

## **Project Report and Documentation**

# **PERSONAL EXPENSE TRACKER APPLICATION**

## **1. INTRODUCTION**

### **a. Project Overview**

Personal expense tracker entails all financial decisions and activities. It makes our life easier by helping us to manage our expenses efficiently. Using this application, We can know where our money goes and can ensure that our money is used wisely. It helps you track all transactions like bills, refunds, payrolls, receipts, taxes, etc., on a daily, weekly, and monthly basis.

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.

### **b. Purpose**

This app helps to keep an accurate record of your money inflow and outflow. It not only helps with budgeting and accounting but also give helpful insights about money management.

- ✓ It helps achieve business goals
- ✓ Has an easy user interface for interaction
- ✓ Generated detailed reports to give insights about profits, loss, budgets, income, balance sheets, etc.
- ✓ Helps having a smart budget plan
- ✓ Has enhanced AI based bot for answering all queries and addressing user needs
- ✓ Avoiding data loss
- ✓ Alert sent to user if spent more than the threshold level

## 2. LITERATURE SURVEY

### a. Existing problem

- Lack of time for proper noting down of expenses
- Data entry errors
- Improper management of money
- Lack of visibility of expenses
- Lost receipts
- Internet issues(In case of online web apps)
- Security
- Unavailable balance due to less saving of money

### b. References

**i. eExpense: A Smart Approach to Track Everyday Expense** by *Shahed Anzarus Sabab, Sadman Saumik Islam, Md. Jewel Rana, Monir Hossain*

Methodology: eExpense can manage daily expense much faster than any other traditional app in the market which takes manual input. The app performs poorly if the input image from the receipt includes a lot of noises

**ii. Time Scheduling and Finance Management: University Student Survival Kit** by *J.L. Yeo, P.S. JosephNg, K.A. Alezabi, H.C. Eaw, K.Y. Phan*

Methodology: Explanatory sequential design, Mixed Mode, Comparative Analysis

**iii. Expense Manager Application** by *Velmurugan A, Albert Mayan J, Niranjana P, Richard Francis*

Methodology: A mobile app that keeps track of all of your daily transactions, keeps track of your money lent or borrowed ,suggests you with the most effective investment options, offers your discounts in popular categories , view exchange and to read latest authenticated financial news.

**iv. Mobile Bookkeeper: Personal Financial Management Application with Receipt Scanner Using Optical Character Recognition** by *Manuel Garcia, Julius Claour*

Methodology: Financial enthusiasts can just scan the receipt using the inbuilt camera of any smartphone and details will be automatically transcribed using Optical Character Recognition (OCR).

### c. Problem Statement Definition

It takes a lot of time for users to manually keep track of their day to day expenses. It sometimes frustrates them and they fail to regularly update and make note of them. To avoid this, an expense tracker application can be used to regularly track and maintain the expenses and also solve the budget planning problems.

## 3. IDEATION & PROPOSED SOLUTION

### a. Empathy Map Canvas



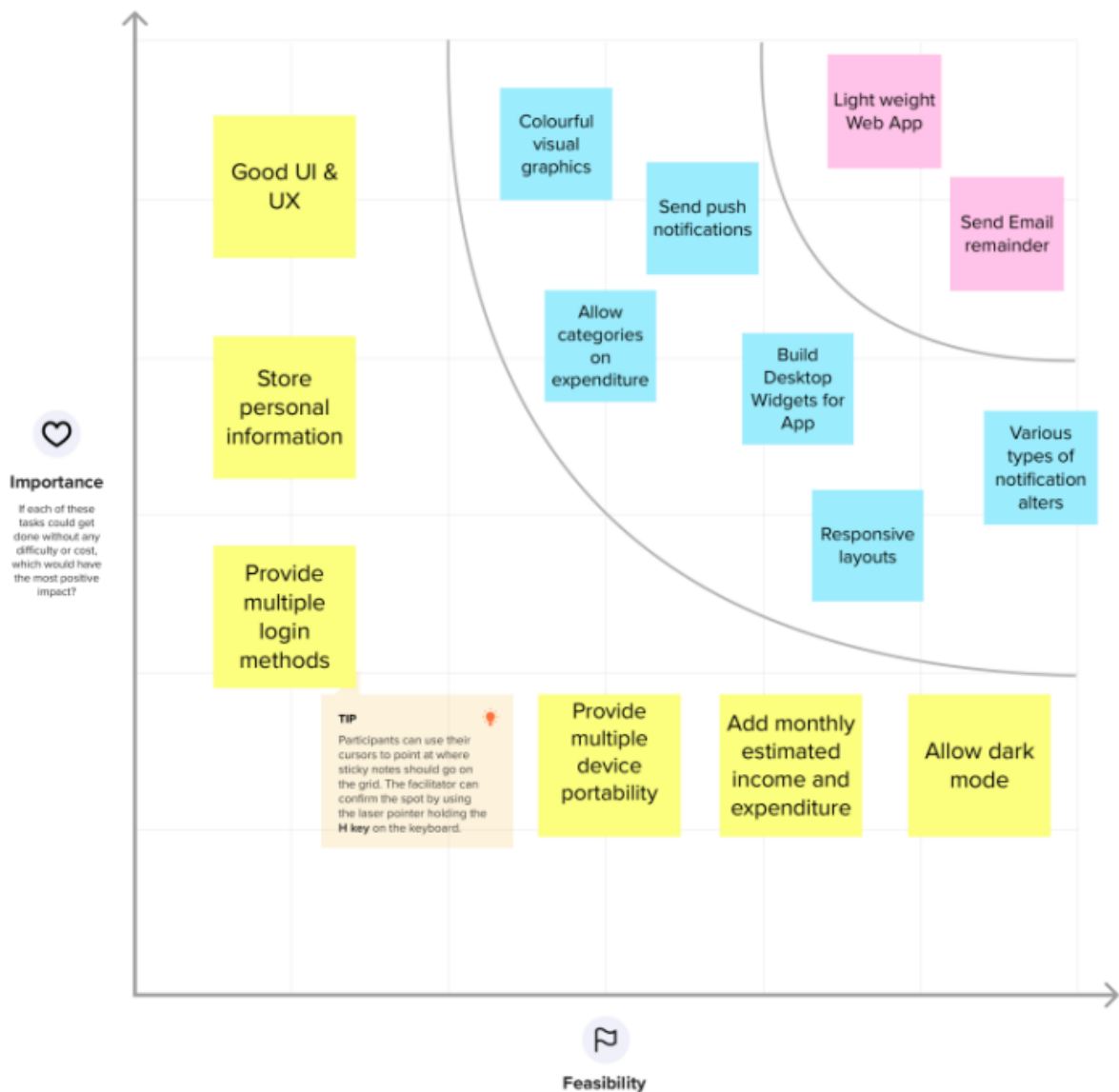
## b. Ideation & Brainstorming

### Ideation



### Brainstorming





c. Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	<ul style="list-style-type: none"><li>• Lack of proper planning of our income</li><li>• User has to keep a expense details in a diary or in a computer</li><li>• All the calculations needs to be manually done by the user</li><li>• Hard to keep track of daily expenses</li></ul>
2.	Idea / Solution description	<ul style="list-style-type: none"><li>• To know where the money is going and spend only on priorities</li><li>• To save money for pre-defined expenses</li><li>• To track the performance of investments</li><li>• To track day-to-day expense</li></ul>
3.	Novelty / Uniqueness	<ul style="list-style-type: none"><li>• To notify you of upcoming bill due dates</li><li>• To generate profit and loss report</li><li>• Categorize expenses</li><li>• Smart budget planning</li></ul>
4.	Social Impact / Customer Satisfaction	<ul style="list-style-type: none"><li>• This application helps user to be financially responsible</li><li>• Free of cost</li><li>• Designed to be dynamic to produce prediction</li><li>• Can reduce burden of manual calculation</li><li>• Create awareness among people</li></ul>
5.	Business Model (Revenue Model)	<ul style="list-style-type: none"><li>• Application is provided free of cost, but it will have some advertisement</li><li>• In premium version, there is no advertisement and have some additional features</li></ul>
6.	Scalability of the Solution	<ul style="list-style-type: none"><li>• This application can handle large number of user data</li><li>• Provides high performance and security</li><li>• Easily available all kinds of devices</li></ul>

#### d. Problem Solution fit

Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <b>CS</b> <ul style="list-style-type: none"> <li>People above the age of 16 who earn or spend money</li> <li>Customers who find it difficult to keep track of their expenses.</li> <li>Customers who want to wisely handle their savings and money</li> </ul>	<b>6. CUSTOMER CONSTRAINTS</b> <b>CC</b> <ul style="list-style-type: none"> <li>All data should be entered manually by the user.</li> <li>Privacy and security</li> <li>Network issues</li> <li>Not compatible in all devices</li> <li>Not enough balance due to lavish spending</li> </ul>	<b>5. AVAILABLE SOLUTIONS</b> <b>AS</b> <ul style="list-style-type: none"> <li>Excel sheet</li> <li>Expense diary</li> <li>Expense tracker app with minimal features. It has less security features and no customer-support.</li> </ul>	Explore AS, differentiate
	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <b>J&amp;P</b> <ul style="list-style-type: none"> <li>To keep track of daily expenses.</li> <li>Alert when a threshold limit is reached</li> <li>Categorizing expenses to have a good visualization of it.</li> <li>Difficult to track monthly expenses manually.</li> <li>Remembering of expenses is difficult</li> </ul>	<b>9. PROBLEM ROOT CAUSE</b> <b>RC</b> <ul style="list-style-type: none"> <li>Due to many platforms which uses online payment, expenses are more and untracked</li> <li>Reckless spending</li> <li>Forgetting payments</li> <li>Linking of financial account to the application</li> </ul>	<b>7. BEHAVIOUR</b> <b>BE</b> <ul style="list-style-type: none"> <li>Have a proper record of all the expenses.</li> <li>Set up monthly limit in the expenses.</li> <li>Would prefer a graphical representation of their daily, monthly and yearly expenses.</li> <li>Start saving money and reduce unwanted expenses.</li> </ul>	Focus on J&P, fit into BE, understand RC
Identify strong TR & EM	<b>3. TRIGGERS</b> <ul style="list-style-type: none"> <li>Reduces time and manual effort</li> <li>Insufficient money during emergency</li> <li>Excessive spending</li> <li>Saving money effectively</li> </ul> <b>4. EMOTIONS: BEFORE / AFTER</b> <p>Before</p> <ul style="list-style-type: none"> <li>Confused</li> <li>Frustrated</li> <li>Stressed</li> </ul> <p>After</p> <ul style="list-style-type: none"> <li>Customers get clarity on the expenses</li> <li>Confident</li> </ul>	<b>10. YOUR SOLUTION</b> <ul style="list-style-type: none"> <li>A personal expense tracker application to handle and keep track of the monthly income and daily/monthly/yearly expenses.</li> <li>Alerting the user when expenses exceed a particular limit</li> <li>Providing a graphical view with proper categorization of the spent amount.</li> <li>An application with good security and real-time tracking of expenses.</li> </ul>	<b>8. CHANNELS of BEHAVIOUR</b> <p><b>8.1 ONLINE</b></p> <ul style="list-style-type: none"> <li>The application comes with a lot of advertisements which can be irritating to the customers.</li> <li>Stealing of private data can be easy in online</li> <li>Data can be stored in cloud which can be secure</li> <li>Accurate graphical representation</li> <li>Untracked expenses if not manually updated</li> </ul> <p><b>8.2 OFFLINE</b></p> <ul style="list-style-type: none"> <li>No real time tracking</li> <li>Backup cannot be guaranteed</li> <li>Difficult In visualization of the amount spent</li> </ul>	Identify strong TR & EM

## 4. REQUIREMENT ANALYSIS

#### a. Functional requirement

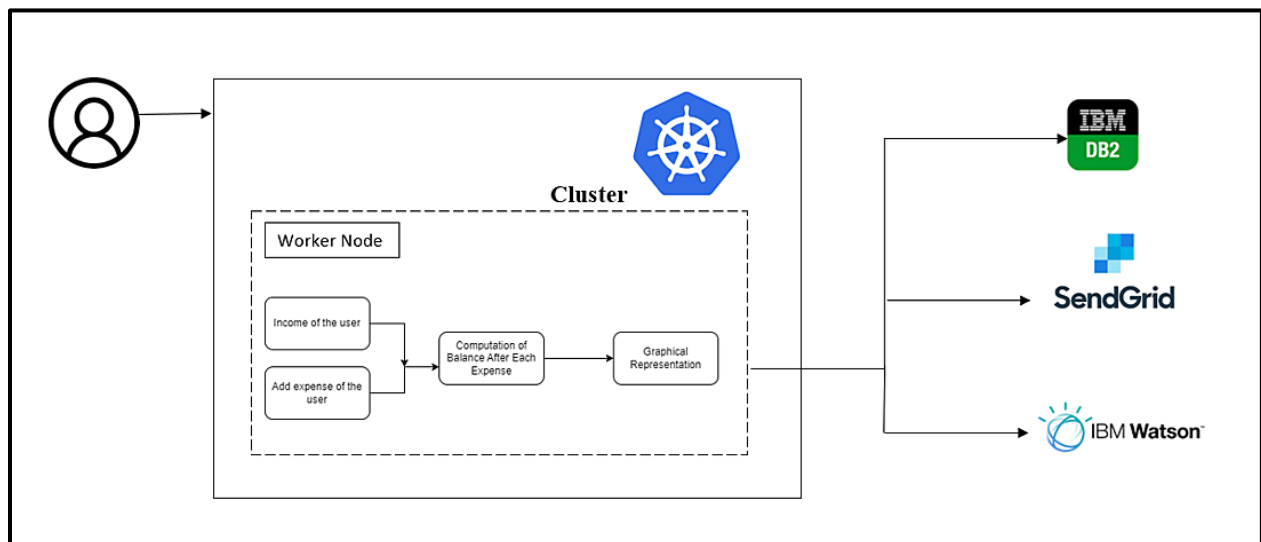
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Application Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User monthly expense tentative data	Data to be registered in the app
FR-4	User monthly income data	Data to be registered in the app
FR-5	Alert/ Notification	Alert through E-mail Alert through SMS
FR-6	User Budget Plan	Planning and Tracking of user expense vs budget limit

b. Non-Functional requirements

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Effectiveness, efficiency and overall satisfaction of the user while interacting with our application.
NFR-2	Security	Authentication, authorization, encryption of the application.
NFR-3	Reliability	Probability of failure-free operations in a specified environment for a specified time.
NFR-4	Performance	How the application is functioning and how responsive the application is to the end-users.
NFR-5	Availability	Without near 100% availability, application reliability and the user satisfaction will affect the solution.
NFR-6	Scalability	Capacity of the application to handle growth, especially in handling more users.

## 5. PROJECT DESIGN

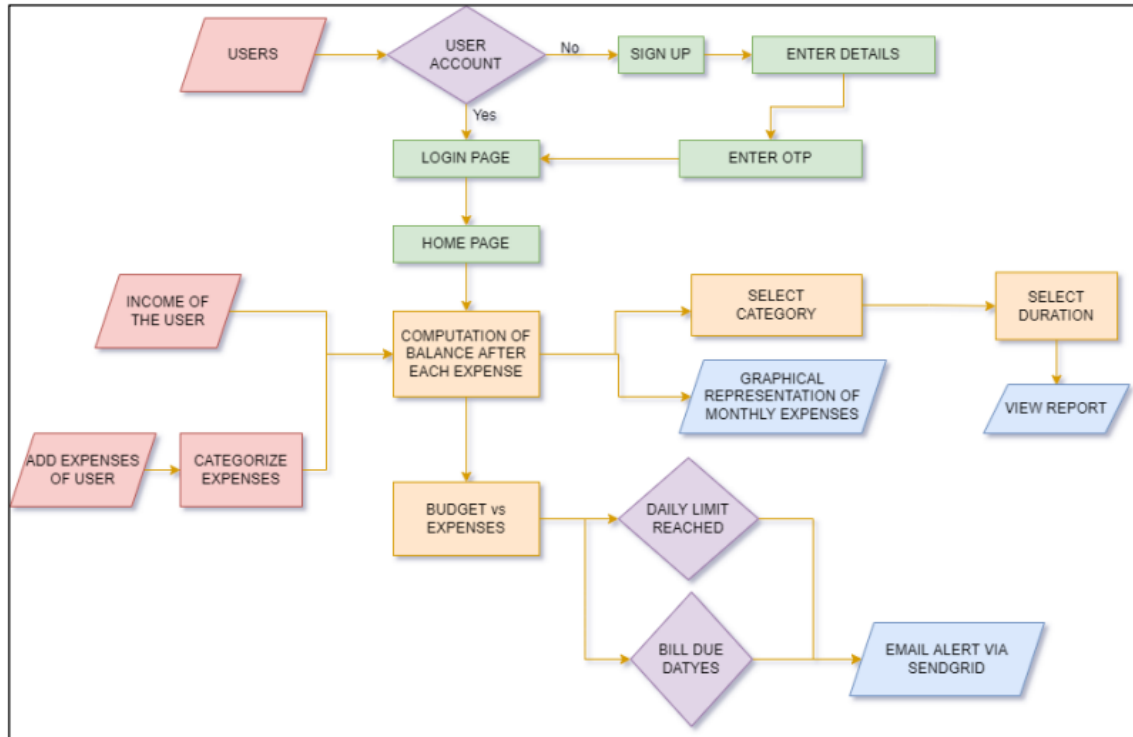
a. Data Flow Diagrams



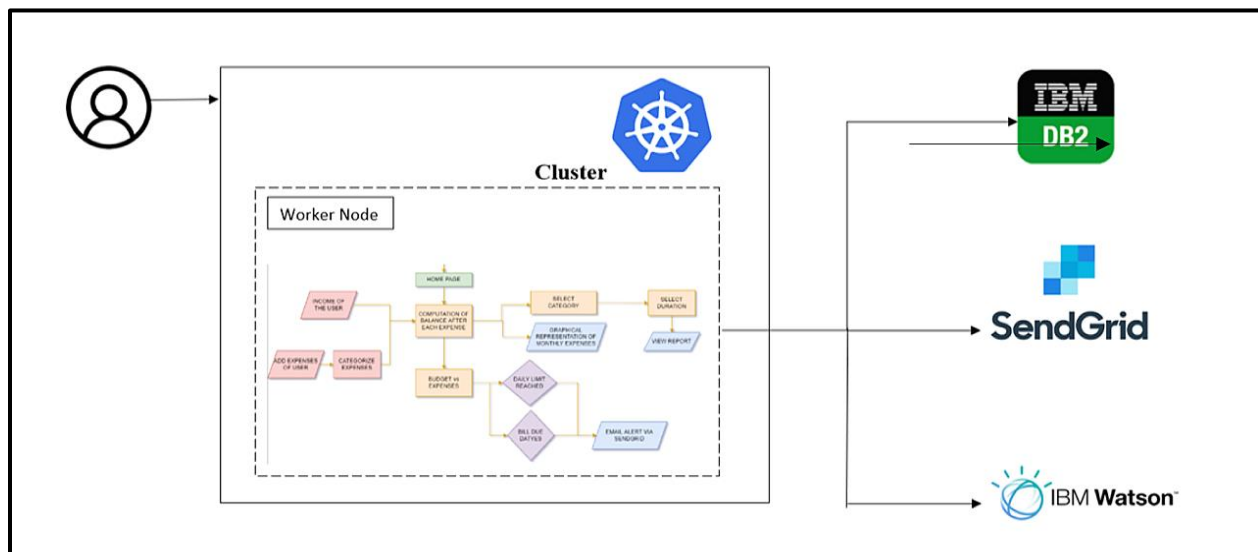


## b. Solution & Technical Architecture

### Solution Architecture



### Technical Architecture



### c. 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 application through Gmail	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can login in the application	High	Sprint-1
	Adding details	USN-6	As a user, I can add in my new expenses in daily/monthly basis.	I can organize my expenses	High	Sprint-2
	Removing details	USN-7	As a user, I can remove my previously added expenses.	I can rearrange my expense details	Medium	Sprint-2
	View Insights	USN-8	As a user, I can view my expenses in the form of graphs and get insights.	I can have better visualization of my expenses	High	Sprint-2
	Alert Notification	USN-9	As a user, I will get alert message if I exceed my target spending amount.	I can have better control on spending money	High	Sprint-1
Administrator	Add/remove user	USN-10	As a admin, I can add or remove user details on DB2 manually.	As a admin I can manage all the user details	High	Sprint-2

## 6. PROJECT PLANNING & SCHEDULING

### a. 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 , new password and confirming the same password.	4	High	Harini K Sudhirkumar S
		USN - 2	As a user , I will receive confirmation email once I have registered for the application.	3	Low	
	Login	USN - 3	As a user , I can log into the application by entering email and password / Google OAuth.	4	High	Monashree K Srinidhi S
	Dashboard	USN - 4	Logging in takes the user to their dashboard.	2	Low	
Sprint - 2		USN - 5	As a user ,I will update my salary at the start of each month.	2	Medium	Harini K Sudhirkumar S
		USN - 6	As a user , I will set a target/limit to keep track of my expenditure.	2	Medium	Monashree K Srinidhi S
	Workspace	USN - 7	Workplace for personal expense tracking	5	Medium	Harini K Sudhirkumar S
	Charts	USN - 8	Graphs to show weekly and everyday expenditure	5	High	Monashree K Srinidhi S
		USN - 9	As a user , I can export raw data as csv file.	2	Medium	Harini K Sudhirkumar S

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint - 3	IBM DB2	USN -10	Linking database with dashboard	3	High	Monashree K Srinidhi S
		USN -11	Making dashboard interactive with JS	3	High	Harini K Sudhirkumar S
	Watson Assistant	USN -12	Embedding Chatbot to clarify user's queries.	4	Low	Monashree K Srinidhi S
	BCrypt	USN -13	Using BCrypt to store passwords securely.	2	Medium	Harini K Sudhirkumar S
	SendGrid	USN -14	Using SendGrid to send mail to the user. (To alert or remind)	2	Medium	Monashree K Srinidhi S
Sprint - 4	Integration	USN -15	Integrating frontend and backend.	3	High	Monashree K Srinidhi S
	Docker	USN -16	Creating Docker image of web app.	2	High	Harini K Sudhirkumar S
	Cloud Registry	USN -17	Uploading docker image to IBM cloud registry.	3	High	Monashree K Srinidhi S
	Kubernetes	USN -18	Creating container using docker and hosting the webapp.	5	High	Monashree K Srinidhi S
	Exposing Deployment	USN -19	Exposing IP/Ports for the site.	1	Medium	Harini K Sudhirkumar S

## b. Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	28 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	6 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	18 Nov 2022

## c. Reports from JIRA

T	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due
	PETA-14	Integrating Front end and Back end	Sudhirkumar S	Monashree K	==	<span>DONE</span>	Done	14/Nov/22	19/Nov/22	
	PETA-13	Kubernetes	Monashree K	Monashree K	==	<span>DONE</span>	Done	14/Nov/22	19/Nov/22	
	PETA-12	Cloud Registry	Srinidhi S	Monashree K	==	<span>DONE</span>	Done	14/Nov/22	19/Nov/22	
	PETA-11	Docker	Harini K	Monashree K	==	<span>DONE</span>	Done	14/Nov/22	19/Nov/22	
	PETA-10	Connecting to ibm DB	Monashree K	Monashree K	==	<span>DONE</span>	Done	07/Nov/22	12/Nov/22	
	PETA-9	Sendgrid	Sudhirkumar S	Monashree K	==	<span>DONE</span>	Done	07/Nov/22	12/Nov/22	
	PETA-8	Watson Assistant	Srinidhi S	Monashree K	==	<span>DONE</span>	Done	07/Nov/22	12/Nov/22	
	PETA-7	Daily/Monthly/Yearly Report	Monashree K	Monashree K	==	<span>DONE</span>	Done	31/Oct/22	05/Nov/22	
	PETA-6	Target Limit	Sudhirkumar S	Monashree K	==	<span>DONE</span>	Done	31/Oct/22	05/Nov/22	
	PETA-5	History of expenses	Harini K	Monashree K	==	<span>DONE</span>	Done	31/Oct/22	05/Nov/22	
	PETA-4	Add expense	Srinidhi S	Monashree K	==	<span>DONE</span>	Done	31/Oct/22	05/Nov/22	
	PETA-3	Connect to ibm db	Monashree K	Monashree K	==	<span>DONE</span>	Done	24/Oct/22	29/Oct/22	
	PETA-2	Create Signup page	Harini K	Monashree K	==	<span>DONE</span>	Done	24/Oct/22	29/Oct/22	
	PETA-1	Create Login page	Sudhirkumar S	Monashree K	==	<span>DONE</span>	Done	24/Oct/22	29/Oct/22	

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

### a. Feature 1 - Watson Assistant

It uses artificial intelligence that understands customers in context to provide fast, consistent, and accurate answers across any application, device, or channel. It remove the frustration of long wait times, tedious searches, and unhelpful chatbots.

### b. Feature 2 - Sendgrid

This feature helps user to receive emails directly generated from the application. The emails are sent if the user spends more than a particular limit. This helps in having a control over the amount spent and saving money wisely.

```
#limit
@app.route("/limit" )
def limit():
    return redirect('/limitn')

@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
    if request.method == "POST":
        number= request.form['number']
        sql1 = "SELECT * FROM LIMITS WHERE ID=?"
        stmt = ibm_db.prepare(conn,sql1)
        ibm_db.bind_param(stmt,1,session['id'])
        ibm_db.execute(stmt)
        acc = ibm_db.fetch_assoc(stmt)
        if(acc):
            sql = "update limits set limit=? where ID=?"
            stmt = ibm_db.prepare(conn,sql)
            ibm_db.bind_param(stmt,1,number)
            ibm_db.bind_param(stmt,2,session['id'])
            ibm_db.execute(stmt)
        else:
            sql = "INSERT INTO LIMITS VALUES (?,?)"
            stmt = ibm_db.prepare(conn,sql)
            ibm_db.bind_param(stmt,1,session['id'])
            ibm_db.bind_param(stmt,2,number)
            ibm_db.execute(stmt)
        return redirect('/limitn')

@app.route("/limitn")
def limitn():
    sql = "SELECT * FROM LIMITS WHERE ID=?"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)

    acc = ibm_db.fetch_assoc(stmt)

    return render_template("limit.html" , y= acc)
```

c. Feature 3 - Add expenses

This feature helps to add everyday expenses according to different categories. It stores the details such as the amount spent in ibm\_db database.

```
#ADDING---DATA
@app.route("/add")
def adding():
    return render_template('add.html')

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

    date = request.form['date']
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category = request.form['category']
    sql = "INSERT INTO EXPENSES VALUES (?, ?, ?, ?, ?, ?)"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.bind_param(stmt,2,date)
    ibm_db.bind_param(stmt,3,expensename)
    ibm_db.bind_param(stmt,4,amount)
    ibm_db.bind_param(stmt,5,paymode)
    ibm_db.bind_param(stmt,6,category)
    ibm_db.execute(stmt)

    return redirect("/display")
```

d. Feature 4 - History

This feature helps to view the history of expenses for a particular user. It helps us to make a proper planning on how to reduce expenses and save money effectively.

e. Feature 5 - Report

This feature helps to view report of the expenses regularly as different categories. It displays the report date wise, month wise and year wise.

```

#REPORT
@app.route("/today")
def today():
    sql = "SELECT * FROM EXPENSES WHERE ID=? and DATE(date) = DATE(NOW())"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)

    acc = ibm_db.fetch_assoc(stmt)
    expense = []
    while acc != False:
        expense.append(acc)
        acc = ibm_db.fetch_assoc(stmt)
    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0

    for x in expense:
        total += x['AMOUNT']
        if x['CATEGORY'] == "food":
            t_food += x['AMOUNT']

        elif x['CATEGORY'] == "entertainment":
            t_entertainment += x['AMOUNT']

        elif x['CATEGORY'] == "business":
            t_business += x['AMOUNT']
        elif x['CATEGORY'] == "rent":
            t_rent += x['AMOUNT']

        elif x['CATEGORY'] == "EMI":
            t_EMI += x['AMOUNT']

        elif x['CATEGORY'] == "other":
            t_other += x['AMOUNT']
    sql = "SELECT * FROM LIMITS WHERE ID=?"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    acc = ibm_db.fetch_assoc(stmt)
    s = ""
    if acc['LIMIT'] <= total:
        s = "YOU WERE EXCEEDED YOUR CURRENT EXPENSE LIMIT"

    return render_template("today.html", expense = expense, total = total ,
                           t_food = t_food,t_entertainment = t_entertainment,
                           t_business = t_business, t_rent = t_rent,
                           t_EMI = t_EMI, t_other = t_other , alert = s)

```

```
@app.route("/month")
def month():
    sql = "SELECT * FROM EXPENSES Where ID=? AND MONTH(date)= MONTH(now())"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)

    acc = ibm_db.fetch_assoc(stmt)
    expense = []
    while acc != False:
        expense.append(acc)
        acc = ibm_db.fetch_assoc(stmt)
    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0

    for x in expense:
        total += x['AMOUNT']
        if x['CATEGORY'] == "food":
            t_food += x['AMOUNT']

        elif x['CATEGORY'] == "entertainment":
            t_entertainment += x['AMOUNT']

        elif x['CATEGORY'] == "business":
            t_business += x['AMOUNT']
        elif x['CATEGORY'] == "rent":
            t_rent += x['AMOUNT']

        elif x['CATEGORY'] == "EMI":
            t_EMI += x['AMOUNT']

        elif x['CATEGORY'] == "other":
            t_other += x['AMOUNT']
    return render_template("today.html", expense = expense, total = total ,
        t_food = t_food,t_entertainment = t_entertainment,
        t_business = t_business, t_rent = t_rent,
        t_EMI = t_EMI, t_other = t_other )
```

```

@app.route("/year")
def year():
    sql = "SELECT * FROM EXPENSES Where ID=? AND YEAR(DATE(date))= YEAR(now())"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)

    acc = ibm_db.fetch_assoc(stmt)
    expense = []
    while acc != False:
        expense.append(acc)
        acc = ibm_db.fetch_assoc(stmt)
    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0

    for x in expense:
        total += x['AMOUNT']
        if x['CATEGORY'] == "food":
            t_food += x['AMOUNT']

        elif x['CATEGORY'] == "entertainment":
            t_entertainment += x['AMOUNT']

        elif x['CATEGORY'] == "business":
            t_business += x['AMOUNT']
        elif x['CATEGORY'] == "rent":
            t_rent += x['AMOUNT']

        elif x['CATEGORY'] == "EMI":
            t_EMI += x['AMOUNT']

        elif x['CATEGORY'] == "other":
            t_other += x['AMOUNT']
    return render_template("today.html", expense = expense, total = total ,
        t_food = t_food,t_entertainment = t_entertainment,
        t_business = t_business, t_rent = t_rent,
        t_EMI = t_EMI, t_other = t_other )

```



## 8. TESTING

### a. Test Cases

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status
LoginPage_TC_001	Functional	Home Page	Verify user is able to see the Login/Signup button when user entered the website		1.Enter URL and click go 2.Click on Login/Signup button displayed on home page 3.Verify login/Signup page displayed or not	(website URL)	Login/Signup page should display	Working as expected	Pass
LoginPage_TC_002	UI	Home Page	Verify the UI elements in Login/Signup button		1.Enter URL and click go 2.Click on Signup button 3.Verify Signup button with below UI elements: a. email text box b. password text box c. Create password	(website URL)	Application should show below UI elements: a. email text box b. password text box c. Create password	Working as expected	Pass
LoginPage_TC_003	Functional	Home page	Verify user is able to create an account in the application		1.Enter URL(https://shopenczer.com/) and click go 2.Click on Signup button 3.Enter Valid email in Email text box 4.Enter username in username text box 5.Enter valid password in password text box 6.Click on Create account	Username: sudhikumar@gmail.com password: sudhi3009	User should display an Account created message	Working as expected	Pass
LoginPage_TC_004	Functional	Home page	Verify user is able to log into application with Valid credentials		1.Enter URL(https://shopenczer.com/) and click go 2.Click on Login button 3.Enter Valid email in Email text box 4.Enter valid password in password text box 5.Click on Login	Username: sudhikumar@gmail.com password: sudhi3009	User should navigate to user account homepage	Working as expected	Pass
LoginPage_TC_005	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL(https://shopenczer.com/) and click go 2.Click on Login button 3.Enter wrong email in Email text box 4.Enter wrong password in password text box 5.Click on Login	Username: sudhikumar@gmail.com password: Testing123	Application should show 'Incorrect email or password' validation message.	Working as expected	Pass
LoginPage_TC_006	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL(https://shopenczer.com/) and click go 2.Click on Login button 3.Enter wrong email in Email text box 4.Enter valid password in password text box 5.Click on Login	Username: sudhikumar@gmail.com password: Testing123678686786876 876	Application should show 'Incorrect email or password' validation message.	Working as expected	Pass
LoginPage_TC_007	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL(https://shopenczer.com/) and click go 2.Click on Login button 3.Enter wrong email in Email text box 4.Enter wrong password in password text box 5.Click on Login	Username: chalam password: Testing123678686786876 876	Application should show 'Incorrect email or password' validation message.	Working as expected	Pass

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status
HomePage_TC_001	Functional	Home page	Verify user is able to see Add expense, History, Limit, Report buttons in the Menu		1.Enter URL(https://shopenczer.com/) and click go 2.Enter the valid login credentials and click Login 3. Verify Add expense, History, Limit, Report buttons are displayed in the Menu	(website Homepage URL)	Application should show below UI elements: a. Add Expense button b. History button c. History button	Working as expected	Pass
HomePage_TC_002	Functional	Add Expense	Verify user is able to add his expense in the Add Expense Menu.		1.Enter URL(https://shopenczer.com/) and click go 2.Enter the valid login credentials and click Login 3.Go to Add expense tab in the menu. 4.Enter Date, Expense name, Expense Amount, Payment mode, Category. 5.Click Add Expense	Date: 12-11-2022T11:40Eip	Application should push the data into the IBM database created and display the details in the display tab.	Working as expected	Pass
HomePage_TC_003	UI	History	Verify user is able to view the total history of the expenses made by the user.		1.Enter URL(https://shopenczer.com/) and click go 2.Enter the valid login credentials and click Login 3.Click the History button in the menu.	(website History URL)	Application should display the total history of the expenses made by the user.	Working as expected	Pass
HomePage_TC_004	Functional	Limit	Verify user is able to set the limit for his expenses.		1.Enter URL(https://shopenczer.com/) and click go 2.Enter the valid login credentials and click Login 3.Go to the Limit tab in the menu. 4.Enter the Expense limit amount in the text box 5.Click Enter	(website Limit URL)	Application should push the Expense limit amount in the database and display the Current Expense limit in the same tab itself.	Working as expected	Pass
HomePage_TC_005	Functional	Limit	Verify user is able to receive an alert message when the expense exceeds the limit.		1.Enter URL(https://shopenczer.com/) and click go 2.Enter the valid login credentials and click Login 3.Add your day to day expenses in the Add expense tab. 4. When the total expense of a month exceeds the Expense limit, the user should receive an alert message.	(website Limit URL)	Application should send an alert message will be sent to the registered mail, when the expense exceeds the limit.	Working as expected	Pass
HomePage_TC_006	Functional	Report	Verify user is able to see the statistics of his own expenses on the basis of a day, a month or a year.		1.Enter URL(https://shopenczer.com/) and click go 2.Enter the valid login credentials and click Login 3.Click the Report dropdown button in the Menu. 4. Select the required button as Today, a month, or a year button to display the statistics.	(website Report URL)	Application should display the statistics of his own expenses on the basis of a day, a month or a year.	Working as expected	Pass

b. User Acceptance Testing

**Defect Analysis**

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	10	2	6	25	43
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	23	14	15	30	84

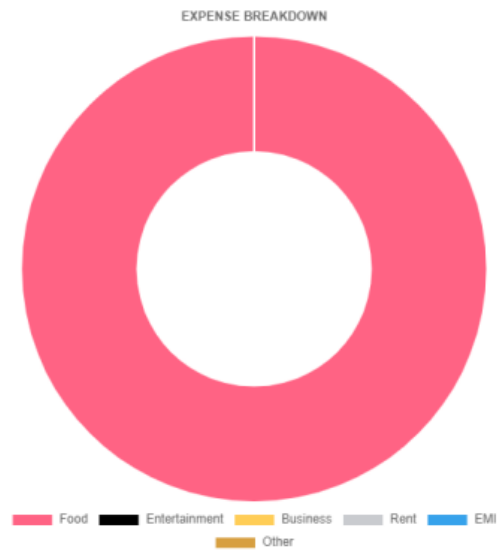
**Test Case Analysis**

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

## 9. RESULTS

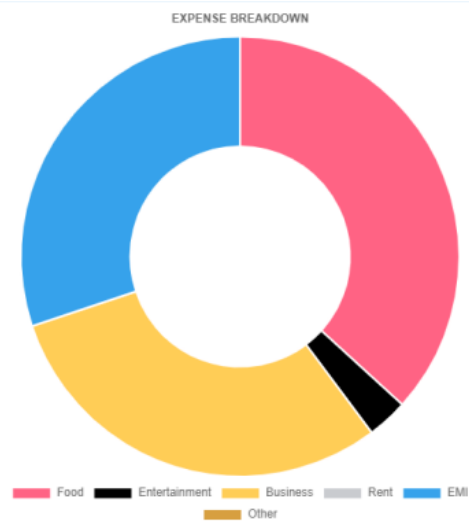
### Expense Breakdown BY Category

Food	5100.0
Entertainment	0
Business	0
Rent	0
EMI	0
Other	0
Total	₹ 5100.0



### Expense Breakdown BY Category

Food	6100.0
Entertainment	500.0
Business	5000.0
Rent	0
EMI	5000.0
Other	0
Total	₹ 16600.0



## 10. ADVANTAGES & DISADVANTAGES

### Advantages:

- This application helps to allocate money to different priorities and manage them wisely.

- It helps us cut down on unnecessary spending.
- Helps us to save a lot of money.
- Alerts the user if he/she is spending more than a limited amount.

#### Disadvantages:

- Existence of the application is usually forgotten all together.
- Error in tracking of spending amount
- Can be time consuming

## 11. CONCLUSION

Monitoring your everyday expenses can set aside your cash and track everything online. We can also see where a few reductions and bargains can be made. Personal Expense Tracker helps us to keep an eye on our day-to-day expenditures and record our money everyday. The project what we have created is more proficient than any other income and expense tracker. It effectively keeps away from the manual figuring the pay and cost each month. It is a user-friendly application.

## 12. FUTURE SCOPE

- It will have various options and categories readily available to keep track of expenses (for example Food, Travelling Fuel, Salary etc.)
- Send SMS notifications for our daily expenditure.
- An improvised AI based chatbot

## 13. APPENDIX

Source Code

**app.py**

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

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30376;SECURITY=S
```

```

SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=lfx34122;PWD=jmQDS9wCaxqRIlQ
d",',')

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

#HOME--PAGE
@app.route("/home")
def home():
    return render_template("homepage.html")

@app.route("/")
def add():
    return render_template("home.html")

#SIGN--UP--OR--REGISTER

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

@app.route('/register', methods = ['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

        sql = "SELECT * FROM USERS WHERE USERNAME=?"
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        acc = ibm_db.fetch_assoc(stmt)
        if acc:
            msg = "Account already exists !!"
        elif not re.match(r'^@]+@[^@]+\.[^@]+',email):
            msg = "Invalid Email address"
        elif not re.match(r'[A-Za-z0-9]+',username):
            msg = "Name must contain only characters and numbers !!"
        else:

```

```

        sql = "INSERT INTO USERS VALUES (?, ?, ?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, email)
        ibm_db.bind_param(stmt, 3, password)
        ibm_db.execute(stmt)
        msg = "Successgully registered !!Login to continue"
        return render_template('login.html', msg = msg)
    elif request.method == 'POST':
        msg = "Please fill out the form !"
        return render_template('register.html', msg = msg)

#LOGIN--PAGE

@app.route("/signin")
def signin():
    return render_template("login.html")

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

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

        sql = "SELECT * FROM USERS WHERE EMAIL=? AND PASSWORD=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        acc = ibm_db.fetch_assoc(stmt)
        if acc:
            session['loggedin'] = True
            session['id'] = acc['USERNAME']
            userid = acc['USERNAME']
            session['username'] = acc['USERNAME']
            msg = acc['USERNAME']

            return render_template('homepage.html', msg = msg)
        else:
            msg = "Incorrect username/password!!"
    return render_template('login.html', msg = msg)

```

```

#ADDING----DATA
@app.route("/add")
def adding():
    return render_template('add.html')

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

    date = request.form['date']
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category = request.form['category']
    sql = "INSERT INTO EXPENSES VALUES (?, ?, ?, ?, ?, ?)"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.bind_param(stmt,2,date)
    ibm_db.bind_param(stmt,3,expensename)
    ibm_db.bind_param(stmt,4,amount)
    ibm_db.bind_param(stmt,5,paymode)
    ibm_db.bind_param(stmt,6,category)
    ibm_db.execute(stmt)

    return redirect("/display")

#DISPLAY---graph
@app.route("/display")
def display():
    sql = "SELECT * FROM EXPENSES WHERE ID=?"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)

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

#limit
@app.route("/limit" )

```

```

def limit():
    return redirect('/limitn')

@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
    if request.method == "POST":
        number= request.form['number']
        sql1 = "SELECT * FROM LIMITS WHERE ID=?"
        stmt = ibm_db.prepare(conn,sql1)
        ibm_db.bind_param(stmt,1,session['id'])
        ibm_db.execute(stmt)
        acc = ibm_db.fetch_assoc(stmt)
        if(acc):
            sql = "update limits set limit=? where ID=?"
            stmt = ibm_db.prepare(conn,sql)
            ibm_db.bind_param(stmt,1,number)
            ibm_db.bind_param(stmt,2,session['id'])
            ibm_db.execute(stmt)
        else:
            sql = "INSERT INTO LIMITS VALUES (?,?)"
            stmt = ibm_db.prepare(conn,sql)
            ibm_db.bind_param(stmt,1,session['id'])
            ibm_db.bind_param(stmt,2,number)
            ibm_db.execute(stmt)
        return redirect('/limitn')

@app.route("/limitn")
def limitn():
    sql = "SELECT * FROM LIMITS WHERE ID=?"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)

    acc = ibm_db.fetch_assoc(stmt)

    return render_template("limit.html" , y= acc)

#REPORT
@app.route("/today")
def today():
    sql = "SELECT * FROM EXPENSES WHERE ID=? and DATE(date) = DATE(NOW())"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)

```



```

acc = ibm_db.fetch_assoc(stmt)
expense = []
while acc != False:
    expense.append(acc)
    acc = ibm_db.fetch_assoc(stmt)
total=0
t_food=0
t_entertainment=0
t_business=0
t_rent=0
t_EMI=0
t_other=0

for x in expense:
    total += x['AMOUNT']
    if x['CATEGORY'] == "food":
        t_food += x['AMOUNT']

    elif x['CATEGORY'] == "entertainment":
        t_entertainment += x['AMOUNT']

    elif x['CATEGORY'] == "business":
        t_business += x['AMOUNT']
    elif x['CATEGORY'] == "rent":
        t_rent += x['AMOUNT']

    elif x['CATEGORY'] == "EMI":
        t_EMI += x['AMOUNT']

    elif x['CATEGORY'] == "other":
        t_other += x['AMOUNT']
sql = "SELECT * FROM LIMITS WHERE ID=?"
stmt = ibm_db.prepare(conn,sql)
ibm_db.bind_param(stmt,1,session['id'])
ibm_db.execute(stmt)
acc = ibm_db.fetch_assoc(stmt)
s = ""
if acc['LIMIT'] <= total:
    s = "YOU WERE EXCEEDED YOUR CURRENT EXPENSE LIMIT"

return render_template("today.html", expense = expense, total = total ,
                        t_food = t_food,t_entertainment = t_entertainment,
                        t_business = t_business, t_rent = t_rent,
                        t_EMI = t_EMI, t_other = t_other , alert = s)

```

[illegible]

```

@app.route("/year")
def year():
    sql = "SELECT * FROM EXPENSES Where ID=? AND YEAR(DATE(date))= YEAR(now())"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)

    acc = ibm_db.fetch_assoc(stmt)
    expense = []
    while acc != False:
        expense.append(acc)
        acc = ibm_db.fetch_assoc(stmt)
    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0

    for x in expense:
        total += x['AMOUNT']
        if x['CATEGORY'] == "food":
            t_food += x['AMOUNT']

        elif x['CATEGORY'] == "entertainment":
            t_entertainment += x['AMOUNT']

        elif x['CATEGORY'] == "business":
            t_business += x['AMOUNT']
        elif x['CATEGORY'] == "rent":
            t_rent += x['AMOUNT']

        elif x['CATEGORY'] == "EMI":
            t_EMI += x['AMOUNT']

        elif x['CATEGORY'] == "other":
            t_other += x['AMOUNT']
    return render_template("today.html", expense = expense, total = total ,
                           t_food = t_food,t_entertainment = t_entertainment,
                           t_business = t_business, t_rent = t_rent,
                           t_EMI = t_EMI, t_other = t_other )

#log-out

```

```
@app.route('/logout')

def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    return render_template('home.html')

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

GitHub & Project Demo Link

GitHub - <https://github.com/IBM-EPBL/IBM-Project-25312-1659958275>