V.S.B. ENGINEERING COLLEGE
(Approved by AICTE, New Delhi, Affiliated to Anna University)
An ISO 9001:2015 Certified Institution
Accredited by NAAC, NBA Accredited Courses

PROJECT

PERSONAL EXPENSE TRACKER APPLICATION

DONE BY

TEAM ID: PNT2022TMID33480

TEAM LEADER : CHANDRU N (922519106017)

TEAM MEMBER 1 : DEVAPRASATH S (922519106023)

TEAM MEMBER 2 : GOKULPRASANTH J (922519106036)

TEAM MEMBER 3 : DEEPAN J (922519106020)

TABLE OF CONTENTS

1. INTRODUCTON

1.1 Project Overview:

This project is based on an expense and income tracking system. This project aims to create an easy, faster and smooth tracking system between the expense and the income .This project also offers some opportunities that will help the user to how to manage the expenses in efficient way and also set have an option to set a limit for the amount to be used for that particular month. So, for the better expense tracking system, we developed our project that will help the users a lot.

1.2 Purpose :

An expense tracking app is an exclusive suite of services for people who seek to handle their earnings and plan their expenses and savings. When you track your spending, you know where your money goes and you can ensure that your money is used wisely. Tracking your expenditures also allows you to understand why you're in debt and how you got there. This will then help you design a be fitting strategy of getting out of debt. Many people in India live on a fixed income, and they find that towards the end of the month they don't have sufficient money to meet their needs

2. Literature Survey

2.1 Existing Problem :

In existing, we need to maintain the Excel sheets, CSV files for the user daily, weekly and monthly expenses and there is no as such complete solution to keep a track of its daily expenses easily. To do so a person as to keep a log in a diary or in a computer system, also all the calculations need to be done by the user which may sometimes results in mistakes leading to losses. The existing system is not user friendly because data is not maintained perfectly. A writing audit is a study of insightful sources on a particular research. We found various similar products that have already been developed in the market. Unlike all those products, Personal Expense tracker (PET) provides security and graphical results. We provide the user to enter their wish-list before any purchase. It generates notifications to notify user about their money management and put an limit to weekly, monthly, expenses.

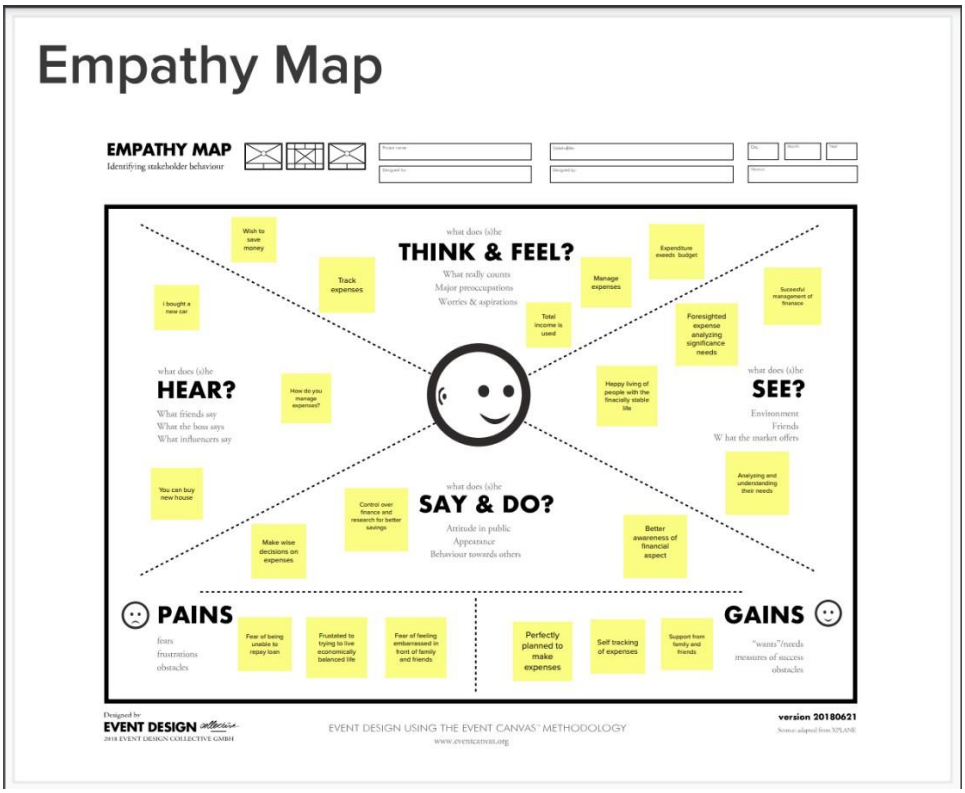
2.2 Problem Statement Definition :

Every earning people are mostly obsessed at the end of the month as they cannot remember where all of their money have gone when they have spent and ultimately have to sustain in little money minimizing their essential needs. There is no as such complete solution present easily to keep track of its daily expenditure easily and notify them if they are going to have money shortage. 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 alert. the main purpose of our application is to track the user's expenses.

3 .Ideation and Proposed Solution

3.1 Empathy Map Canvas :

An empathy map canvas helps brands provide a better experience for users by helping teams understand the perspectives and mindset of their customers. Using a template to create an empathy map canvas reduces the preparation time and standardizes the process so you create empathy map canvases of similar quality



Ideation and Proposed Solution

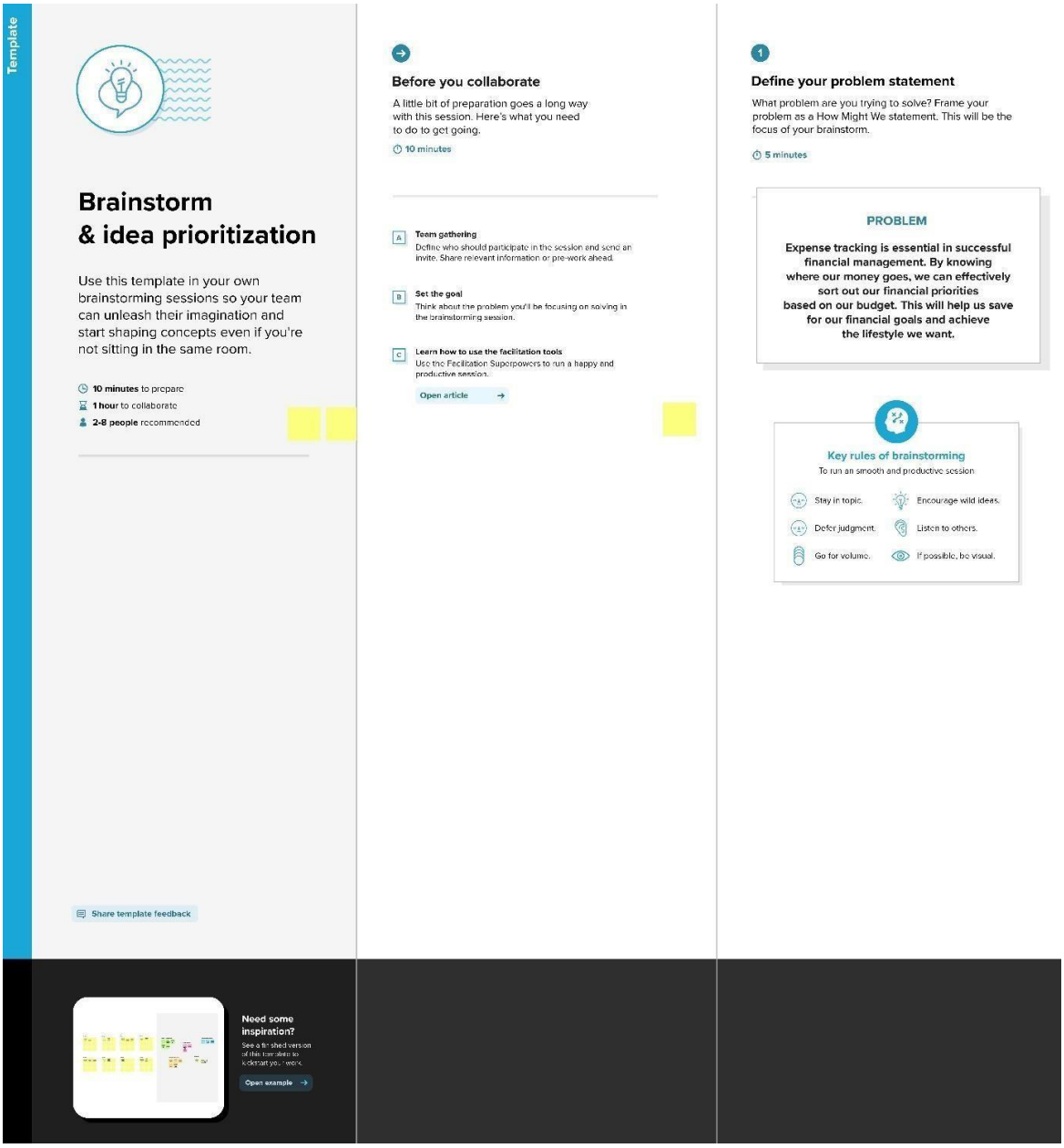
3.2 Ideation and Brainstorming :

Brainstorming is a group activity where everyone comes together to discuss strategies for growth and improvement. You can exchange ideas, share important information and use these meetings as informal catch-up sessions with your co-workers. Brainstorming combines a relaxed, informal approach to problem solving with lateral thinking. It encourages people to come up with thoughts and ideas that can, at first, seem a bit crazy. Some of these ideas can be crafted into original, creative solutions to a problem, while others can spark even more ideas.

Ideation is the process where you generate ideas and solutions through sessions such as Sketching, Prototyping, Brainstorming, Brainwriting, Worst Possible Idea, and a wealth of other ideation techniques. Ideation is also the third stage in the Design Thinking process.

As you can see, ideation is not just a one-time idea generation or a brainstorming session. In fact, we can divide ideation in these three stages: generation, selection, and development.

i.

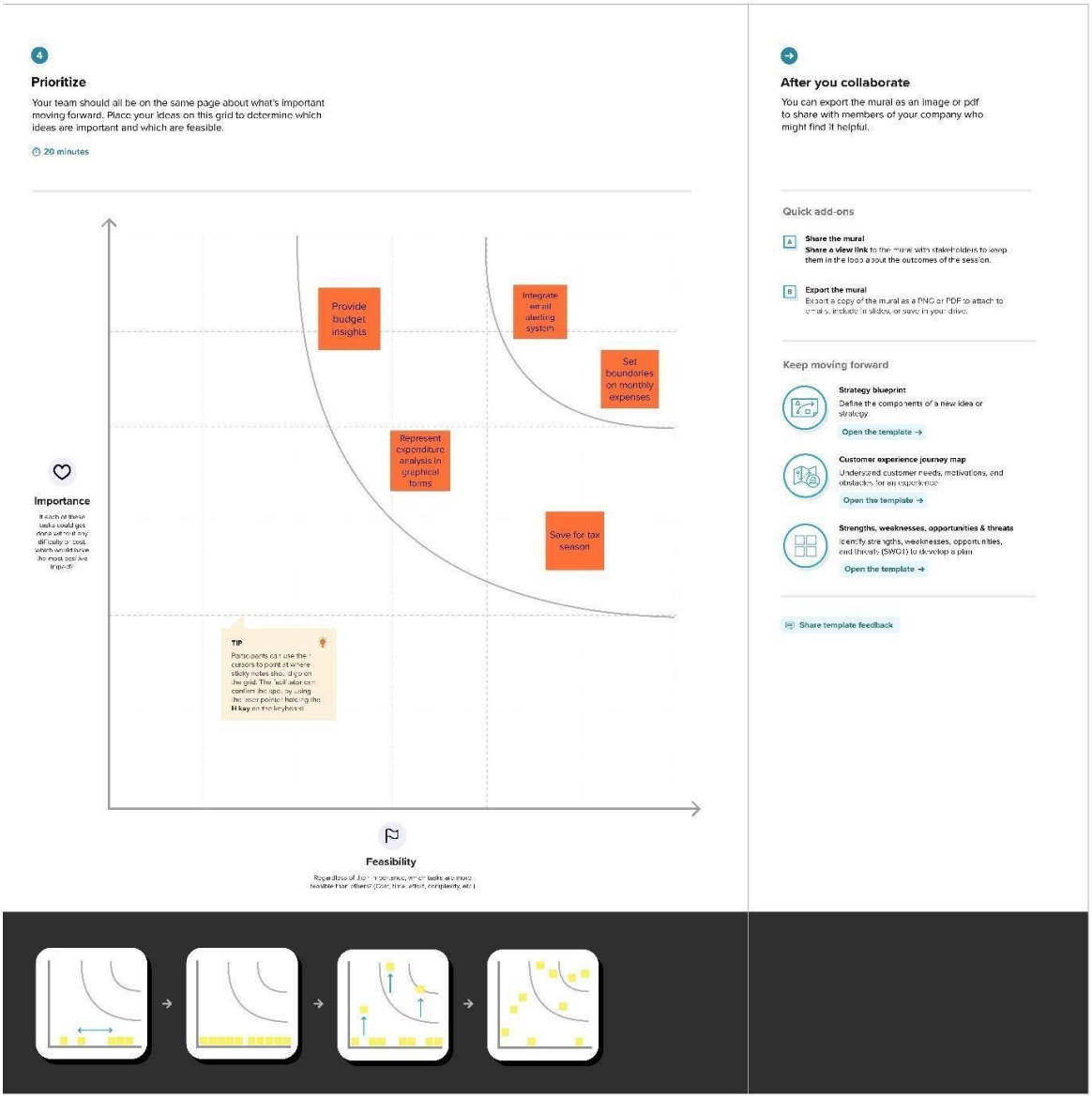

Template
Brainstorm
& idea prioritization
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.
10 minutes to prepare
1 hour to collaborate
2-8 people recommended
Share template feedback
Before you collaborate
A little bit of preparation goes a long way with this session. Here's what you need to do to get going.
10 minutes
A Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.
B Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.
C Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.
Open article
1
Define your problem statement
What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.
5 minutes
PROBLEM
Expense tracking is essential in successful financial management. By knowing where our money goes, we can effectively sort out our financial priorities based on our budget. This will help us save for our financial goals and achieve the lifestyle we want.
Key rules of brainstorming
To run an smooth and productive session
Stay in topic.
Encourage wild ideas.
Defer judgment.
Listen to others.
Go for volume.
If possible, be visual.
Need some inspiration?
See a finished version of this template to kickstart your work.
Open example

Ideation and Proposed Solution

ii.

Ideation and Proposed Solution

iii.



Ideation and Proposed Solution

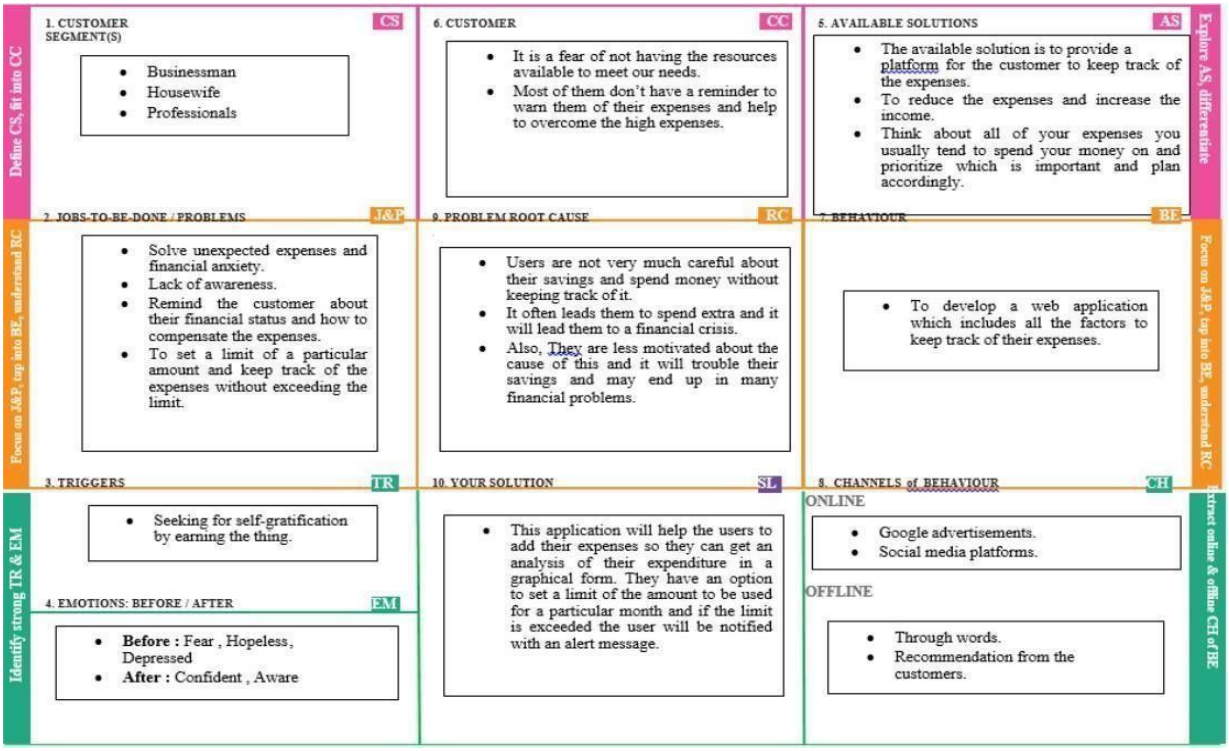
3.3 Proposed Solution :

Expense Tracker is going to be a mobile application so that It can be accessed any time required. This application will have a two-tier architecture: first one is the database tier, where all the data and financial data will be stored. Second it will be the user interface which will support the application user communicate with the system and also store Information in the database. The proposed system should operate offline so it can be accessed at any time without internet availability. The proposed system should provide different categories for the user to select from and they can enter the amount and mode of payment. This system should be able to analyze the information, provide analytics on which category did the user spent most of their money. The proposed system should provide a user interface where the user could store and observe their past expenses. To develop a personal finance application which allows users to add their expenses and based on their expense wallet balance will be updated and displayed. 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 and if the limit is exceeded the users will be notified with an email alert

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Keeping Proper track of our daily expenses is becoming challenging in today's world. Without the proper money management knowledge people overspend on their wants instead of focusing on their needs. Especially when using online applications for purchasing their requirements consumers tend to over spend. This problem leads to improper distribution of their daily expenses. Without proper knowledge on managing money poor are becoming poorer and rich are becoming richer.

2.	Idea / Solution description	An attempt to develop an app to manage our daily expenses and give us insights on managing our money would be a good idea. This app will be able to track expenses on various online platforms and apps. The app can help with proper budgeting and give alerts when the user over spends or crosses the limit previously set by them. This will lead to proper spending habits and make them knowledgeable about money management. IBM cloud can be used to handle the data safely.
3.	Novelty / Uniqueness	The speciality for the app will be the data security with IBM cloud being used for data storage and this app genuinely helps with the money management. The proper and personalized budgeting of the user's money leads them to trust the app and they wouldn't have to worry about their expenditure on unnecessary things.
4.	Social Impact / Customer Satisfaction	People using the app will be becoming better at their spending habits and will be able to save more than their peers who are not using the app. This application aims to improve the users' savings sustainably and steadily which leads them to trust the app without worrying about their money
5.	Business Model (Revenue Model)	This application leads to a business model, theuser can be suggested the right products to buybased on their budget and this can lead to targeted business approaching the right consumers. The model leads to systematic andstructured expenses of the user and also leads to predictive analysis of the future expenses ofthe consumer. This model makes the user more careful with expenses as they are provided with the money management insights
6.	Scalability of the Solution	This application can be created as a multi user model nationwide. The model can also be modified based on the country's law on applications and data security which leads to international implementation of this application by maintaining proper gateway rules. This app when developed for multiple nations can be modified to their requirements. The app can also be modified for a particular group of people or organization

3.4 Problem Fit:



4. Requirement Analysis

4.1 Functional Requirments :

1. Dashboard panel

The system shall authenticate the user and then display panel based on the particular identified user.

2. Add bill

The system shall allow the user to add bill details based on the user's need to track the type of expenses.

3.Expense planner

The system should graphically represent the current month figure based current month expenses and user's own budget share.

4. Expense tracker

The system should graphically represent the yearly expense numbers in form of report

5.Add notes

The system shall allow users to add notes to their expenses.

4.2 Non-Functional Requirments:

1. Usability

There is a consistency in all the modules and webpages. To ease the navigation there is a back tab to provide access to previous page. There is proper instruction on each page.

2. Reliability

Each data record is stored on a well-built efficient database schema. There is no risk ofdata loss. The internal evaluation of data is well coded.

3. Supportability

The system is well built to support any machine. Maintainability of the system is easy.

4. Performance

In order to ease the accessibility, the types of expenses are categorized along with an option to name on the own. Throughput of the system is increased due to light weight database support.

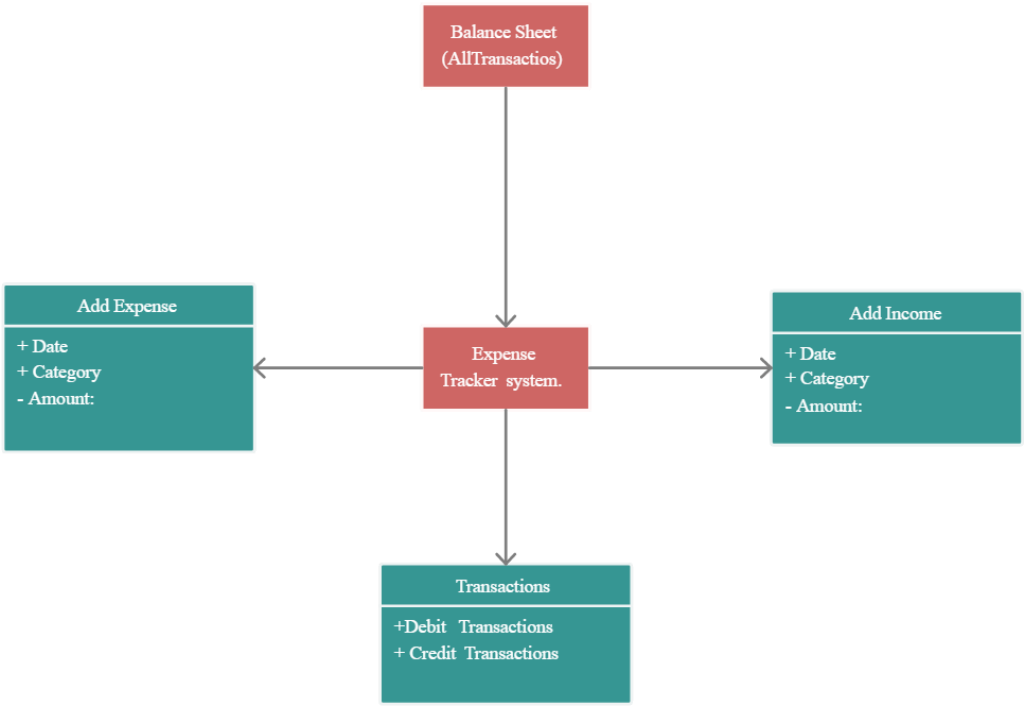
5. Availability

The system is available all the time, no time constraint.

5. Project Design

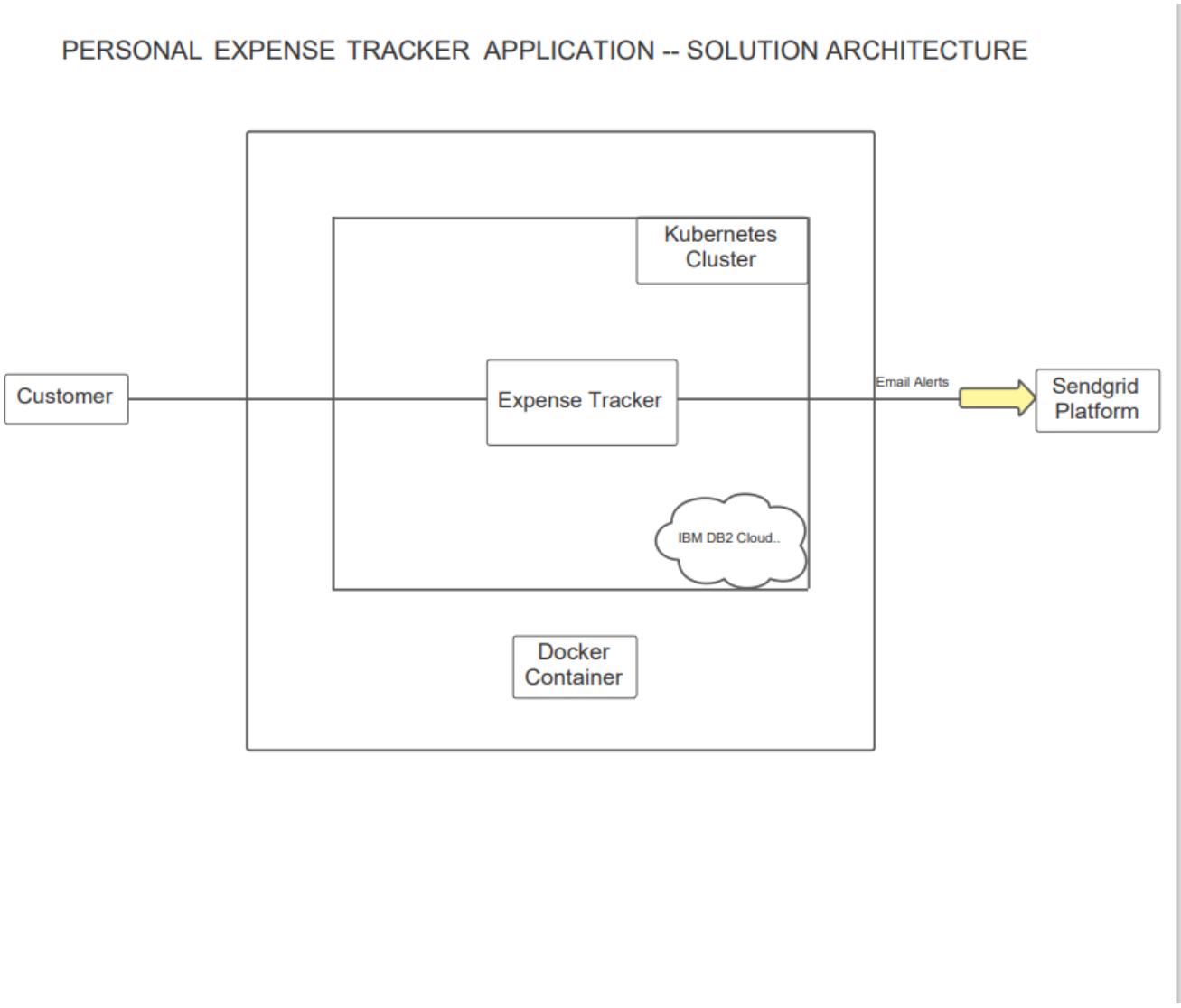
5.1 Data flow Diagrams :

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. Data flow diagrams provide a straightforward, efficient way for organizations to understand, perfect, and implement new processes or systems. They're visual representations of your process or system, so they make it easy to understand and prune.



5.2 Solution & Technical Architecture:

Technical Architecture (TA) is a form of IT architecture that is used to design computer systems. It involves the development of a technical blueprint with regard to the arrangement, interaction, and interdependence of all elements so that system-relevant requirements are met.



6. Project Planning & Scheduling

6.1 Sprint Plainning & Estimation:

Sprint	Functional Requirement (Epic)	User Story / Task
Sprint 1	Registration	As a user, I can register for the application by entering my email, password, and confirming my password.
		As a user, I will receive confirmation email once I have registered for the application
	Login	As a user, I can log into the application by entering email &password
	Dashboard	Logging in takes to the dashboard for the logged user.
Sprint 2	Workspace	Workspace for personal expense tracking
	Charts	Creating various graphs and statistics of customer's data
	Connecting to IBM DB2	Linking database with dashboard
		Making dashboard interactive with JS

Sprint-3		Wrapping up the server side works of frontend
	Watson Assistant	Creating Chat bot for expense tracking and for clarifying user's query
	Send Grid	Using Send Grid to send mail to the user about their expenses
		Integrating both frontend and backend
Sprint-4	Docker	Creating image of website using docker /
	Cloud Registry	Uploading docker image to IBM Cloud registry
	Kubernetes	Create container using the docker image and hosting the site
	Exposing	Exposing IP/Ports for the site

6.2 Sprint Delivery Schedule:

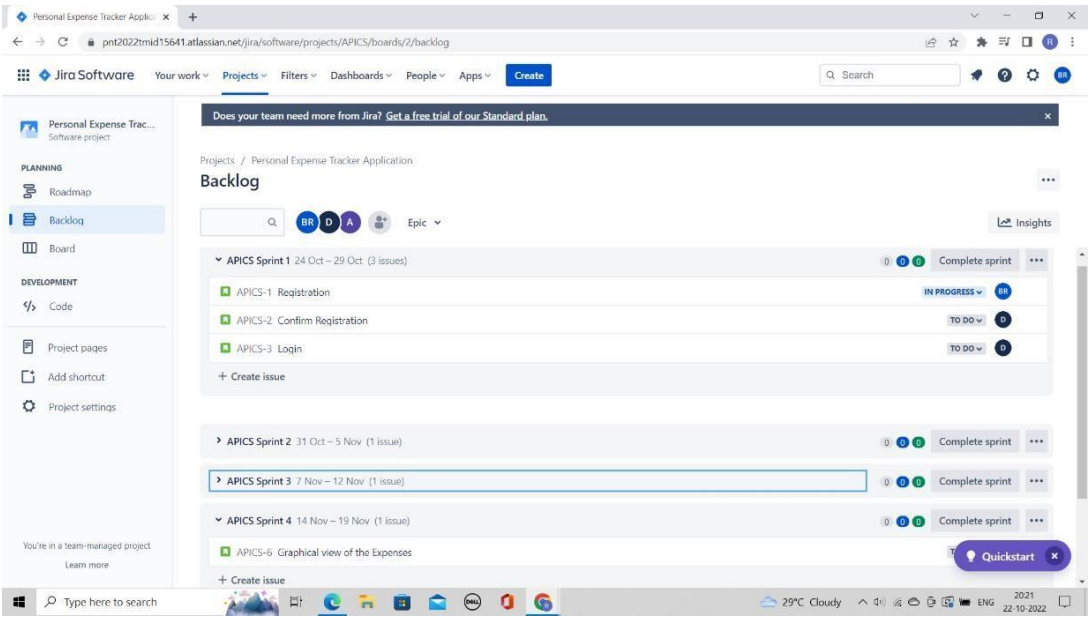
A sprint schedule is a document that outlines sprint planning from end to end. It's one of the first steps in the agile sprint planning process—and something that requires adequate research, planning, and communication.

Sprint	Total Story Points	Sprint Start Date	Sprint End Date	Story Points Collected	Sprint Release Date
Sprint 1	20	23 Oct	27 Oct	20	28 Oct
Sprint 2	20	30 Oct	02 Nov	20	03 Nov
Sprint 3	20	06 Nov	10 Nov	20	11 Nov
Sprint 4	20	13 Nov	18 Nov	20	19 Nov

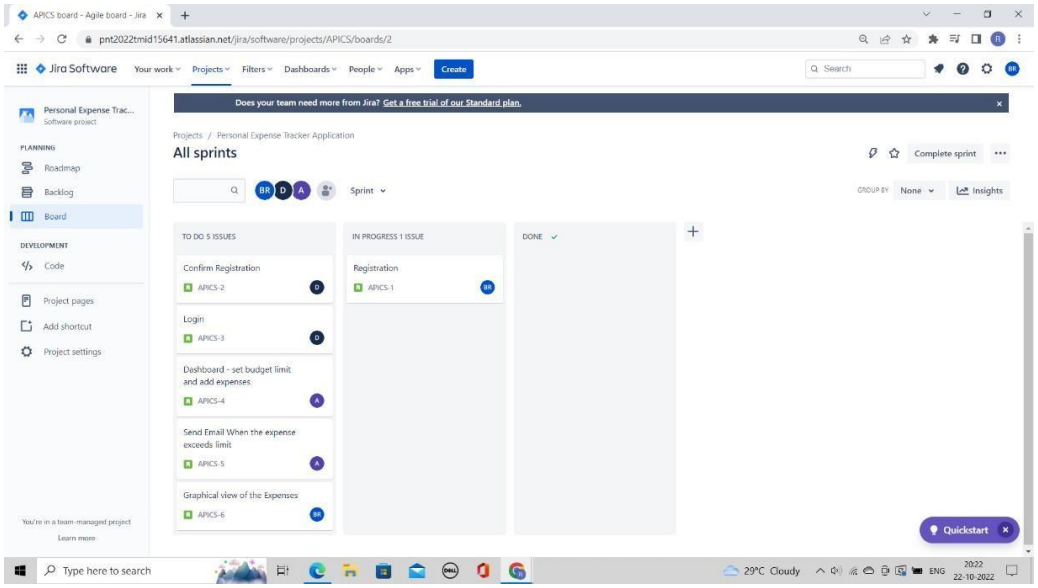
6.3 Reports from JIIRA:

JIIRA is a software application used for issue tracking and project management. The tool, developed by the Australian software company Atlassian , has become widely used by agile development teams to track bugs, stories, epics, and other tasks

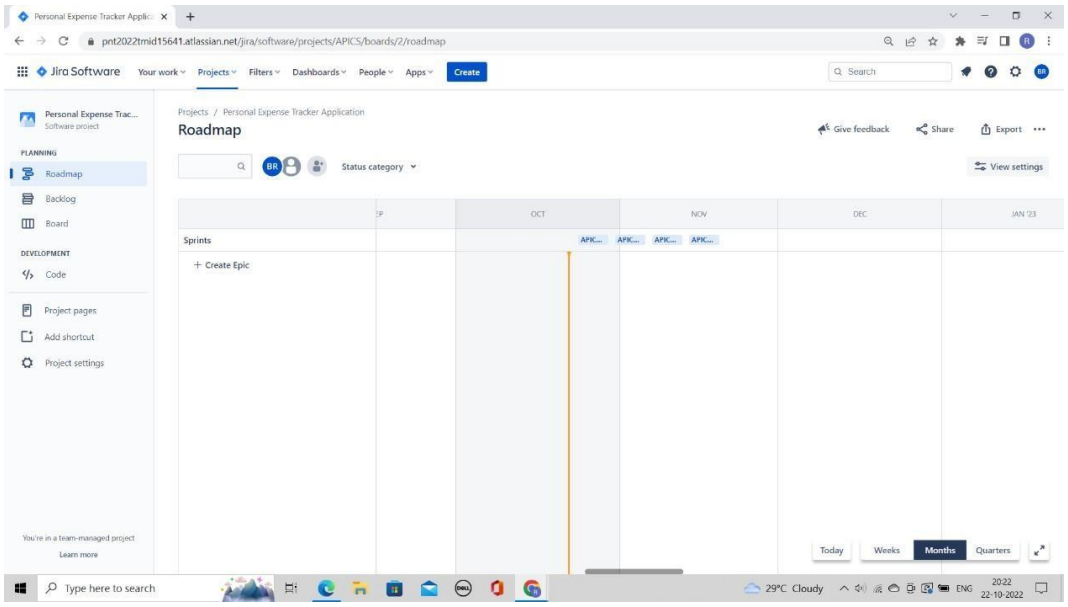
i.


Personal Expense Tracker Applic
pnt2022tmid15641.atlassian.net/jira/software/projects/APICS/boards/2/backlog
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Personal Expense Trac...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Does your team need more from Jira? Get a free trial of our Standard plan.
Projects / Personal Expense Tracker Application
Backlog
Epic
Insights
APICS Sprint 1 24 Oct – 29 Oct (3 issues)
Complete sprint
APICS-1 Registration
IN PROGRESS
APICS-2 Confirm Registration
TO DO
APICS-3 Login
TO DO
Create issue
APICS Sprint 2 31 Oct – 5 Nov (1 issue)
Complete sprint
APICS Sprint 3 7 Nov – 12 Nov (1 issue)
Complete sprint
APICS Sprint 4 14 Nov – 19 Nov (1 issue)
Complete sprint
APICS-6 Graphical view of the Expenses
Create issue
Quickstart
Type here to search
29°C Cloudy
ENG
2021
22-10-2022

ii.

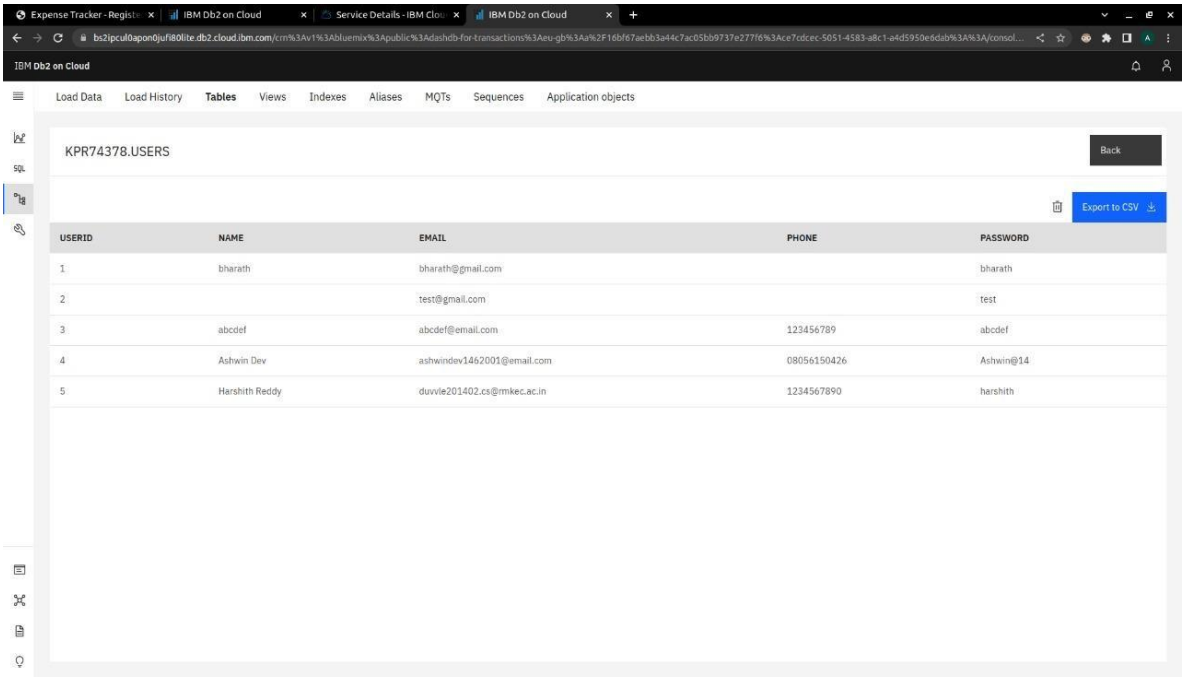

APICS board - Agile board - Jira
pnt2022tmid15641.atlassian.net/jira/software/projects/APICS/boards/2
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Personal Expense Trac...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Does your team need more from Jira? Get a free trial of our Standard plan.
Projects / Personal Expense Tracker Application
All sprints
Complete sprint
Sprint
GROUP BY None
Insights
TO DO 5 ISSUES
IN PROGRESS 1 ISSUE
DONE
Confirm Registration
APICS-2
Login
APICS-3
Dashboard - set budget limit and add expenses
APICS-4
Send Email When the expense exceeds limit
APICS-5
Graphical view of the Expenses
APICS-6
Registration
APICS-1
Quickstart
Type here to search
29°C Cloudy
ENG
2022
22-10-2022

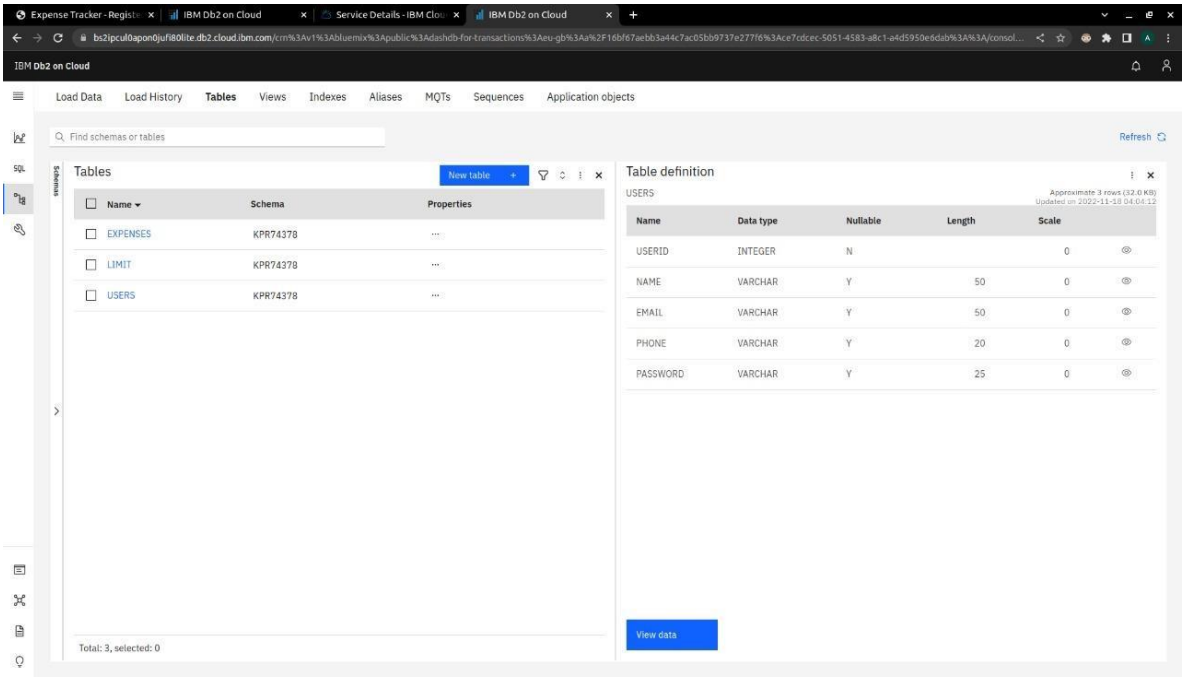
iii.

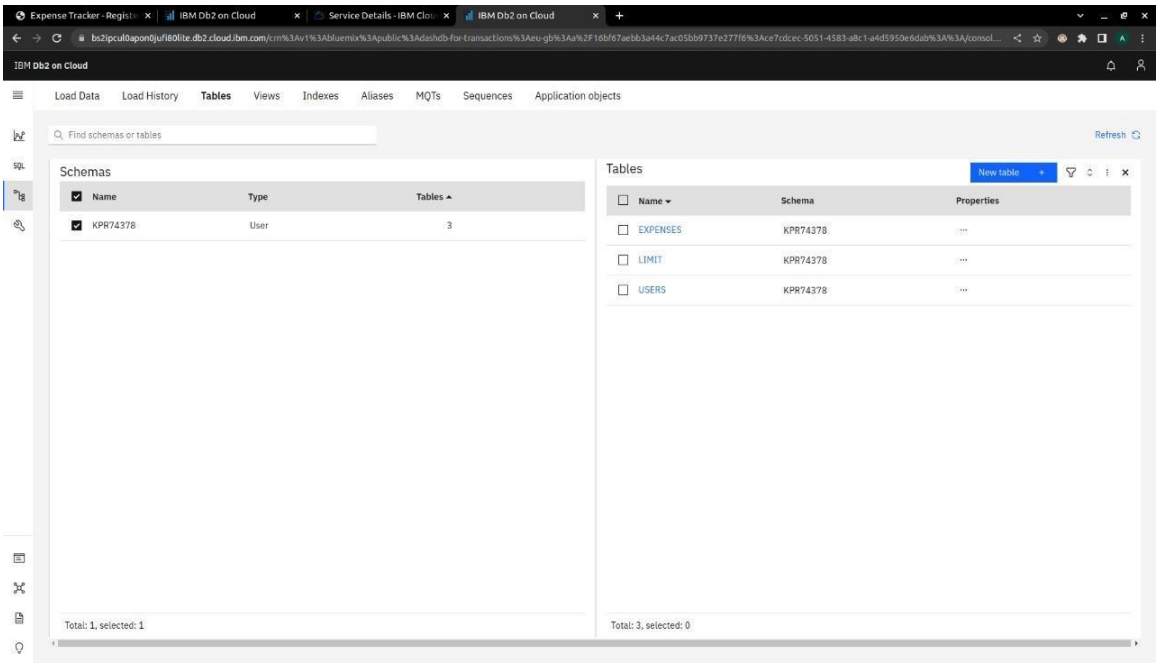

Personal Expense Tracker Applic
pnt2022tmid15641.atlassian.net/jira/software/projects/APICS/boards/2/roadmap
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Personal Expense Trac...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / Personal Expense Tracker Application
Roadmap
Give feedback
Share
Export
Status category
View settings
OCT
NOV
DEC
JAN '23
Sprints
Create Epic
Today
Weeks
Months
Quarters
Type here to search
29°C Cloudy
ENG
20:22
22-10-2022

7. Coding & Solutioning

7.1 Database Schema :





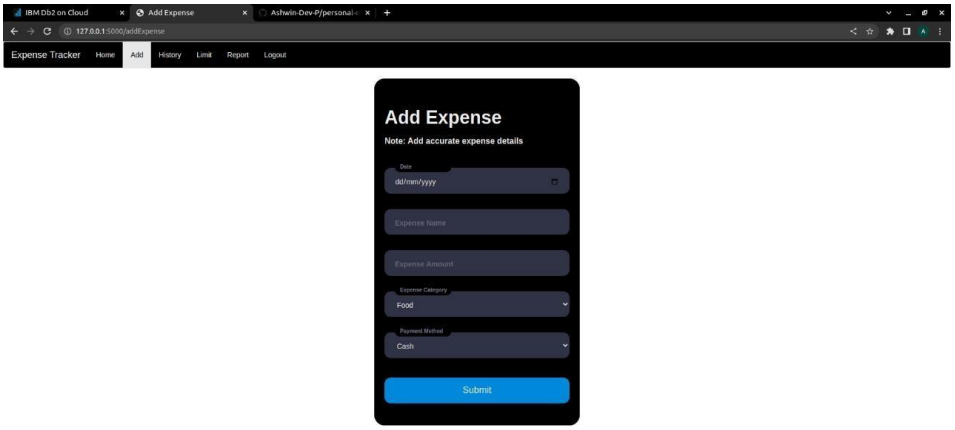

Expense Tracker - Regist
IBM Db2 on Cloud
Service Details - IBM Clou
IBM Db2 on Cloud
IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
Find schemas or tables
Refresh
Schemas
Name
Type
Tables
KPR74378
User
3
Total: 1, selected: 1
Tables
New table
Name
Schema
Properties
EXPENSES
KPR74378
LIMIT
KPR74378
USERS
KPR74378
Total: 3, selected: 0

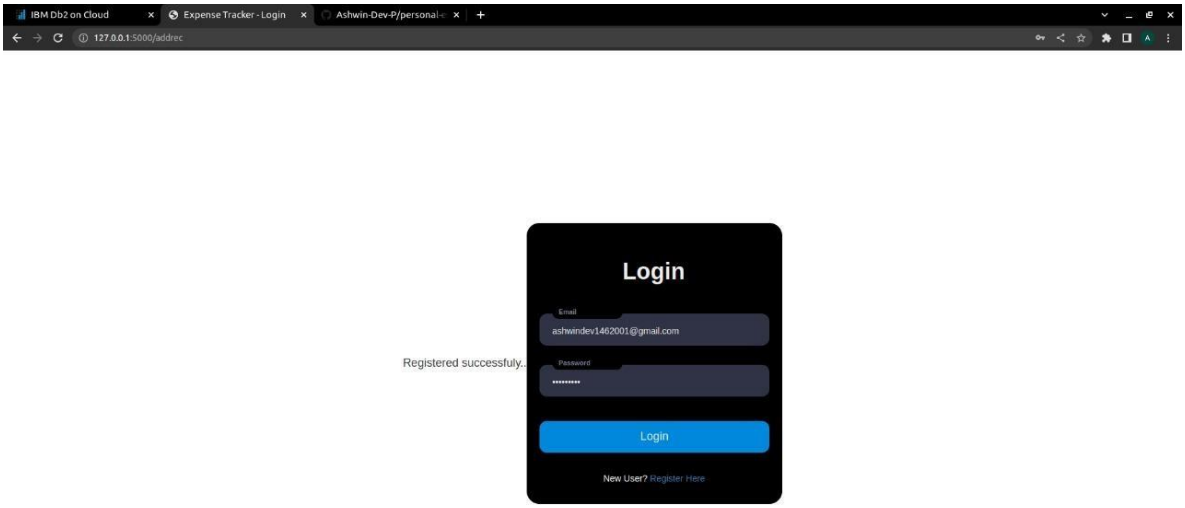
8. Testing

Testing is the process of evaluation a software item to detect differences between given input and expected output. Testing is a process should be done during the development phase.

Test Case Id	Test Description	Input Test Data	Expected Result	Actual Result	Remarks
TC-1	Install PET app in android phone	Transfer PET app	Open Application with it home page	Application executed with home page	Pass
TC-2	Enter valid data in username and password field	Chandru	Show home page for user Chandru	Displayed home page for user Chandru	Pass
TC-3	Enter a valid data in username and leave password field empty	Chandru	Show error	Didn’t show any error	Pass

9. Results


IBM Db2 on Cloud
Add Expense
127.0.0.1:5000/addExpense
Expense Tracker
Home
Add
History
Limit
Report
Logout
Add Expense
Note: Add accurate expense details
Date
dd/mm/yyyy
Expense Name
Expense Amount
Expense Category
Food
Payment Method
Cash
Submit


IBM Db2 on Cloud
Expense Tracker - Login
127.0.0.1:5000/addrec
Registered successfuly..
Login
Email
ashwindev1462001@gmail.com
Password
Login
New User? Register Here

IBM Db2 on Cloud | 127.0.0.1:5000/signin | Ashwin-Dev-P/personal

127.0.0.1:5000/signin

Expense Tracker | Home | Add | History | Limit | Report | Logout

Welcome abcdef!!

PERSONAL EXPENSE TRACKER

Add Expense

IBM Db2 on Cloud | 127.0.0.1:5000/limit | Ashwin-Dev-P/personal

127.0.0.1:5000/limit

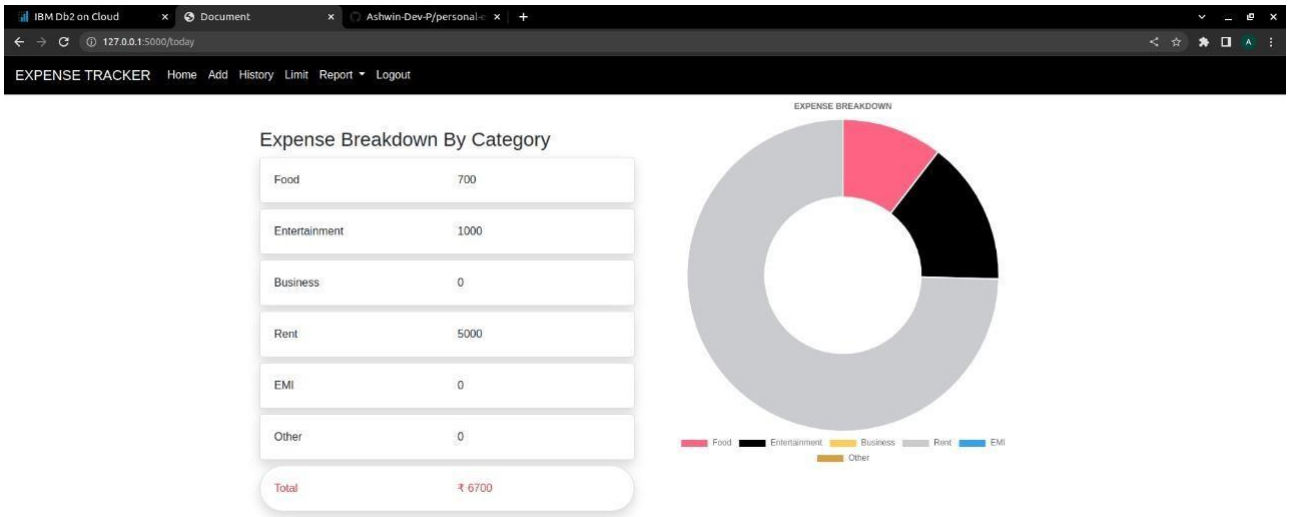
Expense Tracker | Home | Add | History | Limit | Report | Logout

ENTER the MONTHLY LIMIT to avoid over EXPENSES

ENTER

Date	Expense Name	Expense Amount	Expense Category	Payment Method
2022-11-18	Apple	100	food	cash
2022-11-18	Apple	100	food	cash
2022-11-18	Apple	100	food	cash
2022-11-18	Apple	100	food	cash
2022-11-18	Apple	100	food	cash
2022-11-18	Apple	1000	entertainment	debitCard
2022-11-20	Apple	5000	miscellaneous	debitCard
2022-11-20	Apple	5000	miscellaneous	debitCard
2022-11-18	Apple	5000	rent	creditCard
2022-11-06	Apple	100	food	cash
2022-11-18	Apple	100	food	cash
2022-11-18	Apple	100	food	cash

TOTAL Expense :
16800



Source Code:

#app.py

```
import ibm_db
from flask import Flask, redirect, render_template, request, session,
url_for
from markupsafe import escape

#mail
from flask import Flask
from flask_mail import Mail, Message



app = Flask(_name_)
mail = Mail(app)

# configuration of mail
app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] =  "chandrun0509@gmail.com "
app.config['MAIL_PASSWORD'] = "123456789"
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
mail = Mail(app)
```

```
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=824dfd4d-99de-
440d-9991-
629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=
30119;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=
kpr74378;PWD=0ijMNdNTAeNHpl5E",'','')

app.secret_key='a'


def send_mail(recipient_mail):
  msg = Message(
          'Expense tracker',
          sender ='chandrun0509@gmail.com',
          recipients = [recipient_mail]
          )
  msg.body = 'Your Expense Limit has Exceeded'
  mail.send(msg)
  return True

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

```
@app.route('/register')
def register():
  return render_template('register.html')
@app.route('/index')
def index():
  return render_template('index.html')

@app.route('/header')
def header():
    return render_template('header.html')

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



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

@app.route('/addExpense')
def addExpense():
    return render_template('AddExpense.html')
```

```
#for mail
'''
@app.route('/send_mail', methods=['GET', 'POST'])
def send_mail():
    print("method")
    print(request.method)
    if request.method == 'POST':

      print("post method activated")
      print(request.form)
      recipient = request.form['recipient']
      print(recipient)
      msg = Message('Twilio SendGrid Test Email', recipients=[recipient])
      msg.body = ('Congratulations! You have sent a test email with '
                  'Twilio SendGrid!')
      msg.html = ('<h1>Twilio SendGrid Test Email</h1>'
                  '<p>Congratulations! You have sent a test email with '
                  '<b>Twilio SendGrid</b>!</p>')
      mail.send(msg)
      flash(f'A test message was sent to {recipient}.')
      #return redirect(url_for('index'))
    return render_template('home.html')
```

```
#end for mail
'''


@app.route('/addrec',methods = ['POST', 'GET'])
def addrec():
  if request.method == 'POST':

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

    sql = "SELECT * FROM USERS WHERE NAME =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,name)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)

    if account:
      return render_template('login.html', msg="You are already a member,
please login using your details")
    else:
      insert_sql = "INSERT INTO USERS (Name,email,phone,password)
```

```
VALUES (?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, name)
    ibm_db.bind_param(prep_stmt, 2, email )
    ibm_db.bind_param(prep_stmt, 3, phone)
    ibm_db.bind_param(prep_stmt, 4, password)
    ibm_db.execute(prep_stmt)


  return render_template('login.html', msg="Registered successfuly..")



@app.route('/signin', methods =['GET', 'POST'])
def signIn():
  global userid
  msg = ''
  if request.method == 'POST':
    email = request.form['email']
    password = request.form['password']
    sql ="SELECT * FROM USERS WHERE  email = ? AND password = ?"
```

```
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt,1,email)
ibm_db.bind_param(stmt,2,password)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)
print(account)

if account:
    session['loggedin'] = True
    session['id'] = account['USERID']
    userid=account['USERID']
    session['username']=account['NAME']

    #session["name"] = request.form.get("name")

    #session['username'] = account['Name']
    msg = 'Welcome'+" "+session['username']+"!!"
    return render_template('home.html', msg = msg)
else:
    msg = 'Incorrect username / password !'
return render_template('login.html', msg = msg)
```

```
@app.route('/add', methods =['GET', 'POST'])
def add():
    global id
    if request.method=='POST':
      date=request.form['date']
      name=request.form['expenseName']
      amount=request.form['expenseAmount']
      category=request.form['expenseCategory']
      paymethod=request.form['payMethod']
      id=session['id']
      print("session id",id)




      insert_sql = "INSERT INTO EXPENSES
(USERID,DATE,NAME,AMOUNT,CATEGORY,PAYMENTMETHOD) VALUES
(?,?,?,?,?,?)"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt, 1, id)
      ibm_db.bind_param(prep_stmt, 2, date )
```

```
ibm_db.bind_param(prep_stmt, 3, name)
ibm_db.bind_param(prep_stmt, 4, amount)
ibm_db.bind_param(prep_stmt, 5, category)
ibm_db.bind_param(prep_stmt, 6, paymethod)
ibm_db.execute(prep_stmt)

limit="SELECT AMOUNT FROM LIMIT WHERE USERID=?"
stmt = ibm_db.prepare(conn, limit)
ibm_db.bind_param(stmt,1,id)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)
print(account)
limit_amount=account['AMOUNT']
print(limit_amount)

total="SELECT SUM(AMOUNT) FROM EXPENSES WHERE USERID=?"
stmt = ibm_db.prepare(conn, total)
ibm_db.bind_param(stmt,1,id)
ibm_db.execute(stmt)
account= ibm_db.fetch_assoc(stmt)

print("account")
print( account)
```

```
  print("account id")
  print(id)
  total_amount=account['1']
  print("total_amount")
  print(total_amount)

  if (int(total_amount)>int(limit_amount)):
    print("Limit exceeded")
    account_stmt="SELECT EMAIL FROM USERS WHERE USERID=?"
    stmt = ibm_db.prepare(conn, account_stmt)
    ibm_db.bind_param(stmt,1,id)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    print(account)

    send_mail(account['EMAIL'])
    return render_template("limitwarn.html")
  else:
    print(total_amount, limit_amount)
return render_template('AddExpense.html')
```

```
@app.route('/history')
def history():
  global id
  id=session['id']
  print(session['username'])
  students = []
  total=0
  sql = "SELECT * FROM EXPENSES where USERID=?"
  stmt = ibm_db.prepare(conn, sql)
  ibm_db.bind_param(stmt,1,id)
  ibm_db.execute(stmt)
  dictionary = ibm_db.fetch_both(stmt)

  '''limit="SELECT AMOUNT FROM LIMIT WHERE USERID=?"
  stmt = ibm_db.prepare(conn, limit)
  ibm_db.bind_param(stmt,1,id)
  ibm_db.execute(stmt)
  account = ibm_db.fetch_assoc(stmt)
  print(account)
```

```
 amount=account['AMOUNT']
 print(amount)'''
 while dictionary != False:
  # print ("The Name is : ",  dictionary)
  students.append(dictionary)
  total+=int(dictionary[3])
  dictionary = ibm_db.fetch_both(stmt)



 if students:
  return render_template("history.html", students = students,total=total)


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



@app.route('/limitnum' , methods = ['POST' ])
def limitnum():
   if request.method == "POST":
```

```
        number= request.form['number']
        sql = "INSERT INTO LIMIT(USERID,AMOUNT) VALUES(?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,session['id'])
        ibm_db.bind_param(stmt,2,number)
        ibm_db.execute(stmt)
        return render_template("today.html")

@app.route('/logout')
def logout():
  session.pop('loggedin', None)
  session.pop('id', None)
  session.pop('username', None)
  return render_template('register.html')

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



    sql = "SELECT * FROM EXPENSES  WHERE userid =? AND date =
DATE(NOW())"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
```

```
    list2=[]
    texpense=ibm_db.fetch_tuple(stmt)
    print(texpense)



    sql = "SELECT * FROM EXPENSES WHERE USERID=? AND DATE(date) =
DATE(NOW())"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    list1=[]
    expense = ibm_db.fetch_tuple(stmt)
    while(expense):
    list1.append(expense)
      expense = ibm_db.fetch_tuple(stmt)

    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0
```

```
for x in list1:
  total +=int(x[3])
  if x[4] == "food":
    t_food += int(x[3])

  elif x[4] == "entertainment":
      t_entertainment  += int(x[3])

  elif x[4] == "business":
       t_business  += int(x[3])
  elif x[4] == "rent":
    t_rent  += int(x[3])

  elif x[4] == "emi":
    t_EMI  += int(x[3])

  elif x[4] == "Miscellaneous":
    t_other  += int(x[3])
```

```
    return render_template("today.html", texpense = list1, 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():
    sql = "SELECT MONTHNAME(DATE),SUM(AMOUNT) FROM EXPENSES
WHERE USERID=? GROUP BY MONTHNAME (DATE)"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    list2=[]
    texpense = ibm_db.fetch_tuple(stmt)
    while(texpense):
    list2.append(texpense)
     texpense = ibm_db.fetch_tuple(stmt)
    print(list2)
```

```
    sql = "SELECT * FROM EXPENSES WHERE USERID=? AND
MONTH(date)=MONTH(DATE(NOW()))"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    list1=[]
    expense = ibm_db.fetch_tuple(stmt)
    while(expense):
    list1.append(expense)
      expense = ibm_db.fetch_tuple(stmt)
    print(list1)
    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
    t_EMI=0
    t_other=0



    for x in list1:
```

```
  total += int(x[3])
  if x[4] == "food":
    t_food +=int(x[3])

  elif x[4] == "entertainment":
    t_entertainment  += int(x[3])

  elif x[4] == "business":
  t_business += int(x[3])
  elif x[4] == "rent":
  t_rent  += int(x[3])

  elif x[4] == "emi":
    t_EMI  += int(x[3])

  elif x[4] == "Miscellaneous":
    t_other  += int(x[3])

print(total)

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

```
    print(t_EMI)
    print(t_other)



    return render_template("month.html", texpense = list2, expense =
expense,  total = total ,
                    t_food = t_food,t_entertainment =  t_entertainment,
                    t_business = t_business,  t_rent =  t_rent,
                    t_EMI =  t_EMI,  t_other =  t_other )



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



    sql = "SELECT YEAR(DATE),SUM(AMOUNT) FROM EXPENSES WHERE
USERID=? GROUP BY YEAR(DATE)"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    list2=[]
    texpense = ibm_db.fetch_tuple(stmt)
```

```
    while(texpense):
      list2.append(texpense)
      texpense = ibm_db.fetch_tuple(stmt)
    print(list2)




    sql = "SELECT * FROM EXPENSES WHERE USERID=? AND
YEAR(date)=YEAR(DATE(NOW()))"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    list1=[]
    expense = ibm_db.fetch_tuple(stmt)
    while(expense):
    list1.append(expense)
      expense = ibm_db.fetch_tuple(stmt)

    total=0
    t_food=0
    t_entertainment=0
    t_business=0
    t_rent=0
```

```
t_EMI=0
t_other=0


for x in list1:
  total += int(x[3])
  if x[4] == "food":
    t_food +=int(x[3])

  elif x[4] == "entertainment":
    t_entertainment  += int(x[3])

  elif x[4] == "business":
  t_business += int(x[3])
  elif x[4] == "rent":
  t_rent  += int(x[3])

  elif x[4] == "emi":
    t_EMI  += int(x[3])

  elif x[4] == "Miscellaneous":
    t_other  += int(x[3])
```

```
    print(total)

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



    return render_template("year.html", texpense = list2, 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 )

if _name_ =='_main_':
   app.run(host='0.0.0.0',debug=True)
```

Advantages

1. You have no control over your money If you don't check your spending and create a budget, you will have no control whatsoever on your money. Instead, money will control you, and you will either have perpetual lack of funds or you will end up steeped in debt. A money manager app helps you decide between short-term and long-term spending .

2.You have no financial goals If you are spending money frivolously, you will not have money to set financial goals. However, when you have a daily expense manager, you will be able to work with limited resources and use your money in a wise manner so that you can create financial goals and ensure you meet them.

3. You are unaware what is happening with your money If you are clueless about how much is your inflow and how much you are spending, you will not know at the end of the month what happened to your money. An expense tracker helps you figure out what is happening to your money, and whether you can afford something you want.

4. You spend and save in a haphazard manner

If you don't have great financial management skills, you will not know how to categorize your expenses. However, tracking your expenses and budgeting them will help you become aware of how much you have to allocate to each expense category, and if you are short, you will be able to make adjustments with ease.

5. You have no clue about making your money work for you, In this day and age, when expenses are going through the roof, it has become crucial that you learn to make your money work for you so that you can create a nest egg for the future.

6. You don't have funds for emergencies Remember, emergencies come when you least expect. Hence, if you don't have money stashed away for a rainy day, you will end up borrowing from family and friends. This way you could get into debt that will be difficult to pay back due to your poor money management skills

Disadvantages

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 or 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're 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

Conclusion

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

Github & Project Demo Link:

https://github.com/IBM-EPBL/IBM-Project-41768-1660644713