

# **PERSONAL EXPENSE TRACKER APPLICATION**

## **PROJECT REPORT**

Submitted by

**NISHANTHI.R**

**813819104062**

**NITHIN.T**

**813819104063**

**RHUBIKA.M**

**813819104076**

**ROHIT.M**

**813819104077**

# CONTENTS

## **1. INTRODUCTION**

- 1.1 Project Overview
- 1.2 Purpose

## **2. LITERATURE SURVEY**

- 2.1 Existing problem
- 2.2 References
- 2.3 Problem Statement Definition

## **3. IDEATION & PROPOSED SOLUTION**

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming
- 3.3 Proposed Solution
- 3.4 Problem Solution fit

## **4. REQUIREMENT ANALYSIS**

- 4.1 Functional requirement
- 4.2 Non-Functional requirements

## **5. PROJECT DESIGN**

- 5.1 Data Flow Diagrams
- 5.2 Solution & Technical Architecture
- 5.3 User Stories

## **6. PROJECT PLANNING & SCHEDULING**

- 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 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 Link

# **1.INTRODUCTION**

## **1.1. PROJECT OVERVIEW**

In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management.

Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert.

## **1.2 PURPOSE**

Tracking your expense involves identifying your expenditures throughout the month. It's an essential activity that you should ideally do every day throughout the month. It may seem like a lot of work to itemize your expenses when you first begin, but understanding why it's important to track expenses and how to do so with minimal effort can help you successfully commit to the activity and become more aware of your spending. Tracking your expenses is one of the key factors in making your budget work for you. If you do not know how much you have spent each month, you cannot tell when you have overspent. Even the small expenses can cause you to blow your budget. One of the simplest is a written ledger or tracking system. It may be even easier to choose budgeting software that works with an app to track expenses on your phone. It will allow you to keep up while on the go. It can also help you become more aware of what you are spending and where you are spending it. That can help you identify the areas where you need to make changes.

## **2. LITERATURE SURVEY**

### **2.1 Existing problem**

In existing, we need to maintain the Excel sheets, CSV etc. files for the user daily and monthly expenses. In existing, there is no as such complete solution to keep a track of its daily expenditure easily. To do so a person has to keep a log in a diary or in a computer, also all the calculations need to be done by the user which may sometimes result in errors leading to losses.

There can be many disadvantages of using a manual accounting system. Accounting, for any business, can be a complex undertaking. A manual accounting system requires you to understand the accounting process in a way that may be unnecessary with a computerized accounting system. This can be an advantage or a disadvantage, depending on the person doing the bookkeeping; often, a specially trained professional is needed to ensure that accounting is done properly. Unraveling the complexity of your financial records by hand may be time consuming. Since it takes time to generate reports.

### **2.2 References**

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

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

[3]. Engineering Association of Palestine. Current Engineering Statistics Book.

Ramallah; 2005.

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

[5]. Chen JW, Yen M. Engineering Accreditation: A Foundation for Continuing Quality Improvement. 2005 Mar 1–5; Tainan. Exploring Innovation in Education and Research.

## **2.3 Problem Statement Definition**

Expense Tracker is an Application which can help the user to keep track of their Expenses. Nowadays, people can do various things by using a mobile and so, they can also use it for Budgeting and planning their expense in the mobile instead of doing it manually. For this purpose, an application can be developed to satisfy the needs of the customer. This application can help the user to keep track of their expenses in an organized way and to maintain a proper balance between expenditure and savings. In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management. Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert.

## 3. IDEATION AND PROPOSED SOLUTION

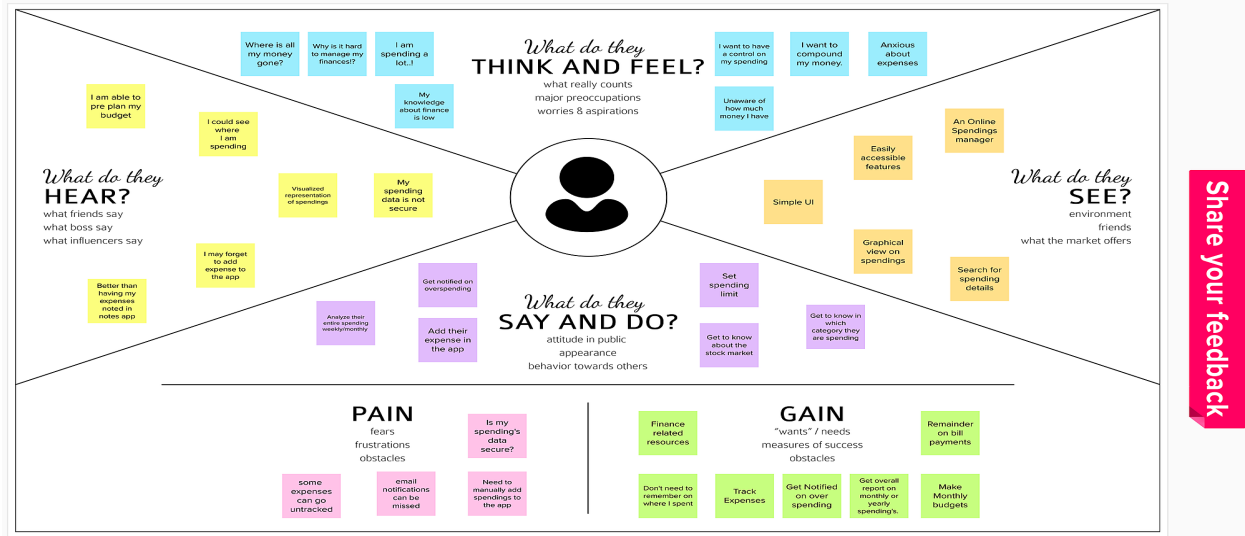
### 3.1 Empathy map canvas

Edit this template  
Right-click to unlock

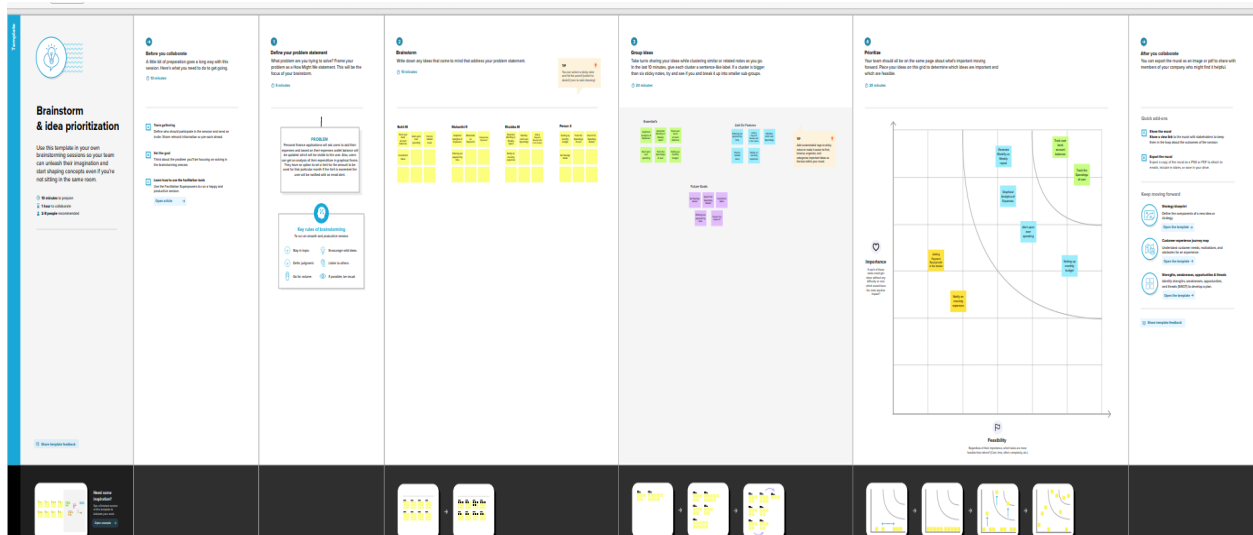
# Empathy Map Canvas

Gain insight and understanding on solving customer problems.

1 Build empathy and keep your focus on the user by putting yourself in their shoes.



### 3.2 Ideation and Brainstorming



### 3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To help the customer keep tracking of his/her personal expenses periodically.
2.	Idea / Solution description	To build a new innovative solution through which a user can get graphical analysis of expenses and generate monthly reports.
3.	Novelty / Uniqueness	Notification on recurring expenses and setting saving goals can be done easily.
4.	Social Impact / Customer Satisfaction	Customer can easily search for spending details and payment receipt is added with the tracker.
5.	Business Model (Revenue Model)	Subscription model and financial related advertisement.
6.	Scalability of the Solution	More number of users can be managed effectively, since entire application is hosted on cloud.

## 3.4 Problem Solution Fit

Problem-Solution fit canvas 2.0		Purpose / Vision	
Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span> Customers use different methods to track their income and expense by using paper and pen, spreadsheets and budgeting apps	<b>6. CUSTOMER CONSTRAINTS</b> <span>CC</span> The constraints that the customer face while using this application is used in a low cost and easy way.	<b>5. AVAILABLE SOLUTIONS</b> <span>AS</span> The solution which we have proposed are if the expense exceeded than specified limit, the application will show you an alert message in form of a pie chart.
	<div>Explore AS, differentiate</div>		
Focus on J&P, tap into BE, understand RC	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <span>J&amp;P</span> Customers can able to track their income and expenses easily	<b>9. PROBLEM ROOT CAUSE</b> <span>RC</span> Due to unawareness and also there is no right software so that the customer finds hard to track their expense.	<b>7. BEHAVIOUR</b> <span>BE</span> They can able to make decision whether they can avoid some unwanted expenses.
	<div>Focus on J&amp;P, tap into BE, understand RC</div>		
Identify strong TR & EM	<b>3. TRIGGERS</b> <span>TR</span> Some of the triggers are advertisements in the television and information from the experts.	<b>10. YOUR SOLUTION</b> <span>SL</span> This application tracks your every expenses anywhere and anytime without using the paper work. Just click and enter your expenditure to avoid data loss, quick settlements and reduce human error. To provide the pie chart or graph lines in this application.	<b>8. CHANNELS of BEHAVIOUR</b> <span>CH</span> <b>8.1 ONLINE</b> with the help of various online channel customers track their expenses regularly.
	<b>4. EMOTIONS: BEFORE / AFTER</b> <span>EM</span> With traditional tracker system customers find difficult to track their expenses but after using personal expense they track their expense easily	<b>8.2 OFFLINE</b> Personal Expense Tracker Application enables customers to reduce spending their expenses for unwanted things.	
<div>Extract online &amp; offline CH of BE</div>			



## 4.REQUIREMENT ANALYSIS

### 4.1 Functional Requirements

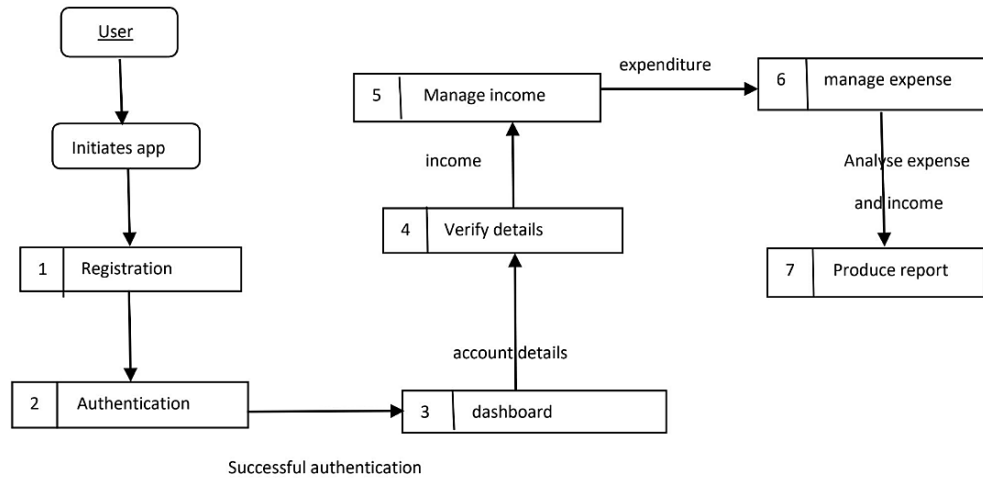
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration via g-mail or phone number
FR-2	User Confirmation	Confirmation via OTP Confirmation via g-mail
FR-3	Tracking of expenses	The user can keep track of his/her expenses
FR-4	Notify on recurring expenses	The user gets notification for his/her monthly recurring expenses
FR-5	Recommend investment ideas	Recommending ideas based on their expenses, salary, etc.
FR-6	Alert on over spending	It gives an alert message if the user spends more than his salary.

### 4.2 Non Functional Requirements

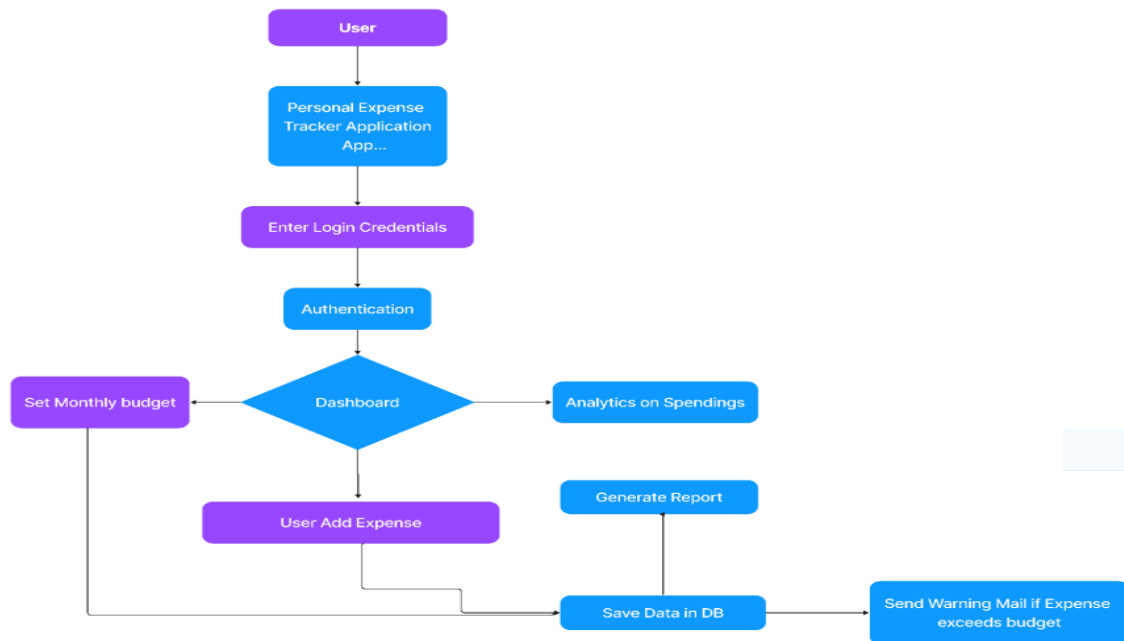
FR No.	Non-Functional Requirement	Description
NFR-1	<b>Usability</b>	It is very user friendly since they can easily store their expense details.
NFR-2	<b>Security</b>	It can be accessed only by the authorized user with the registered login credentials.
NFR-3	<b>Reliability</b>	The application is more reliable since it allows only the authorized user.
NFR-4	<b>Performance</b>	The system perform well in reporting the expenses, remainders, alerts, etc.
NFR-5	<b>Availability</b>	The expense tracker application can be accessed at any time.
NFR-6	<b>Scalability</b>	The application will have the data backup mechanism.

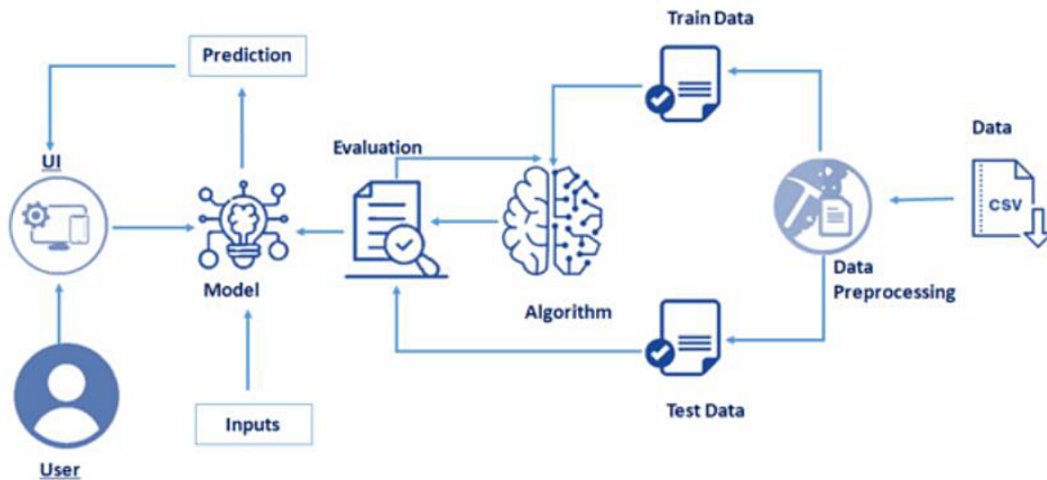
## 5.PROBLEM DESIGN

### 5.1 Data Flow Diagrams



### 5.2 Solution and Technical Architecture





## 5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story I 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-I
		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-I
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register & access the dashboard with Gmail Login	Medium	Sprint-I
	Login	USN-5	As a user, I can log into the application by entering email & password	I can get a detailed report	High	Sprint-I
	Dashboard	USN-6	As a user,I can upload my income and expenditure to get a detail report.	I can get my detailed report	High	Sprint-I

Customer (Web user)	Registration	USN-7	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-I
Customer Care Executive	Communica te customer	USN-8	As a customer care executive ,I can able to address customer issues and resolve them in timely and efficient manner	I can solve the customer problem	High	Sprint-I
Administrat or		USN-9	As a system administrator I want to be able to configure user settings so that I can control access	I can access my user details	Low	Sprint-2

Customer (Personel expense tracking application)	Identify skills	USN-IO	As a user ,I am able to identify our skills and eligibility	I can able to identify the skills	High	Sprint- 1
	Produce details of income and expendeture	USN-II	As a user, I am able to produce the required details	I can produce details	High	Sprint-I

## 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.	7	High	Rhubika, Nishanthi
Sprint-1	Login	USN-2	As a user, I can login to the application by entering the registered username and password	6	High	Nithin, Nishanthi
Sprint-1	Dashboard	USN-3	As a user, I can use the facilities that are provided.	3	Medium	Rhubika, Rohit
Sprint-1	Add expense	USN-4	As a user, I can add daily expense	4	Medium	Rohit
Sprint-2	Set Budget	USN-5	As a user, I can set budgets for monthly expenses	8	High	Nithin, Rhubika
Sprint-2	Alerts users on overspending	USN-6	As a user, I get alert users on overspending	4	Medium	Rohit, Nishanthi
Sprint-2	Analysis on expense	USN-7	As a user, I can analyse the daily, monthly and annual	8	High	Nithin

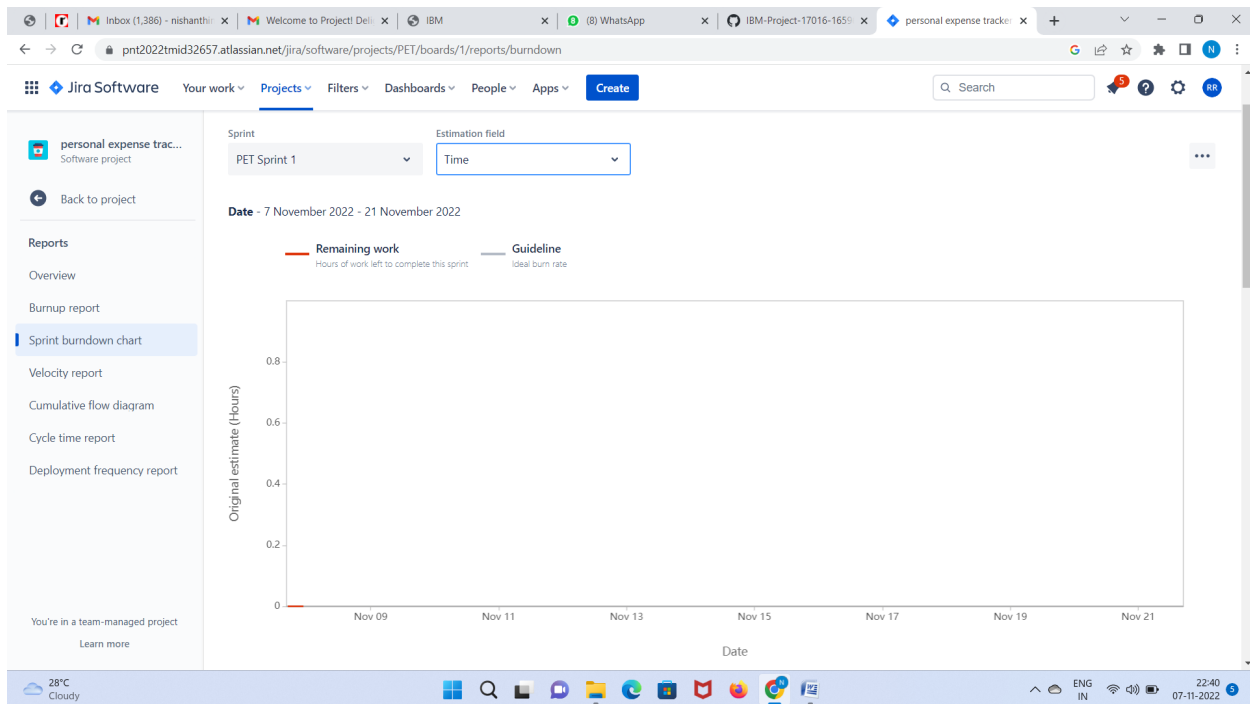
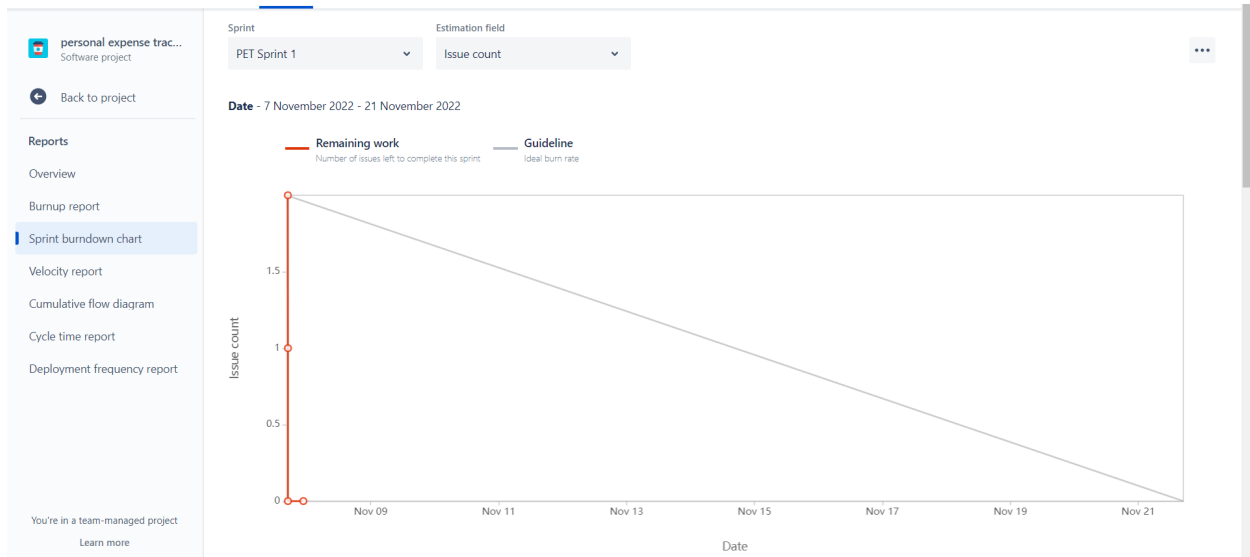
			expenses.			
Sprint-3	Creating filter based on category	USN-8	As a user, I can see the category-based history of expenses.	6	Medium	Rhubika
Sprint-3	Creating filter based on payment mode	USN-8	As a user, I can see the payment mode-based history of expenses.	6	Medium	Nithin
Sprint-3	Creating filter based on time	USN-8	As a user, I can see the time-based history of expenses.	6	Medium	Rohit
Sprint-3	Generate pdf report on expense	USN-9	As a user, I can get report of the overall expenses in pdf format.	2	Medium	Nishanthi
Sprint-4	Testing	USN-10	Testing the application with various tools.	10	High	Nishanthi, Rhubika
Sprint-4	Deployment	USN-11	Deployment of the application.	10	High	Rohit,Nithin

## 6.2 Sprint Delivery Schedule

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

## 6.3 Report from JIRA









personal expense trac...  
Software project



Back to project

## Reports

Overview

Burnup report

Sprint burndown chart

Velocity report

Cumulative flow diagram

Cycle time report

Deployment frequency report

You're in a team-managed project  
[Learn more](#)

Sprint

Estimation field

PET Sprint 1

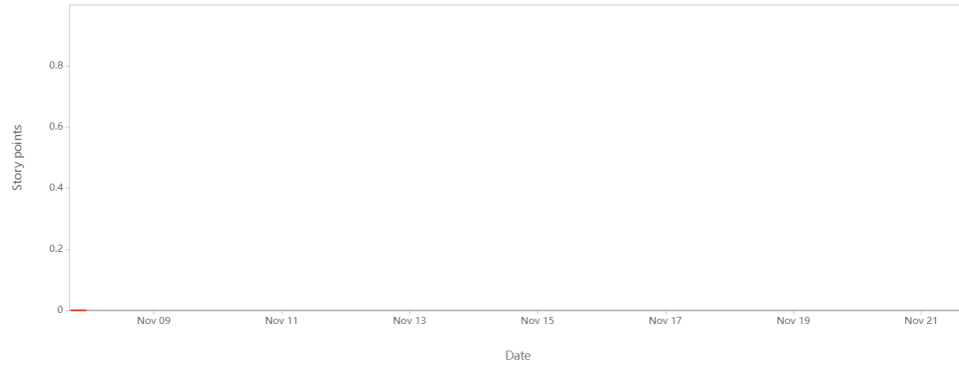
Story points



Date - 7 November 2022 - 21 November 2022

**Remaining work**  
Number of story points left to complete this sprint

**Guideline**  
Ideal burn rate



## 7.CODING & SOLUTIONING

### 7.1 Feature 1

#### **feature.py**

```
from flask import Blueprint, redirect, render_template, url_for, session, request,
send_file
import ibm_db
from io import BytesIO
from reportlab.lib.utils import ImageReader
from reportlab.platypus import Table
from reportlab.pdfgen.canvas import Canvas
from reportlab.pdfbase import pdfmetrics
from reportlab.pdfbase.ttfonts import TTFont
from reportlab.lib.pagesizes import A4
from reportlab.lib import colors
from reportlab.lib.units import cm
from datetime import datetime
from .connect import connection
from .email import sendMail
feature = Blueprint("feature", __name__, url_prefix="/")
con = connection
@feature.route("/pdf")
def pdfGenerator():
    if(request.method == 'GET'):
        if 'auth' in session:
            if session['auth']:
                qry = "SELECT * FROM EXPENSE WHERE USER_ID=? ORDER BY EXP_ID;"

                stmt = ibm_db.prepare(con,qry)
                uid = session['uid']
                ibm_db.bind_param(stmt,1,uid)

                resp = ibm_db.execute(stmt)

                data = ibm_db.fetch_assoc(stmt)
```

```

result = [ ]

while data:
    t = [ ]

    t.append(data['TITLE'])
    t.append(data['AMOUNT'])
    t.append(data['CATEGORY'])
    t.append(data['PAYMENT_MODE'])
    t.append(str(data['DATE_TIME'][:10]))

    result.append(t)

    data = ibm_db.fetch_assoc(stmt)

# print(result)

buffer = create_pdf(result)

buffer.seek(0)

return send_file(buffer, as_attachment=True,
mimetype='application/pdf',download_name="{name}-
report.pdf".format(name=str(session['name'])))

return redirect(url_for('auth.login'))
return redirect(url_for('auth.login'))

def create_pdf(expenseList):
    buffer = BytesIO()
    canvas = Canvas(buffer, pagesize=A4)
    WIDTH, HEIGHT = A4
    MARGIN = 0.5 * cm
    canvas.translate(MARGIN, HEIGHT-MARGIN)

```

```
canvas.setFont("Helvetica-Bold", 15)
canvas.drawString(WIDTH//3+10, 0, "Expense Report")
```

```
canvas.setFont("Helvetica", 12)
canvas.drawString(430, -0.9*cm, "MyMoney Pvt Ltd")
canvas.setStrokeGray(0)
canvas.line(0, -1*cm, WIDTH - 2*MARGIN, -1*cm)
```

```
canvas.line(0, -26.6*cm, WIDTH - 2*MARGIN, -26.6*cm)
canvas.drawString(430, -26.5*cm, "MyMoney Pvt Ltd")
```

```
txt_obj = canvas.beginText(14, -2* cm)
txt_obj.setFont("Helvetica-Bold", 18)
txt_obj.setWordSpace(3)
txt_obj.textOut("Expense information")
txt_obj.moveCursor(0, 16)
txt_obj.setFont("Helvetica", 15)
txt_lst = [
    "Name : " + str(session['name']),
    "Email : " + str(session['email']),
    "User ID : " + str(session['uid']),
    "Budget : Rs. " + str(session['actualBudget']),
    "Current Balance : Rs. " + str(session['balance']),
]
for line in txt_lst:
    txt_obj.textOut(line)
    txt_obj.moveCursor(0, 16)
canvas.drawText(txt_obj)
```

```
canvas.setFont("Helvetica", 15)
canvas.setFont("Helvetica-Bold", 15)
canvas.drawString(0, -6.5*cm, "Expense List : ")
canvas.setFont("Helvetica", 15)
table_data = [['Paid For', 'Amount( in Rs )', 'Category', 'Payment Mode', 'Date']]
```

```

for e in expenseList:
    table_data.append(e)

t = Table(table_data, colWidths=[180, 100, 90, 120, 60], rowHeights=30)
style = [('BACKGROUND',(0,1),(4,(len(table_data))),colors.lightblue),
        ('ALIGN',(0,-1),(-1,-1),'CENTER'),
        ('BOX', (0,0), (-1,-1), 0.25, colors.black),
        ('INNERGRID', (0,0), (-1,-1), 0.25, colors.black),
        ('FONTSIZE', (0,0), (-1,-1), 10),
        ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
        ('VALIGN', (0, 0), (-1, -1), 'MIDDLE')]
t.setStyle(tblstyle=style)
t.wrapOn(canvas, 10, 10)
t.drawOn(canvas=canvas, x=20, y=-15.5*cm)

canvas.showPage()
canvas.save()
return buffer

```

## **9.RESULTS**

### **9.1 Performance Matrics**

The new system has overcome most of the limitations of the existing system and works according to the design specification given. The project what we have developed is work more efficient than the other income and expense tracker. The project successfully avoids the manual calculation for avoiding calculating the income and expense per month. The modules are developed with efficient and also in an attractive manner. The developed systems dispense the problem and meet the needs of by providing reliable and comprehensive information. All the requirements projected by the user have been met by the system. The newly developed system consumes less processing time and all the details are updated and processed immediately. Since the screen provides online help messages and is very userfriendly, any user will get familiarized with its usage. Module s are designed to be highly flexible so that any failure requirements can be easily added to the modules without facing many problems. The best organizations have a way of tracking and handling these reimbursements. This ideal practice guarantees that the expenses tracked are accurately and in a timely manner. From a company perspective, timely settlements of these expenses when tracked well will certainly boost your employees' morale Quickly Determine How Profitable Business Is Effective expense tracking and reporting to avoid conflict

## **10.ADVANTAGES & DISADVANTAGES**

### **Advantages:**

It's simple to set up and use. When you're creating your own method of tracking your finances, you first have to figure out how you're going to do that. Are you going to use pen and paper, or software, or an excel spreadsheet? What are you going to track? How are you going to input that data, and how often are you going to do it? With an automated app, it tracks everything for you in real time. It has a wealth of information, so no matter what data you feel is important to track, it is all there and available for you – you just need to take a look to see it. There's an easy user interface for everything as well. Whether you want to set up a budget, track a type of expense, or look over your financial history, there's a tab or an option ready and waiting for you. it's instantaneous

The application will track all of your data for you. It doesn't do it once a week or once a month, like you might if you were doing it manually. All the information is automatically brought right into your account as soon as it is available. That means that you have a day by day way of checking to ensure that you're on track and moving in the right direction. I've had it where Mint.com actually helped me react very quickly to a fraudulent transaction from an ATM machine. Someone had gotten a hold of our bank account details through a card skimmer (most likely), and they withdrew a bunch of cash. Within a few hours, I had already been made aware of it through Mint, contacted my bank, and started the process of getting my money back, all thanks to the automated system that was already in place.

### **Disadvantage**

Your information is less secure, and probably being used and sold. If the service is free, then the product is you. Mint.com, like other financial apps, is a free service. They have to pay their bills somehow, so regardless of what their privacy policy may or may not say, just assume that your spending history and trends are going to be recorded and analyzed, by someone, somewhere. Now, you shouldn't have to worry about credit card fraud identity theft these companies are large enough and secure enough that you'll never have to worry about something like that. Just recognize that your information, most likely anonymous, will be used and potentially even sold. Personally, I have no problem with that, but if you do, then make sure you avoid these types of services.

Automating everything to do with your finances can make you financially lazy. If your

bills are paid automatically and your finances are track automatically, then what is there left for you to do? Not a lot, to be honest. So you might stop caring about what you are spending and where your money is going. Eventually you may look at your Mint data and realize that you've blown your budget over the last two months, but by then it is too late. So if you do choose to use this program, ensure that you are also being diligent in checking in on your finances. Set up a weekly or biweekly check for yourself to go through your finances and hit on all the important points.



## **11.CONCLUSION**

"Personal Expense Tracker" is an application. Daily Expense Tracker is a gadget that being developed to help customers in budget planning. It offers end customers to file their earnings and costs within the finances that have been planned beforehand. Furthermore, customers are capable to hold song on their spending so they are not wasting their money barring doubt. Last however clearly not the least, they are able to get right of entry to the gadget anytime and anywhere that. After making this application we guarantee that this software will assist its customers to manipulate the value of their daily expenditure. It will show to be beneficial for the human beings who are pissed off with their day-by-day price range management, because of amount of costs and needs to control cash and to preserve the report of each day fee which may also be useful to trade their way of spending money. In short, this application will help its customers to overcome the wastages of money.

## **12.FUTURE SCOPE**

The Future Enhancements of the application can be allowed to support in all the upcoming android versions . History can be set to view all the details in the app even if the particular data is deleted from the database. Statistics could be prepared based on the Income, Expense details of the user. Sharing files via Bluetooth, WhatsApp can be allowed. Printing the details of the particular income or expense details can be made. Some of the extra components are like enabling users to register to the application using existing email or social network account, it will synchronize the users profile data to the application

## 13.APPENDIX

### Source Code

#### feature.py

```
from flask import Blueprint, redirect, render_template, url_for, session, request, send_file
import ibm_db
from io import BytesIO
from reportlab.lib.utils import ImageReader
from reportlab.platypus import Table
from reportlab.pdfgen.canvas import Canvas
from reportlab.pdfbase import pdfmetrics
from reportlab.pdfbase.ttfonts import TTFont
from reportlab.lib.pagesizes import A4
from reportlab.lib import colors
from reportlab.lib.units import cm
from datetime import datetime
from .connect import connection
from .email import sendMail
feature = Blueprint("feature", __name__, url_prefix="/")
con = connection
```

```
@feature.route('/dashboard')
def dashboard():
    if 'auth' in session:
        if session['auth']:
            qry="SELECT * FROM ACCOUNT WHERE user_id=?"
            stmt1=ibm_db.prepare(con, qry)
            ibm_db.bind_param(stmt1,1,session['uid'])
            ibm_db.execute(stmt1)
            resp2=ibm_db.fetch_assoc(stmt1)

            uid = session['uid']
            q = "SELECT * from EXPENSE WHERE user_id=? ORDER BY EXP_ID DESC"
            s = ibm_db.prepare(con,q)
            ibm_db.bind_param(s,1,uid)
            ibm_db.execute(s)

            r = ibm_db.fetch_assoc(s)
            expList = []
            while r!=False:
```

```

li=[r['TITLE'],r['AMOUNT'], r['DATE_TIME'],r['PAYMENT_MODE']]
expList.append(li)
r = ibm_db.fetch_assoc(s)

# print(expList)

result = [ ]
paymentmethods = [ ]
pay = {}
total_balance = 0
while resp2 != False:
    t = [resp2['NAME'],resp2['BALANCE'],resp2['ACC_NO']]
    name=resp2['NAME']
    acno = str(resp2['ACC_NO'])
    accid = str(resp2['ACC_ID'])
    session[accid]=resp2['BALANCE']
    upi = resp2['UPI']
    cc = resp2['CREDITCARD']
    dc = resp2['DEBITCARD']
    cheque = resp2['CHEQUE']
    temp = [ ]
    if upi:
        temp.append("UPI")
        paymentmethods.append([name+"-UPI",resp2['ACC_ID']])
    if cc:
        temp.append("Credit Card")
        paymentmethods.append([name+"-Credit",resp2['ACC_ID']])
    if dc:
        temp.append("Debit Card")
        paymentmethods.append([name+"-Debit",resp2['ACC_ID']])
    if cheque:
        temp.append("Cheque")
        paymentmethods.append([name+"-Cheque",resp2['ACC_ID']])
    if acno=="00":
        temp.append("Cash")
    temp.append(accid)
    pay[name+'-' +acno] = temp
    total_balance+=float(resp2['BALANCE'])
    result.append(t)
    resp2 = ibm_db.fetch_assoc(stmt1)
# print(result)

```

```

        # print(pay)
        # print(paymentmethods)
        session['accounts'] = result
        session['balance'] = total_balance
        session['payment'] = paymentmethods
        session['pay'] = pay
        session['expense'] = expList
        return render_template('dashboard.html')
    else:
        return redirect(url_for('auth.login'))
return redirect(url_for('auth.login'))

@feature.route('/addAccount', methods=['GET','POST'])
def addAccountDetail():
    if(request.method == 'POST'):
        if 'auth' in session:
            if session['auth']:

                print(request.form)

                name = request.form['accname']
                accno = request.form['accno']
                balance = request.form['balance']
                upi = True if bool(request.form['upi']) if 'upi' in request.form else False else False
                cc = True if bool(request.form['cc']) if 'cc' in request.form else False else False
                dc = True if bool(request.form['dc']) if 'dc' in request.form else False else False
                cheque = True if bool(request.form['Cheque']) if 'Cheque' in request.form else False else
False
                uid = session['uid']

                qry2 = "INSERT INTO
ACCOUNT(name,balance,acc_no,upi,creditCard,debitCard,cheque,user_id) VALUES(?,?,?,?,?,?,?)"
                stmt2 = ibm_db.prepare(con,qry2)
                ibm_db.bind_param(stmt2,1,name)
                ibm_db.bind_param(stmt2,2,balance)
                ibm_db.bind_param(stmt2,3,accno)
                ibm_db.bind_param(stmt2,4,upi)
                ibm_db.bind_param(stmt2,5,cc)
                ibm_db.bind_param(stmt2,6,dc)
                ibm_db.bind_param(stmt2,7,cheque)
                ibm_db.bind_param(stmt2,8,uid)

```

```

resp2 = ibm_db.execute(stmt2)

# print(resp2)
return redirect(url_for('feature.dashboard'))
return render_template("login.html")
elif request.method == 'GET':
    return render_template("accountDetails.html")

@feature.route('/addExpense', methods=['GET', 'POST'])
def addExpense():
    if(request.method == 'POST'):
        if 'auth' in session:
            if session['auth']:

                title = request.form['title']
                amount = request.form['price']
                description = request.form['description']
                paymenttype = request.form['paymenttype']
                category = request.form['category']
                categoryList = ['Food and Drinks','Transportation','Entertainment','Mobile','Investment']
                if category not in categoryList:
                    category = 'Entertainment'

                accdet = paymenttype.split('--')
                accid = accdet[-1]
                paymode = accdet[0]
                uid = session['uid']
                now = datetime.now()
                dt_string = now.strftime("%d/%m/%Y %H:%M:%S")

                currentMonth = datetime.now().month
                currentYear = datetime.now().year

                if float(session[accid])<float(amount):
                    return render_template("dashboard.html", message="Insufficient Funds..Choose
different account")

                qry = "INSERT INTO

```

```
EXPENSE(title,description,amount,payment_mode,date_time,acc_id,user_id,category,month,year) VALUES(?,?,?,?,?,?,?,?,?,?)"
```

```
stmt = ibm_db.prepare(con,qry)
f = request.files.get('img',)
```

```
ibm_db.bind_param(stmt,1,str(title))
ibm_db.bind_param(stmt,2,description)
ibm_db.bind_param(stmt,3,amount)
ibm_db.bind_param(stmt,4,paymode)
ibm_db.bind_param(stmt,5,dt_string)
ibm_db.bind_param(stmt,6,accid)
ibm_db.bind_param(stmt,7,uid)
ibm_db.bind_param(stmt,8,category)
ibm_db.bind_param(stmt,9,currentMonth)
ibm_db.bind_param(stmt,10,currentYear)
```

```
resp = ibm_db.execute(stmt)
# print(resp)
```

```
qry2 = "UPDATE ACCOUNT SET BALANCE=BALANCE-? WHERE ACC_ID=?"
# print(qry)
```

```
budget = float(session['budget'])
amt = float(amount)
```

```
if(budget-amt)<=0:
    message = "
```

Hey {0},

Warning!!.. your Expenses have exceeded the planned budget.

Please Plan your future expenses accordingly!

Regards

MyMoney Team

```
".format(session['name'])
sc = sendMail("Alert Budget Exceeded !!",message,session['email'])
# print("Mail Status : "+str(sc))
budget-=amt
session['budget'] = str(budget)
```

```

stmt2 = ibm_db.prepare(con,qry2)
ibm_db.bind_param(stmt2,1,amount)
ibm_db.bind_param(stmt2,2,accid)

resp2 = ibm_db.execute(stmt2)
# print(resp2)

return redirect(url_for('feature.dashboard'))
return redirect(url_for('auth.login'))
return redirect(url_for('auth.login'))
return redirect(url_for('auth.login'))

```

```
@feature.route('/charts')
```

```
def charts():
```

```
    if(request.method == 'GET'):
```

```
        if 'auth' in session:
```

```
            if session['auth']:
```

```
                uid = session['uid']
```

```
                categoryData=[0,0,0,0,0]
```

```
                # categoryData=['0','0','0','0','0']
```

```
                qry="SELECT category,SUM(amount) as AMOUNT FROM EXPENSE WHERE USER_ID=?
GROUP BY category"
```

```
                stmt = ibm_db.prepare(con,qry)
```

```
                ibm_db.bind_param(stmt,1,uid)
```

```
                resp = ibm_db.execute(stmt)
```

```
                var = ibm_db.fetch_assoc(stmt)
```

```
                while var!=False:
```

```
                    # print(var)
```

```
                    if(var['CATEGORY']=='Investment'):
```

```
                        categoryData[4]=int(var['AMOUNT'])
```

```
                    elif(var['CATEGORY']=='Transportation'):
```



```

        categoryData[1]=int(var['AMOUNT'])
    elif(var['CATEGORY']=='Entertainment'):
        categoryData[2]=int(var['AMOUNT'])
    elif(var['CATEGORY']=='Mobile'):
        categoryData[3]=int(var['AMOUNT'])
    elif(var['CATEGORY']=='Food and Drinks'):
        categoryData[0]=int(var['AMOUNT'])
    var = ibm_db.fetch_assoc(stmt)

paymentType = [0,0,0,0,0]
# paymentType = ['0','0','0','0','0']

qry1 = "SELECT payment_mode, SUM(AMOUNT) as AMOUNT from EXPENSE WHERE
user_id=? GROUP BY payment_mode"

stmt2 = ibm_db.prepare(con,qry1)

ibm_db.bind_param(stmt2,1,uid)

resp2 = ibm_db.execute(stmt2)

data = ibm_db.fetch_assoc(stmt2)

while data:

    # print(data)

    if(data['PAYMENT_MODE']=='UPI'):
        paymentType[1] = int(data['AMOUNT'])
    elif(data['PAYMENT_MODE']=='Cash'):
        paymentType[0] =int(data['AMOUNT'])
    elif(data['PAYMENT_MODE']=='Debit Card'):
        paymentType[2] = int(data['AMOUNT'])
    elif(data['PAYMENT_MODE']=='Credit Card'):
        paymentType[3] = int(data['AMOUNT'])
    elif(data['PAYMENT_MODE']=='Cheque'):
        paymentType[4] = int(data['AMOUNT'])

    data= ibm_db.fetch_assoc(stmt2)

# print(categoryData)

```

```

# print(paymentType)

qry2 = "SELECT * from EXPENSE WHERE user_id=? ORDER BY EXP_ID DESC"
# qry2 = "SELECT * FROM EXPENSE WHERE USER_ID=? ORDER BY EXP_ID DESC"

stmt3 = ibm_db.prepare(con,qry2)

ibm_db.bind_param(stmt3,1,uid)

resp3 = ibm_db.execute(stmt3)
dt = ibm_db.fetch_assoc(stmt3)
expHistory = []

while dt:
    t = []
    t.append(dt['TITLE'])
    t.append(dt['AMOUNT'])
    t.append(dt['CATEGORY'])
    t.append(dt['PAYMENT_MODE'])
    t.append(str(dt['DATE_TIME']))

    expHistory.append(t)

    dt = ibm_db.fetch_assoc(stmt3)

# print(expHistory)

return
render_template("chart.html",categoryData=categoryData,paymentType=paymentType,expense
History=expHistory)
return redirect(url_for('auth.login'))

@feature.route('/profile', methods=['GET', 'POST'])
def profile():
    if(request.method == 'POST'):
        if 'auth' in session:
            if session['auth']:

                name = request.form['name']
                password = request.form['password']

```

```
budget = request.form['budget']
uid = session['uid']
qry = "UPDATE USER SET NAME=?,PASSWORD=?,BUDGET=? WHERE USER_ID=?"
```

```
stmt = ibm_db.prepare(con,qry)
```

```
ibm_db.bind_param(stmt,1,name)
ibm_db.bind_param(stmt,2,password)
ibm_db.bind_param(stmt,3,budget)
ibm_db.bind_param(stmt,4,uid)
```

```
resp = ibm_db.execute(stmt)
```

```
# print(resp)
```

```
session['name'] = name
session['budget'] = budget
session['actualBudget']=budget
```

```
return redirect(url_for('feature.dashboard'))
return redirect(url_for('auth.login'))
return redirect(url_for('auth.login'))
```

```
elif request.method=='GET':
    return render_template('profile.html')
```

```
@feature.route("/pdf")
```

```
def pdfGenerator():
```

```
    if(request.method == 'GET'):
        if 'auth' in session:
            if session['auth']:
```

```
                qry = "SELECT * FROM EXPENSE WHERE USER_ID=? ORDER BY EXP_ID;"
```

```
                stmt = ibm_db.prepare(con,qry)
                uid = session['uid']
                ibm_db.bind_param(stmt,1,uid)
```

```
                resp = ibm_db.execute(stmt)
```

```

data = ibm_db.fetch_assoc(stmt)

result = []

while data:
    t = []

    t.append(data['TITLE'])
    t.append(data['AMOUNT'])
    t.append(data['CATEGORY'])
    t.append(data['PAYMENT_MODE'])
    t.append(str(data['DATE_TIME'][:10]))

    result.append(t)

    data = ibm_db.fetch_assoc(stmt)

# print(result)

buffer = create_pdf(result)

buffer.seek(0)

return send_file(buffer, as_attachment=True,
mimetype='application/pdf',download_name="{name}-
report.pdf".format(name=str(session['name'])))

return redirect(url_for('auth.login'))
return redirect(url_for('auth.login'))

def create_pdf(expenseList):
    buffer = BytesIO()
    canvas = Canvas(buffer, pagesize=A4)
    WIDTH, HEIGHT = A4
    MARGIN = 0.5 * cm
    canvas.translate(MARGIN, HEIGHT-MARGIN)

    canvas.setFont("Helvetica-Bold", 15)

```

```
canvas.drawString(WIDTH//3+10, 0, "Expense Report")
```

```
canvas.setFont("Helvetica", 12)
canvas.drawString(430, -0.9*cm, "MyMoney Pvt Ltd")
canvas.setStrokeGray(0)
canvas.line(0, -1*cm, WIDTH - 2*MARGIN, -1*cm)
```

```
canvas.line(0, -26.6*cm, WIDTH - 2*MARGIN, -26.6*cm)
canvas.drawString(430, -26.5*cm, "MyMoney Pvt Ltd")
```

```
txt_obj = canvas.beginText(14, -2* cm)
txt_obj.setFont("Helvetica-Bold", 18)
txt_obj.setWordSpace(3)
txt_obj.textOut("Expense information")
txt_obj.moveCursor(0, 16)
txt_obj.setFont("Helvetica", 15)
txt_lst = [
    "Name : " + str(session['name']),
    "Email : " + str(session['email']),
    "User ID : " + str(session['uid']),
    "Budget : Rs. " + str(session['actualBudget']),
    "Current Balance : Rs. " + str(session['balance']),
]
```

```
for line in txt_lst:
    txt_obj.textOut(line)
    txt_obj.moveCursor(0, 16)
canvas.drawText(txt_obj)
```

```
canvas.setFont("Helvetica", 15)
canvas.setFont("Helvetica-Bold", 15)
canvas.drawString(0, -6.5*cm, "Expense List : ")
canvas.setFont("Helvetica", 15)
table_data = [['Paid For', 'Amount( in Rs )', 'Category', 'Payment Mode', 'Date']]
```

```
for e in expenseList:
    table_data.append(e)
```

```
t = Table(table_data, colWidths=[180, 100, 90, 120, 60], rowHeights=30)
style = [('BACKGROUND',(0,1),(4,(len(table_data))),colors.lightblue),
        ('ALIGN',(0,-1),(-1,-1),'CENTER'),
```

```

        ('BOX', (0,0), (-1,-1), 0.25, colors.black),
        ('INNERGRID', (0,0), (-1,-1), 0.25, colors.black),
        ('FONTSIZE', (0,0), (-1,-1), 10),
        ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
        ('VALIGN', (0, 0), (-1, -1), 'MIDDLE')]
t.setStyle(tblstyle=style)
t.wrapOn(canvas, 10, 10)
t.drawOn(canvas=canvas, x=20, y=-15.5*cm)

canvas.showPage()
canvas.save()
return buffer

```

## app.py

```

# from flask import Flask,render_template,request,session,redirect,url_for
# from connect import get_db_connection

# from controllers.auth import auth

# app = Flask(__name__)
# app.secret_key = 'abcde'
# con = 0

# @app.route('/')
# def home():
#     if 'auth' in session:
#         if not session['auth']:
#             return redirect(url_for('login'))
#     return render_template("home.html")

# def connection():
#     return con

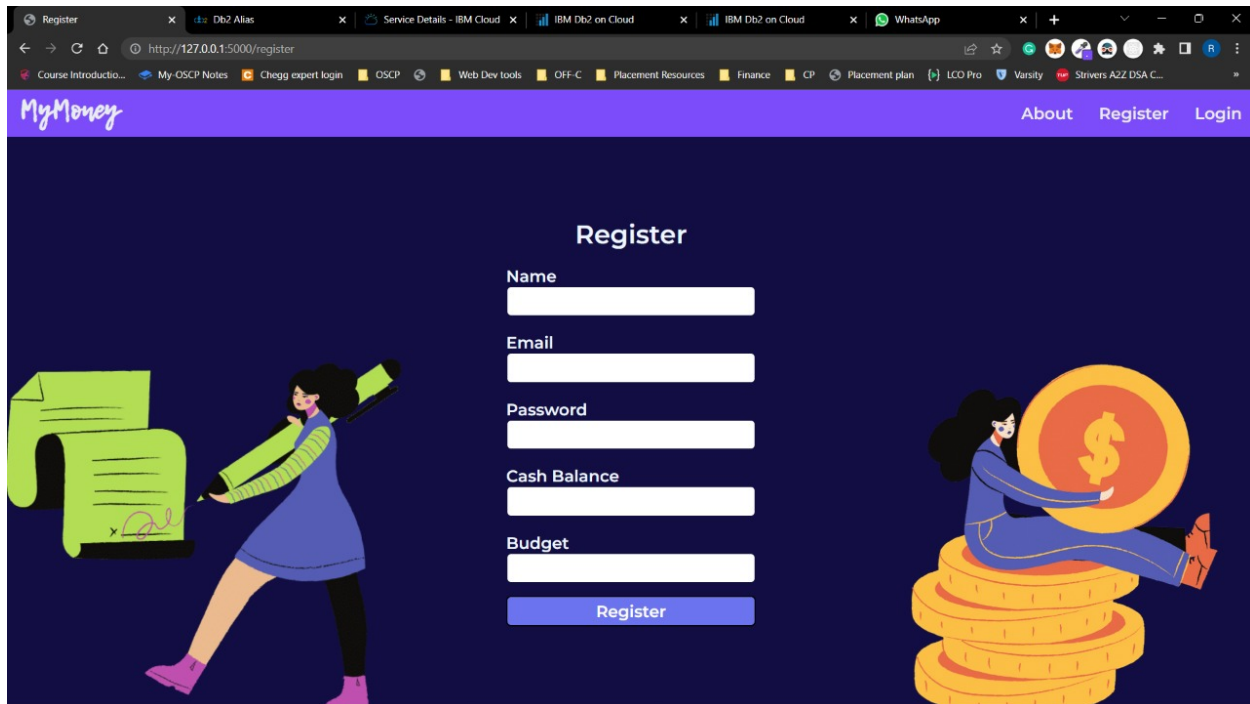
from web import createApp

if(__name__=="__main__"):
    # con = get_db_connection()

```

```
# app.register_blueprint(auth, url_prefix="/")
app = createApp()
app.run(host="0.0.0.0",port=int("5000"),debug=True)
```

## OUTPUT



The screenshot shows a web browser window with the URL `http://127.0.0.1:5000/register`. The page has a purple header with the "MyMoney" logo and navigation links for "About", "Register", and "Login". The main content area is dark blue and features a "Register" heading. Below the heading are five input fields: "Name", "Email", "Password", "Cash Balance", and "Budget". A blue "Register" button is positioned below the "Budget" field. The page is decorated with two illustrations: on the left, a woman in a blue dress and pink shoes is writing on a large green document; on the right, a woman in a blue suit is sitting on a stack of gold coins, holding a large gold coin with a dollar sign.

Register

Name

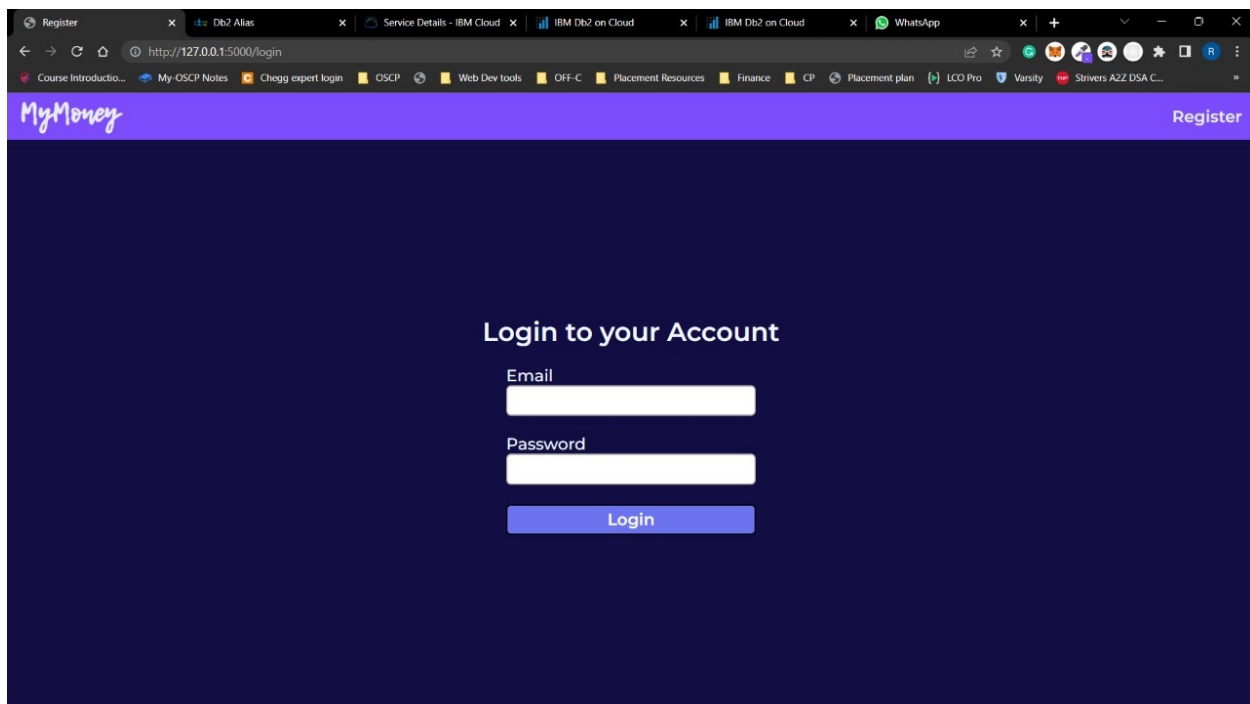
Email

Password

Cash Balance

Budget

Register



The screenshot shows the same web browser window with the URL `http://127.0.0.1:5000/login`. The page has a purple header with the "MyMoney" logo and a "Register" link. The main content area is dark blue and features a "Login to your Account" heading. Below the heading are two input fields: "Email" and "Password". A blue "Login" button is positioned below the "Password" field.

Register

Login to your Account

Email

Password

Login

Account Details

Db2 Alias

Service Details - IBM Cloud

IBM Db2 on Cloud

IBM Db2 on Cloud

WhatsApp

http://127.0.0.1:5000/addAccount

Course Introductio...My OSCP NotesChegg expert loginOSCPWeb Dev toolsOFF-CPlacement ResourcesFinanceCPPlacement planLCO ProVarsityStrivers A2Z DSA C...

MyMoney

ProfileAdd AccountLogout

## Add Account Details

Account Title

Account Number

Account Balance (in Rupees)

Choose Payment Options that are enabled in this account

UPI

☒

Credit Card

☐

Debit Card

☒

Cheque

☐

Add Account

Dashboard

Db2 Alias

Service Details - IBM Cloud

IBM Db2 on Cloud

IBM Db2 on Cloud

http://127.0.0.1:5000/dashboard

Course Introductio...My OSCP NotesChegg expert loginOSCPWeb Dev toolsOFF-CPlacement ResourcesFinanceCPPlacement planLCO ProVarsityStrivers A2Z DSA C...

MyMoney

ProfileAdd AccountLogout

### Balance Available

Cash : Rs. 12242  
CUB : Rs. 44498  
Total : Rs.56740.0

### Expense History

Macbook pro 14/11/2022 12:45:39	Rs 90000
Burger 14/11/2022 15:31:24	Rs 258
Test Chart 16/11/2022 14:27:30	Rs 500

### Add Expense

Title

Amount in Rs

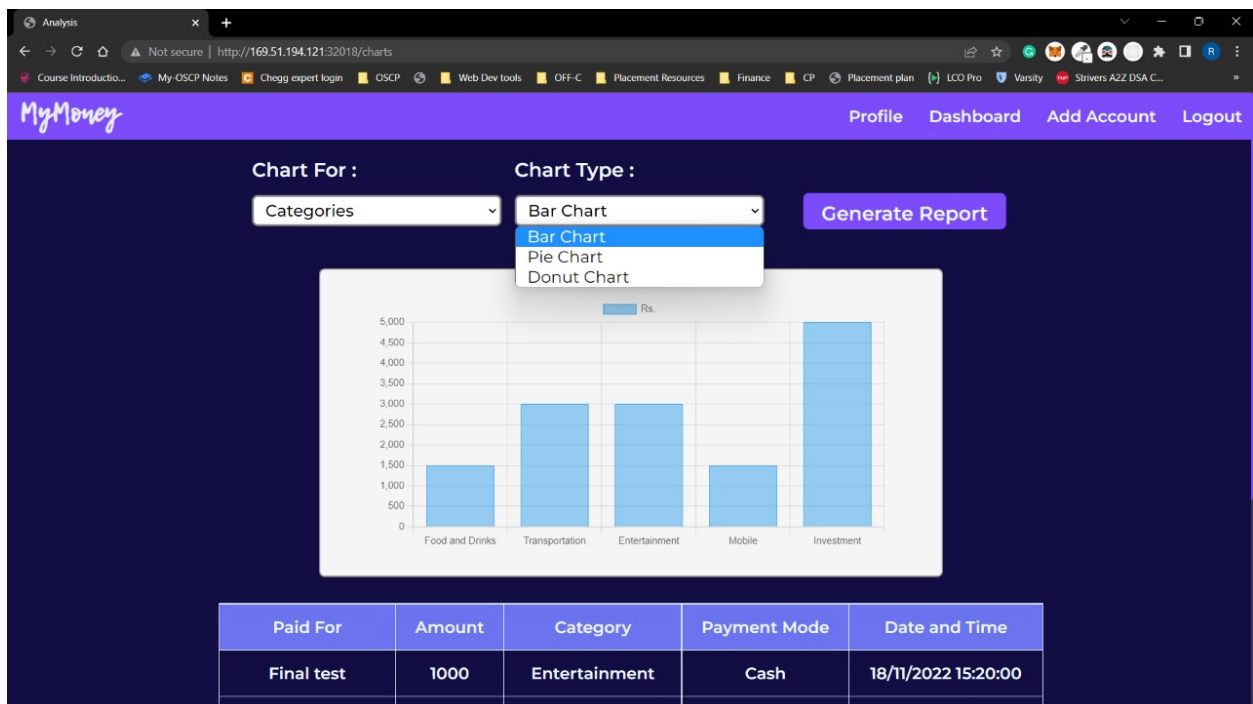
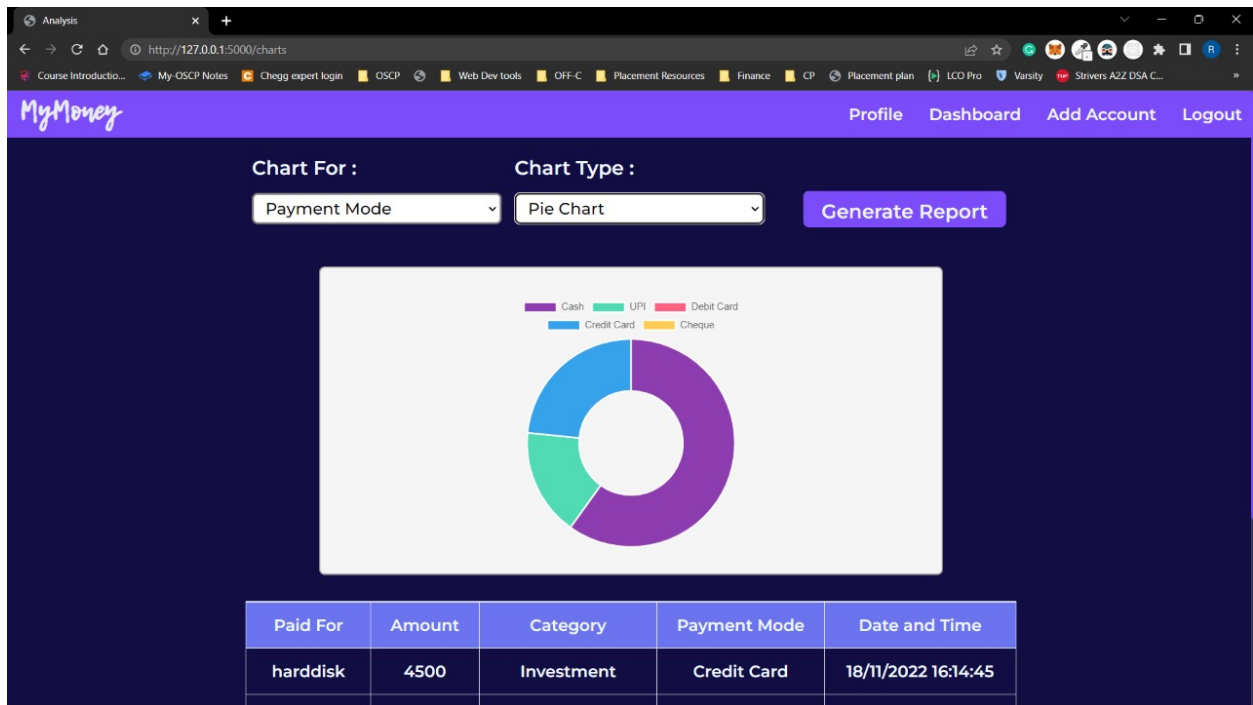
Description

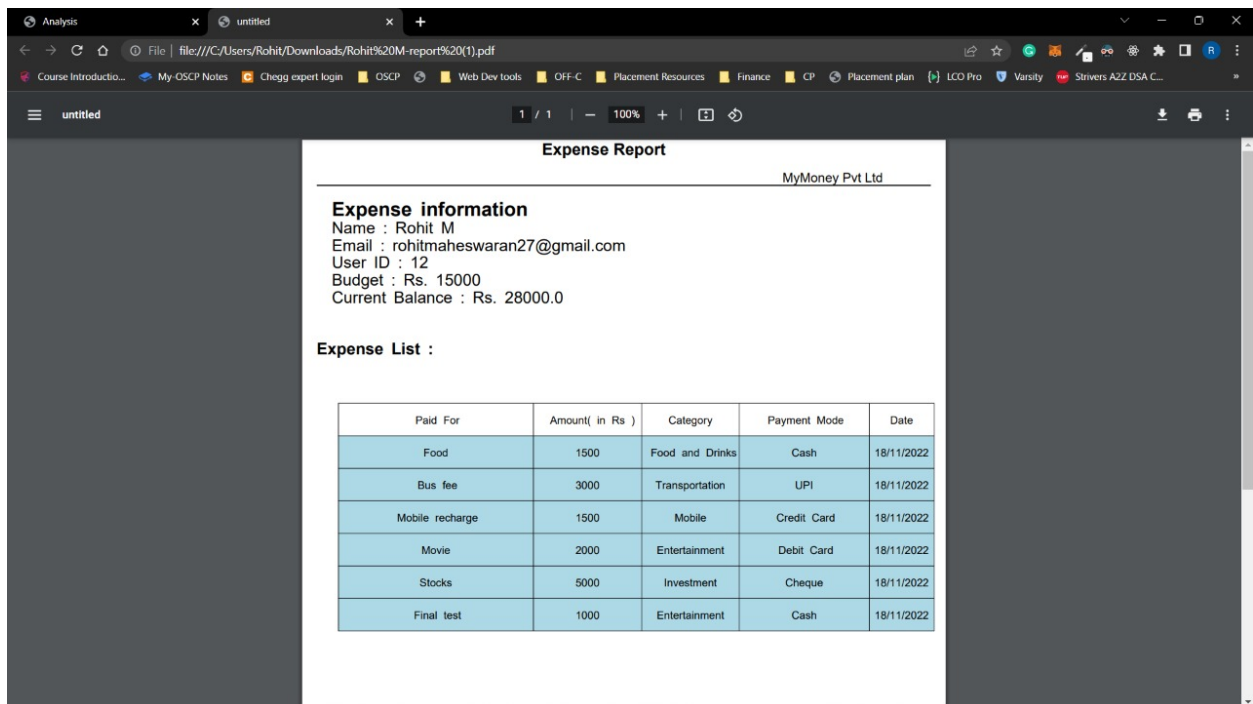
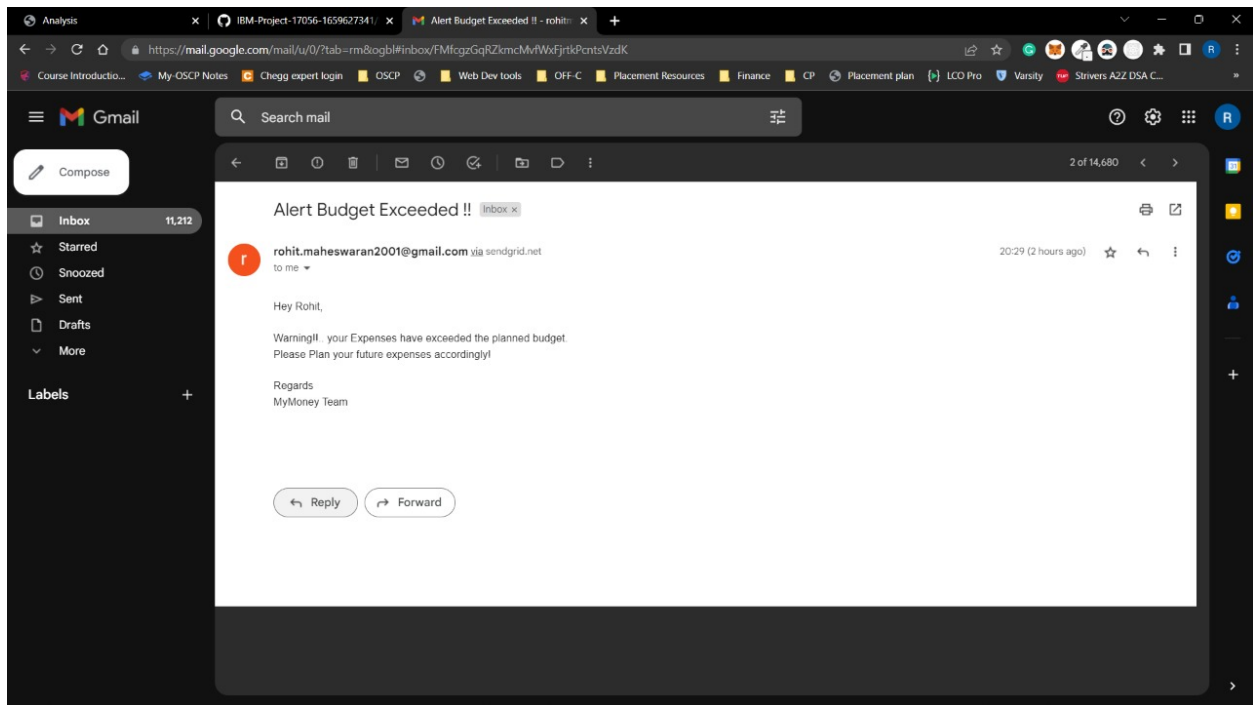
Food and Drinks

Choose Mode of Payment

Add







**GITHUB LINK**

<https://github.com/IBM-EPBL/IBM-Project-17056-1659627341>

**PROJECT DEMO LINK**

[https://youtu.be//q9B\\_hvYI\\_YM](https://youtu.be//q9B_hvYI_YM)

**Deployed Site**

<https://169.51.194.121:32018>