TLS=False,
    MAIL_USE_SSL=True,
    MAIL_USERNAME = '<admin-mail>',
    MAIL_PASSWORD = '<admin-mail-password>'   #stored as environment variable.
)
mail = Mail(app)
conn=ibm_db.connect('DATABASE=bludb;HOSTNAME=<HOSTNAME>;PORT=<PORT>;SECURITY
=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=<UID>;PWD=<PWD>',",")




def admin_access_required(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if session['username'] == 'admin':
            return f(*args, **kwargs)
        else:
            flash("Access Denied, login as admin", "danger")
            return redirect(url_for('login_page'))
    return wrap

def login_required(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if 'logged_in' in session:
            return f(*args,**kwargs)
        else:
            flash('You need to login first!', "warning")
            return redirect(url_for('login_page'))
    return wrap
```

```
def already_logged_in(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if 'logged_in' in session:
            flash("You are already logged in!", "success")
            return redirect(url_for('dashboard'))
        else:
            return f(*args, **kwargs)
    return wrap

@app.route('/logout/')
@login_required
def logout():
    flash("You have been logged out!", "success")
    session.clear()
    gc.collect()
    return redirect(url_for('main'))

def verify(_username, _password):
    try:
        print("Database connected")
        username=_username
        password=_password
        sql = 'select * from profiles where username='+'\"'+username+'\"'
        stmt = ibm_db.exec_immediate(conn, sql)
        dictionary = ibm_db.fetch_assoc(stmt)
        if dictionary != False:
            if sha256_crypt.verify(_password,dictionary["PASSWORD"]):
                return True
            else:
                flash("Invalid Credentials, password isn't correct!", "danger")
                return False
```

```
        else:
            flash("No such user found with this username", "warning")
            return False
    except:
        print ("Database error",ibm_db.conn_errormsg())
        return False


@app.route('/', methods=['GET','POST'])
def main():
    return render_template('main.html')


@app.route('/dashboard/',methods=['GET','POST'])
@login_required
def dashboard():
    expensedetails = []
    sql = "SELECT AMOUNT,DETAILS,CHAR(DATE(DANDT),USA) AS DATEADDED,
CHAR(TIME(DANDT),USA) AS TIMEADDED FROM USERDATA WHERE USERID = ?"
    stmt = ibm_db.prepare(conn, sql)
    print(session["username"])
    ibm_db.bind_param(stmt, 1, session["username"])
    ibm_db.execute(stmt)
    details = ibm_db.fetch_assoc(stmt)
    while details != False:
        expensedetails.append(details)
        details = ibm_db.fetch_assoc(stmt)

    label = [row['DATEADDED'] for row in expensedetails]


    amountlabel = [row['AMOUNT'] for row in expensedetails]
```

```
    print(request.args.get('success'))

    sql2 = "SELECT SUM(AMOUNT) AS TOTALVAL FROM USERDATA WHERE USERID = ?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2, 1, session["username"])
    ibm_db.execute(stmt2)
    totalexpense = ibm_db.fetch_assoc(stmt2)
    if totalexpense['TOTALVAL'] is None:
        totalexpense['TOTALVAL'] = 0


    sql3 = "SELECT SUM(AMOUNT) AS TOTALVAL FROM WALLET WHERE WALLETID = ?"
    stmt3 = ibm_db.prepare(conn, sql3)
    ibm_db.bind_param(stmt3, 1, session["username"])
    ibm_db.execute(stmt3)
    walletbalance = ibm_db.fetch_assoc(stmt3)
    if walletbalance['TOTALVAL'] is None:
        walletbalance['TOTALVAL'] = 0
        availablebalance = 0
    else:
        availablebalance = int(
            walletbalance['TOTALVAL']) - int(totalexpense['TOTALVAL'])
    if (availablebalance <= 50):

        flash("Your balance is too low!!!")
    elif (availablebalance > 50 and availablebalance <= 200):
        sql4="SELECT * from profiles where username=?"
        stmt3 = ibm_db.prepare(conn, sql4)
        ibm_db.bind_param(stmt3, 1, session["username"])
        ibm_db.execute(stmt3)
        profiles = ibm_db.fetch_assoc(stmt3)
        msg = Message('Hello, Reminder from Personal expense manager app!', sender = '<admin-mail-d>',
recipients = [profiles["EMAIL"]])
```

```
        msg.body = "Your balance is getting low so take care of your expenses...!!!\nAvailable Balance:Rs.{}\nThanks & Regards,\n\nPersonal Expense Tracker".format(availablebalance)
        mail.send(msg)
        flash("Your balance is getting low so take care of your expenses...!!!","warning")
    return render_template('dashboard.html', dashboard='active', name=session["username"], success=request.args.get('success'), danger=request.args.get('danger'), expensedetails=expensedetails, totalexpense=totalexpense['TOTALVAL'], walletbalance=availablebalance,label=label, amountlabel=amountlabel)

@app.route('/addmoney', methods=['POST'])
@login_required
def addmoney():
    try:
        amount = request.form['walletamount']
        print("Balance:")
        print(int(amount))
        if (int(amount) == 0):
            flash("Please enter some amount", "danger")
            return redirect(url_for('dashboard'))

        else:
            sql="select * from wallet where walletid=?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, session["username"])
            ibm_db.execute(stmt)
            account = ibm_db.fetch_assoc(stmt)
            print(account)
            if account!=False:
                sql = "update wallet set amount=amount+? where walletid=?"
                stmt = ibm_db.prepare(conn, sql)
                ibm_db.bind_param(stmt, 1, int(amount))
                ibm_db.bind_param(stmt, 2, session["username"])
                print("Executing...")
```

```
            ibm_db.execute(stmt)
            print("Executed!!")
            flash("Money added successfully","success")
            return redirect(url_for('dashboard'))
        else:
            sql = "INSERT INTO WALLET(WALLETID,AMOUNT) VALUES(?,?)"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, session["username"])
            ibm_db.bind_param(stmt, 2, int(amount))
            ibm_db.execute(stmt)
            flash("Money added successfully","success")
            return redirect(url_for('dashboard'))
    except Exception as e:
        print(e)
        flash("Unexpexcted error occured")
        return redirect(url_for('dashboard'))

@app.route('/addexpense/<balance>', methods=['POST'])
@login_required
def addexpense(balance):
    amount = request.form['amount']
    detail = request.form['details']

    if (int(amount) == 0):
        flash("Please enter some amount", "danger")
        return redirect(url_for('dashboard'))

    else:
        sql = "INSERT INTO USERDATA(USERID,AMOUNT,DETAILS) VALUES(?,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1,session["username"])
        ibm_db.bind_param(stmt, 2, amount)
        ibm_db.bind_param(stmt, 3, detail)
```

```
        ibm_db.execute(stmt)
        print("Balance")
        print(balance)
        flash("Expense added successfully", "success")
        return redirect(url_for('dashboard'))

@app.route('/deleteexpense/<val>/<amount>')
@login_required
def deleteexpense(val, amount):

    sql = "DELETE USERDATA WHERE USERID=? AND CHAR(TIME(DANDT),USA)= ? AND AMOUNT=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, session["username"])
    ibm_db.bind_param(stmt, 2, val)
    ibm_db.bind_param(stmt, 3, amount)
    ibm_db.execute(stmt)
    flash("Deleted Succesfully","success")
    return redirect(url_for('dashboard'))

@app.route('/login/', methods=['GET','POST'])
@already_logged_in
def login_page():
    try:
        form = LoginForm(request.form)
        if request.method == 'POST':
            # to create data base first!
            _username = form.username.data
            _password = form.password.data

            # check if username and password are correct
            if verify(_username, _password) is False:
                return render_template('login.html', form=form)
```

```
            session['logged_in'] = True
            session['username'] = _username

            gc.collect()
            return redirect(url_for('dashboard',success='Login Successfull'))

        return render_template('login.html', form=form)
    except Exception as e:
        return render_template('error.html',e=e)

@app.route('/register/', methods=['GET','POST'])
def register_page():
    try:
        form = RegistrationForm(request.form)
        if request.method == 'POST' and form.validate():
            _username = form.username.data
            _name = form.name.data
            _email = form.email.data
            _password = sha256_crypt.encrypt(str(form.password.data))
            sql = 'select * from profiles where username='+'\"'+_username+'\"
            stmt = ibm_db.exec_immediate(conn, sql)
            dictionary = ibm_db.fetch_assoc(stmt)
            if dictionary == False:
                try:
                    insertValueQuery='insert into profiles values(?,?,?,?)'
                    print(_username+' '+_name+' '+_password+' '+_email)
                    param=_name,_username,_email,_password
                    stmt = ibm_db.prepare(conn, insertValueQuery)
                    ibm_db.execute(stmt, param)
                    flash("Thank you for registering!", "success")
                    gc.collect()
                    session['logged_in'] = True
                    session['username'] = _username
```

```
                session.modified = True
                return redirect(url_for('dashboard'))
            except Exception as e:
                print(e)
                return render_template('error.html',e=e)
        else:
            flash('User Already registered with  username {}'.format(_username), "warning")
            return render_template('register.html', form=form)
    return render_template('register.html', form=form)
  except Exception as e:
    return render_template('error.html',e=e)

@app.route('/forget_password/', methods=['GET', 'POST'])
def forget_password():
  _email = None
  try:
    if request.method=="POST":
      if request.form['submit'] == "Send Email":
        #check if email matches in database
        _email = request.form['email']
        sql = 'select * from profiles where email='+'\"'+_email+'\"
        stmt = ibm_db.exec_immediate(conn, sql)
        dictionary = ibm_db.fetch_assoc(stmt)
        if dictionary == False:
          flash('Email is not registered with us', "danger")
          _email = None
        else:
          session['username'] = dictionary["USERNAME"]
          msg = Message('Hello, your personal expense manager app!', sender = '<admin-mail-id>',
recipients = [_email])
          secret_key = random.randint(1000,10000)
          session['otp'] = secret_key
          session.modified = True
```

```
                msg.body = "Your One Time password: {}. \n Valid till half an hour from the generation of the OTP.".format(secret_key)
                mail.send(msg)
                flash("OTP is sent to your registered mail Id.", "success")
            return render_template('forget_password.html')
        if request.form['submit'] == "Verify OTP":
            otp = request.form['otp']
            if 'username' in session:
                if int(otp) == session['otp']:
                    session['logged_in'] = True
                    return redirect(url_for('dashboard'))
                else:
                    flash("OTP is incorrect. Try again!", "warning")
                    return render_template('forget_password.html')
            else:
                flash("First enter email!")
                return render_template('forget_password.html')
    else:
        return render_template('forget_password.html')
  except Exception as e:
    return render_template('error.html', e=e)

@app.errorhandler(500)
@app.errorhandler(404)
def page_not_found(e):
  return render_template('error.html', e=e)


if __name__ == "__main__":
  app.run(debug=True)
```

GITHUB & PROJECT DEMO LINK

Github Link :

https://github.com/IBM-EPBL/IBM-Project-24260-1659940730

Demo Link:

https://youtu.be/oBlp1MNYcsg