CLOUD APPLICATION DEVELOPMENT – BANKING & FINANCE

PERSONAL EXPENSE TRACKER APPLICATION

PROJECT REPORT

Submitted by:

Gatharine Kerenhap E (950019104013)

Narmatha B (950019104032)

Sabana Fathima S (950019104038)

Shrimathi S (950019104042)

ANNA UNIVERSITY REGIONAL CAMPUS – TIRUNELVELI

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AUGUST 2022 – NOVEMBER 2022

TABLE OF CONTENT

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

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

7.1 Feature 1

7.2 Feature 2

7.3 Feature 3

7.4 Feature 4

7.5 Database Schema (if Applicable)

8. TESTING

8.1 Test Cases

8.2 User Acceptance Testing

9. RESULTS

9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo

CHAPTER-1

INTRODUCTION

1.1 Project Overview

Personal finance entails all the financial decisions and activities that a Finance app makes life easier by helping the user to manage their finances efficiently. A personal finance app will not only help them with budgeting and accounting but also give helpful insights about money management. In this world, people spend lots of money than earning it. Most of the time they spend more money on unwanted things. By this activity they are facing so much struggles to run their family at the end of the month or year. By solving this issue, an application which is used to add the expenses of a user and spend money according to that plan. If a user spend additional money, this application notify them through their mail. Also by developing this application financial issue in a family will not be arise anymore.

1.2 Purpose

The purpose of this expense tracker application is to make people to spend money on useful things and to avoid many financial problems in a family. Financial issue occur when people spend lots of money on buying unwanted stuffs and things. They do not have any plan to spend it. So to avoid these issues this application is created and it is very useful for all kind of people.

CHAPTER-2

LITERATURE SURVEY

2.1 Existing Problem

1.Spendee

Spendee is a finance tracking app which get your money under control with easy to use finance manager in your pocket and also it connect with your online banking, E-Wallet and see your wealth in one place.

Advantages:

• This app seeing all your financial habits enables you to stick to your goals .

• It be organised in what’s important to your finance.

• It takes responsibility and knows exactly where your money goes.

• The data is categorized, displayed in stylish graphs that helps on savings.

• It helps in proper financial health and maintaining positive cash flow.

Disadvantages:

• The entries in the app may take some time to process.

• It is very slow while syncing to cloud.

• The payment tracking is convenient but not all the time of emergency

2.Budget App – Expense Tracker

Budget app is a expense tracker app which tracks and records your expenses with a help of a balance calculator.

Advantages:

• This app planning the daily income by recording the expenses.

• The app has automatic generation of money statistics charts.

• It set a budget for each month in this app

• The balance can be calculated and displayed automatically.

• The data of the expenses can be export to an Excel file.

Disadvantages:

• The balance of your budget for that month is not in the statistics section.

• The data may vary while taking backup.

• It missed to show all records on same chart.

3.Wallet : Budget Expense Tracker

Wallet is all-in-one personal finance app which has features of budget planner, bill manager, and money saver in it.

Advantages:

• The app plan, manage and get a report of your finances.

• This app help to keep full control of your finances.

• It provide detailed information about financial use.

Disadvantages:

• The app has no option for removing unnecessary bank data.

• The app is very slow to process.

2.2 References

https://www.spendee.com/

https://play.google.com/store/apps/details?id=com.hg.moneymanager.budgetapp&hl=en_IN&gl=US&pli=1

https://play.google.com/store/apps/details?id=com.droid4you.application.wallet&hl=en_IN&gl=US

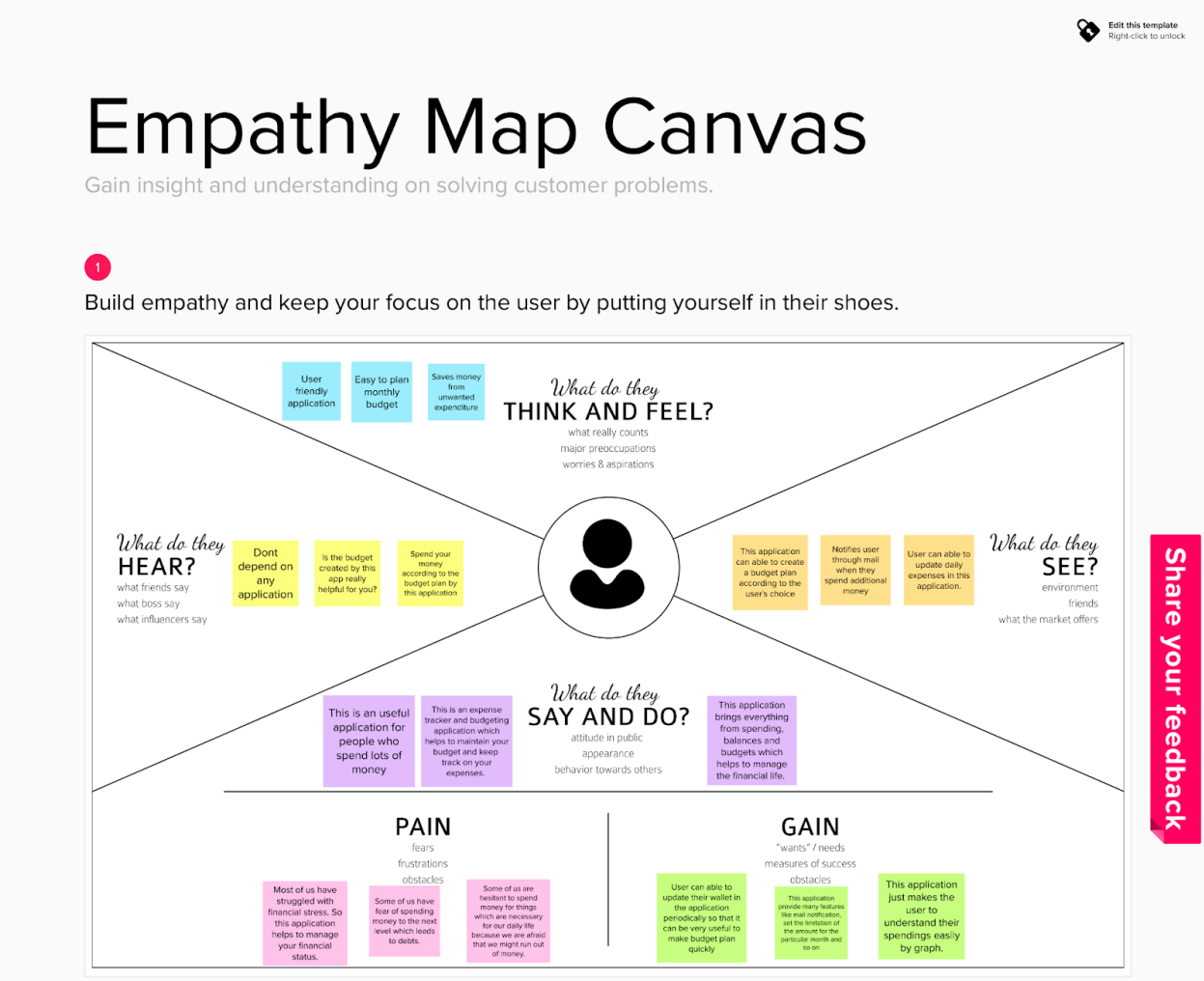
2.3 Problem Statement Definition

In this world, people spend lots of money than earning it. Most of the time they spend more money on unwanted things. By this activity they are facing so much struggles to run their family at the end of the month or year. By solving this issue, an application which is used to add the expenses of a user and spend money according to that plan. If a user spend additional money, this application notify them through their mail. Also by developing this application financial issue in a family will not be arise anymore.

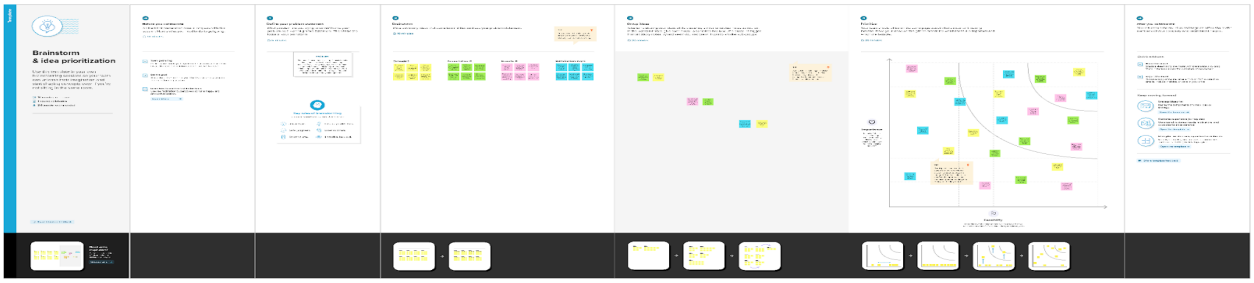
CHAPTER-3

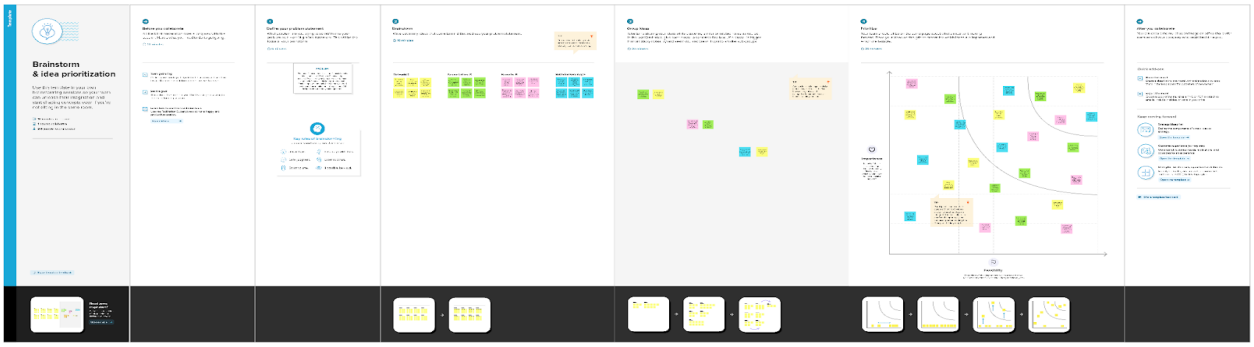
IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming





3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	It is challenging for the people to manage their cash flow, day-to-day. They always prepare a monthly budget for their expenses manually.
2.	Idea / Solution description	When it comes to accounting, sometimes manual calculation may lead to fluctuations in spending. Such problems can be overcome by developing an application where users can add their income in the wallet. When expenses exceeds the income in the wallet, the user will be notified through email.
3.	Novelty / Uniqueness	Users can set limit to some specified categories like Education, Groceries, Pharmacy etc.
4.	Social Impact / Customer Satisfaction	It will help the people to track their expenses.
5.	Business Model (Revenue Model)	The details of the day-to-day expenditure of the user can be recorded in the spreadsheet which will be easily recognized by the user.
6.	Scalability of the Solution	This application can able to withstand many number of users.

3.4 Problem Solution Fit

CHAPTER-4

REQUIREMENT ANALYSIS

4.1 Functional Requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Login	Registration through Gmail
FR-2	User Confirmation	Confirmation via Gmail
FR-3	User Password	Confirmation via comparing database details
FR-4	User Details for Wallet	Registration through Register Form
FR-5	User Details Confirmation	Confirmation via password
FR-6	Wallet update	Add income in the wallet
FR-7	Email Alert	Alert Message through Gmail if wallet amount exceeds

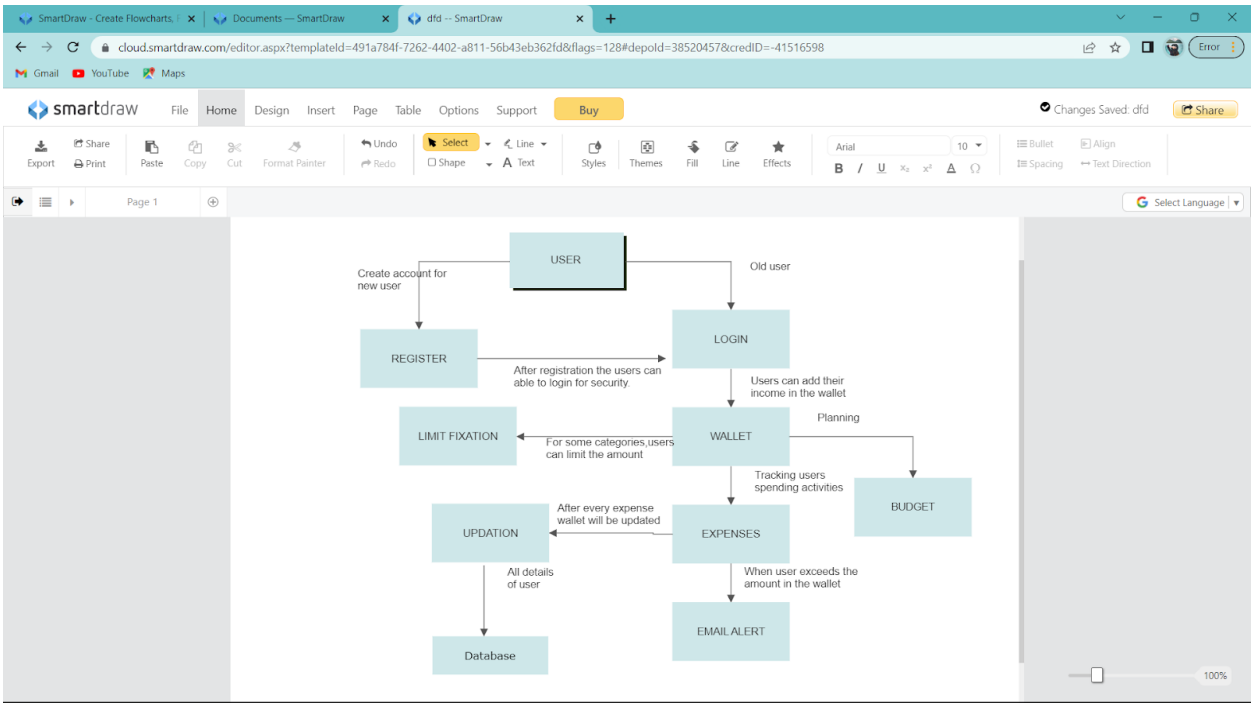
4.2 Non-Functional Requirements

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	User can tracker their expense in graphical manner Which helps to understand their expenditure in easy manner.
NFR-2	**Security**	We only store the information needed to save user from the trouble of syncing or updating financial information manually. Application also has a security feature that lets users set a password to access their account.
NFR-3	**Reliability**	The database update process can rollback to all related details in case of problem arise in updating
NFR-4	**Performance**	The application can perform well user can experience the fast while using the application
NFR-5	**Availability**	Wallet alert message will be send to user when they exceed their limit. User can set limit to some specific categories like Education, Pharmacy, Groceries etc.
NFR-6	**Scalability**	This application can able to with stand many number of users

CHAPTER-5

PROJECT DESIGN

5.1 Data Flow Diagram

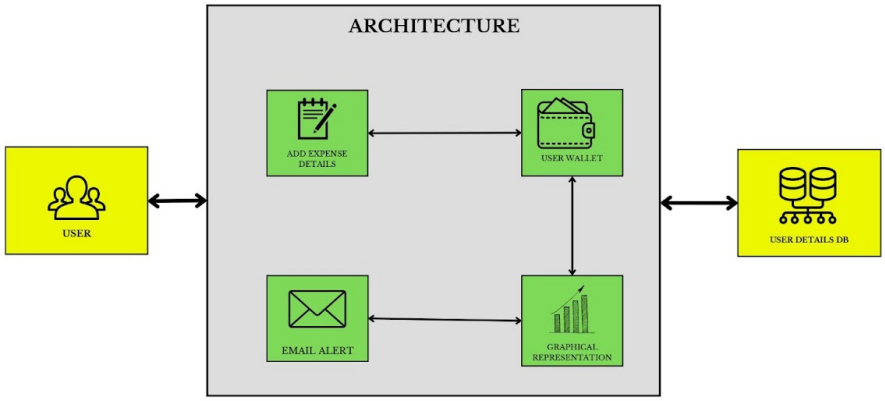


5.2 Solution & Technical Architecture

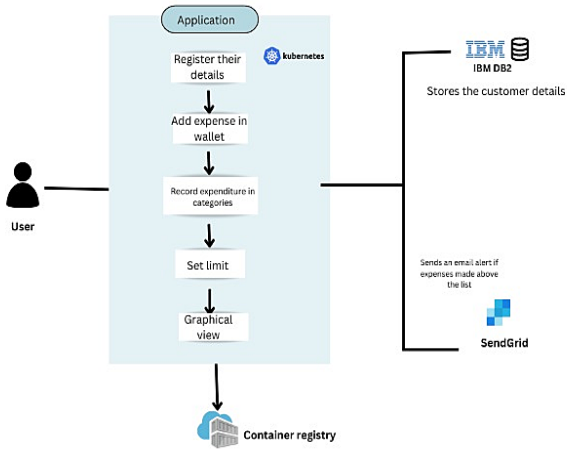
Solution Architecture

TEAM ID	PNT2022TMID49592
PROJECT	PERSONAL EXPENSE TRACKER APPLICATION

SOLUTION ARCHITECTURE



Technical Architecture



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
	Login	USN-3	As a user, I can log into the application by entering email & password	I can login by giving email and password	Low	Sprint-1
	Wallet page	USN-4	As a user, I can add my monthly income in the wallet section	I can see the amount in the wallet which was added	High	Sprint-1

	Update and graph view	USN-5	As a user, I spend the money in the wallet and the wallet will be updated periodically and can see the monthly expenses as a graph	I can able to see the updated wallet and graph format of my monthly expenses	Low	Sprint-2
	Email alert	USN-6	As a user, if i am exceeding the money in the wallet then i will be getting an alert through my email	I can get the email alert	High	Sprint-2
	Fixing limit	USN-7	As a user, I can set the limit in the wallet amount for unwanted spendings	I can set the limit in the wallet	Low	Sprint-2
Customer (Web user)	Registration	USN-8	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-2
		USN-9	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-3

	Login	USN-10	As a user, I can log into the application by entering email & password	I can login by giving email and password	Low	Sprint-3
	Wallet page	USN-11	As a user, I can add my monthly income in the wallet section	I can see the amount in the wallet which was added	High	Sprint-3
	Update and graph view	USN-12	As a user, I spend the money in the wallet and the wallet will be updated periodically and can see the monthly expenses as a graph	I can able to see the updated wallet and graph format of my monthly expenses	Low	Sprint-4
	Email alert	USN-13	As a user, if i am exceeding the money in the wallet then i will be getting an alert through my email	I can get the email alert	High	Sprint-4
	Fixing limit	USN-14	As a user, I can set the limit in the wallet amount for unwanted spendings	I can set the limit in the wallet	Low	Sprint-4

CHAPTER-6

PROJECT PLANNING AND SCHEDULING

6.1 Sprint Planning and Estimation

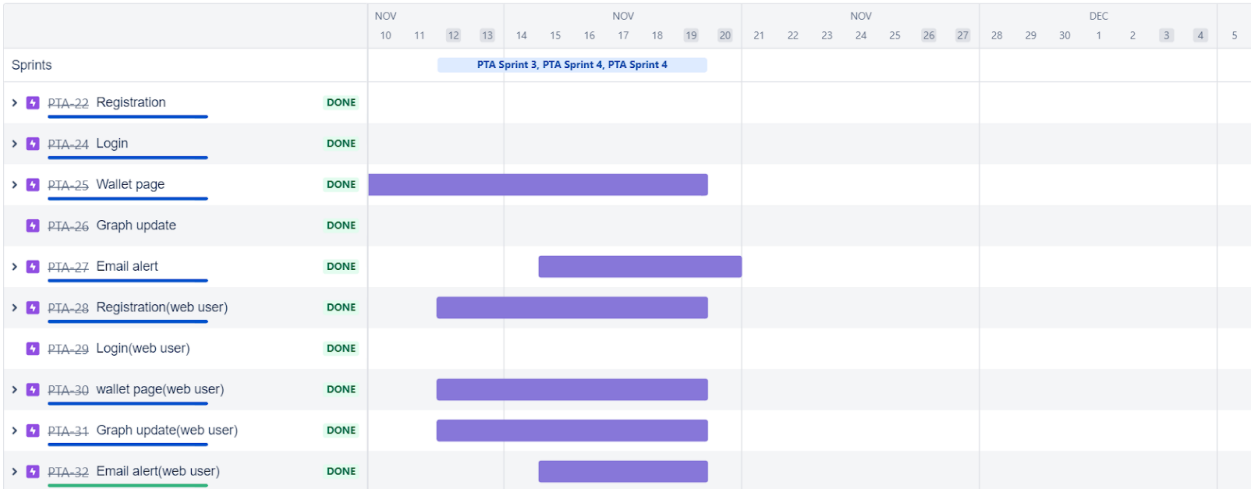
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	3	High	Sabana Fathima, Shrimathi
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	3	High	Sabana Fathima, Shrimathi
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	2	Low	Sabana Fathima, Shrimathi
Sprint-1	Wallet page	USN-4	As a user, I can add my monthly income in the wallet section	4	High	Sabana Fathima, Shrimathi
Sprint-2	Update and graph view	USN-5	As a user, I spend the money in the wallet and the wallet will be updated periodically and can see the monthly expenses as a graph	4	Low	Narmatha, Gatharine
Sprint-2	Email alert	USN-6	As a user, if i am exceeding the money in the wallet then i will be getting an alert through my email	4	High	Narmatha, Gatharine

Sprint-2		USN-7	As a user, I can set the limit in the wallet amount for unwanted spendings	4	Low	Narmatha, Gatharine
Sprint-3	Registration(web user)	USN-8	As a user, I can register for the application by entering my email, password, and confirming my password.	3	High	Sabana Fathima, Shrimathi
Sprint-3		USN-9	As a user, I will receive confirmation email once I have registered for the application	3	High	Sabana Fathima, Shrimathi
Sprint-3	Login(web user)	USN-10	As a user, I can log into the application by entering email & password	2	Low	Sabana Fathima, Shrimathi
Sprint-3	Wallet page(web user)	USN-11	As a user, I can add my monthly income in the wallet section	4	High	Sabana Fathima, Shrimathi
Sprint-4	Update and graph view (web user)	USN-12	As a user, I spend the money in the wallet and the wallet will be updated periodically and can see the monthly expenses as a graph	4	Low	Narmatha, Gatharine
Sprint-4	Email alert(web user)	USN-13	As a user, if i am exceeding the money in the wallet then i will be getting an alert through my email	4	High	Narmatha, Gatharine
Sprint-4		USN-14	As a user, I can set the limit in the wallet amount for unwanted spendings	4	Low	Narmatha, Gatharine

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	12	6 Days	24 Oct 2022	29 Oct 2022	12	29 Oct 2022
Sprint-2	12	6 Days	31 Oct 2022	05 Nov 2022	12	05 Nov 2022
Sprint-3	12	6 Days	07 Nov 2022	12 Nov 2022	12	12 Nov 2022
Sprint-4	12	6 Days	14 Nov 2022	19 Nov 2022	12	19 Nov 2022

6.3 Reports From JIRA

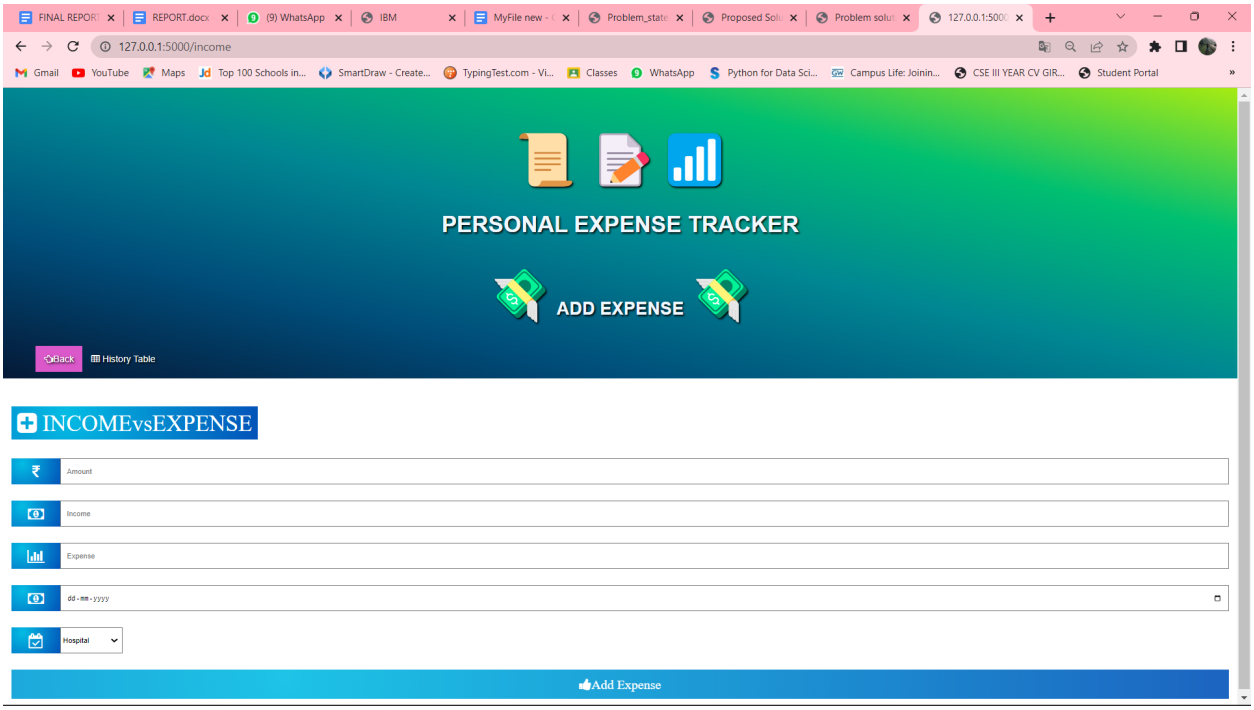


CHAPTER-7

CODING & SOLUTIONING

7.1 Feature 1

Addexpense page



User can able to add their daily expenses in this add expense page and they can also mention the kind of expenditure with the particular date.

app.py:

```
@app.route('/income', methods =['GET', 'POST'])
def income():
    if 'id' in session and request.method == 'POST' and ('amount' in request.form) and ('income' in request.form) and ('expense' in request.form) and ('expensedate' in request.form )and ('category' in request.form):
        msg = ''
```

```
        uid = session['id']
        amount = request.form['amount']
        income = request.form['income']
        expense = request.form['expense']
        expensedate = request.form['expensedate']
        category = request.form['category']

        limit = ibm_db.prepare(conn, "SELECT limit from limit WHERE uid = ?")
        ibm_db.bind_param(limit, 1, session['id'])
        ibm_db.execute(limit)

        data = ibm_db.fetch_tuple(limit)
        if data != False:
            data = data[0]
            sum = 0
            all_expenses = ibm_db.prepare(conn, "SELECT expense from expense
WHERE uid = ?")
            ibm_db.bind_param(all_expenses, 1, session['id'])
            ibm_db.execute(all_expenses)
            expense_data = ibm_db.fetch_tuple(all_expenses)

            while expense_data != False:
                sum += int(expense_data[0])
                expense_data = ibm_db.fetch_tuple(all_expenses)

        sum += int(expense)
        email = (session['email'] + '@gmail.com').lower()
```

```
        if sum >= data:
            #Limit Exceeded
            msg = Message('Limit Exceeded', sender='shrikumar13102001@gmail.com',
recipients=[email])
            msg.body = "You Have Exceeded Your Expense Limit"
            mail.send(msg)
            msg = "Exceeded Limit"
            return render_template('add.html', a = msg)

        prep_stmt = ibm_db.prepare(conn, "INSERT INTO
expense(uid,amount,income,expense,expensedate,category) VALUES (?, ?, ?, ?, ?, ?)")
        ibm_db.bind_param(prep_stmt, 1, uid)
        ibm_db.bind_param(prep_stmt, 2, amount)
        ibm_db.bind_param(prep_stmt, 3, income)
        ibm_db.bind_param(prep_stmt, 4, expense)
        ibm_db.bind_param(prep_stmt, 5, expensedate)
        ibm_db.bind_param(prep_stmt, 6, category)
        ibm_db.execute(prep_stmt)
        msg = 'You have successfully added !'
        return render_template('add.html', a = msg)
    return render_template('add.html')
```

addexpense.html:

```
<html>
<body>
<form action="{{ url_for('income')}}" method="post">
<div class="header">
```

```
<center><span style='font-size:100px;'>&#128220;</span><span style='font-
size:100px;'>&#128221;</span><span style='font-
size:100px;'>&#128246;</span></center>
  <h2> PERSONAL EXPENSE TRACKER</h2>
  <h3><span style='font-size:100px;'>&#128184;</span>ADD EXPENSE<span
style='font-size:100px;'>&#128184;</span></h3>
<div id="navbar">
  <a class="active" href="/button"><i class="fa fa-hand-o-left"></i>Back</a>
  <a href="/display"><i class="fa fa-table"></i> History Table</a>
</div>
</div>

<div class="content">

    <form style="max-width:500px;margin:auto">
        <h2><i class="fa fa-plus-square icon"> INCOMEvsEXPENSE</i></h2>
        <div class="input-container">
            <i class="fa fa-rupee icon"></i>
          <input class="input-field" type="number" placeholder="Amount"
name="amount">
        </div>

        <div class="input-container">
            <i class="fa fa-money icon"></i>
          <input class="input-field" type="number" placeholder="Income" name="income">
        </div>
```

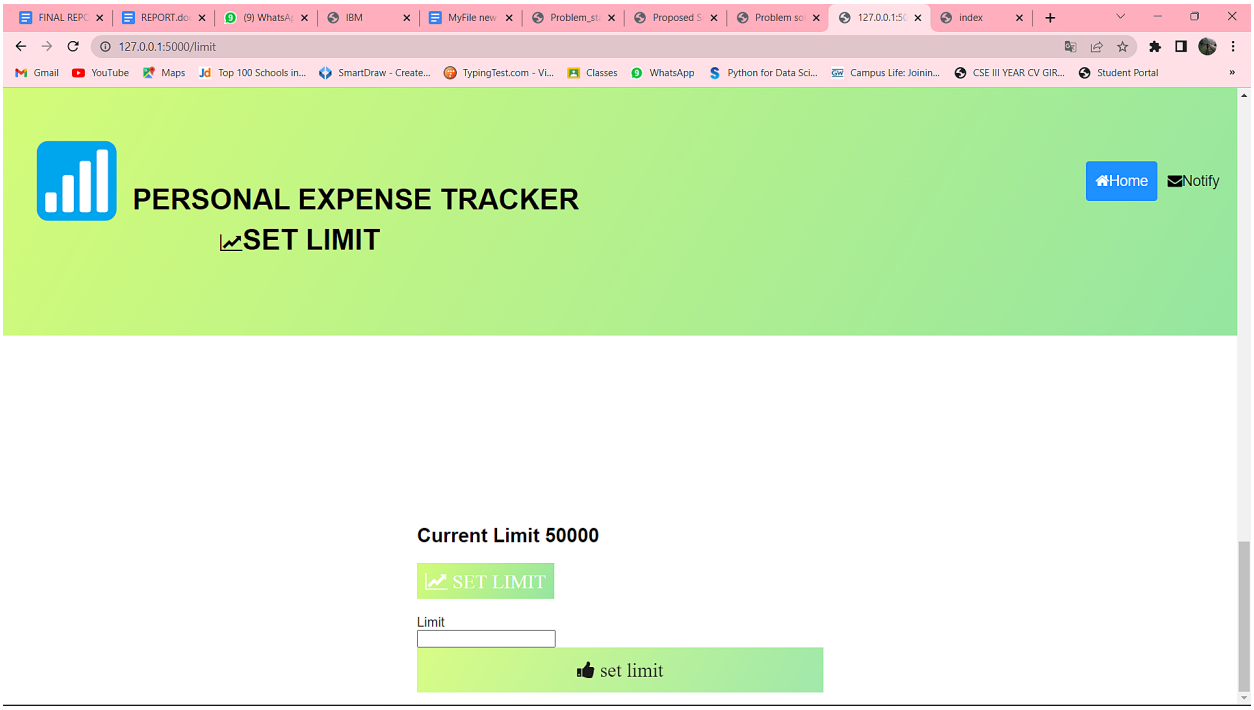
```
        <div class="input-container">
            <i class="fa fa-bar-chart-o icon"></i>
          <input class="input-field" type="number" placeholder="Expense"
name="expense">
        </div>
        <div class="input-container">
          <i class="fa fa-money icon"></i>
        <input class="input-field" type="date" placeholder="date" name="expensedate">
       </div>
        <!-- <p> Expense Details<br>(like Grocery,Education,Hospital,etc.....)</p> -->
        <div class="input-container">
            <i class="fa fa-calendar-check-o icon"></i>
              <select name="category">
                        <option value="hos">Hospital</option>
                        <option value="ent">Entertainment</option>
                        <option value="shop">Shopping</option>
                        <option value="groc">Grocery</option>
                        <option value="edu">Eduacation</option>
              </select>
            </div>

        <button type="submit" class="btn"><i class="fa fa-thumbs-up" style="font-
size:24px">Add Expense</i></button>
      </form></div>
</form>
```

</body>

</html>

7.2 Feature 2

Limit page



In this limit page, users can able to set the amount which they are going to spend in the whole month. When they exceeds the money in the wallet page, then they will be notified through email alert.

app.py:

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

def limit():

```
    if 'id' in session and 'email' in session:
        uid = session['id']
        exist = ibm_db.prepare(conn, 'SELECT uid, limit FROM limit WHERE uid = ?')
        ibm_db.bind_param(exist,1, session['id'])
        ibm_db.execute(exist)
        exist = ibm_db.fetch_tuple(exist)
        if request.method == "POST":
            print("Executing INSERT into LIMIT")
            uid = session['id']
            stmt = ""
            if exist == False:
              print("Creating New")
              stmt = ibm_db.prepare(conn, 'INSERT INTO limit (limit, uid) VALUES (?, ?)')
            else:
             print("Updating")
             stmt = ibm_db.prepare(conn, 'UPDATE limit SET \
             limit = ? \
             WHERE uid = ?')



            ibm_db.bind_param(stmt, 1,  request.form['limit'])
            ibm_db.bind_param(stmt, 2,  uid)
            ibm_db.execute(stmt)

            return render_template('limit.html', status="Success",
limit=int(request.form['limit']))
        else:
```

```
            print(exist[1])
            limit = int(exist[1])
            return render_template('limit.html', limit=limit)
    return 'Not Authed'
```

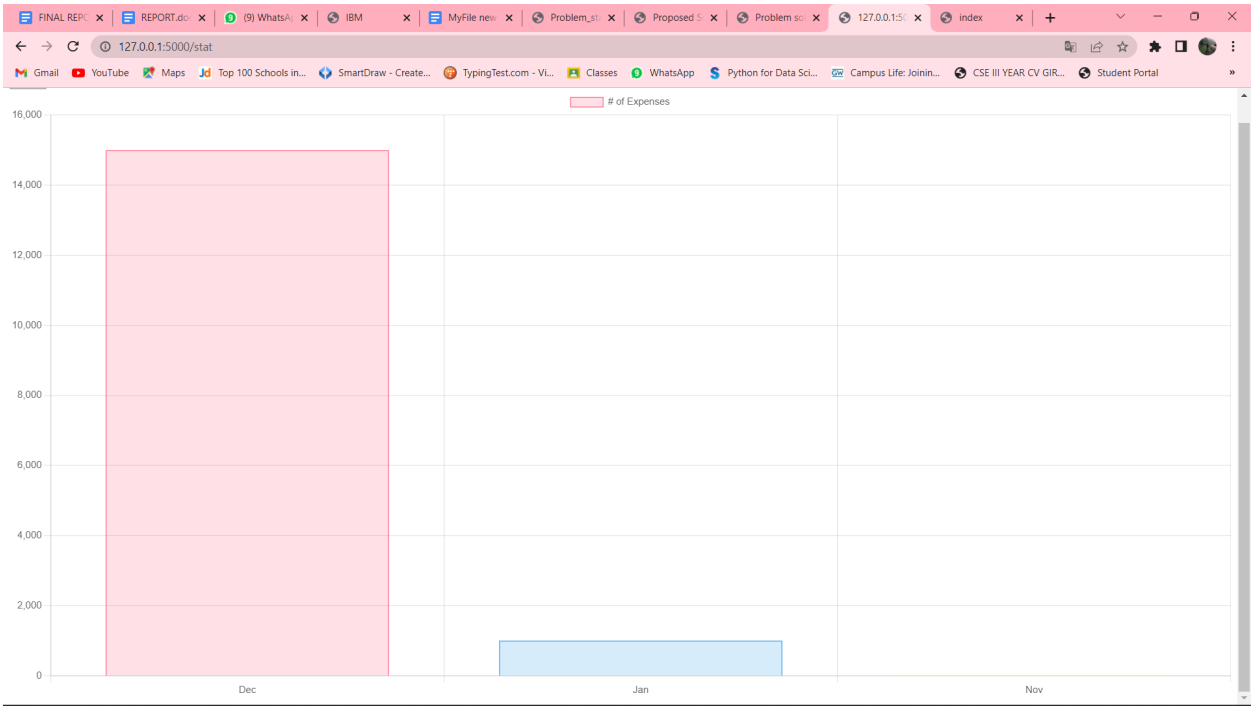
limit.html:

```
<html>
<body>
<div id="navbar">
  <a href="#default" id="logo"><span style='font-
size:100px;'>&#128246;</span>PERSONAL EXPENSE TRACKER<br><br><i
class="fa fa-line-chart" style="font-size:24px"></i>SET LIMIT</a>
  <div id="navbar-right">
    <a class="active" href="#index"><i class="fa fa-home"></i>Home</a>
    <a href="#contact"><i class="fa fa-envelope"></i>Notify</a>
  </div>
</div>

<div style="margin-top:210px;padding:15px 15px 2500px;font-size:30px">

</div>
<body>
<div class="content">

        {{status}}
    <form action="{{ url_for('limit') }}" method="post" style="max-
width:500px;margin:auto">
```

```
            <h2>Current Limit  {{limit}}</h2>
            <h2><i class="fa fa-line-chart icon"> SET LIMIT</i></h2>

        <label for="limit">Limit</label> <br>
        <input name="limit" type="number"/>
            <button type="submit" class="btn"><i class="fa fa-thumbs-up" style="font-size:24px"> set limit</i></button>
    </form>
</div>

</body>
</html>
```

7.3 Feature 3

Graphical representation page



In this page, users can able to view their monthly expenses as a graphical representation. From this view, they will find a solution of how they are spending their money?

app.py:

```
@app.route('/stat')
def stat():
    if 'id' in session:
        stmt = ibm_db.prepare(conn, 'SELECT expense, expensedate FROM expense WHERE uid = ?')
        ibm_db.bind_param(stmt,1,session['id'])
        ibm_db.execute(stmt)
        tb = ibm_db.fetch_tuple(stmt)

        months = {}
        while tb != False:
            sliced = tb[1].strftime("%b")
            if sliced in months:
                months[sliced] += tb[0]
            else:
                months[sliced] = tb[0]
            tb = ibm_db.fetch_tuple(stmt)

        return render_template('stat.html', data=months)
    return 'Not Authed Please Login'
```

stat.html:

```
<body>
```

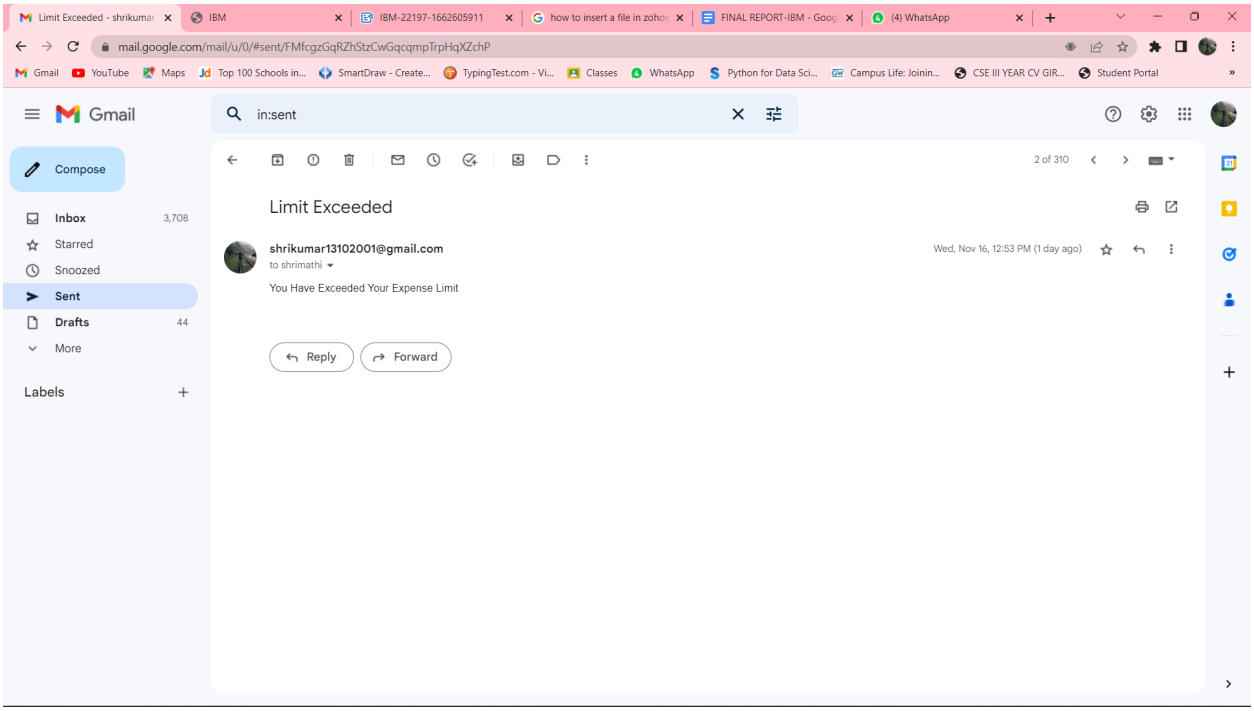
```
    <header>
      <button href="/button">Back</button>
    </header>
  </body>


<canvas id="myChart"></canvas>
<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.9.1/chart.min.js"></script>
<script>
const ctx = document.getElementById('myChart').getContext('2d');

const months = JSON.parse('{{ data | tojson | safe}}');

const monKeys = [];
const vals = [];
for(let val in months){
    monKeys.push(val);
    vals.push(months[val]);
}
// vals.reverse();
// monKeys.reverse();
console.log(months);
const myChart = new Chart(ctx, {
    type: 'bar',
    data: {
        labels: monKeys,
        datasets: [{
```

```
        label: '# of Expenses',
        data: vals,
        backgroundColor: [
            'rgba(255, 99, 132, 0.2)',
            'rgba(54, 162, 235, 0.2)',
            'rgba(255, 206, 86, 0.2)',
            'rgba(75, 192, 192, 0.2)',
            'rgba(153, 102, 255, 0.2)',
            'rgba(255, 159, 64, 0.2)'
        ],
        borderColor: [
            'rgba(255, 99, 132, 1)',
            'rgba(54, 162, 235, 1)',
            'rgba(255, 206, 86, 1)',
            'rgba(75, 192, 192, 1)',
            'rgba(153, 102, 255, 1)',
            'rgba(255, 159, 64, 1)'
        ],
        borderWidth: 1
    }]
},
options: {
    scales: {
        y: {
            beginAtZero: true
        }
    }
```

```
    }
});
</script>
```

7.4 Feature 4

Email alert



When user exceeds the money in the wallet page or when they add expenses more than the limit amount, then they will be notified throough email alert.

app.py:

```
 email = (session['email'] + '@gmail.com').lower()
        if sum >= data:
            #Limit Exceeded
            msg = Message('Limit Exceeded', sender='shrikumar13102001@gmail.com', recipients=[email])
```

```
msg.body = "You Have Exceeded Your Expense Limit"
mail.send(msg)
msg = "Exceeded Limit"
return render_template('add.html', a = msg)
```

7.5 Feature 5

Database Schema

Table name: REGI

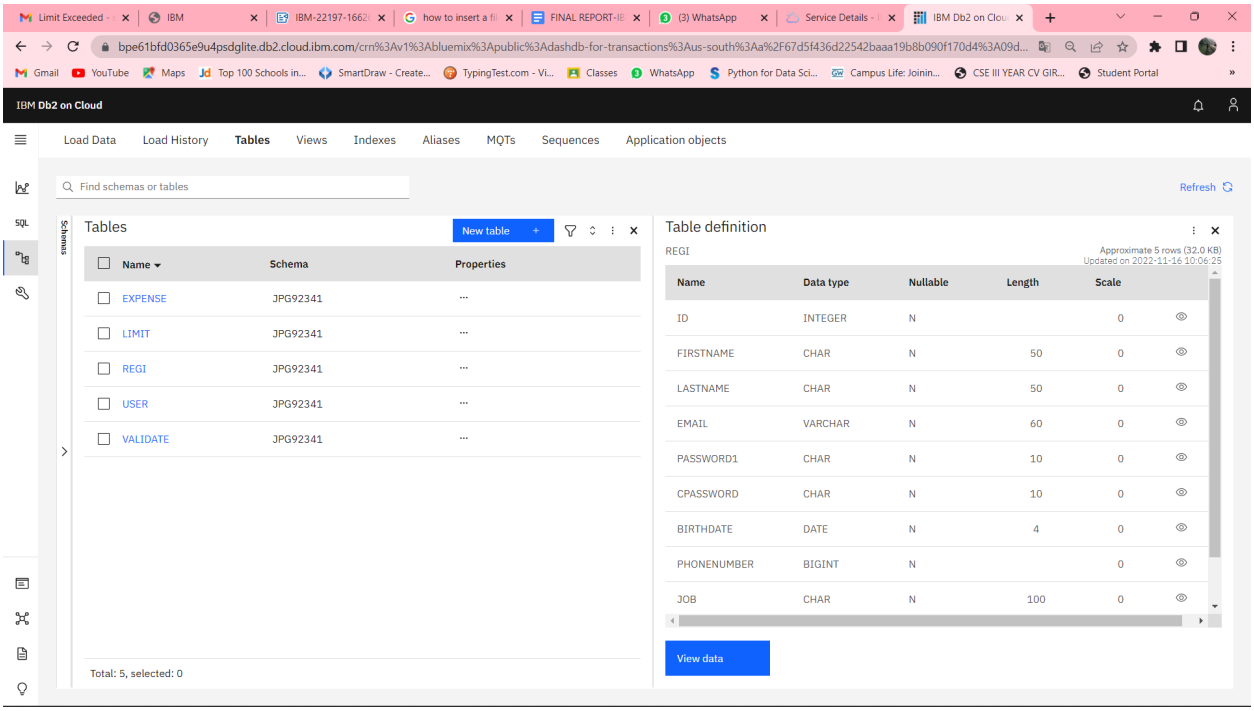


Table name: EXPENSE

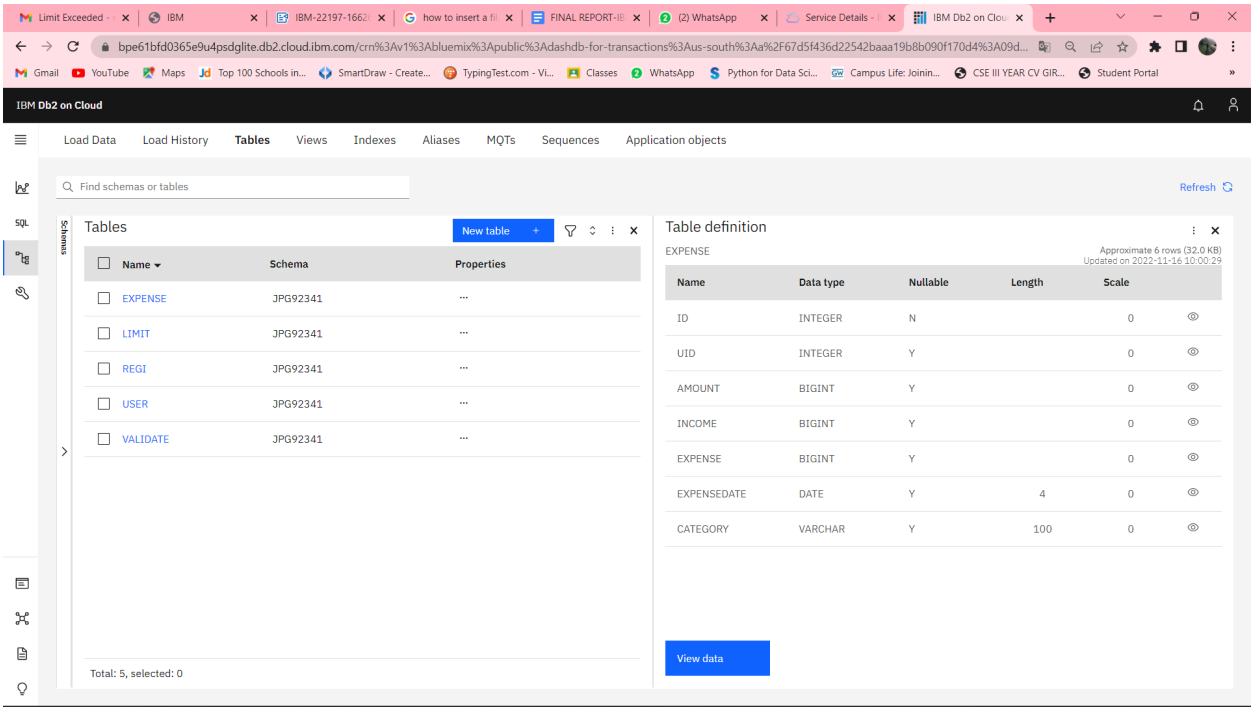


Table name: LIMIT

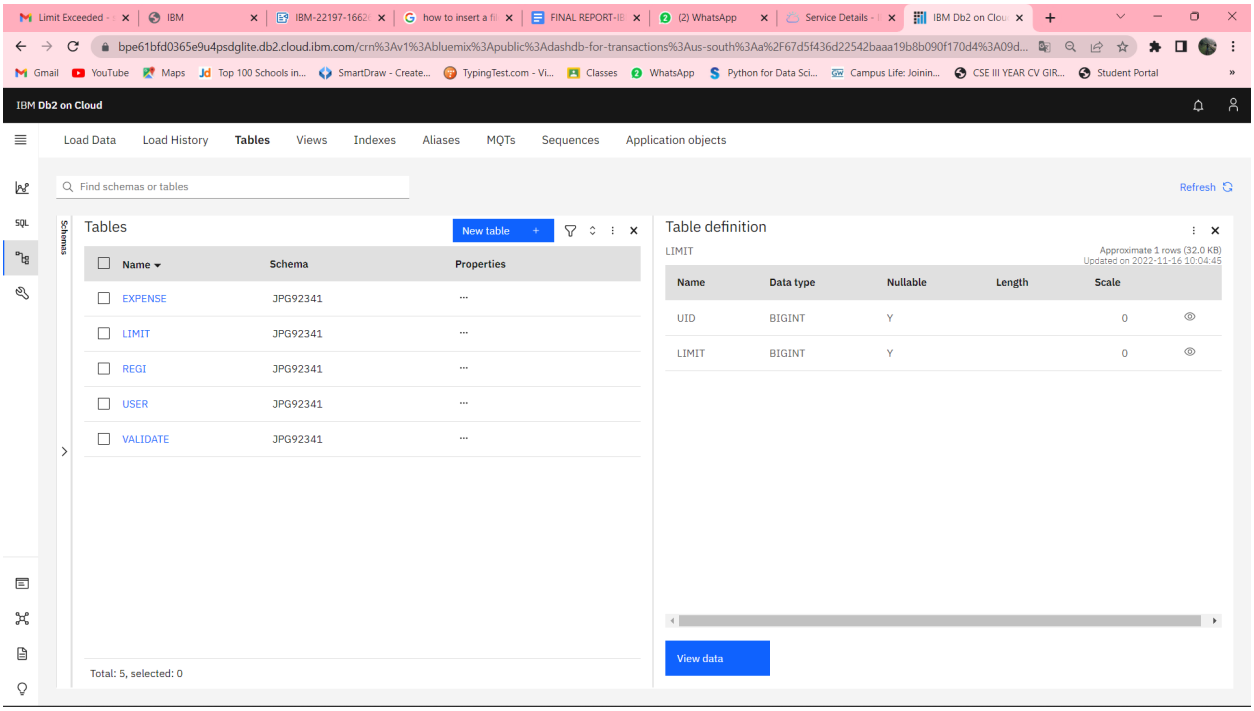
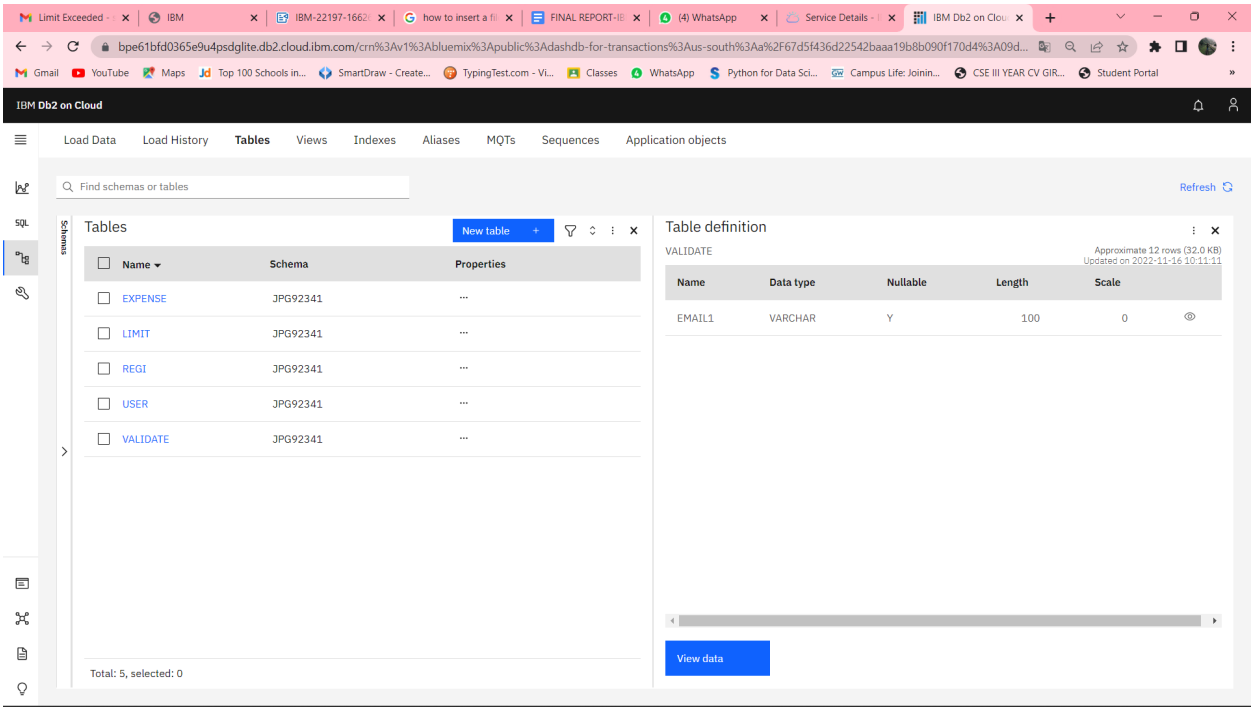


Table name: VALIDATE



CHAPTER-8

TESTING

8.1 Test Cases

				Date	17-Nov-22			
				Team ID	PNT2022TMID49592			
				Project Name	Project - Personal expense tracke			
				Maximum Marks	4 marks			
Test case ID	**Feature Type**	**Componen t**	**Test Scenario**	**Pre-Requisite**	**Steps To Execute**	**Test Data**	**Expected Result**	**Actual Result**
IndexPage_TC_OO 1	Functional	Home Page	Verify user is able to see the Login/Register when user click menu option		1.Enter URL and click go 2.Click on menu option 3.Verify Login/Register is displayed or not	https://localhost:5000	Login/Register should display	Working a expectec
LoginPage_TC_OO 2	UI	Login Page	Verify the UI elements in Login page		1.Enter URL and click go 2.Verify login with below UI elements: a.email text box b.password text box c.Login button d.Don't have an account? Creat Your Account	https://locall.host/5000/	Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.Don't have an account? Creat Your Account	Working a expectec
LoginPage_TC_OO 3	Functional	Login page	Verify user is able to login into application with Valid credentials		1.Enter URL and click go 2.Enter Valid email in Email text box 3.Enter valid password in password text box 4.Click on login button	email: retest12@gmail.com password:retest	User should navigate to user profile page	Working a expectec
LoginPage_TC_OO 4	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL and click go 2.Enter Valid email in Email text box 3.Enter valid password in password text box	email:mathi@gmail.com password:mathi	Application should show 'Incorrect email or password ' validation message.	Working a expectec
LoginPage_TC_OO 5	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL and click go 2.Enter Valid email in Email text box 3.Enter valid password in password text box	email: shrimathi@gmail.com password:mathi1234	Application should show 'Incorrect email or password ' validation message.	Working a expectec
					1.Enter URL and click go 2.Verify register with below UI	URL: firstname:test123	Application should show below UI elements:	

Shopenzer Testcases | Testscearnios | Explore

				Date	17-Nov-22			
				Team ID	PNT2022TMID49592			
				Project Name	Project - Personal expense tracke			
				Maximum Marks	4 marks			
Test case ID	**Feature Type**	**Componen t**	**Test Scenario**	**Pre-Requisite**	**Steps To Execute**	**Test Data**	**Expected Result**	**Actual Result**
RegisterPage_TC_0 06	UI	Register page	Verify user is able to register into application with Valid credentials		1.Enter URL and click go 2.Verify register with below UI elements: a.firstname text box b.lastname text box c.email text box d.password text box e.confirm password box f.date of birth box g.phone number text box h.job text box i.income in rs text box j. register button	URL: firstname:test123 lastname:test1 email:retest12@gmail.com password:retest confirm password:retest date of birth: 16-09-2002 phone numeber: 9361132795 job:Teacher income:70000	Application should show below UI elements: 1.Enter URL and click go 2.Verify register with below UI elements: a.firstname text box b.lastname text box c.email text box d.password text box e.confirm password box f.date of birth box g.phone number text box h.job text box i.income in rs text box j. register button with blue color	Working a expectec
VerifyPage_TC_00 7	UI	Register page	Verify user is able to submit their emailid		1.verify the page with below UI elements: a.email text box b.submit button	email:retest12@gmail.com	Application should show below UI elements: a.email text box b.submit button with white color	Working a expectec
OTPPage_TC_008	UI	Verify page	Verify user is able to submit OTP		1.verify the page with below UI elements: a.otp text box b.continue button	otp:	Application should show below UI elements: a.OTP text box b.Continue button with blue color	Working a expectec
OTPPage_TC_009	Functional	Verify page	Verify user submitted their vaild OTP		1.Enter URL and click go 2.Enter Valid OTP in OTP text box	OTP:	User should navigate to login page	Working a expectec
UserProfilePage_T	Functional	Profile	Verify user is able to see the Home/Expense when user enters		1.Enter URL and click go 2.Verify Home/Expense is displayed		Home/Expense should display	Working a

Shopenzer Testcases | Testscearnios | Explore

				Date	17-Nov-22			
				Team ID	PNT2022TMID49592			
				Project Name	Project - Personal expense tracke			
				Maximum Marks	4 marks			
Test case ID	**Feature Type**	**Component**	**Test Scenario**	**Pre-Requisite**	**Steps To Execute**	**Test Data**	**Expected Result**	**Actual Result**
ButtonPage_TC_011	Functional	Button page	Verify user is able to see the add Expense/view graph result/set limit when user enters the button page		1.Enter URL and click go 2.Verify add Expense/view graph result/set limit button is displayed or not		add Expense/view graph result/set limit should display	Working a expectec
ButtonPage_TC_012	UI	Button page	Verify user able to click on the buttons to enter their requried page		1.Enter the URL and click go 2.if user wants to add their expense means click on add expense button 3.if user wants to view the graph means click on view graph result button 4.if user wants to set their limit means click on set limit button		User should navigate to add expense page/graph page/set limit page based on user selection	Working a expectec
AddExpensePage_TC_013	UI	Expense page	Verify user is able to see the IncomevsExpense form		1.Enter the URL and click go 2.Verify register with below UI elements: a.amount text box b.income text box c.expense text box d. date box e. category box f.add expense button		IncomevsExpense form should display with below UI elements: a.Enter the amount in text box b.Enter the income in text box c.Enter the expense in text box d.Enter the date in box e.Enter the category in box f.add expense button with blue color	Working a expectec
AddExpensePage_TC_014	Functional	Expense page	Verify user able to store the data in database		1.Enter the URL and click go 2.Verify register with below UI elements: a.amount text box b.income text box c.expense text box	amount:2000 income:1000 expense:1000 date:2022-01-01 category:hospital	Application should show 'Successfully stored '	Working a expectec

Shopenzer Testcases | Testscearnios | Explore

				Date	17-Nov-22			
				Team ID	PNT2022TMID49592			
				Project Name	Project - Personal expense tracke			
				Maximum Marks	4 marks			
Test case ID	**Feature Type**	**Component**	**Test Scenario**	**Pre-Requisite**	**Steps To Execute**	**Test Data**	**Expected Result**	**Actual Result**
GraphPage_TC_015	Functional	Graph page	verify the graph change based on the user data get form the IncomevsExpense form		1.Go to the Button page 2.click on view graph result button		Application should show the changes in the graph	Working a expectec
SetLimitPage_TC_016	UI	Limit page	Verify user is able to see the set limit form		1.Go to the Button page 2.click on set limit button 3.verify the page with below UI elements: a.current limit b.limit box c.set limit button		Application should show the'current limit with set limit form' with the below UI elements: a.Enter the limit in box b.sete limit button with green color	Working a expectec
SetLimitPage_TC_017	Functional	Limit page	Get the limit value form the user		1.Go to the Button page 2.click on set limit button 3.verify the page with below UI elements: a.current limit b.limit box c.set limit button	limit:30000	Application should show ' limit set successfully'	Working a expectec
SetLimitPage_TC_018	Functional	Limit page	verify the user limit with the IncomevsExpense form data and alert the user with alert mail		Based on the limit form the set limit form and IncomevsExpense form it compare the Expense and Limit if the expense increase above limit it gives alert mail to user		Application should send the alert mail to user mail	Working a expectec
LogoutPage_TC_019	Functional	logout page	verify user successfully logout from their profile page		click on the logout button		click on the logout button to logout from the account	Working a expectec

Shopenzer Testcases | Testscearnios | Explore

8.2 User Acceptance Testing

Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

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	10	4	2	8	15
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	9	2	4	11	20
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	0	1	8
Totals	22	14	11	22	51

Test Case Analysis

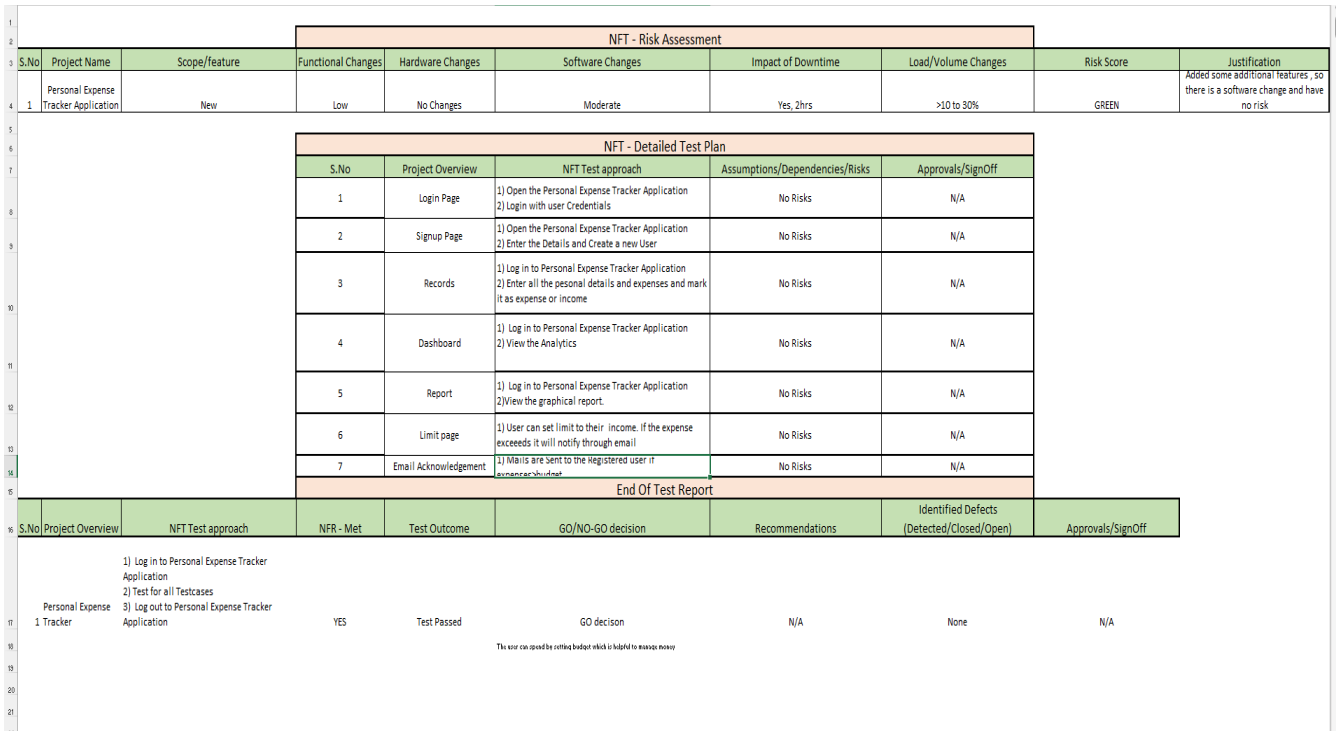
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Interface	7	0	0	7
Homepage	23	0	0	23
Signup	8	0	0	8
Login	33	0	0	33
Limit	3	0	0	3
Logout	2	0	0	2
Final Report Output	4	0	0	4
Version Control	2	0	0	2

CHAPTER-9

RESULTS

9.1 Performance Metrics



NFT - Risk Assessment

S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes	Risk Score	Justification
1	Personal Expense Tracker Application	New	Low	No Changes	Moderate	Yes, 2hrs	>10 to 30%	GREEN	Added some additional features , so there is a software change and have no risk

NFT - Detailed Test Plan

S.No	Project Overview	NFT Test approach	Assumptions/Dependencies/Risks	Approvals/SignOff
1	Login Page	1) Open the Personal Expense Tracker Application 2) Login with user Credentials	No Risks	N/A
2	Signup Page	1) Open the Personal Expense Tracker Application 2) Enter the Details and Create a new User	No Risks	N/A
3	Records	1) Log in to Personal Expense Tracker Application 2) Enter all the pesonal details and expenses and mark it as expense or income	No Risks	N/A
4	Dashboard	1) Log in to Personal Expense Tracker Application 2) View the Analytics	No Risks	N/A
5	Report	1) Log in to Personal Expense Tracker Application 2)View the graphical report.	No Risks	N/A
6	Limit page	1) User can set limit to their income. If the expense exceeeds it will notify through email	No Risks	N/A
7	Email Acknowledgement	1) Mails are Sent to the Registered user if expenses>budget	No Risks	N/A

End Of Test Report

S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff
1	Personal Expense Tracker	1) Log in to Personal Expense Tracker Application 2) Test for all Testcases 3) Log out to Personal Expense Tracker Application	YES	Test Passed	GO decison	N/A	None	N/A

The user can spend be setting budget which is helpful to manage money

CHAPTER-10

ADVANTAGES AND DISADVANTAGES

Advantages:

- Through this app, the user can create a budget and track their income and spending.
- The user can categorize their payments into different types such as groceries,education,etc.
- It reduces time consuming and accurate.
- The user can able to see their history of expenditure.
- To easily understand about the spending,the user can view through graph.
- In this app,the user can set limit to manage their expense,if the expense exceeds the limit it will notify the user.
- In this way,the user can easily realize where their bulk of money goes.

Disadvantages:

- This app is not fully secured,the app may get hacked.

CHAPTER-11

CONCLUSION

In conclusion, developing a personal budget and tracking all expenses and spending is acrucial aspect of personal finances. This budgeting system provides management with a means of controlling its activities and of monitoring actual performance and comparing it to budget goals.The importance of actually seeing the user spending on their budget sheet was enlightening.

CHAPTER-12

FUTURE SCOPE

Money management is the highly specialised wing of management that focuses on efficient financial planning for daily use. Unlike the traditional approach that was merely restricted to fundraising, in the modern corporate world, the budget planning is responsible for the spending, strategic planning, direction, and control of financial undertakings in human life.

In future,it is sure that it must be a life changeable one for human.They can easily manage their budget. Even the illeterate people can able yo use use this app.It is user friendly.So every user spend their money for essential needs.Through this app,the user can able to realize where their money goes.

CHAPTER-13

APPENDIX

Source code:

app.py:

```
from flask import Flask, render_template, request, redirect, url_for,
session
import ibm_db
import re
from random import *
from clarifai_grpc.grpc.api import service_pb2, resources_pb2
from clarifai_grpc.grpc.api.status import status_code_pb2
from clarifai_grpc.channel.clarifai_channel import ClarifaiChannel
from clarifai_grpc.grpc.api import service_pb2_grpc
from flask_mail import Mail, Message
app = Flask(__name__)
mail = Mail(app) # instantiate the mail class

# configuration of mail
app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'shrikumar13102001@gmail.com'
app.config['MAIL_PASSWORD'] = 'mlzuhzieegwenrhc'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
mail = Mail(app)
otp = randint(000000,999999)
from clarifai_setup import (
    DOG_IMAGE_URL,
    GENERAL_MODEL_ID,
```

```
    NON_EXISTING_IMAGE_URL,
    RED_TRUCK_IMAGE_FILE_PATH,
    both_channels,
    metadata,
    raise_on_failure,
    post_model_outputs_and_maybe_allow_retries,
)

def test_predict_image_url():
    stub = service_pb2_grpc.V2Stub(ClarifaiChannel.get_grpc_channel())

    req = service_pb2.PostModelOutputsRequest(
        model_id=GENERAL_MODEL_ID,
        inputs=[
            resources_pb2.Input(

data=resources_pb2.Data(image=resources_pb2.Image(url=DOG_IMAGE_URL))
            )
        ],
    )

    response = post_model_outputs_and_maybe_allow_retries(stub, req,
metadata=metadata())
    print(response)
    raise_on_failure(response)

    assert len(response.outputs[0].data.concepts) > 0
app.secret_key = 'a'
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SEC
URITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=jpg92341;PWD=5H
WT2lmD7POkuLnv",'','')
@app.route("/")
@app.route('/home')
def home():
   return render_template('index.html')
```

```
@app.route('/login', methods =['GET', 'POST'])
def login():
    global userid
    msg = ''
    if request.method == 'POST'and 'email' in request.form and 'password1'
in request.form:
        email = request.form['email']
        password1 = request.form['password1']
        stmt = ibm_db.prepare(conn,'SELECT * FROM regi WHERE email = ? AND
password1 = ?')
        ibm_db.bind_param(stmt,1,email)
        ibm_db.bind_param(stmt,2,password1)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_tuple(stmt)
        if account:
            session['id'] = account[0]
            userid =  account[0]
            session['email'] = account[1]
            msg = 'Logged in successfully !'
            return redirect(url_for('userprofile'))
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)
@app.route('/button')
def button():
    return render_template('button.html')

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

    if 'id' in session and 'email' in session:
        uid = session['id']
        exist = ibm_db.prepare(conn, 'SELECT uid, limit FROM limit WHERE
uid = ?')
        ibm_db.bind_param(exist,1, session['id'])
        ibm_db.execute(exist)
```

```
        exist = ibm_db.fetch_tuple(exist)
        if request.method == "POST":
              print("Executing INSERT into LIMIT")
              uid = session['id']
              stmt = ""
              if exist == False:
                 print("Creating New")
                 stmt = ibm_db.prepare(conn, 'INSERT INTO limit (limit,
uid) VALUES (?, ?)')
              else:
                print("Updating")
                stmt = ibm_db.prepare(conn, 'UPDATE limit SET \
                limit = ? \
                WHERE uid = ?')


              ibm_db.bind_param(stmt, 1,  request.form['limit'])
              ibm_db.bind_param(stmt, 2,  uid)
              ibm_db.execute(stmt)

              return render_template('limit.html', status="Success",
limit=int(request.form['limit']))
        else:
               if exist == False:
                   return render_template('limit.html', limit=0)
               else:
                      return render_template('limit.html',
limit=int(exist[1]))
    return 'Not Authed'

@app.route('/stat')
def stat():
    if 'id' in session:
        stmt = ibm_db.prepare(conn, 'SELECT expense, expensedate FROM
expense WHERE uid = ?')
        ibm_db.bind_param(stmt,1,session['id'])
```

```
        ibm_db.execute(stmt)
        tb = ibm_db.fetch_tuple(stmt)

        months = {}
        while tb != False:
            sliced = tb[1].strftime("%b")
            if sliced in months:
                months[sliced] += tb[0]
            else:
                months[sliced] = tb[0]
            tb = ibm_db.fetch_tuple(stmt)

        return render_template('stat.html', data=months)
    return 'Not Authed Please Login'
@app.route('/display')
def display():
    if 'id' in session :
        stmt = ibm_db.prepare(conn, 'SELECT amount, income, expense,
expensedate, category FROM expense WHERE uid = ?')
        ibm_db.bind_param(stmt,1,session['id'])
        ibm_db.execute(stmt)

        tb = ibm_db.fetch_assoc(stmt)
        data = []
        while tb != False:
            data.append(tb)
            tb = ibm_db.fetch_assoc(stmt)
        print(data)
        return render_template('display.html',data=data)

    return 'Not Authed'

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

```
def income():
    if 'id' in session and request.method == 'POST' and ('amount' in
request.form) and ('income' in request.form) and ('expense' in
request.form) and ('expensedate' in request.form )and ('category' in
request.form):
            msg = ''
            uid = session['id']
            amount = request.form['amount']
            income = request.form['income']
            expense = request.form['expense']
            expensedate = request.form['expensedate']
            category = request.form['category']

            limit = ibm_db.prepare(conn, "SELECT limit from limit WHERE
uid = ?")
            ibm_db.bind_param(limit, 1, session['id'])
            ibm_db.execute(limit)

            data = ibm_db.fetch_tuple(limit)
            if data != False:
                data = data[0]
                sum = 0
                all_expenses = ibm_db.prepare(conn, "SELECT expense from
expense WHERE uid = ?")
                ibm_db.bind_param(all_expenses, 1, session['id'])
                ibm_db.execute(all_expenses)
                expense_data = ibm_db.fetch_tuple(all_expenses)

                while expense_data != False:
                    sum += int(expense_data[0])
                    expense_data = ibm_db.fetch_tuple(all_expenses)

            sum += int(expense)
            email = (session['email'] + '@gmail.com').lower()
            if sum >= data:
                #Limit Exceeded
```

```
                msg = Message('Limit Exceeded',
sender='shrikumar13102001@gmail.com', recipients=[email])
                msg.body = "You Have Exceeded Your Expense Limit"
                mail.send(msg)
                msg = "Exceeded Limit"
                return render_template('add.html', a = msg)

            prep_stmt = ibm_db.prepare(conn, "INSERT INTO
expense(uid,amount,income,expense,expensedate,category) VALUES (?, ?, ?,
?, ?, ?)")
            ibm_db.bind_param(prep_stmt, 1, uid)
            ibm_db.bind_param(prep_stmt, 2, amount)
            ibm_db.bind_param(prep_stmt, 3, income)
            ibm_db.bind_param(prep_stmt, 4, expense)
            ibm_db.bind_param(prep_stmt, 5, expensedate)
            ibm_db.bind_param(prep_stmt, 6, category)
            ibm_db.execute(prep_stmt)
            msg = 'You have successfully added !'
            return render_template('add.html', a = msg)
    return render_template('add.html')

@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST':
        firstname = request.form['firstname']
        lastname = request.form['lastname']
        email = request.form['email']
        password1 = request.form['password1']
        cpassword = request.form['cpassword']
        birthdate = request.form['birthdate']
        phonenumber = request.form['phonenumber']
        job = request.form['job']
        income = request.form['income']
        sql = "SELECT * FROM regi WHERE email = ? "
        stmt = ibm_db.prepare(conn,sql)
```

```
        ibm_db.bind_param(stmt,1,email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', firstname):
            msg = 'firstname must contain only characters and numbers !'
        elif not email or not firstname or not password1:
            msg = 'Please fill out the form !'
        else:
            insert_sql = "INSERT INTO
regi(firstname,lastname,email,password1,cpassword,birthdate,phonenumber,job
,income) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)"
            stmt = ibm_db.prepare(conn,insert_sql)
            ibm_db.bind_param(stmt, 1, firstname)
            ibm_db.bind_param(stmt, 2, lastname)
            ibm_db.bind_param(stmt, 3, email)
            ibm_db.bind_param(stmt, 4, password1)
            ibm_db.bind_param(stmt, 5, cpassword)
            ibm_db.bind_param(stmt, 6, birthdate)
            ibm_db.bind_param(stmt, 7, phonenumber)
            ibm_db.bind_param(stmt, 8, job)
            ibm_db.bind_param(stmt, 9, income)
            ibm_db.execute(stmt)
            msg = 'You have successfully registered !'
            return render_template('email.html')
    elif request.method == 'POST':
        msg = 'Please fill out the form !'

    return render_template('register.html', msg = msg)
@app.route('/userprofile', methods =['GET', 'POST'])
def userprofile():
    if 'id' in session:
```

```
        stmt = ibm_db.prepare(conn, 'SELECT firstname,email,income FROM
regi WHERE id = ?')
        ibm_db.bind_param(stmt,1,session['id'])
        ibm_db.execute(stmt)
        tb = ibm_db.fetch_assoc(stmt)
        data = []
        while tb != False:
            data.append(tb)
            tb = ibm_db.fetch_assoc(stmt)
        print(data)
        return render_template('userprofile.html',data=data)
    return render_template('userprofile.html')
@app.route('/verify', methods=['GET', 'POST'])
def verify():
    if request.method == 'POST':
        email1 = request.form['email1']
        sql = "SELECT * FROM validate WHERE email1 = ? "
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,email1)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account already exists !'
        else:
           insert_sql = "INSERT INTO validate VALUES (?)"
           stmt = ibm_db.prepare(conn,insert_sql)
           ibm_db.bind_param(stmt, 1, email1)
           ibm_db.execute(stmt)
           msg = Message('Hello',sender
='shrikumar13102001@gmail.com',recipients = [email1])
           msg.body = str(otp)
           mail.send(msg)
           return render_template('verify.html')
    return render_template('verify.html')
@app.route('/validate',methods=['GET', 'POST'])
```

```
def validate():
 user_otp = request.form['otp']
 if otp == int(user_otp):
    return render_template('index.html')
 return render_template('index.html')

@app.route('/logout', methods=['GET'])
def logout():
    if 'email' in session:
        del session['email']
        del session['id']
        return redirect('/')
    return redirect('/')


if __name__ == '__main__':
    app.run(debug = True, host='0.0.0.0', port = 8080)
```

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-22197-1659807715.git

DEMO LINK:

https://drive.google.com/drive/folders/1B5FRdTZ6O3lErbkTtFbgIxIIJ4nzb6Up?usp=sharing