Personal Expense Tracker Application

Project Report

Team ID	**PNT2022TMID06539**
Project Name	**Personal Expense Tracker Application**
Team Members	**VIGNESHWARAN A** **NELSON J** **SANTHOSH E** **VERJIN V**

CONTENTS

INTRODUCTION

a. Project overview

Mobile applications are top in user convenience and have over passed the web applications in terms of popularity and usability. There are various mobile applications that provide solutions to manage personal and group expense but not many of them provide a comprehensive view of both cases. In this paper, we develop a mobile application developed for the android platform that keeps record of user personalexpenses, his/her contribution in group expenditures, top investment options,view of the current stock market, read authenticated financial news and grab the best ongoingoffers in the market in popular categories. With our application can manage their expenses and decide on theirbudget more effectively.

b. Purpose

It also known as expense manager and money manager, an expensetracker is a software or application that helps to keep an accurate recordof your money inflow and outflow. Many people in India live on a fixed income, and they find that towards the end of the month they don'thave sufficient money to meet theirneeds.

2.LITERATURE SURVEY

a. Existing Problem

The problem of current generation population is that they can’t remember where all of the money they earned have gone and ultimately have to live while sustaining the little money they have left for their essential needs. In this time there is no such perfect solution which helps a person to track theirdaily expenditure easilyand efficiently and notify them about the money shortage they have. For doing so have to maintain long ledgers or computer logs to maintain such data and the calculation is done manually by the user, which may generate error leading to losses.

b. Reference

i. https://nevonprojects.com/daily-expense-tracker-system/

ii. https://data-flair.training/blogs/expense-tracker-python/

iii. https://phpgurukul.com/daily-expense-tracker-using-php-and-mysql/

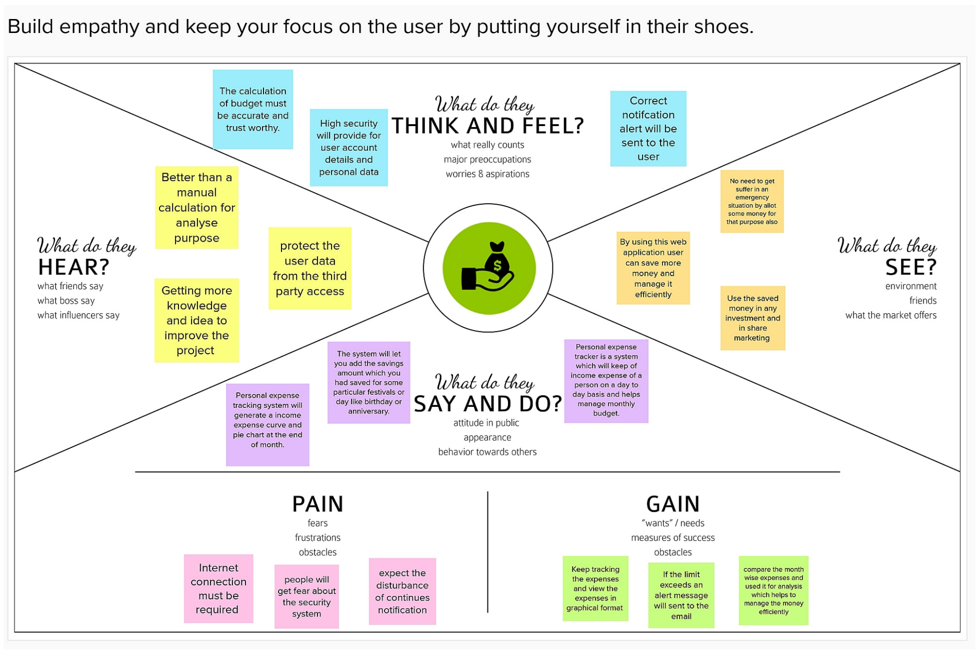
iv. https://ijarsct.co.in/Paper391.pdf

Problem statement definition:

This Expense Tracker is a web application that facilitates the users to keep trackand manage their personal as well as business expenses. This application helpsthe users to keep a digital diary. It will keep track of a user's income and expenses on a daily basis. The user will be able to add his/her expendituresinstantly and can review them anywhere and anytime with the help of the internet. He/she can easily import transactions from his/her mobile wallets without risking his/her information and efficiently protecting his/her privacy. This expense tracker provides a complete digital solutionto this problem.Excel sheets do very little to help in tracking Furthermore, they don't have the advanced functionality of preparing graphical visuals automatically. Not only it will save the time of the peoplebut also it will assure error free calculations. The user just has to enter the income and expenditures and everything else will be performedby the system. Keywords: Expense Tracker, budget, planning, savings, graphical visualization of expenditure.

IDEATION & PROPOSED SOLUTION

i. Empathy Map canvas



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

What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations

The calculation of budget must be accurate and trust worthy.

High security will provide for user account details and personal data

Correct notifcation alert will be sent to the user

What do they HEAR?
what friends say
what boss say
what influencers say

Better than a manual calculation for analyse purpose

Getting more knowledge and idea to improve the project

protect the user data from the third party access

What do they SEE?
environment
friends
what the market offers

No need to get suffer in an emergency situation by allot some money for that purpose also

By using this web application user can save more money and manage it efficiently

Use the saved money in any investment and in share marketing

What do they SAY AND DO?
attitude in public
appearance
behavior towards others

Personal expense tracking system will generate a income expense curve and pie chart at the end of month.

The system will let you add the savings amount which you had saved for some particular festivals or day like birthday or anniversary.

Personal expense tracker is a system which will keep of income expense of a person on a day to day basis and helps manage monthly budget.

PAIN
fears
frustrations
obstacles

Internet connection must be required

people will get fear about the security system

expect the disturbance of continues notification

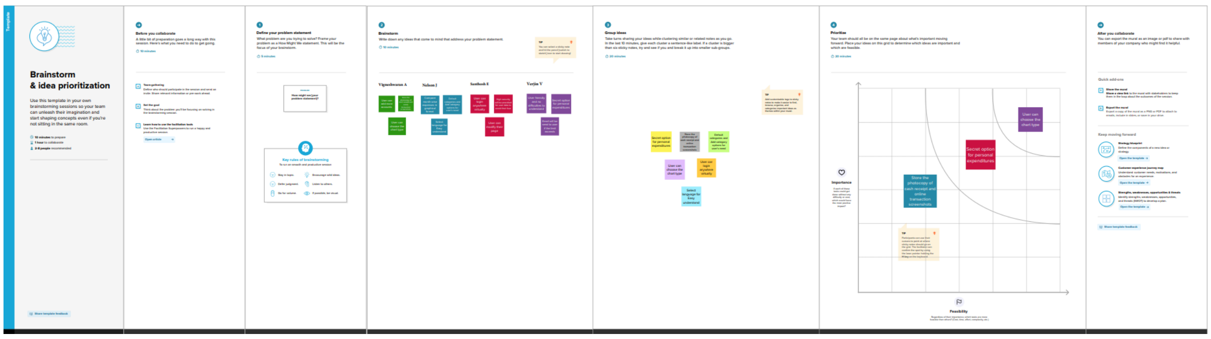
GAIN
"wants" / needs
measures of success
obstacles

Keep tracking the expenses and view the expenses in graphical format

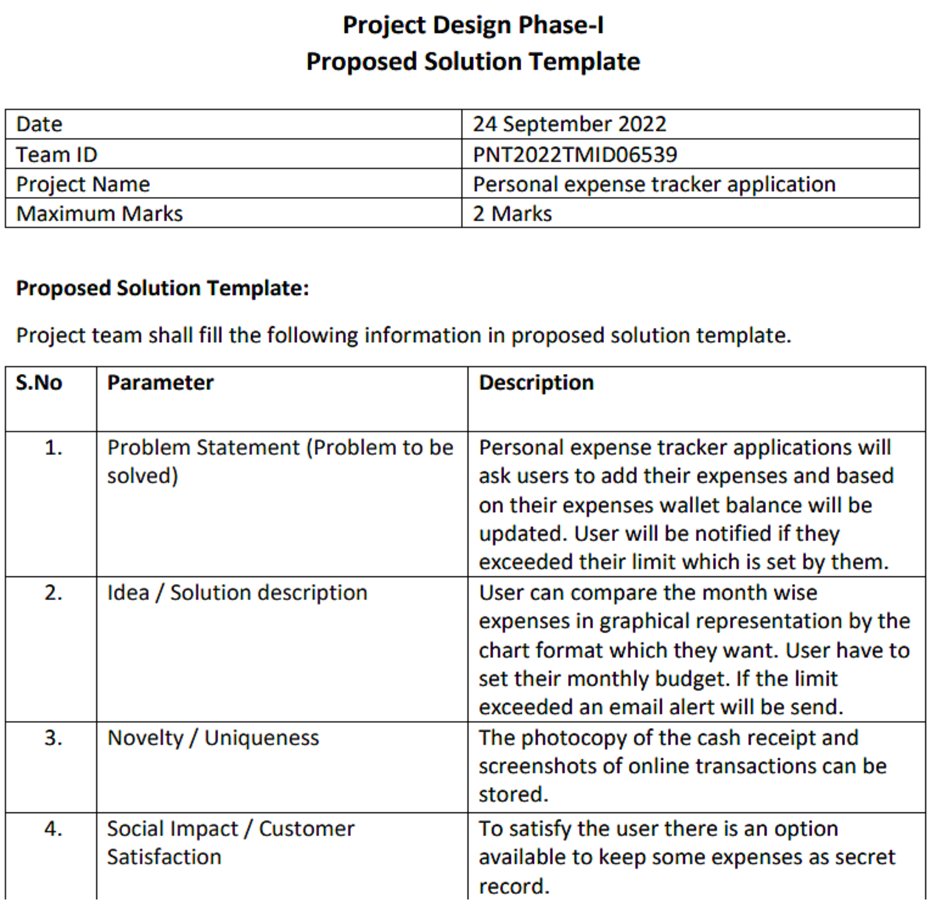
If the limit exceeds an alert message will sent to the email

compare the month wise expenses and used it for analysis which helps to manage the money efficiently

ii. Ideation & Brainstorming



Proposed Solution



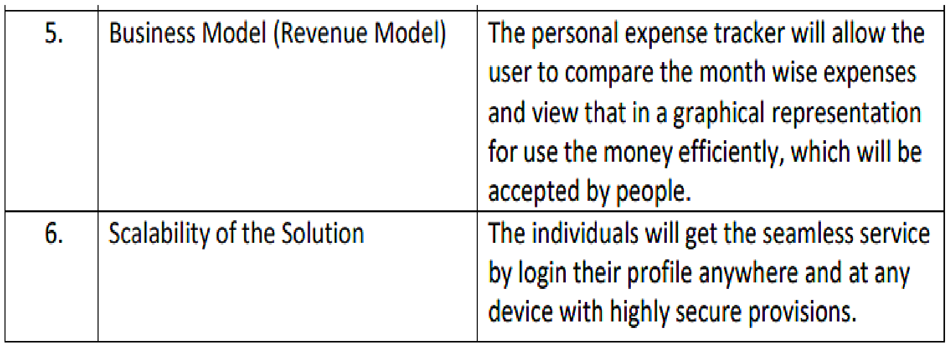
Project Design Phase-I
Proposed Solution Template

Date	24 September 2022
Team ID	PNT2022TMID06539
Project Name	Personal expense tracker application
Maximum Marks	2 Marks

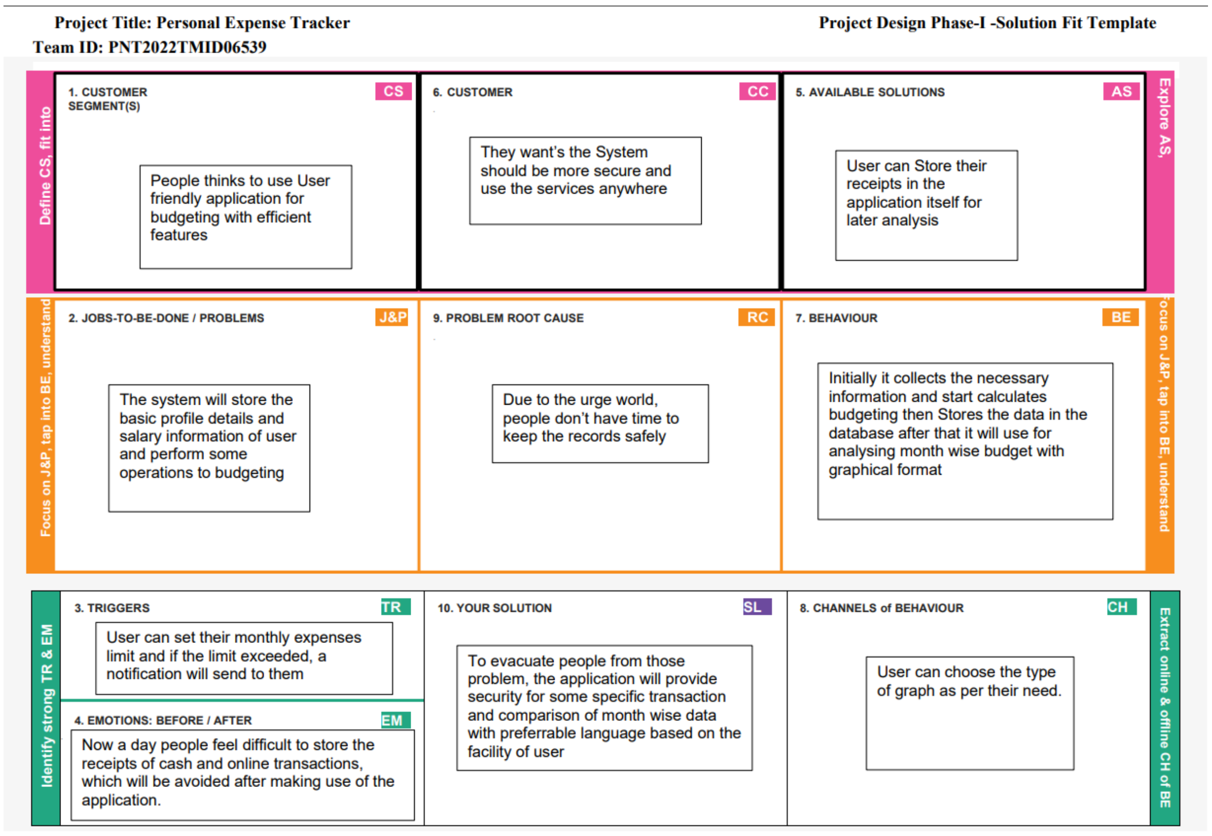
Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	Personal expense tracker applications will ask users to add their expenses and based on their expenses wallet balance will be updated. User will be notified if they exceeded their limit which is set by them.
2.	Idea / Solution description	User can compare the month wise expenses in graphical representation by the chart format which they want. User have to set their monthly budget. If the limit exceeded an email alert will be send.
3.	Novelty / Uniqueness	The photocopy of the cash receipt and screenshots of online transactions can be stored.
4.	Social Impact / Customer Satisfaction	To satisfy the user there is an option available to keep some expenses as secret record.
5.	Business Model (Revenue Model)	The personal expense tracker will allow the user to compare the month wise expenses and view that in a graphical representation for use the money efficiently, which will be accepted by people.
6.	Scalability of the Solution	The individuals will get the seamless service by login their profile anywhere and at any device with highly secure provisions.



a. Proposed Solution Fit



Project Title: Personal Expense Tracker **Project Design Phase-I -Solution Fit Template**

Team ID: PNT2022TMID06539

Define CS, fit into

1. CUSTOMER SEGMENT(S) CS

People thinks to use User friendly application for budgeting with efficient features

6. CUSTOMER CC

They want's the System should be more secure and use the services anywhere

5. AVAILABLE SOLUTIONS AS

User can Store their receipts in the application itself for later analysis

Explore AS,

Focus on J&P, tap into BE, understand

2. JOBS-TO-BE-DONE / PROBLEMS J&P

The system will store the basic profile details and salary information of user and perform some operations to budgeting

9. PROBLEM ROOT CAUSE RC

Due to the urge world, people don't have time to keep the records safely

7. BEHAVIOUR BE

Initially it collects the necessary information and start calculates budgeting then Stores the data in the database after that it will use for analysing month wise budget with graphical format

ocus on J&P, tap into BE, understand

Identify strong TR & EM

3. TRIGGERS TR

User can set their monthly expenses limit and if the limit exceeded, a notification will send to them

4. EMOTIONS: BEFORE / AFTER EM

Now a day people feel difficult to store the receipts of cash and online transactions, which will be avoided after making use of the application.

10. YOUR SOLUTION SL

To evacuate people from those problem, the application will provide security for some specific transaction and comparison of month wise data with preferrable language based on the facility of user

8. CHANNELS of BEHAVIOUR CH

User can choose the type of graph as per their need.

Extract online & offline CH of BE

b. Requirement analysis

Project Design Phase-II
Solution Requirements (Functional & Non-functional)

Date	16 October 2022
Team ID	PNT2022TMID06539
Project Name	Personal Expense Tracker Application
Maximum Marks	4 Marks

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	User Confirmation	Confirmation via Email
FR-3	Tracking Expense	Helpful insights about money management
FR-4	Alert Message	Give alert mail if the amount exceeds the budget limit

Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	You will able to allocate money to different priorities and also help you to cut down on unnecessary spending
NFR-2	**Security**	It employs the latest security and technology measures to keep customers personal and financial information safe
NFR-3	**Reliability**	• Used to manage his/her expense so that the user is the path of financial stability. • It is categorized by week, month, and year and also helps to see more expenses made. • Helps to define their own categories.
NFR-4	**Performance**	Help to gain control of your finance, pay down debt, grow your net worth, help to upload receipts, track mileage
NFR-5	**Availability**	Able to track business expense and monitor important for maintaining healthy cash flow but also qualifying for deductions that could reduce your taxable income
NFR-6	**Scalability**	To know where money goes and you can ensure that money is used widely

5. PROJECT DESIGN

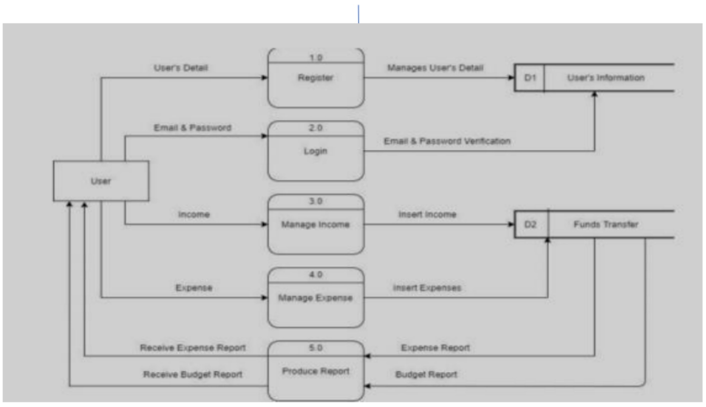
a. Data Flow Diagrams

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 wheredata is store.

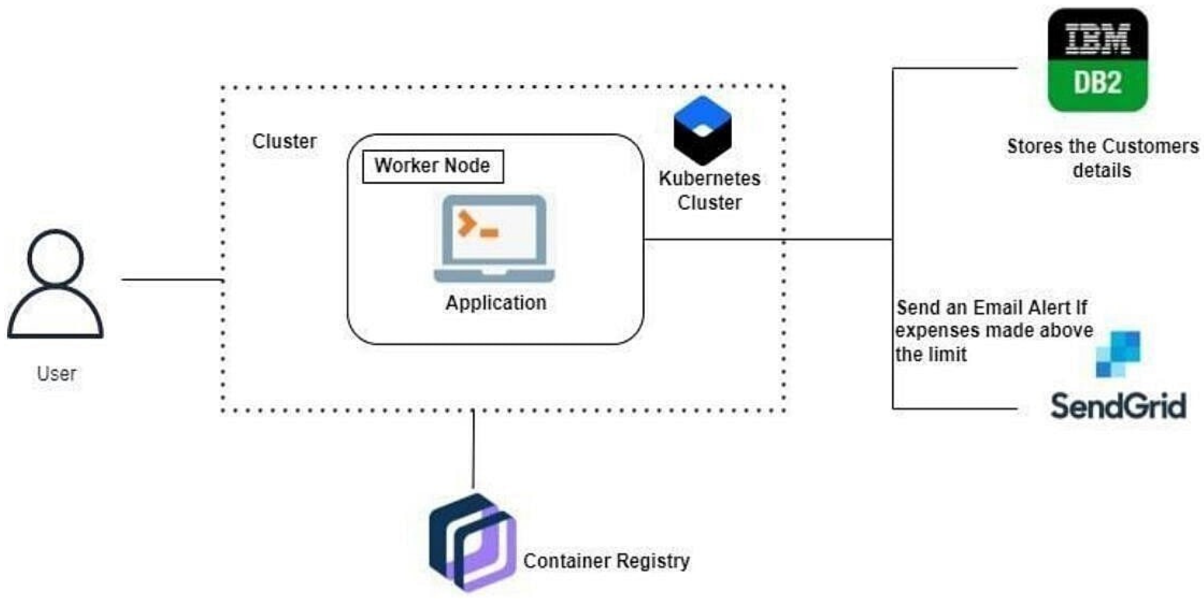
Project Design Phase-II
Data Flow Diagram & User Stories

Date	16 October 2022
Team ID	PNT2022TMID06539
Project Name	Personal Expense Tracker Application
Maximum Marks	4 Marks

Data Flow diagram:



b. Solution & Technical Architecture


Cluster
Worker Node
Application
Kubernetes Cluster
User
Stores the Customers details
Send an Email Alert If expenses made above the limit
Container Registry

c. User Stories

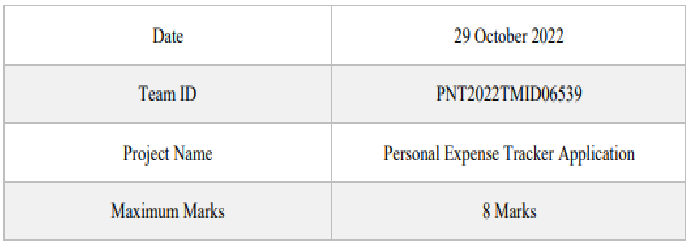
User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user and web 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 form	I can register by entering the details	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access my dashboard	High	Sprint-1
	Dashboard	USN-6	As a user ,I can log into the dashboard and manage income	I can add, delete and modify the income	High	Sprint-1
		USN-7	As a user, I can log into the dashboard and manage expense	I can add, delete and modify the expenses	High	Sprint-1
		USN-8	As a user, I can get a report is based on the details	I can manage my money by viewing this report	Medium	Sprint-1
Administrator	Alert message	USN-9	As a user, I can get an email if the money level is above the limit	I can receive alert email	High	Sprint-1
	Database	USN-10	As a user , I can't able to see the database but the details are automatically stored on the database	Based on the details on the database, I can get the details of money monthly through email	High	Sprint-1

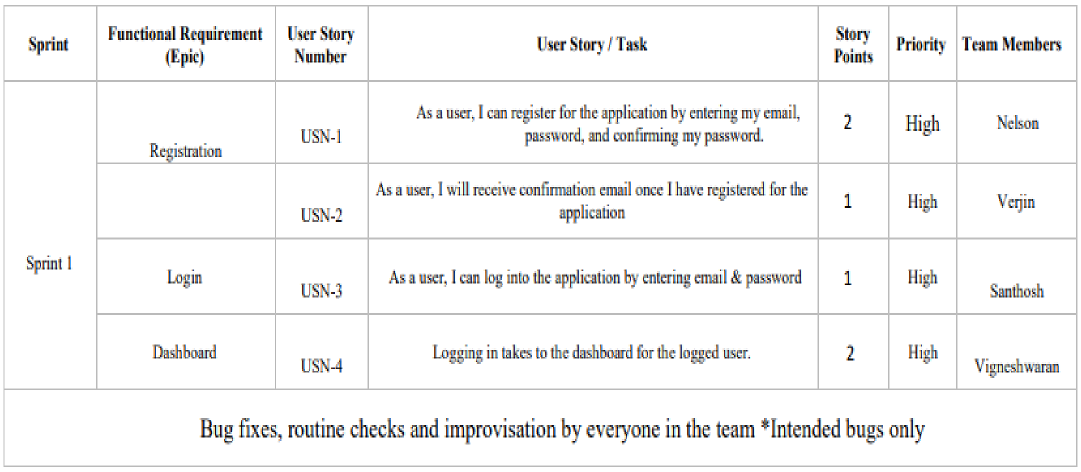
6.PROJECT PLANNING & SCHEDULING

a. Sprint Planning & Estimation



Date	29 October 2022
Team ID	PNT2022TMID06539
Project Name	Personal Expense Tracker Application
Maximum Marks	8 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)



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	Nelson
		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Verjin
	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Santhosh
	Dashboard	USN-4	Logging in takes to the dashboard for the logged user.	2	High	Vigneshwaran
Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only						

Sprint 2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Santhosh
	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Verjin
	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Santhosh
		USN-4	Making dashboard interactive with JS	2	High	Nelson
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Nelson
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user's query	1	Medium	Verjin
	SendGrid	USN-3	Using SendGrid to send mail to the user about their expenses	1	Low	Vigneshwaran
		USN-4	Integrating both frontend and backend	2	High	Santhosh
Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only						
Sprint-4	Docker	USN-1	Creating image of website using docker	2	High	Santhosh
	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Nelson
	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Vigneshwaran
	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Verjin

b. Sprint Delivery Schedule

Date	29 October 2022
Team ID	PNT2022TMID06539
Project Name	Personal Expense Tracker Application
Maximum Marks	8 Marks

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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	30 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	06 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	13 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	20 Nov 2022

Velocity

We have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). Calculating the team's average velocity (AV) per iteration unit (story points per day)

7.Coding and Solutioning Features

Feature 1: Add Expense
Feature 2: Update Expense
Feature 3: Delete Expense
Feature 4: Set Limit
Feature 5: Send Alert Emails to users

<u>Other Features</u>

Track your expenses anywhere, anytime. Seamlessly manage your money and budget without any financial paperwork. Just click and submit your invoices and expenditures. Access, submit, and approve invoices irrespective of time and location. Avoid data loss by scanning your tickets andbills and saving in the app. Approvalof bills and expenditures in real-time and get notified instantly. Quicksettlement of claims and reduced human errors with an automatedand streamlined billingprocess.

Coding:

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

import ibm_db
import re
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase


app = Flask(__name__)

app.secret_key = 'a'

conn=ibm_db.connect("DATABASE=bludb;HOSTNAM
E=824dfd4d-99de-440d-9991-
629c01b3832d.bs2io90l08kqb1olcg.databases.appd
omain.cloud;PORT=30119;SECURITY=SSL;SSLServer
Certificate=DigiCertGlobalRootCA.crt;UID=mtq37014;
PWD=W4Sam6RCrj9zDrfD;",",")


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

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

```
#SIGN--UP--OR--REGISTER


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



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




        sql = "SELECT * FROM REGISTER WHERE
USERNAME =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)

        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and
numbers !'
        else:
            sql1="INSERT INTO
REGISTER(USERNAME,PASSWORD,EMAIL)
VALUES(?,?,?)"
            stmt1 = ibm_db.prepare(conn, sql1)

            ibm_db.bind_param(stmt1,1,username)
            ibm_db.bind_param(stmt1,2,password)
            ibm_db.bind_param(stmt1,3,email)
            ibm_db.execute(stmt1)
            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']
        sql = "SELECT * FROM REGISTER WHERE
USERNAME =? AND PASSWORD =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)


        if account:
            session['loggedin'] = True
            session['id'] = account["ID"]
            userid=  account["ID"]
            session['username'] = account["USERNAME"]
            session['email']=account["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',methods=['GET', 'POST'])
def addexpense():

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

    sql = "INSERT INTO
EXPENSES(USERID,DATE,EXPENSENAME,AMOUNT,P
AYMENTMODE,CATEGORY,TIME)
VALUES(?,?,?,?,?,?,?)"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.bind_param(stmt,2,date)
    ibm_db.bind_param(stmt,3,expensename)
    ibm_db.bind_param(stmt,4,amount)
    ibm_db.bind_param(stmt,5,paymode)
    ibm_db.bind_param(stmt,6,category)
    ibm_db.bind_param(stmt,7,time)
    ibm_db.execute(stmt)


    print(date + " " + expensename + " " + amount + " "
+ paymode + " " + category)

    sql1 = "SELECT * FROM EXPENSES WHERE
USERID=? AND
MONTH(date)=MONTH(DATE(NOW()))"
    stmt1 = ibm_db.prepare(conn, sql1)
    ibm_db.bind_param(stmt1,1,session['id'])
    ibm_db.execute(stmt1)
    list2=[]
    expense1 = ibm_db.fetch_tuple(stmt1)
    while(expense1):
        list2.append(expense1)
        expense1 = ibm_db.fetch_tuple(stmt1)
    total=0
    for x in list2:
          total += x[4]

    sql2 = "SELECT EXPLIMIT FROM LIMITS ORDER BY
```

```
LIMITS.ID DESC LIMIT 1"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.execute(stmt2)
    limit=ibm_db.fetch_tuple(stmt2)

    if(total>limit[0]):



        mail_from = '19i304@psgtech.ac.in'
        mail_to = session['email']

        msg = MIMEMultipart()
        msg['From'] = mail_from
        msg['To'] = mail_to
        msg['Subject'] = 'Expense Alert Limit'
        mail_body = """
        Dear User, You have exceeded the specified
monthly expense Limit!!!!

        """
        msg.attach(MIMEText(mail_body))

        try:
            server =
smtplib.SMTP_SSL('smtp.sendgrid.net', 465)
            server.ehlo()
            server.login('apikey',
'SG.abtZTw0XTv6MWJXdiVW2sg.r_1bDQUJUwsDAtc
xaVKQClBW9akQCV0cOy02XtN1Uwo')
```

```
            server.sendmail(mail_from, mail_to,
msg.as_string())
            server.close()
            print("mail sent")
        except:
            print("issue")


    return redirect("/display")




#DISPLAY---graph

@app.route("/display")
def display():
    print(session["username"],session['id'])

    sql = "SELECT * FROM EXPENSES WHERE
USERID=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    list1=[]
    row = ibm_db.fetch_tuple(stmt)
    while(row):
        list1.append(row)
        row = 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 += 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]
```

```
   return render_template('display.html' ,expense =
list1,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)




#delete---the--data

@app.route('/delete/<string:id>', methods = ['POST',
'GET' ])
def delete(id):
    print(id)
    sql = "DELETE FROM expenses WHERE  id =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,id)
    ibm_db.execute(stmt)

    return redirect("/display")




#UPDATE---DATA

@app.route('/edit/<id>', methods = ['POST', 'GET' ])
```

```
def edit(id):


    sql = "SELECT * FROM expenses WHERE  id =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,id)
    ibm_db.execute(stmt)
    row=ibm_db.fetch_tuple(stmt)

    print(row)
    return render_template('edit.html', expenses = row)




@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']
      time=request.form["time"]



      sql = "UPDATE expenses SET date =? ,
expensename =? , amount =?, paymentmode =?,
category =?, time=? WHERE expenses.id =? "
      stmt = ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt,1,date)
      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,time)
      ibm_db.bind_param(stmt,7,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']


         sql = "INSERT INTO LIMITS(USERID,EXPLIMIT)
```

```
VALUES(?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,session['id'])
        ibm_db.bind_param(stmt,2,number)
        ibm_db.execute(stmt)
        return redirect('/limitn')


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


    sql = "SELECT EXPLIMIT FROM LIMITS ORDER BY
LIMITS.ID DESC LIMIT 1"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    row=ibm_db.fetch_tuple(stmt)


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

#REPORT

@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 += 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]





    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)

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 += 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("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 += 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("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 )

#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')



if __name__ == "__main__":
```

```
app.run(debug=True)
```

8.TESTING

a. Testing:

i. Login Page (Functional)
ii. Login Page (UI)
iii. Add Expense Page (Functional)

b. User Acceptance Testing:

1. Purpose of Document

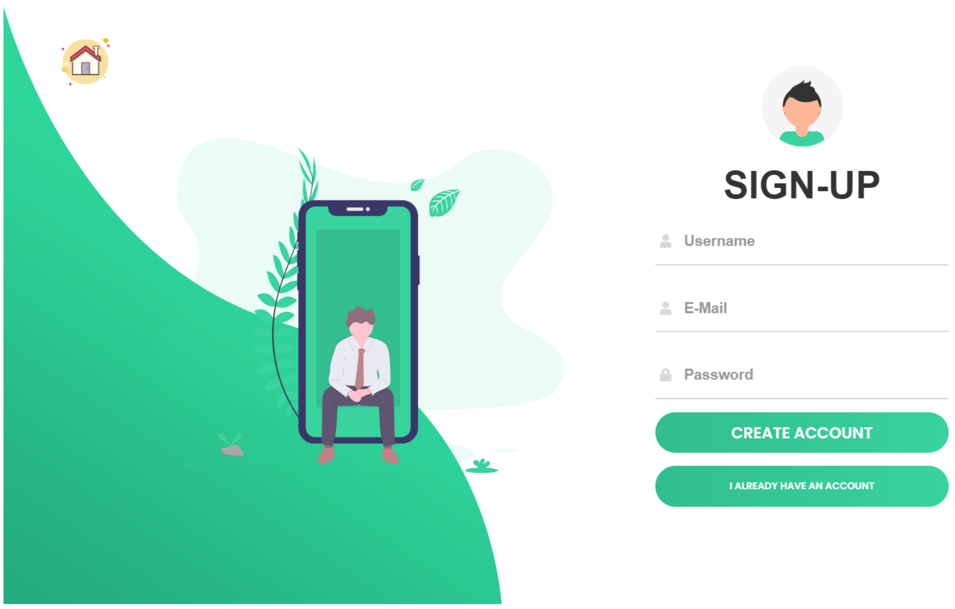
The purpose of this document is to briefly explain the test coverageand openissues of [productname] project time of the release to user acceptance testing (UAT)

2. Defect Analysis

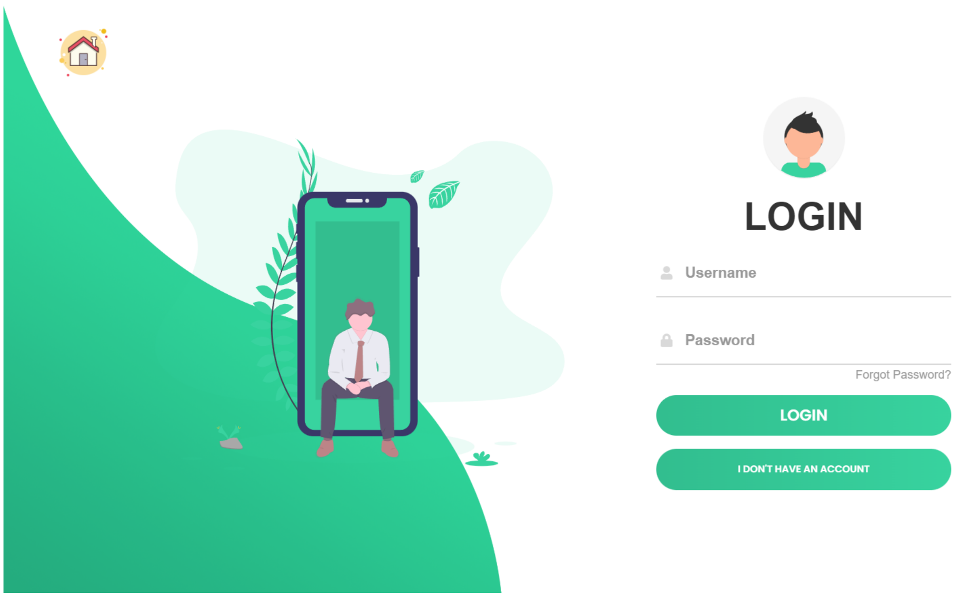
This report shows the numberof resolved or closed bugs at each severity level,and how they are resolved.

c.Screenshots:

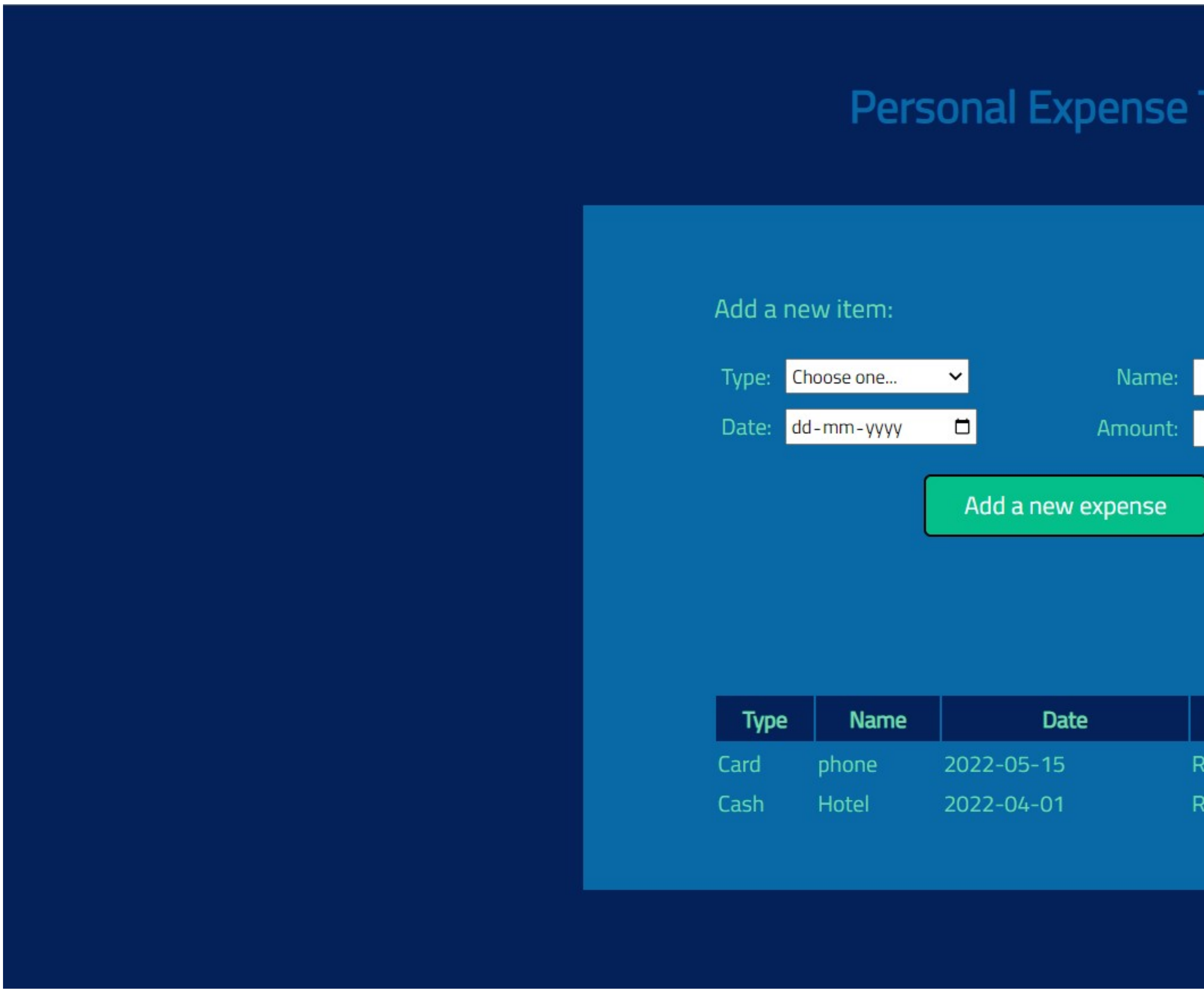
a. **Sign Up Page:**



b. **Login Page:**



c.**<u>Break down of ExpensePage:</u>**



Personal Expense

Add a new item:

Type: Choose one...

Name:

Date: dd-mm-yyyy

Amount:

Add a new expense

Type	Name	Date
Card	phone	2022-05-15
Cash	Hotel	2022-04-01

ADVANTAGES AND DISADVANTAGES

<u>ADVANTAGES:</u>

One of the major pros of trackingspending is alwaysbeing aware of the stateof one's personal finances. Tracking what you spend can help you stick to your budget, not just in a general way, but in each category such as housing, food, transportation and gifts. While a con is that manually tracking all cash that is spent can be irritating as well as time consuming, a pro is that doing this automatically can be quick and simple. Another pro is that many automaticspending tracking softwareprograms are availablefor free. Having the program on a hand-held device can be a main pro since it can be checked before spending occursin order to be sure of the available budget.

DISADVANTAGES:

A con with any system used to track spending is that one may start doing it then taper off until it's forgotten about all together. Yet, this is a risk for any new goal such as trying to lose weight or quit smoking. If a person first makes a budget plan, then places money in savings before spending any each new pay period or month, the tracking goal can help. In this way, tracking spendingand making sure all receiptsare accounted for only needsto be done once or twice a month. Even with constant tracking of one's spending habits, there is no guarantee that financial goals will be met.Although this can be considered to be a con of tracking spending, it could be changed into a pro if one makes up his or her mind to keep trying to properly manage all finances.

CONCLUSION

A comprehensive money management strategy requires clarity and conviction for decision- making. You will need a defined goal and a clear vision for grasping the business and personal finances. That’s when an expense tracking app comes into the picture. An expense tracking app is an exclusive suite of services for people who seek to handle their earnings andplantheir expenses and savings efficiently. It helps you track all transactions like bills, refunds, payrolls, receipts, taxes, etc., on a daily, weekly, and monthly basis.

FUTURE SCOPE

a. Achieve your business goals with a tailored mobileapp that perfectly fits your business.
b. Scale-up at the pace your business is growing.
c. Deliver an outstanding customerexperience through additionalcontrol over the app.
d. Control the securityof your business and customer data.
e. Open direct marketing channelswith no extra costs with methods suchas push notifications.
f. Boost the productivity of all the processes withinthe organization.
g. Increase efficiency and customer satisfaction with an app alignedtotheir needs.
h. Seamlessly integrate with existing infrastructure.
i. Ability to provide valuable insights.
j. Optimize sales processes to generate more revenue through enhanceddata collection.
k. **Chats:** Equip your expensetracking app with a bot that can understandand answer all user queries and address their needs such as account balance, credit score, etc.
l. **Prediction:** With the help of AI, your mobile app can predict your next purchase, according to your spending behavior. Moreover, it can recommend products and provide unique insights on saving money. It bringsout the factors causing fluctuations in your expenses.