PROJECT REPORT

Team ID	**TEAM ID - PNT2022TMID48497**
Project Name	**PERSONAL EXPENSE TRACKER APPLICATION**

1. INTRODUCTION

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.

1.1 Project Overview

The personal expense tracker application is designed to track the users expense on a daily basis. This system splits your income based on your daily and monthly expense. If the daily & monthly expenses limit are exceeded, the application sends an alert email to the users mail. The personal expense tracker application produces a report at the end of the month/day and displays the chart for the expenses.

1.2 Purpose

- The main purpose of this personal expense tracker application is to reduce the difficulties in managing money in our day to day life.

- Most of the people cannot track their expense manually so this motivates the users to use an application that tracks their expenses and set limits for their expenses so that they are well aware of their expenses with the limit alert easily.

2. LITERATURE SURVEY

2.1 Existing problem

In the existing system, the data are stored in the local storage of the device and data handling is a tedious process. There is no proper assistance in the current system and virtualization does not provide full interoperability to the user. In this existing system traditional statistical approach is used. Email alert is not sent to the user when he exceeds the limit for the expense. In this existing system, month end statement is in .CVS file format.

2.2 References

PAPER– 1

TOPIC : A Smart Approach to Track Daily Expenses

AUTHOR : UP Singh, AK Gupta, Dr. B. Balamurugan

OVERVIEW :

In this paper, a Java GUI based application was proposed to assure that it will help its users to manage the cost of their daily expenditure. It will guide them and aware them of their daily expenses. The proposed design contained the basic modules for adding and viewing expenses, managing expense categories. Supports CRUD operations on expense data.

ADVANTAGES :

- Category-wise management of expenses.
- Daily, monthly, annual basis tracking.
- Simple and user-friendly.

DISADVANTAGES :

- Lack of visual analytics for expense data.
- Lack of support for splitting group expenses.
- Supports manual data monitoring only.

PAPER– 2

TOPIC : Expense Tracker

AUTHOR : Prof Miriam Thomas, Lekshmi P, and Dr. Mahalekshmi T

OVERVIEW :

Daily Expense Tracker System is designed to keep a track of Income-Expense of an organization on a day-to-day basis. This System divides the Income based on daily expenses. If exceed day's expense, system will calculate income and will provide new daily expense allowed amount. Daily expense tracking System will generate report at the end of month to show Income-Expense graph. And employees send reports to the manager for verification. Manager send final reports to administrator .Based on the final reports system predict the next month expense . It will helps to manage over all expense and income.

ADVANTAGES :

- Maintenance of expense data in the form of Excel sheets, CSV files, thereby avoiding entering individual expenses manually.
- Better visual analytics of data for various timelines.
- Supports handling for reimbursements.
- Least squares regression, a statistical procedure, is used to predict the expense limits.

DISADVANTAGES :

- Suitable for organization scale, too complex for personal use.
- Expense prediction is not really necessary for small transactions made on personal use.
- Involves the participation of 3 roles Admin, Manager, Employee.

PAPER– 3

TOPIC : Expense Tracker Application

AUTHOR : Velmurugan.R , Mrs.P.Usha

OVERVIEW :

This is an android based application that allows user to maintain a computerized diary to track expenses on a day-to-day basis to stay on budget and know expenses that is represented via a graphical representation with special features of distributing expenses in different categories

suitable for the user. Java, XML, MySQL is used. Filtering transaction views, view analytics and PDF report are also included.

ADVANTAGES :

- Has various components of updating and viewing users expenditure.
- User can track his expenses by choosing a day and using various filtering options to study expenses.
- Visualization using pie chart with percentage view shows graphical representation.

DISADVANTAGES :

- Doesn't support upcoming android versions.
- If a particular data is deleted, it cannot be viewed again.
- Statistics about income and expense detail of user can be prepared.

PAPER – 4

TOPIC : Online Income and Expense Tracker

AUTHOR : S. Chandini, T. Poojitha, D. Ranjith, V.J. Mohammed Akram, M.S. Vani, V. Rajyalakshmi

OVERVIEW :

It is a web application which is helpful to manage out income and expense as a daily or periodically or else whenever we want to remind and acts as an indicator or reminder example in the fastest world which we can't able to remember what are the things we have to do for the end of month and what are the payments we have to pay for the particular month.

ADVANTAGES :

- User friendly and data is maintained efficiently.
- Generates report at the end of week or month to show Income.
- Expense via multiple graphs.
- There is also an option to view owe and lend expenses which adds or gets deducted from the overall budget according without bothering the user.

DISADVANTAGES :

- Does not provide any option to handle shared expense of a group.
- Effort has to be made to include each and every transaction into the input field.

PAPER – 5

TOPIC : A Review on Budget Estimator Android Application

AUTHOR : Namita Jagtap, Priyanka Joshi, Aditya Kamble

OVERVIEW :

The system known as Budget Estimator is designed to manage the application user ‘s daily expenses in a more efficient and manageable way. This project is about mobile application Expenses system with geo location tracking, based on the location of the user, it using Google Places, to check, the available store in the area, provides a notification for offers purpose, In term of security design, this system may implement a login authentication such as OTP message to your mobile device, this function may bring more security confidence to user. To reduce manual calculations, we propose an application which is developed by android. This application allows users to maintain a digital automated diary.

ADVANTAGES :

- In this paper, an algorithm was proposed to show offers in nearby places using geo location tracking.
- This mobile application with 2-step verification method provides the security to the users.

DISADVANTAGES :

- Suitable for only Personal use.
- It does not provide any analytics.

2.3 Problem Statement Definition

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.

Problem Statement (PS)	**I am (Customer)**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	Financier	Use the expense calculation without using internet	It is impossible	It requires very high speed internet connection	Frustrated
PS-2	User	Make monthly financial calculations in my android application	There is no user friendliness	It requires more steps to add new extra table	Disappointed
PS-3	Accountant	Save my monthly calculations in dataset for future references	No storage or backup facility in other expense applications	The expense applications are collaborated with cloud	Anxious
PS -4	Employee	Track my expenses in monthly and in the end of the month I want to know the difference between	None of the applications has to display the difference between two months expense	It has to collaborated with cloud and track each and every year expense	Depressed

		previous month and this month			

3. IDEATION & PROPOSED SOLUTION

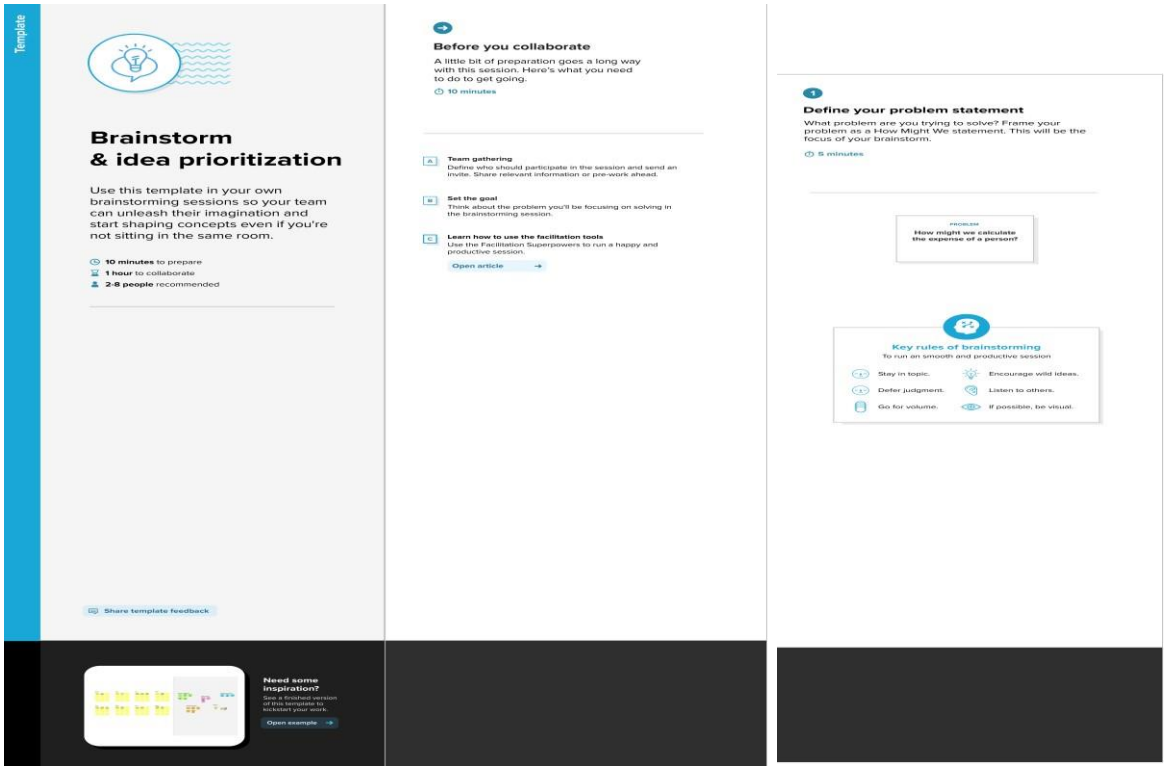
3.1 Empathy Map Canvas

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

3.2 Ideation and Brainstorming

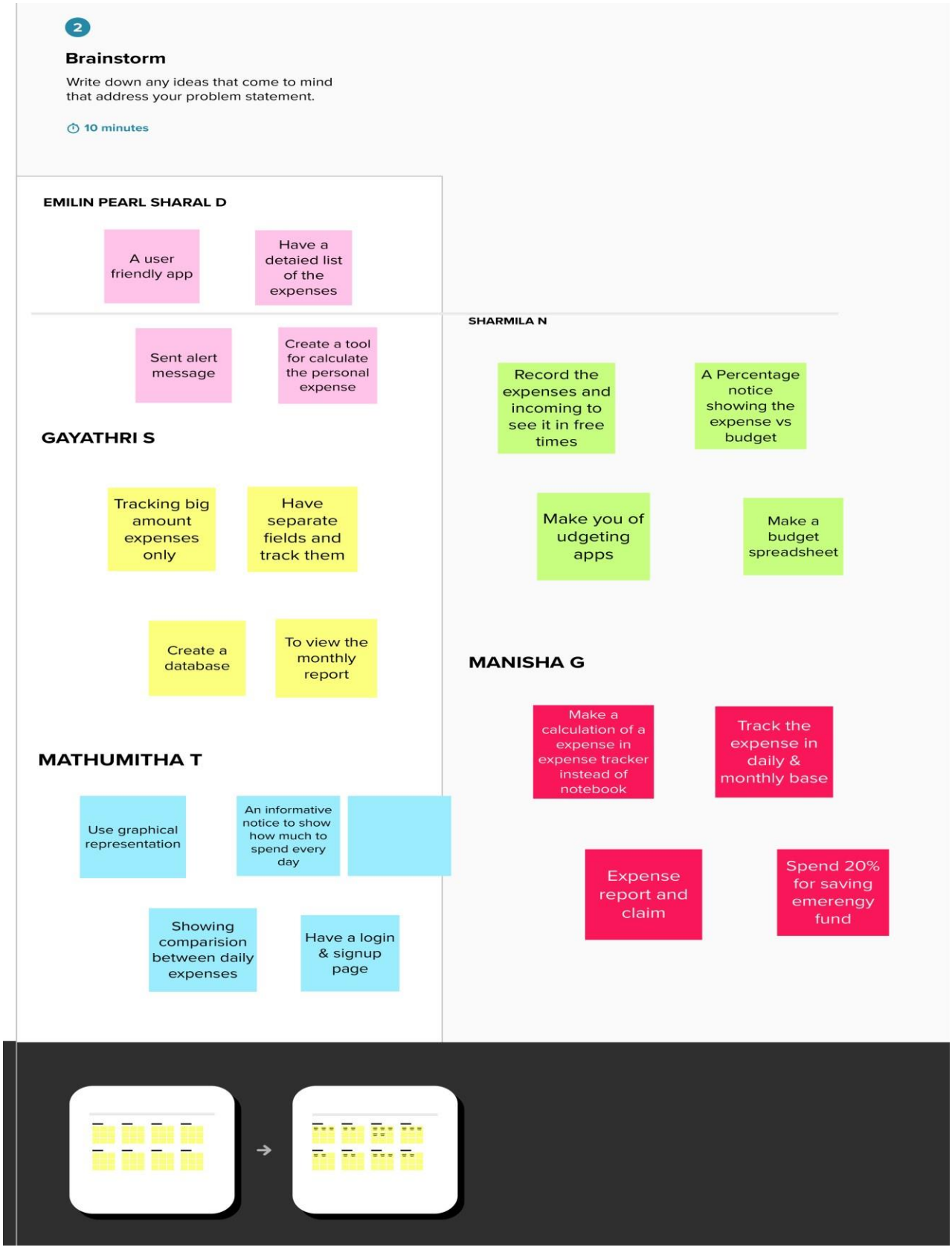
Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Step-1: Team Gathering, Collaboration and Select the Problem Statement



PNT2022TMID48497

Step-2: Brainstorm, Idea Listing and Group



PNT2022TMID48497

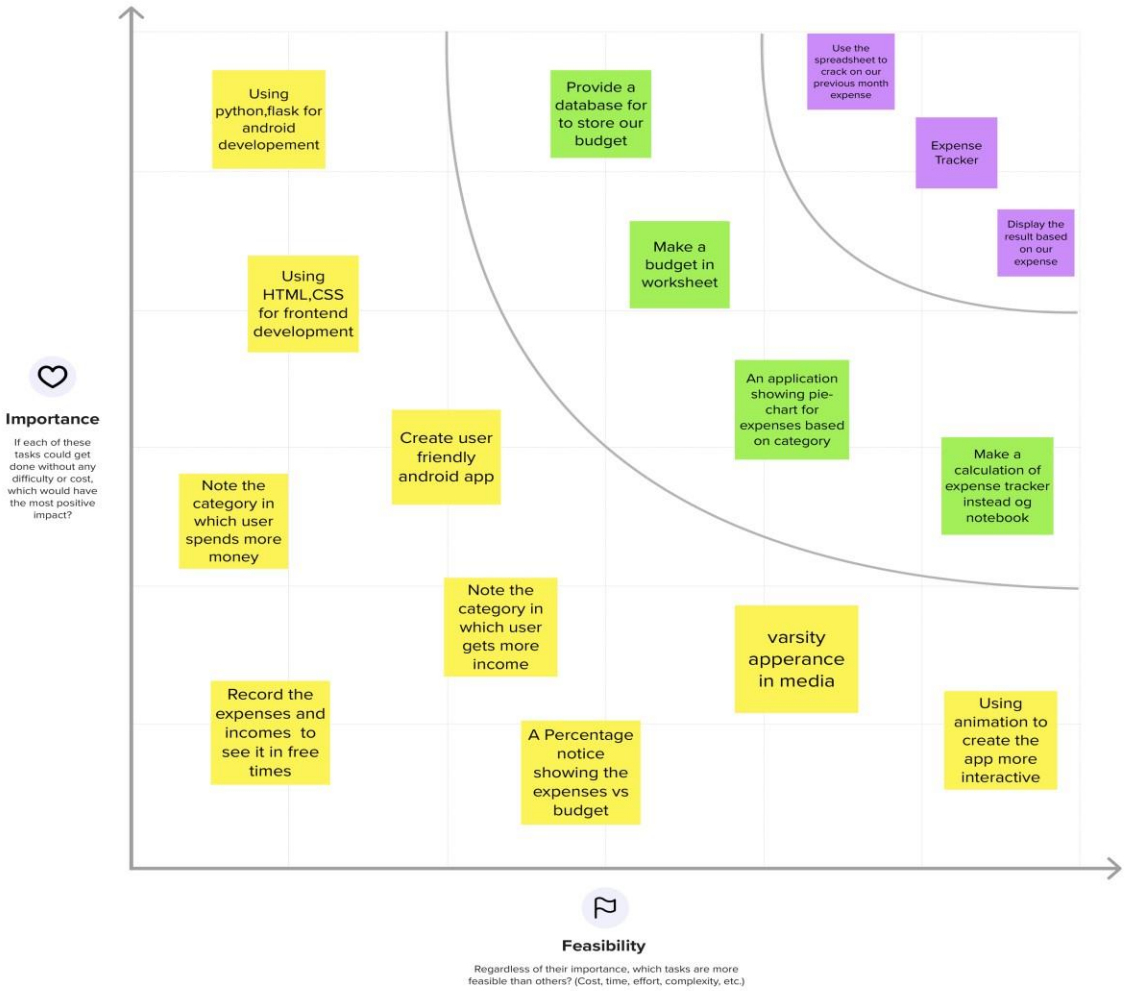
Step-3: Idea Prioritization

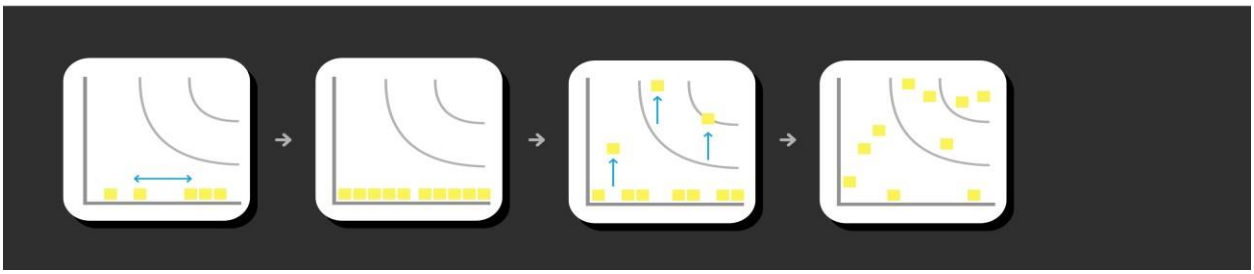
4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes





3.3 Proposed Solution

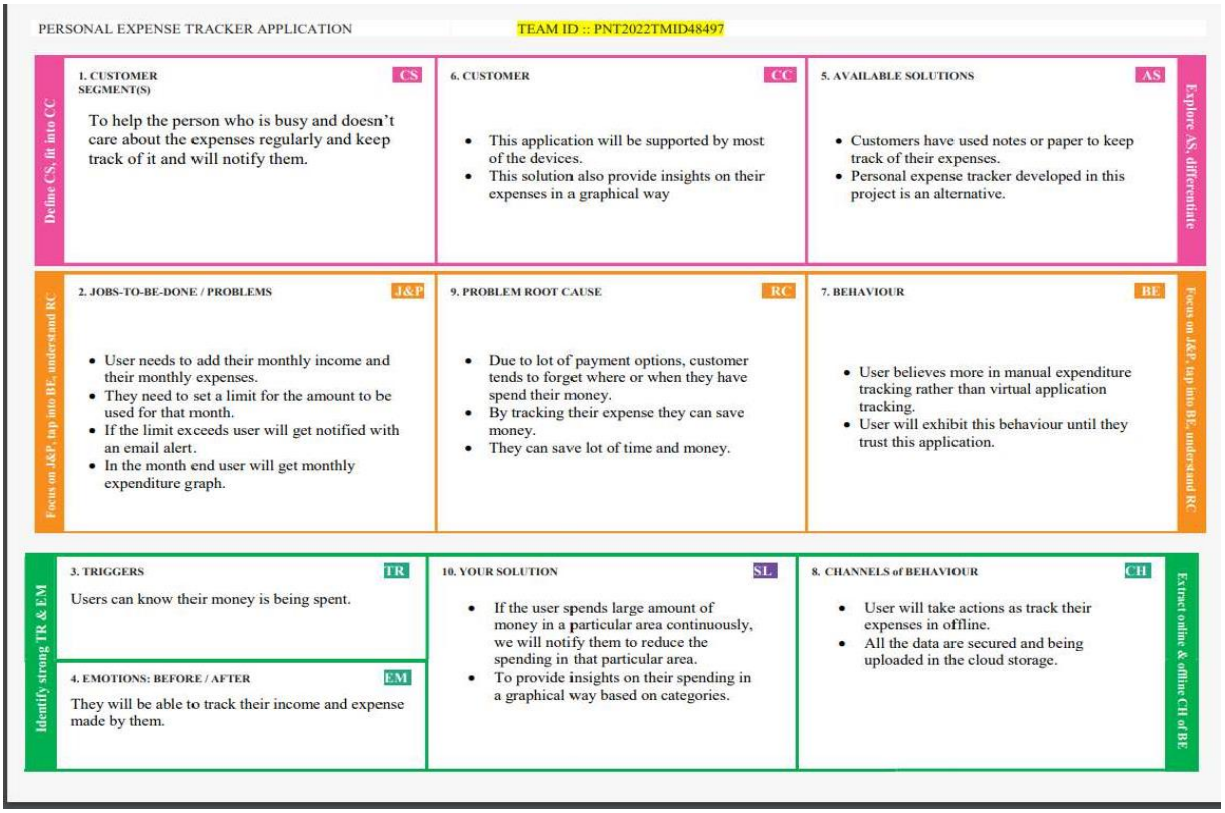
In this personal expense tracker project IBM DB2 cloud is used to store the data instead of storing in local storage. Here containerization is a concept that took over virtualization, which allows the user to run the application uniformly, and consistently on any infrastructure using the Dockers application. IBM Watson Assistant Chat bot is used to guide the user and explain about the application. In this system project backup details is recorded in IBM Cloud Foundry so in case of any failure, the information will be automatically roll backed to the latest checkpoint. Here our project is built using python flask that allows better scalability to this project. If the user exceeds the limit then he will be sent an alert email stating that he has exceeded his expense limit using Send Grid.

S. No.	Parameter	Description
1.	Problem Statement (Problemto be solved)	To track the expense of the individual person in the user friendly way.
2.	Idea / Solution description	Our application is to track the expense of the individual person and to calculate the total final amount to be spending this month and also to display the previous month expense to know the user to reduce the unnecessary expense this month.
3.	Novelty / Uniqueness	This application has to collaborated with cloud. The cloud storage is used to store the monthly expense results and to display to the user if they want.
4.	Social Impact / Customer Satisfaction	The personal expense tracker application has to track the expense and to do the expense calculation in an user friendly manner. It also helpsthe user to avoid the unnecessary wages.

5.	Business Model (Revenue Model)	Free trial for 1 month can be given tothe users, so that a significant user base is created. Following the free trial, the users can be given subscription for 3 months, 6 months or 1 year.
6.	Scalability of the Solution	More number of users can be managed effectively, since entire application is hosted on cloud. It also help us to review the expense calculations in the previous month.

3.4 Problem Solution Fit

The Problem Solution Fit is used to find a problem with your customer and that the solution you have realized for it actually solves the customer's problem.



4. REQUIREMENT ANALYSIS

4.1 Functional requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	Monthly refreshments	Personal expense tracker application shall allow the user to add the data to their expanses.
FR-3	Planner	It is help to show the graphical representation to the users about previous month expense results.
FR-4	Tracker	To track the expense flow is to be decreased or increased compared to the previous month and current month.
PR-5	Category	It will help the user to add new categories or to delete the existing categories in the application.

4.2 Non-Functional requirements

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

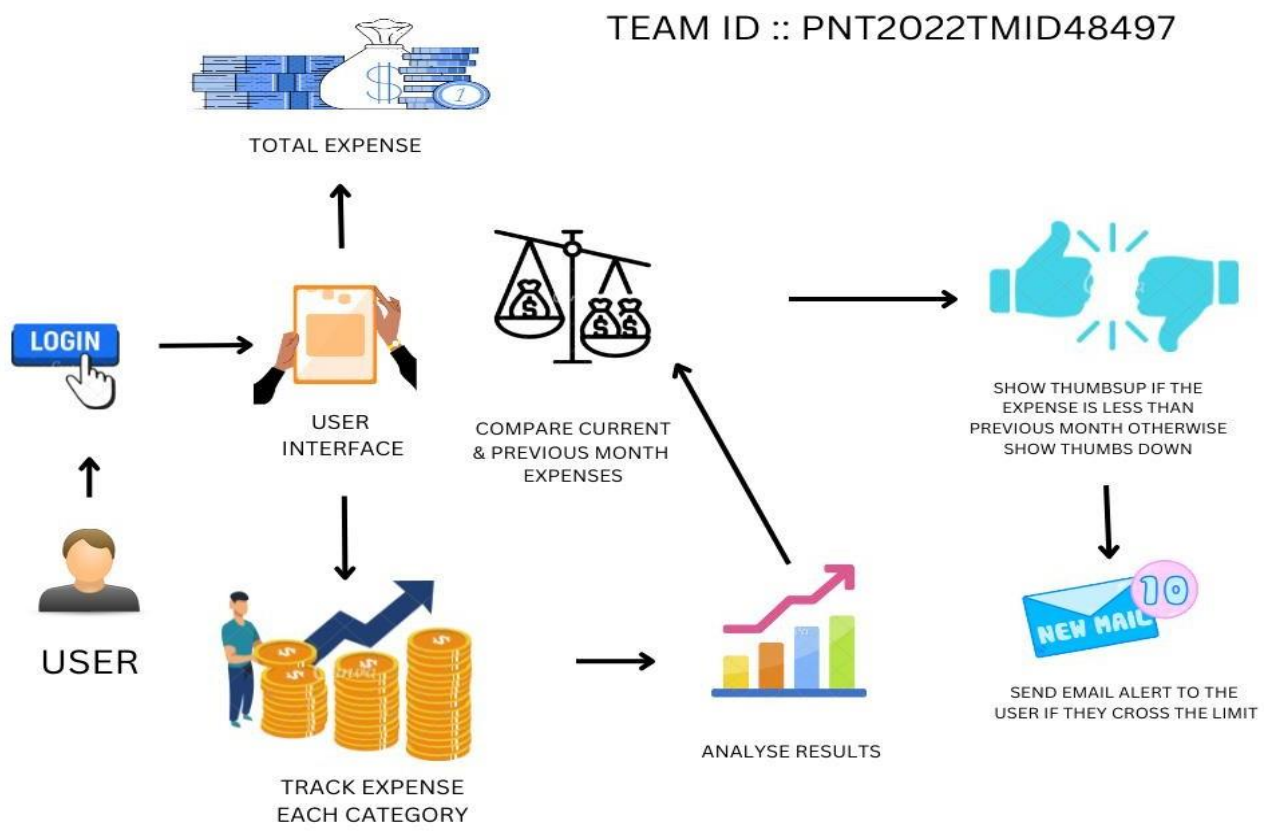
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	It is very much user friendly one. It is easy to calculate our expenses and track the money flow.
NFR-2	**Security**	More data security of the user bank account and payment data sheet.
NFR-3	**Reliability**	Each data is stored in the very well formed database sheet with heavy security.

NFR-4	**Performance**	It has to give the users to the options to add or delete the categories and to track the money flow. And then to alert the users if more money flow in the particular category.
NFR-5	**Availability**	It is the application to be available in offline and to track our money flow in monthly.
NFR-6	**Scalability**	It has the ability to add more than 25 categories in one month with backup option.

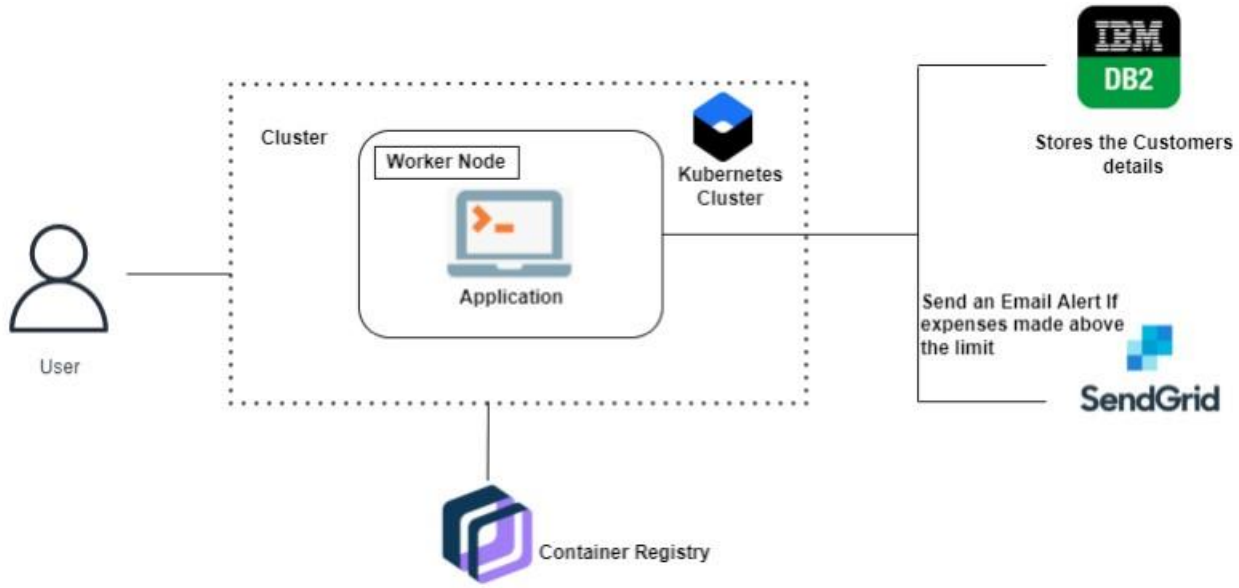
5. PROJECT DESIGN

5.1 Data Flow Diagrams

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



5.2 Solution & 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
		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 access my account through Gmail login	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can receive login confirmation and login credentials	High	Sprint-1

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
	Dashboard	USN-6	As a user, can access dashboard my to manage my expenses	Overall credit outlook	Low	Sprint-1
Customer (Web user)	Web user	USN-7	As a customer, can access the application using the web based platform also	Can have separate web page form	Medium	Sprint-1
Customer Care Executive	Expense management		As a customer care executive, periodically update and maintains expense application	Can have the login access when Admin permits	High	Sprint-1
Administrator	Creates and makes the application into use		As a administrator, is responsible for every expense count management	I can have the direct access to the application	High	Sprint-1

6. PROJECT PLANNING AND SCHEDULING

6.1 Sprit 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.	2	High	2
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	1
Sprint-1		USN-3	As a user, I can register for the application through Gmail	1	High	1
Sprint-1	Login	USN-4	As a user, I can log into the application by entering email & password	3	High	3

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint -2	Login	USN-5	As a user, after logging in, I will have to update my profile by providing all the required details	5	High	5
Sprint-3	Expense update	USN-6	As a user, I can add wallet balance and add or delete my expense	5	Medium	5
Sprint-3	Expense update	USN-7	As a user, I can update my expenses in the wallet periodically	2	Low	2
Sprint-4	Email alert	USN-8	As a user, I can fix some limit for spending within the wallet balance	3	High	3
Sprint-1	Email alert	USN-9	As a user, I will receive email notification if I reached the limit	2	High	2
Sprint-2	Verification	USN-10	As a user, I can verify the entered expenses correctly	5	High	5
Sprint-3	Comparison	USN-11	As a user, I can compare current month expense and previous month expense, if it is less than previous month or not.	3	High	3
Sprint-4	Output	USN-12	As a user, I can maintain the privacy with the help of username and password	3	High	5

6.2 Sprit Planning and Estimation

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	50 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

Velocity:
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

7. CODING & SOLUTIONING

7.1 Features

7.1.1 Python Flask

We have used python flask to develop our project. Python flask is a web framework and a python module that lets you develop web applications easily.

7.1.2 IBM Cloud DB2

When the new user registers into the application and the details of the user gets stored in IBM Cloud DB2 . We have connected the DB2 with our project using the below code

```
app.config['database'] = 'bludb'app.config['hostname'] = '3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'app.config['port'] =
'31498'app.config['protocol'] = 'tcpip'app.config['uid'] = 'sbb93800'app.config['pwd'] =
'wobsVLm6ccFxcNLe'app.config['security'] = 'SSL'try:
mysql = DB2(app)

conn_str='database=bludb;hostname=2d46b6b4-cbf6-40eb-bbce-
6251e6ba0300.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=32328;protocol=tcpip;\

uid=lsc91268;pwd=dlWyz6qJK3v27xP6;security=SSL'ibm
_db_conn = ibm_db.connect(conn_str,",")

print("Database connected without any error
!!")except:
print("IBM DB Connection error   :     " + DB2.conn_errormsg())
```

7.1.3 Send Grid

When the new user registers, the confirmation mail is sent to the user's mail using the Send Grid and when the user exceeds the limit the alert email is send to the user using this Send Grid service.

```
app =
Flask(__name__)app.config['SECRET_K
EY'] = 'top-
secret!'app.config['MAIL_SERVER'] =
'smtp.sendgrid.net'app.config['MAIL_PO
RT'] = 587app.config['MAIL_USE_TLS']
= Trueapp.config['MAIL_USERNAME']
= 'apikey'
app.config['MAIL_PASSWORD'] = 'SG.PU_eO2bJTI-
HAnjene8ngw.P8zB2XEy14FM4Efn0wTV-
5JG98963QWXKZZza_bugb8'
app.config['MAIL_DEFAULT_SENDER'] =
'tarunvinodh@gmail.com'mail = Mail(app)
```

```
total=0
forxinexpense:
total += x[4]

param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER BY id DESC LIMIT 1"
res = ibm_db.exec_immediate(ibm_db_conn, param)
dictionary = ibm_db.fetch_assoc(res)
row = []
s = 0
whiledictionary != False:
temp = []
temp.append(dictionary["LIMITSS"])
row.append(temp)
dictionary = ibm_db.fetch_assoc(res)
s = temp[0]

iftotal>int(s):
msg = "Hello " + session['username'] + " , " + "you have crossed the monthly limit of Rs. " + str(s) + "/- !!!" + "\n" + "Thank you, " + "\n" + "Team Personal Expense Tracker."
sendmail(msg,session['email'])
returnredirect("/display")
```

PNT2022TMID48497

```
importsmtplibimp
ort sendgrid
assgimportos
from sendgrid.helpers.mail import Mail, Email,
To, ContentSUBJECT = "expense tracker"s =
smtplib.SMTP('smtp.gmail.com', 587)

defsendmail(TEXT,email):
print("sorry we cant process your candidature")
s = smtplib.SMTP('smtp.gmail.com', 587)
s.starttls()

s.login("demo123demo987@gmail.com", "taryluhlooidfwvj")
message  = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)

s.sendmail("demo123demo987@gmail.com", email, message)
s.quit()
```

7.1.4 IBM Watson Assistant Chat bot

We have integrated IBM Watson Assistant Chat bot. Here the users can know about the personal expense tracker application using the information given by the bot.

7.1.5 Deploying flask app in Docker

We have deployed our flask app in Docker where they package all the code, libraries, and dependencies together to make it possible for multiple containers to run in the same host and we can run our flask app using this Docker Desktop.

```
FROM python:3.6
WORKDIR /app
ADD . /app
COPY requirements.txt /app
RUN python3 -m pip install -r requirements.txt
RUN python3 -m pip install ibm_db
EXPOSE 5000
CMD ["python","app.py"]
```

7.1.6 IBM Cloud Container Registry

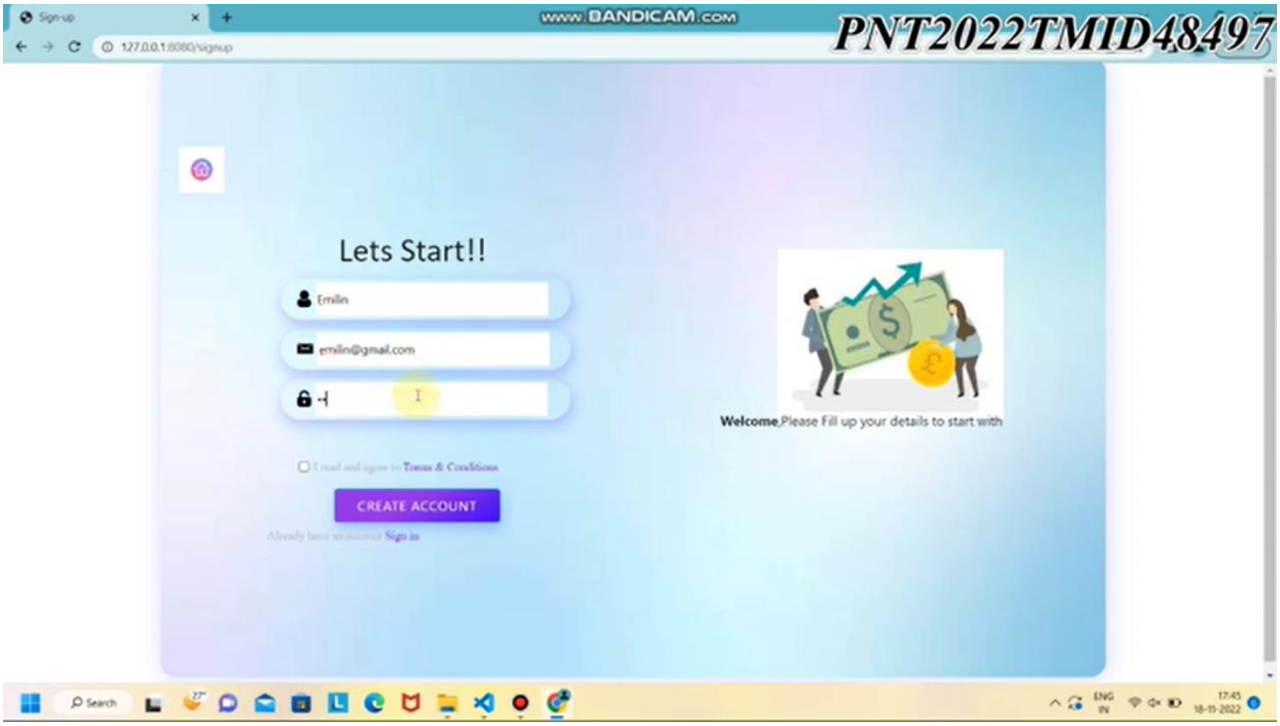
We have deployed our app as Docker image at IBM Cloud Registry.

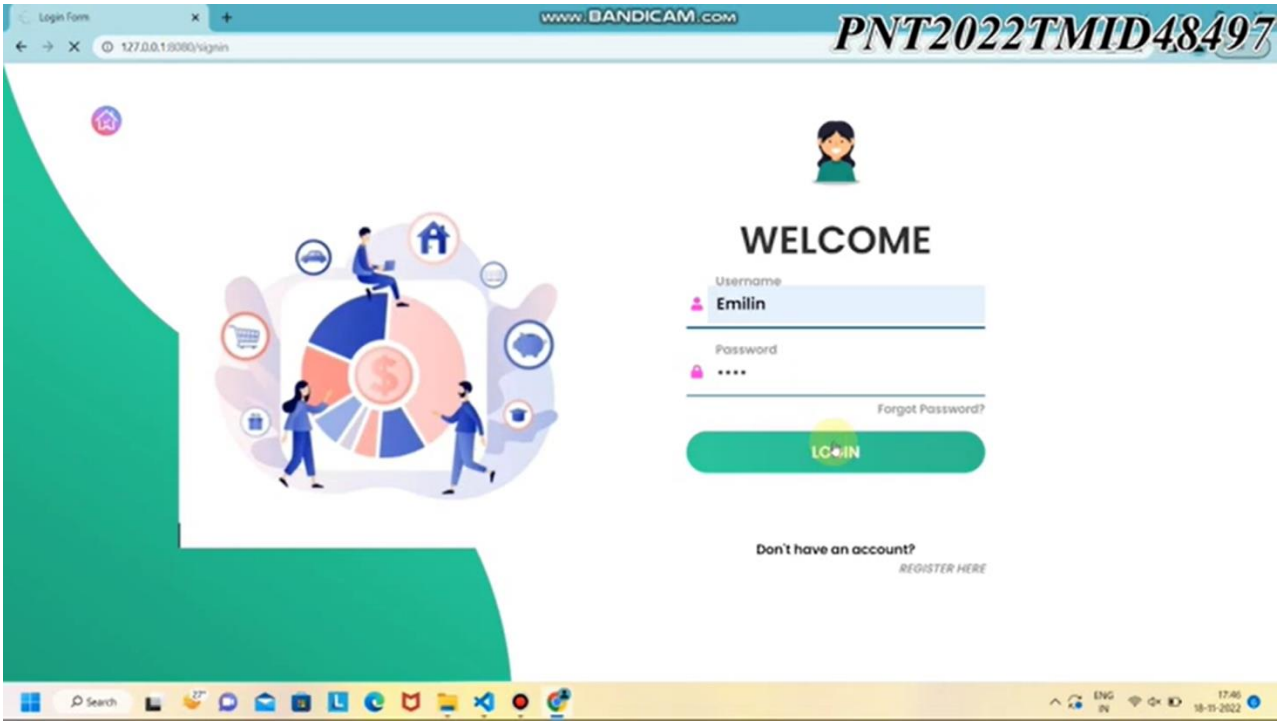
7.1.7 Kubernetes

These containers are managed by Kubernetes which automates the operational tasks of the container.

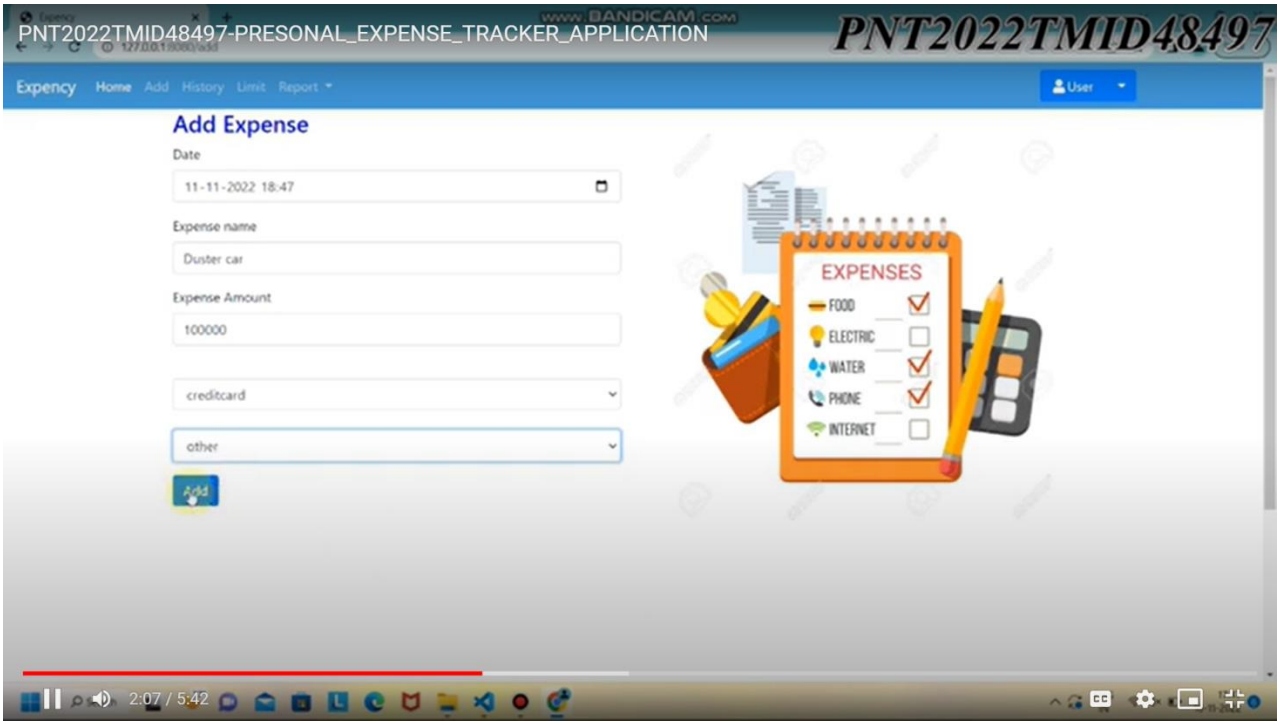
8.TESTING

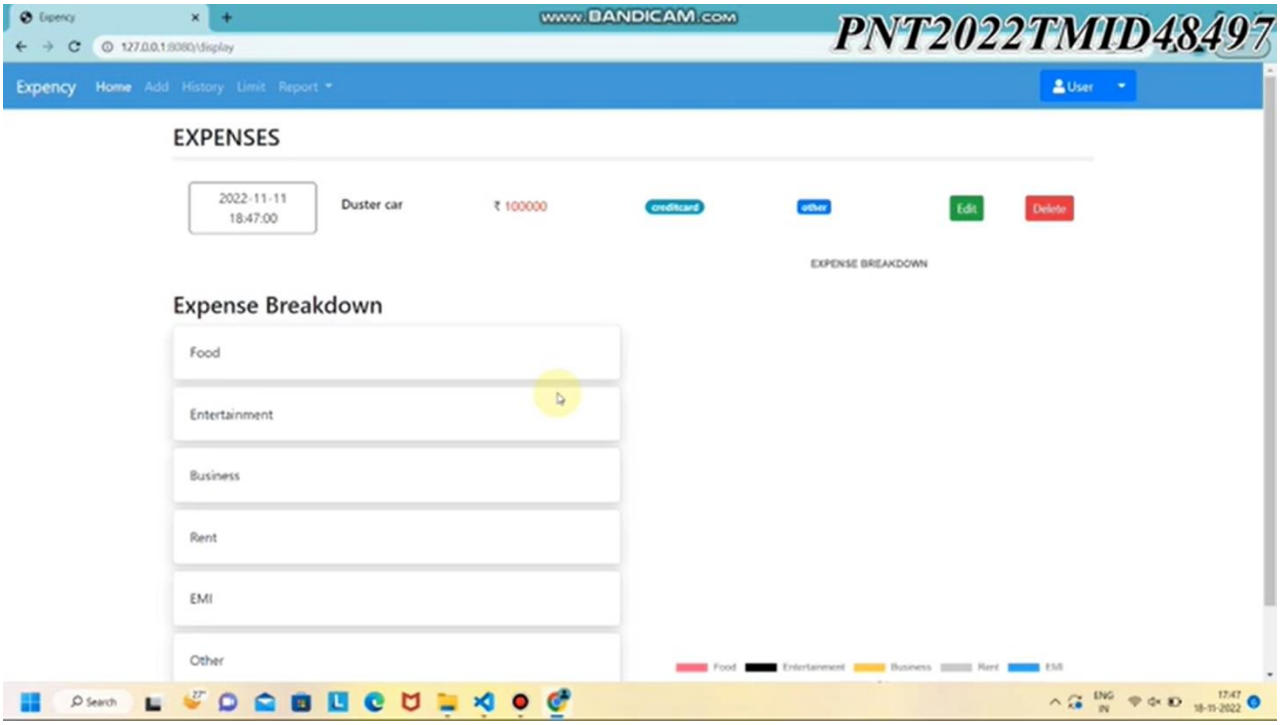
8.1.Test Cases



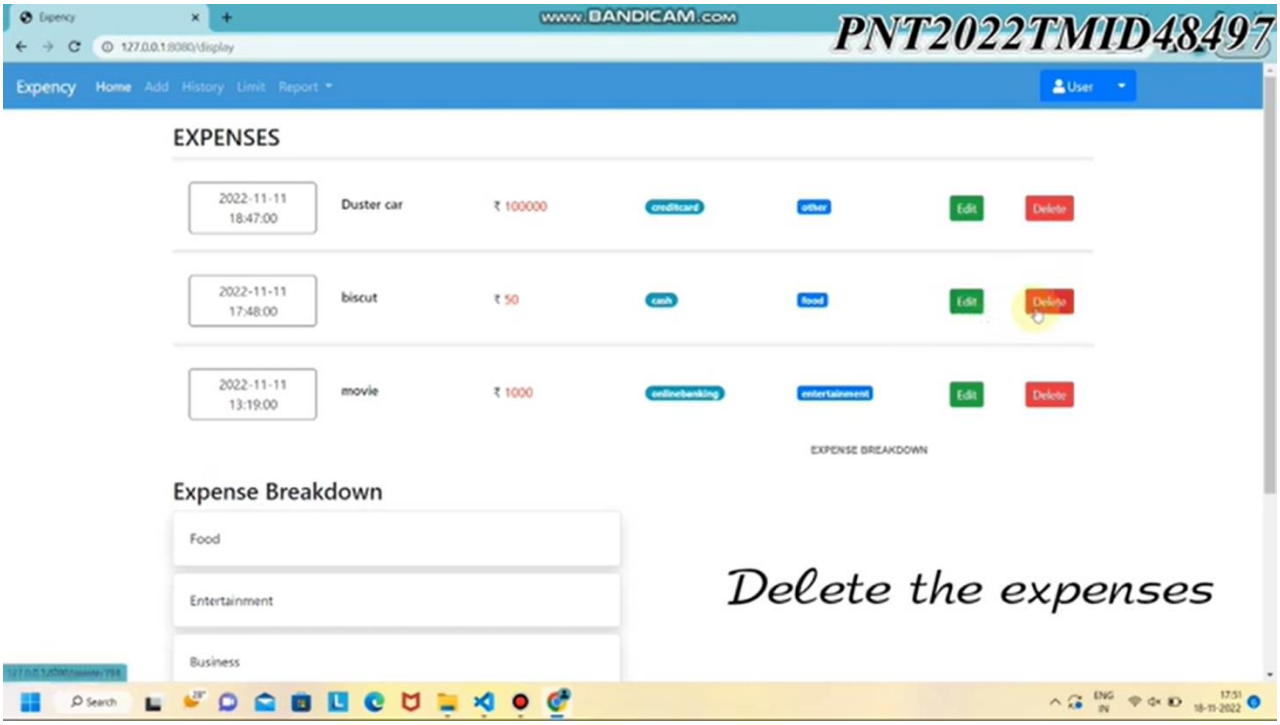


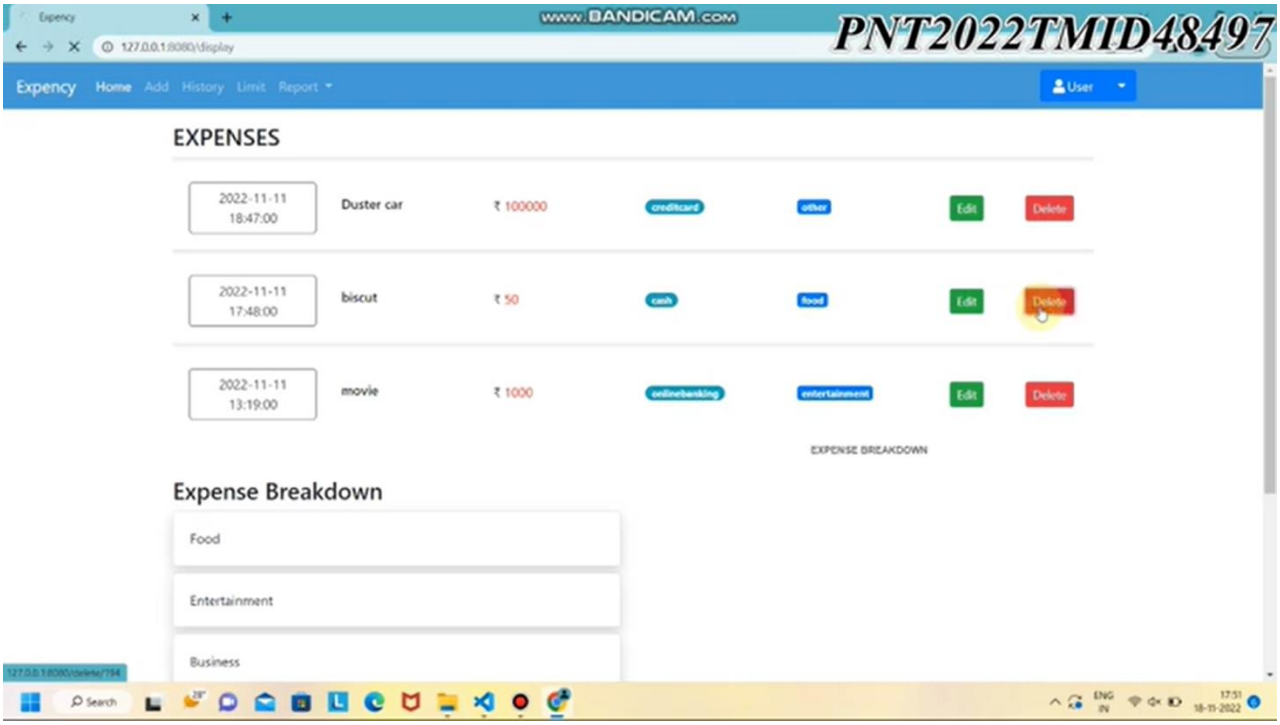
PNT2022TMID48497

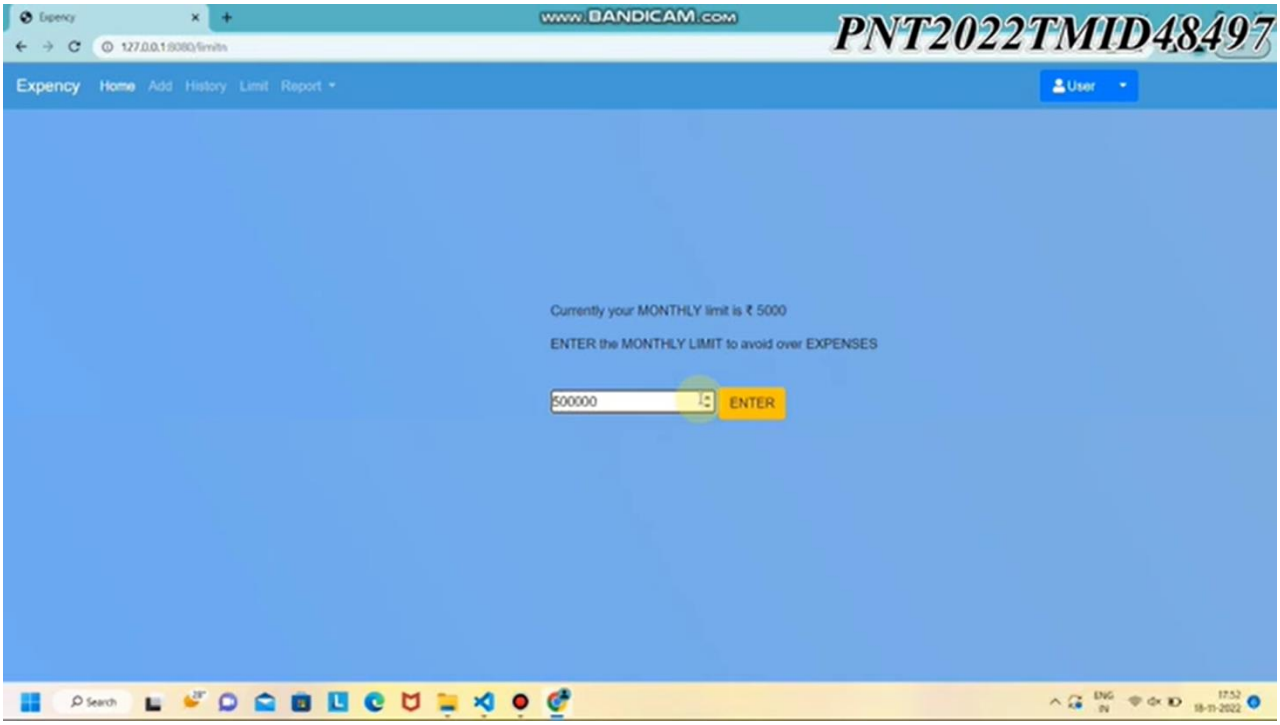


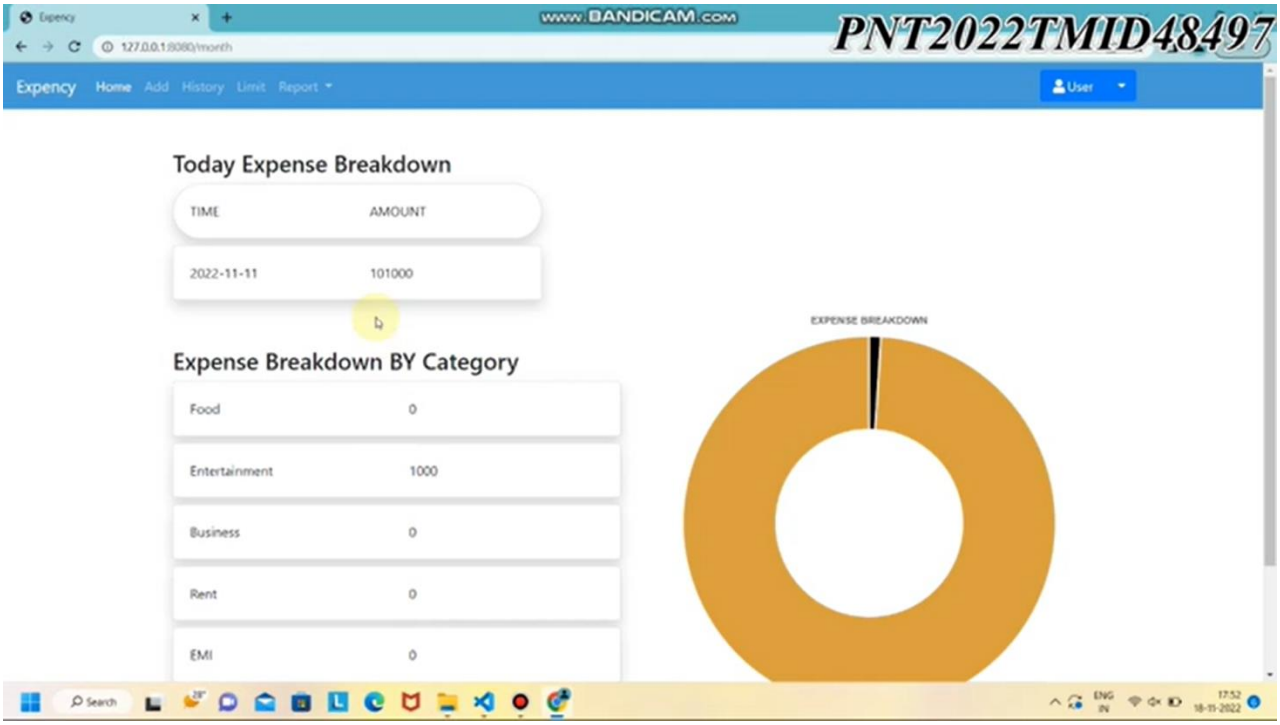


PNT2022TMID48497






www.BANDICAM.com
PNT2022TMID48497
Expency
Home
Add
History
Limit
Report
User
Currently your MONTHLY limit is ₹ 5000
ENTER the MONTHLY LIMIT to avoid over EXPENSES
500000
ENTER


www.BANDICAM.com
PNT2022TMID48497
Expency
Home
Add
History
Limit
Report
User
Today Expense Breakdown
TIME
AMOUNT
2022-11-11
101000
Expense Breakdown BY Category
Food
0
Entertainment
1000
Business
0
Rent
0
EMI
0
EXPENSE BREAKDOWN

8.1.1 Test case report

TEST CASE ID	FEATURE TYPE	COMPONENT	TEST SCENARIO	PRE-REQUISITE	STEPS TO EXECUTE	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	STATUS	COMMENTS	TC FOR AUTOMATION (Y/N)	BUG ID	EXECUTED BY
1	Functional	Login page	Verify users is able to login into the application		• Open the personal expense tracker application • Login with users Credentials • Verify logged in to user account	Username : emilin Password : 1234	Login Successful	Working as excepted	Pass		N		Emilin Pearl Sharal D
2	Functional	Signup page	Verify users is able to sign up into the application		• Open the personal expense tracker application • Enter the details and create the new user • Verify if user is created or not	Username : gayu Password : gayu	Account created successfully	Working as excepted	Pass		N		Gayathri
3	Functional	Dashboard page	Verify all the user details are stored in the Database		• Open the personal expense tracker application • Enter the details and create the new user • Verify if the user is created and stored in the database	Username : madhu Password : madhu	User should navigate to the dashboard	Working as excepted	Pass		N		Madhumitha
4	Functional	Login page	Verify the user is able to log into the application with invalid credentials		• Enter the URL and click go • Click on sign in button • Enter in valid username / email in the input • Click on login button	Username : manisha Password : manisha	Application should show the incorrect username or password	Working as excepted	Pass		N		Manisha
5	Functional	Login Page	Verify the user is able to log into the application with in valid credentials		• Enter the URL and click go • Click on sign in button • Enter in valid username / email in the input • Enter invalid password • Click on login button	Username : sharmi Password : sharmi	Application should show the incorrect username or password	Working as excepted	Pass		N		Sharmila

8.2 User Acceptance Testing

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the s oftware application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done.

1. 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).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

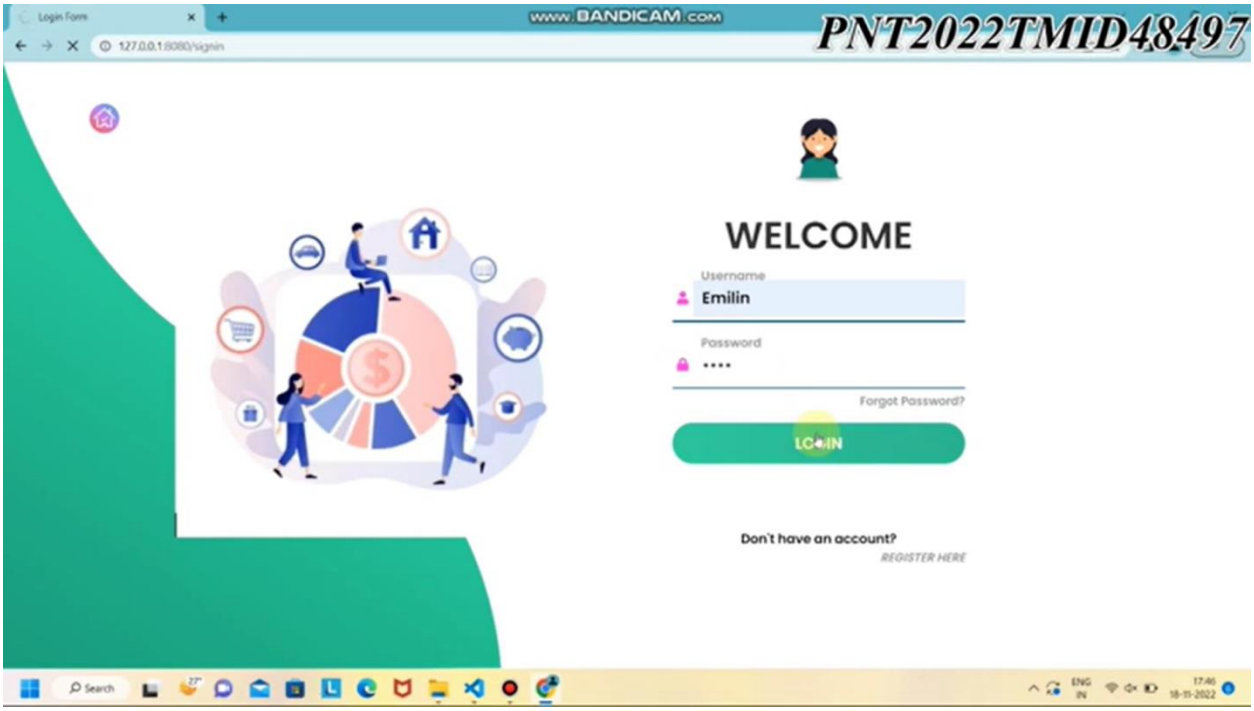
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	8	15
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	9	2	4	11	20
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	0	1	8
Totals	22	14	11	22	51

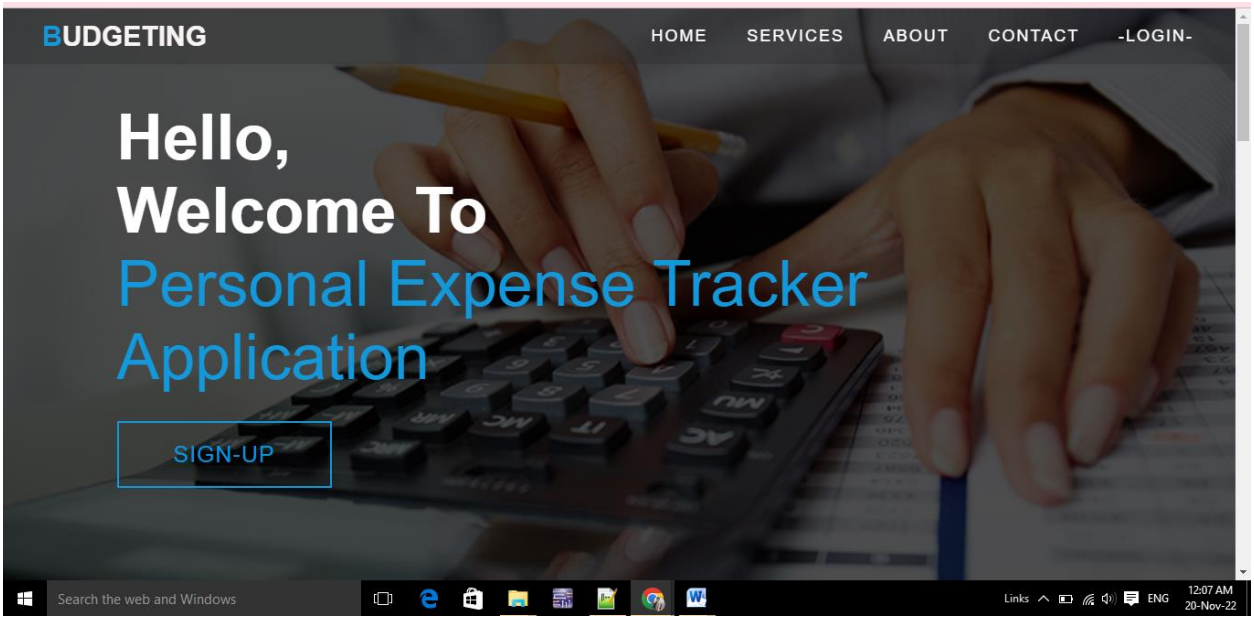
3. Test Case Analysis

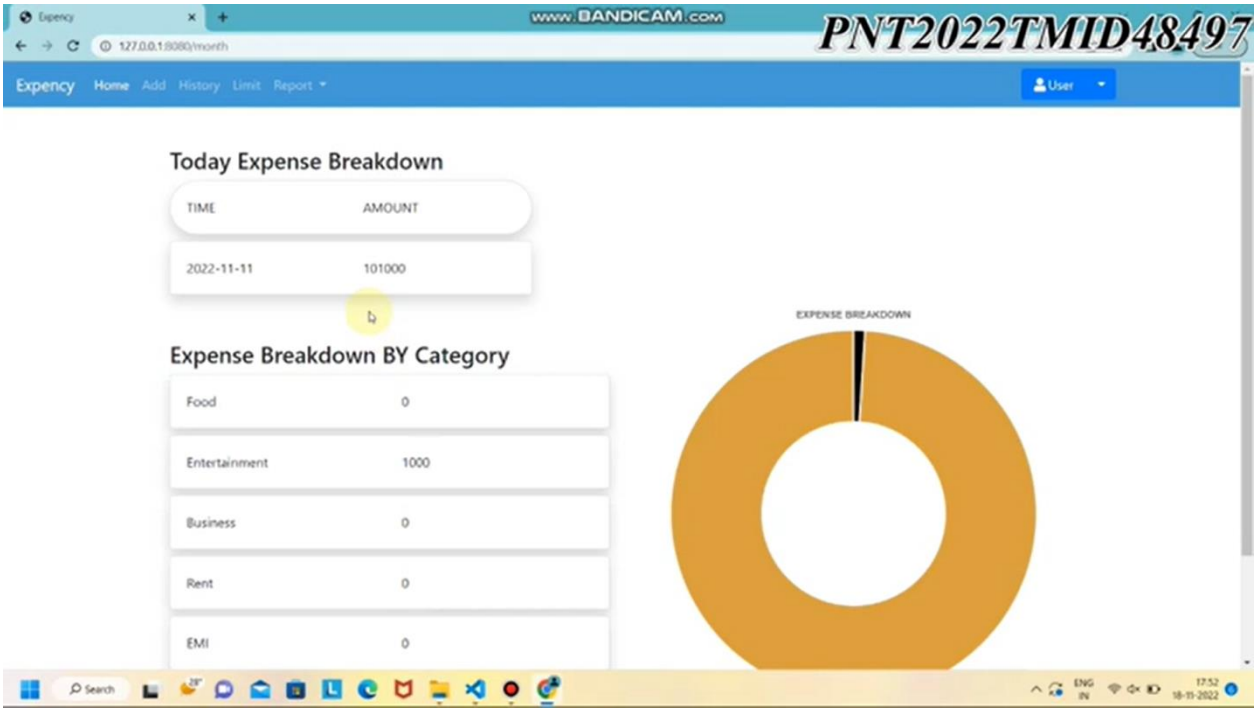
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Interface	7	0	0	7
Login	43	0	0	43
Logout	2	0	0	2
Limit	3	0	0	3
Signup	8	0	0	8
Final Report Output	4	0	0	4

OUTPUT


PNT2022TMID48497
WELCOME
Username
Emilin
Password
Forgot Password?
LOGIN
Don't have an account?
REGISTER HERE


BUDGETING
HOME
SERVICES
ABOUT
CONTACT
-LOGIN-
Hello,
Welcome To
Personal Expense Tracker
Application
SIGN-UP



9. RESULTS

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.

10. ADVANTAGES & DISADVANTAGES

10.1 Advantages

- User can have a control over their money and expenses.
- Users are alerted with an email when they exceed their limit.
- Reports are generated based on the users expenses.

10.2 Disadvantages

- Less Secured
- Limited Accessibility

11. CONCLUSION

Personal Expense Tracker Application is an web based application. We created this application so that a user can accurately calculate his/her daily cost and monthly expense. Using this application, the user will see the amount of his/her income and how much a user is spending, and also notification will be sent to the user's , if he/she exceeds the limit and report is generated.

12. FUTURE SCOPE

Now in our application we covered almost all features but in future we will add some more futures. The features are below

- Multiple account support.
- Include currency converter.
- Multilingual language

13. APPENDIX

13.1 GitHub Link

https://github.com/IBM-EPBL/IBM-Project-35618-1660286896.git

13.2 Video Demo link

https://youtu.be/JLPE40P_DpA

13.3 Source Code

app.py

```
# -*- coding: utf-8 -*-
"""
Spyder Editor
#TEAM ID:
PNT2022TMID48497
from flask import Flask,
render_template, request,
redirect, session
# from flask_mysqldb import
MySQL
# import MySQLdb.cursors
import re

from flask_db2 import DB2
import ibm_db
import ibm_db_dbi
from sendemail import
sendgridmail,sendmail
```

```
# from gevent.pywsgi import
WSGIServer
import os



app = Flask(__name__)


app.secret_key = 'a'


#
app.config['MYSQL_HOST']
= 'remotemysql.com'
#app.config['MYSQL_USER']
= 'D2DxDUPBii'
#app.config['MYSQL_PASS
WORD'] = 'r8XBO4GsMz'
# app.config['MYSQL_DB'] =
'D2DxDUPBii'
"""
dsn_hostname = "3883e7e4-
18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb
1od8lcg.databases.appdomain.
cloud"
dsn_uid = "sbb93800"
dsn_pwd =
"wobsVLm6ccFxcNLe"
dsn_driver = "{IBM DB2
ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "31498"
```

```
dsn_protocol = "tcpip"


dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
    "PORT={3};"
    "PROTOCOL={4};"
    "UID={5};"
    "PWD={6};"
).format(dsn_driver,
dsn_database, dsn_hostname,
dsn_port, dsn_protocol,
dsn_uid, dsn_pwd)
"""
# app.config['DB2_DRIVER']
= '{IBM DB2 ODBC
DRIVER}'
app.config['database'] =
'bludb'
app.config['hostname'] =
'3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb
1od8lcg.databases.appdomain.
cloud'
app.config['port'] = '31498'
app.config['protocol'] = 'tcpip'
app.config['uid'] = 'sbb93800'
app.config['pwd'] =
'wobsVLm6ccFxcNLe'
```

```
app.config['security'] = 'SSL'
try:
    mysql = DB2(app)



conn_str='database=bludb;hos
tname=3883e7e4-18f5-4afe-
be8c-
fa31c41761d2.bs2io90l08kqb
1od8lcg.databases.appdomain.
cloud;port=31498;protocol=tc
pip;\

uid=sbb93800;pwd=wobsVL
m6ccFxcNLe;security=SSL'
    ibm_db_conn =
ibm_db.connect(conn_str,",")

    print("Database connected
without any error !!")
except:
    print("IBM DB Connection
error   :     " +
DB2.conn_errormsg())

# app.config['']

# mysql = MySQL(app)
```

#HOME--PAGE

```
@app.route("/home")
def home():
    return
render_template("homepage.ht
ml")

@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 = ''
    print("Break point1")
    if request.method ==
'POST' :
        username =
request.form['username']
        email =
request.form['email']
```

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

        print("Break point2" +
"name: " + username + "------"
+ email + "------" + password)

        try:
            print("Break point3")
            connectionID =
ibm_db_dbi.connect(conn_str,
", ")
            cursor =
connectionID.cursor()
            print("Break point4")
        except:
            print("No connection
Established")

        # cursor =
mysql.connection.cursor()
        # with app.app_context():
        #     print("Break point3")
        #     cursor =
ibm_db_conn.cursor()
        #     print("Break point4")

        print("Break point5")
```

```
        sql = "SELECT * FROM
register WHERE username =
?"
        stmt =
ibm_db.prepare(ibm_db_conn,
sql)
        ibm_db.bind_param(stmt,
1, username)
        ibm_db.execute(stmt)
        result =
ibm_db.execute(stmt)
        print(result)
        account =
ibm_db.fetch_row(stmt)
        print(account)

        param = "SELECT *
FROM register WHERE
username = " + "\'" +
username + "\'"
        res =
ibm_db.exec_immediate(ibm_
db_conn, param)
        print("---- ")
        dictionary =
ibm_db.fetch_assoc(res)
        while dictionary != False:
            print("The ID is : ",
dictionary["USERNAME"])
            dictionary =
ibm_db.fetch_assoc(res)
```

```
        # dictionary =
ibm_db.fetch_assoc(result)
        # cursor.execute(stmt)

        # account =
cursor.fetchone()
        # print(account)

        # while
ibm_db.fetch_row(result) !=
False:
 #
# account =
ibm_db.result(stmt)
# print(ibm_db.result(result,
"username"))



#print(dictionary["username"]
)
        print("break point 6")
        if account:
            msg = 'Username
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:
            sql2 = "INSERT INTO
register (username,
email,password) VALUES (?,
?, ?)"
            stmt2 =
ibm_db.prepare(ibm_db_conn,
sql2)

ibm_db.bind_param(stmt2, 1,
username)

ibm_db.bind_param(stmt2, 2,
email)

ibm_db.bind_param(stmt2, 3,
password)
            ibm_db.execute(stmt2)
 # cursor.execute('INSERT
INTO register VALUES
(NULL, % s, % s, % s)',
(username,
email,password))
  # mysql.connection.commit()
            msg = 'You have
successfully registered !'
```

```
return
render_template('signup.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']
        # cursor =
mysql.connection.cursor()
# cursor.execute('SELECT *
FROM register WHERE
username = % s AND
password = % s', (username,
password ),)
```

```
        # account =
cursor.fetchone()
        # print (account)

        sql = "SELECT * FROM
register WHERE username = ?
and password = ?"
        stmt =
ibm_db.prepare(ibm_db_conn,
sql)
        ibm_db.bind_param(stmt,
1, username)
        ibm_db.bind_param(stmt,
2, password)
        result =
ibm_db.execute(stmt)
        print(result)
        account =
ibm_db.fetch_row(stmt)
        print(account)

        param = "SELECT *
FROM register WHERE
username = " + "\'" +
username + "\'" + " and
password = " + "\'" +
password + "\'"
        res =
ibm_db.exec_immediate(ibm_
db_conn, param)
```

```
        dictionary =
ibm_db.fetch_assoc(res)

        # sendmail("hello
sakthi","sivasakthisairam@gm
ail.com")

        if account:
            session['loggedin'] =
True
            session['id'] =
dictionary["ID"]
            userid =
dictionary["ID"]
            session['username'] =
dictionary["USERNAME"]
            session['email'] =
dictionary["EMAIL"]

            return redirect('/home')
        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',meth
ods=['GET', 'POST'])
def addexpense():

    date = request.form['date']
    expensename =
request.form['expensename']
    amount =
request.form['amount']
    paymode =
request.form['paymode']
    category =
request.form['category']

    print(date)
    p1 = date[0:10]
    p2 = date[11:13]
    p3 = date[14:]
    p4 = p1 + "-" + p2 + "." +
p3 + ".00"
    print(p4)
    # cursor =
mysql.connection.cursor()
    # cursor.execute('INSERT
INTO expenses VALUES
(NULL,  % s, % s, % s, % s,
% s, % s)', (session['id'] ,date,
expensename, amount,
paymode, category))
```

```
#mysql.connection.commit()
# print(date + " " +
expensename + " " + amount +
" " + paymode + " " +
category)

    sql = "INSERT INTO
expenses (userid, date,
expensename, amount,
paymode, category) VALUES
(?, ?, ?, ?, ?, ?)"
    stmt =
ibm_db.prepare(ibm_db_conn,
sql)
    ibm_db.bind_param(stmt, 1,
session['id'])
    ibm_db.bind_param(stmt, 2,
p4)
    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)

    print("Expenses added")
```

 # email part

```
    param = "SELECT *
FROM expenses WHERE
userid = " + str(session['id']) +
" AND MONTH(date) =
MONTH(current timestamp)
AND YEAR(date) =
YEAR(current timestamp)
ORDER BY date DESC"
    res =
ibm_db.exec_immediate(ibm_
db_conn, param)
    dictionary =
ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []

temp.append(dictionary["ID"])

temp.append(dictionary["USE
RID"])

temp.append(dictionary["DAT
E"])

temp.append(dictionary["EXP
ENSENAME"])
```

```
temp.append(dictionary["AM
OUNT"])

temp.append(dictionary["PAY
MODE"])

temp.append(dictionary["CAT
EGORY"])
        expense.append(temp)
        print(temp)
        dictionary =
ibm_db.fetch_assoc(res)

    total=0
    for x in expense:
          total += x[4]

    param = "SELECT id,
limitss FROM limits WHERE
userid = " + str(session['id']) +
" ORDER BY id DESC
LIMIT 1"
    res =
ibm_db.exec_immediate(ibm_
db_conn, param)
    dictionary =
ibm_db.fetch_assoc(res)
    row = []
    s = 0
    while dictionary != False:
```

```
        temp = []

temp.append(dictionary["LIM
ITSS"])
        row.append(temp)
        dictionary =
ibm_db.fetch_assoc(res)
        s = temp[0]

    if total > int(s):
        msg = "Hello " +
session['username'] + " , " +
"you have crossed the monthly
limit of Rs. " + s + "/- !!!" +
"\n" + "Thank you, " + "\n" +
"Team Personal Expense
Tracker."

sendmail(msg,session['email'])

    return redirect("/display")
#DISPLAY---graph

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

print(session["username"],sess
ion['id'])

    # cursor =
mysql.connection.cursor()
```

```
    # cursor.execute('SELECT
* FROM expenses WHERE
userid = % s AND date
ORDER BY `expenses`.`date`
DESC',(str(session['id'])))
    # expense =
cursor.fetchall()

    param = "SELECT *
FROM expenses WHERE
userid = " + str(session['id']) +
" ORDER BY date DESC"
    res =
ibm_db.exec_immediate(ibm_
db_conn, param)
    dictionary =
ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []

temp.append(dictionary["ID"])

temp.append(dictionary["USE
RID"])

temp.append(dictionary["DAT
E"])

temp.append(dictionary["EXP
ENSENAME"])
```

```
temp.append(dictionary["AM
OUNT"])

temp.append(dictionary["PAY
MODE"])

temp.append(dictionary["CAT
EGORY"])
        expense.append(temp)
        print(temp)
        dictionary =
ibm_db.fetch_assoc(res)

    return
render_template('display.html'
,expense = expense)
```

#delete---the--data

```
@app.route('/delete/<string:id
>', methods = ['POST', 'GET'
])
def delete(id):
    #  cursor =
mysql.connection.cursor()
    #  cursor.execute('DELETE
FROM expenses WHERE  id
= {0}'.format(id))
    #
mysql.connection.commit()
```

```
    param = "DELETE FROM
expenses WHERE  id = " + id
    res =
ibm_db.exec_immediate(ibm_
db_conn, param)

    print('deleted successfully')
    return redirect("/display")
```

#UPDATE---DATA

```
@app.route('/edit/<id>',
methods = ['POST', 'GET' ])
def edit(id):
    # cursor =
mysql.connection.cursor()
    # cursor.execute('SELECT
* FROM expenses WHERE
id = %s', (id,))
    # row = cursor.fetchall()

    param = "SELECT *
FROM expenses WHERE  id
= " + id
    res =
ibm_db.exec_immediate(ibm_
db_conn, param)
    dictionary =
ibm_db.fetch_assoc(res)
    row = []
```

```
    while dictionary != False:
        temp = []

temp.append(dictionary["ID"])

temp.append(dictionary["USE
RID"])

temp.append(dictionary["DAT
E"])

temp.append(dictionary["EXP
ENSENAME"])

temp.append(dictionary["AM
OUNT"])

temp.append(dictionary["PAY
MODE"])

temp.append(dictionary["CAT
EGORY"])
        row.append(temp)
        print(temp)
        dictionary =
ibm_db.fetch_assoc(res)

    print(row[0])
    return
render_template('edit.html',
expenses = row[0])
```

```
@app.route('/update/<id>',
methods = ['POST'])
def update(id):
  if request.method == 'POST'
:

      date = request.form['date']
      expensename =
request.form['expensename']
      amount =
request.form['amount']
      paymode =
request.form['paymode']
      category =
request.form['category']

    #   cursor =
mysql.connection.cursor()
    #
cursor.execute("UPDATE
`expenses` SET `date` = % s ,
`expensename` = % s ,
`amount` = % s, `paymode` =
% s, `category` = % s
WHERE `expenses`.`id` = % s
",(date, expensename, amount,
```

```
str(paymode),
str(category),id))
    #
mysql.connection.commit()

      p1 = date[0:10]
      p2 = date[11:13]
      p3 = date[14:]
      p4 = p1 + "-" + p2 + "." +
p3 + ".00"

      sql = "UPDATE expenses
SET date = ? , expensename =
? , amount = ?, paymode = ?,
category = ? WHERE id = ?"
      stmt =
ibm_db.prepare(ibm_db_conn,
sql)
      ibm_db.bind_param(stmt,
1, p4)
      ibm_db.bind_param(stmt,
2, expensename)
      ibm_db.bind_param(stmt,
3, amount)
      ibm_db.bind_param(stmt,
4, paymode)
      ibm_db.bind_param(stmt,
5, category)
      ibm_db.bind_param(stmt,
6, id)
      ibm_db.execute(stmt)
```

```
    print('successfully
updated')
    return redirect("/display")

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

@app.route("/limitnum" ,
methods = ['POST' ])
def limitnum():
    if request.method ==
"POST":
       number=
request.form['number']
      #  cursor =
mysql.connection.cursor()
      #
cursor.execute('INSERT
INTO limits VALUES
(NULL, % s, % s)
',(session['id'], number))
      #
mysql.connection.commit()

       sql = "INSERT INTO
limits (userid, limitss)
VALUES (?, ?)"
```

```
        stmt =
ibm_db.prepare(ibm_db_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():
    # cursor =
mysql.connection.cursor()
    # cursor.execute('SELECT
limitss FROM `limits`
ORDER BY `limits`.`id`
DESC LIMIT 1')
    # x= cursor.fetchone()
    # s = x[0]

    param = "SELECT id,
limitss FROM limits WHERE
userid = " + str(session['id']) +
" ORDER BY id DESC
LIMIT 1"
```

```
    res =
ibm_db.exec_immediate(ibm_
db_conn, param)
    dictionary =
ibm_db.fetch_assoc(res)
    row = []
    s = " /-"
    while dictionary != False:
        temp = []

temp.append(dictionary["LIM
ITSS"])
        row.append(temp)
        dictionary =
ibm_db.fetch_assoc(res)
        s = temp[0]
return
render_template("limit.html" ,
y= s)
```

#REPORT

```
@app.route("/today")
def today():
    #   cursor =
mysql.connection.cursor()
    #   cursor.execute('SELECT
TIME(date)   , amount FROM
expenses  WHERE userid =
%s AND DATE(date) =
```

```
DATE(NOW())
',(str(session['id'])))
    #   texpense =
cursor.fetchall()
    #   print(texpense)

      param1 = "SELECT
TIME(date) as tn, amount
FROM expenses WHERE
userid = " + str(session['id']) +
" AND DATE(date) =
DATE(current timestamp)
ORDER BY date DESC"
      res1 =
ibm_db.exec_immediate(ibm_
db_conn, param1)
      dictionary1 =
ibm_db.fetch_assoc(res1)
      texpense = []

      while dictionary1 != False:
          temp = []

temp.append(dictionary1["TN
"])

temp.append(dictionary1["AM
OUNT"])
          texpense.append(temp)
          print(temp)
```

```
        dictionary1 =
ibm_db.fetch_assoc(res1)

    #   cursor =
mysql.connection.cursor()
    #   cursor.execute('SELECT
* FROM expenses WHERE
userid = % s AND
DATE(date) =
DATE(NOW()) AND date
ORDER BY `expenses`.`date`
DESC',(str(session['id'])))
    #   expense =
cursor.fetchall()

    param = "SELECT *
FROM expenses WHERE
userid = " + str(session['id']) +
" AND DATE(date) =
DATE(current timestamp)
ORDER BY date DESC"
    res =
ibm_db.exec_immediate(ibm_
db_conn, param)
    dictionary =
ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []

temp.append(dictionary["ID"])
```

```
temp.append(dictionary["USE
RID"])

temp.append(dictionary["DAT
E"])

temp.append(dictionary["EXP
ENSENAME"])

temp.append(dictionary["AM
OUNT"])

temp.append(dictionary["PAY
MODE"])
temp.append(dictionary["CAT
EGORY"])
        expense.append(temp)
        print(temp)
        dictionary =
ibm_db.fetch_assoc(res)


    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[4]
        if x[6] == "food":
            t_food += x[4]

        elif x[6] ==
"entertainment":
            t_entertainment  +=
x[4]

        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]

        elif x[6] == "EMI":
            t_EMI  += x[4]

        elif x[6] == "other":
            t_other  += x[4]

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)
```

```
return
render_template("today.html",
texpense = texpense, 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("/month")
def month():
    #   cursor =
mysql.connection.cursor()
    #   cursor.execute('SELECT
DATE(date), SUM(amount)
FROM expenses WHERE
userid= %s AND
MONTH(DATE(date))=
MONTH(now()) GROUP BY
DATE(date) ORDER BY
DATE(date)
',(str(session['id'])))
    #   texpense =
cursor.fetchall()
    #   print(texpense)
```

```
    param1 = "SELECT
DATE(date) as dt,
SUM(amount) as tot FROM
expenses WHERE userid = " +
str(session['id']) + " AND
MONTH(date) =
MONTH(current timestamp)
AND YEAR(date) =
YEAR(current timestamp)
GROUP BY DATE(date)
ORDER BY DATE(date)"
    res1 =
ibm_db.exec_immediate(ibm_
db_conn, param1)
    dictionary1 =
ibm_db.fetch_assoc(res1)
    texpense = []

    while dictionary1 != False:
        temp = []

temp.append(dictionary1["DT
"])

temp.append(dictionary1["TO
T"])
        texpense.append(temp)
        print(temp)
        dictionary1 =
ibm_db.fetch_assoc(res1)
```

```
    #   cursor =
mysql.connection.cursor()
    #   cursor.execute('SELECT
* FROM expenses WHERE
userid = % s AND
MONTH(DATE(date))=
MONTH(now()) AND date
ORDER BY `expenses`.`date`
DESC',(str(session['id'])))
    #   expense =
cursor.fetchall()

      param = "SELECT *
FROM expenses WHERE
userid = " + str(session['id']) +
" AND MONTH(date) =
MONTH(current timestamp)
AND YEAR(date) =
YEAR(current timestamp)
ORDER BY date DESC"
      res =
ibm_db.exec_immediate(ibm_
db_conn, param)
      dictionary =
ibm_db.fetch_assoc(res)
      expense = []
      while dictionary != False:
          temp = []

temp.append(dictionary["ID"])
```

```
temp.append(dictionary["USE
RID"])

temp.append(dictionary["DAT
E"])
temp.append(dictionary["EXP
ENSENAME"])

temp.append(dictionary["AM
OUNT"])

temp.append(dictionary["PAY
MODE"])

temp.append(dictionary["CAT
EGORY"])
        expense.append(temp)
        print(temp)
        dictionary =
ibm_db.fetch_assoc(res)


    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[4]
        if x[6] == "food":
            t_food += x[4]

        elif x[6] ==
"entertainment":
            t_entertainment  +=
x[4]

        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]

        elif x[6] == "EMI":
            t_EMI  += x[4]

        elif x[6] == "other":
            t_other  += x[4]

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)
```

```
 return
render_template("today.html",
texpense = texpense, 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():
    #   cursor =
 mysql.connection.cursor()
    #   cursor.execute('SELECT
 MONTH(date), SUM(amount)
 FROM expenses WHERE
 userid= %s AND
 YEAR(DATE(date))=
 YEAR(now()) GROUP BY
 MONTH(date) ORDER BY
 MONTH(date)
 ',(str(session['id'])))
    #   texpense =
 cursor.fetchall()
    #   print(texpense)
```

```
    param1 = "SELECT
MONTH(date) as mn,
SUM(amount) as tot FROM
expenses WHERE userid = " +
str(session['id']) + " AND
YEAR(date) = YEAR(current
timestamp) GROUP BY
MONTH(date) ORDER BY
MONTH(date)"
    res1 =
ibm_db.exec_immediate(ibm_
db_conn, param1)
    dictionary1 =
ibm_db.fetch_assoc(res1)
    texpense = []

    while dictionary1 != False:
        temp = []

temp.append(dictionary1["MN
"])

temp.append(dictionary1["TO
T"])
        texpense.append(temp)
        print(temp)
        dictionary1 =
ibm_db.fetch_assoc(res1)
```

```
    #   cursor =
mysql.connection.cursor()
#   cursor.execute('SELECT *
FROM expenses WHERE
userid = % s AND
YEAR(DATE(date))=
YEAR(now()) AND date
ORDER BY `expenses`.`date`
DESC',(str(session['id'])))
    #   expense =
cursor.fetchall()

      param = "SELECT *
FROM expenses WHERE
userid = " + str(session['id']) +
" AND YEAR(date) =
YEAR(current timestamp)
ORDER BY date DESC"
      res =
ibm_db.exec_immediate(ibm_
db_conn, param)
      dictionary =
ibm_db.fetch_assoc(res)
      expense = []
      while dictionary != False:
          temp = []

temp.append(dictionary["ID"])

temp.append(dictionary["USE
RID"])
```

```
temp.append(dictionary["DAT
E"])

temp.append(dictionary["EXP
ENSENAME"])

temp.append(dictionary["AM
OUNT"])

temp.append(dictionary["PAY
MODE"])

temp.append(dictionary["CAT
EGORY"])
        expense.append(temp)
        print(temp)
        dictionary =
ibm_db.fetch_assoc(res)



    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[4]
        if x[6] == "food":
```

```
            t_food += x[4]

        elif x[6] ==
"entertainment":
            t_entertainment  +=
x[4]

        elif x[6] == "business":
            t_business  += x[4]
        elif x[6] == "rent":
            t_rent  += x[4]

        elif x[6] == "EMI":
            t_EMI  += x[4]

        elif x[6] == "other":
            t_other  += x[4]

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)
return
render_template("today.html",
texpense = texpense, 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)
   session.pop('email', None)
   return
render_template('home.html')
port =
os.getenv('VCAP_APP_PORT
', '8080')
if __name__ == "__main__":
    app.secret_key =
os.urandom(12)
    app.run(debug=True,
host='0.0.0.0', port=port)
```

Home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="..\static\css\home.css">
<title>My Website</title>
</head>
<body>
<!-- Header -->
<section id="header">
<div class="header container">
<div class="nav-bar">
<div class="brand">
<a href="#hero">
<h1><span>B</span>udgeting</h1>
</a>
</div>
<div class="nav-list">
<div class="hamburger">
<div class="bar"></div>
</div>
<ul>
<li><a href="#hero" data-after="Home">Home</a></li>
<li><a href="#services" data-after="Service">Services</a></li>
```

```
<li><a href="#about" data-after="About">About</a></li>
<li><a href="#contact" data-after="Contact">Contact</a></li>
<LI><a href="/signin" data-after="Login">-Login-</a></LI>
</ul>
</div>
</div>
</div>
</section>
<!-- End Header -->
<!-- Hero Section -->
<section id="hero">
<div class="hero container">
<div>
<h1>Hello, <span></span></h1>
<h1>Welcome To <span></span></h1>
<h1>Personal Expense Tracker Application <span></span></h1>
<a href="/signup" type="button" class="cta">Sign-up</a>
</div>
</div>
</section>
<!-- End Hero Section -->
<!-- Service Section -->
<section id="services">
<div class="services container">
<div class="service-top">
<h1 class="section-title">Serv<span>i</span>ces</h1>
```

```
<p>MyBudget provides a many services to the customer and industries. Financial solutions to meet your needs whatever your money goals, there is a My Budget solution to help you reach them </p>
</div>
<div class="service-bottom">
<div class="service-item">
<div class="icon"><imgsrc="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
<h2>Personal Expenses</h2>
<p>Budgeting is more than paying bills and setting aside savings. it's about creating a money plan for the life you want</p>
</div>
<div class="service-item">
<div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
<h2>Investments</h2>
<p>Follow your investments and bring your portfolio into focus with support for stocks,bonds,CDs,mutual funds and more</p>
</div>
<div class="service-item">
<div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/services.png" /></div>
<h2>Online Banking</h2>
<p>MyBudget application can automatically download transactions and send payments online from many financial institutions</p>
</div>
<div class="service-item">
<div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/services.png"
```

```
/></div>
<h2>Financial Life</h2>
<p>Get your Complete financial picture at a glance. With MyBudget application you can view your all the financial activities
</p>
</div>
</div>
</div>
</section>
<!-- End Service Section -->
<!-- About Section -->
<section id="about">
<div class="about container">
<div class="col-left">
<div class="about-img">
<img src="..\static\images\img-2.png" alt="img">
<div><h2>Founders, Emilin and Team </h2></div>
</div>
</div>
<div class="col-right">
<h1 class="section-title">About <span>Us</span></h1>
<h2>Financial Solution</h2>
<p>Budgeting is an expenses tracking application. Budgeting provides many services to the customers to meet their needs whatever their money goals,there is a udgeting application help to reach them.You can Contact our service center for further information and also follow our social media for update on new services </p>
<a href="#footer" class="cta">Follow Us</a>
```

```
</div>
</div>
</section>
<!-- End About Section -->
<!-- Contact Section -->
<section id="contact">
<div class="contact container">
<div>
<h1 class="section-title">Contact <span>info</span></h1>
</div>
<div class="contact-items">
<div class="contact-item">
<div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/phone.png" /></div>
<div class="contact-info">
<h1>Phone</h1>
<h2>+91 8056556341</h2>
</div>
</div>
<div class="contact-item">
<div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/new-post.png"
/></div>
<div class="contact-info">
<h1>Email</h1>
<h2>emilindaniel1219@gmail.com</h2>
<h2>emilinpearlsharalece12@gmail.com</h2>
</div>
</div>
```

```
<div class="contact-item">
<div class="icon"><img src="https://img.icons8.com/bubbles/100/000000/map-marker.png"
/></div>
<div class="contact-info">
<h1>Address</h1>
<h2>Chettinad College of Engineering and Technology</h2>
<h2>Tamilnadu</h2>
</div>
</div>
</div>
</div>
</section>
<!-- End Contact Section -->
<!-- Footer -->
<section id="footer">
<div class="footer container">
<div class="brand">
<h1><span>M</span>y <span>B</span>udget</h1>
</div>
<h2>Your Complete Financial Solution</h2>
<div class="social-icon">
<div class="social-item">
<a href="#"><img src="https://img.icons8.com/bubbles/100/000000/facebook-new.png" /></a>
</div>
<div class="social-item">
<a href="#"><img src="https://img.icons8.com/bubbles/100/000000/instagram-new.png" /></a>
</div>
```

```
<div class="social-item">
<a href="#"><img src="https://img.icons8.com/bubbles/100/000000/behance.png" /></a>
</div>
</div>
<p>Copyright © 2022 Emilin. All rights reserved</p>
</div>
</section>
<!-- End Footer -->
<script src="..\static\js\home.js"></script>
</body>
</html>
```

Login.html

```
<!DOCTYPE html>
<html>
<head>
<title>Login Form</title>
<link rel="stylesheet" type="text/css" href="..\static\css\login.css">
<link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">
<script src="https://kit.fontawesome.com/a81368914c.js"></script>
<meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body >
<img class="wave" src="..\static\images\wave.png">
<div class="container">
```

```
<div class="img">
<div id="png"><a href="/" title="HOME"><img style="width:75px; height:75px ; "
src="..\static\images\home-icon.jpg"></a></div>
<img src="..\static\images\dia.jpg">
</div>
<div class="login-content">
<form action='/login' method="POST">
<div class="msg">{{ msg }}</div>
<img src="..\static\images\avatar.jpg">
<h2 class="title">Welcome</h2>
<div class="input-div one">
<div class="i">
<i class="fas fa-user"></i>
</div>
<div class="div">
<h5>Username</h5>
<input type="text" name="username" class="input" required>
</div>
</div>
<div class="input-div pass">
<div class="i">
<i class="fas fa-lock"></i>
</div>
<div class="div">
<h5>Password</h5>
<input type="password" name="password" class="input" required>
</div>
```

```
</div>
<a href="#">Forgot Password?</a>
<input type="submit" class="btn" value="Login">
<div>
<ul>
</ul>
</div>
<div class="app" ><b>Don't have an account?</b><a id="app1" href="\signup">REGISTER
HERE</a></div>
</form>
</div>
</div>
<script type="text/javascript" src="..\static\js\login.js"></script>
</body>
</html>
```

Docker file

```
FROM python:3.6

WORKDIR /app
ADD . /app
COPY requirements.txt /app
RUN python3 -m pip install -r requirements.txt
RUN python3 -m pip install ibm_db
EXPOSE 5000
CMD ["python","app.py"]

# FROM python:3.10-alpine
# WORKDIR /app
# ADD . /app
```

```
# RUN set -e; \
#         apk add --no-cache --virtual .build-deps \
#                 gcc \
#                 libc-dev \
#                 linux-headers \
#                 mariadb-dev \
#                 python3-dev \
#         ;
# COPY requirements.txt /app
# RUN pip3 install -r requirements.txt
# CMD ["python3","app.py"]
```

Deployment.yaml

```
apiVersion: apps/v1
 kind: Deployment
 metadata:
   name: sakthi-flask-node-deployment
 spec:
   replicas: 1
   selector:
     matchLabels:
       app: flasknode
   template:
     metadata:
       labels:
         app: flasknode
     spec:
       containers:
       - name: flasknode
         image: icr.io/sakthi_expense_tracker2/flask-template2
         imagePullPolicy: Always
         ports:
         - containerPort: 5000
```

Sendmail.py

```
import smtplib

import sendgrid as sg

import os
```

```
from sendgrid.helpers.mail import Mail, Email, To, Content

SUBJECT = "expense tracker"

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



def sendmail(TEXT,email):

    print("sorry we cant process your candidature")

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

    s.starttls()

    # s.login("fawwashkhan@gmail.com", "fawwashkhan")

    s.login("tproduct8080@gmail.com", "lxixbmpnexbkiemh")

    message  = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)

    # s.sendmail("fawwashkhan@gmail.com", email, message)

    s.sendmail("il.tproduct8080@gmail.com", email, message)

    s.quit()

def sendgridmail(user,TEXT):



    # from_email = Email("fawwashkhan@gmail.com")

    from_email = Email("tproduct8080@gmail.com")

    to_email = To(user)
```

```
subject = "Sending with SendGrid is Fun"

content = Content("text/plain",TEXT)

mail = Mail(from_email, to_email, subject, content)



# Get a JSON-ready representation of the Mail object

mail_json = mail.get()

# Send an HTTP POST request to /mail/send

response = sg.client.mail.send.post(request_body=mail_json)

print(response.status_code)

print(response.headers)
```

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-35618-1660286896.git

Video Demo Link:

https://youtu.be/JLPE40P_DpA

PNT2022TMID48497

TEAM MEMBERS :

EMILIN PEARL SHARAL D (TEAM LEADER)

GAYATHRI S (TEAM MEMBER – 1)

MATHUMITHA T (TEAM MEMBER – 2)

SHARMILA N (TEAM MEMBER – 3)

MANISHA G (TEAM MEMBER – 4)