IBM – NALAIYA THIRAN PROJECT

PERSONAL EXPENSE TRACKER APPLICATION

INDUSTRY MENTOR: KUSBOO

FACULTY MENTOR: K.NAGARAJ

Submitted by

TEAM LEADER **: PRAVEEN RAJ R**

TEAM MEMBER: **PRAVEEN DAVID L**

TEAM MEMBER : **RAJANGAM R**

TEAM MEMBER : **RAGHUL NATH**

B.E. ELECTRONICS AND COMMUNICATION ENGINEERING

SETHU INSTITUTE OF TECHNOLOGY, PULLOOR

An Autonomous Institution Affiliated to Anna university, Chennai

PULLOOR, KARIAPATTI-626 115.

TABLE OF CONTENT

1.INTRODUCTION

1.1 Project overview

With the launch and increase in sales of smartphones over the last few years, people are using mobile applications to get their work done, which makes their lives easier. Mobile applications comprise various different categories such as Entertainment, Sports, Lifestyle, Education, Games, Food and Drink, Health and Fitness, Finance, etc.

This Expense Tracker application falls in the Finance Category and serves the important purpose of managing finances which is a very important part of one's life.

The software product went through the design, development, and the testing phase as a part of the Software Development Lifecycle.The application's interface is designed using custom art elements, the functionality is implemented using iOS SDK, and the phase of testing the product was accomplished successfully. The application is not much user intensive but just comprises of having them enter the expense amount, date, category, merchant and other optional attributes (taking picture of the receipts, entering notes about the expense, adding subcategories to the categories). With this entered information, the user is able to see the expense details daily, weekly, monthly, and yearly in figures, graphs, PDF format, and can print them as well if a printer is detected or scanned nearby. All these topics have been explained in detail in their respective chapters.

The aim of this project is to provide a solution for users on how to manage finances in any circumstance by keeping track of their expenses everyday. Ultimately, this contributes to societal well-being.

1.2 Purpose

The motivation to work in this project is actually our real-life experience. As a user We face many difficulties in our daily file. In our daily life money is the most important portion and without it we cannot last one day on earth but if we keep on track all financial data then we can overcome this problem. Most of the people cannot track their expenses and income one way they face the money crisis and depression. This situation motivates us to make an android app to track all financial activities. Using the Daily Expense Tracker user can be tracking expenses day to day and making life tension free.

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 and plan their expenses and savings efficiently. It helps you track all transactions like bills, refunds, payrolls, receipts, taxes, etc., on a daily, weekly, and monthly basis.

2. LITERATURE SURVEY

2.1 Existing Problem

This above study shows that the people prefer and are getting comfortable in managing their finances through a mobile app and evolve from the age old paper system. They have understood the benefits in leveraging digital tools to get better insights and a bigger look over their finances.

Parents are worried that their children lack financial literacy unlike their global counterparts. They are looking for applications where their children can learn about spending money. This financial illiteracy is prevalent even among elders.

Existing Solutions :

1.Money View - Expense Manager App

Money View App reads all of the transactional SMS messages and provides you with real-time visibility into your finances. This app unearths the hidden financial data that sits idly in SMS logs and makes excellent use of it.

Key features of the App:

- Check your bank account balances.
- See the most recent banking transactions.
- The Money View app automatically categorises your payments and displays major areas of spending.
- View weekly and monthly summaries to help you avoid overspending and improve the efficiency of your budget planning.
- It keeps track of your spending, sends personalised bill-paying reminders, finds relevant savings opportunities, and much more.
- Track your financial progress by looking at your spending trends over time.

2.Good budget - Budget & Finance App

This personal finance manager app acts as a proactive budget planner, assisting you in staying on top of your budget, bills, and finances. The personal finance app was designed for simple, real-time budget and financial tracking, making it one of the best expense tracker apps in India.

Key Features of the App:

- Data is backed up automatically and securely to Good budget's website.
- Disaggregate expense transactions

● Transactions that are scheduled and envelope fills

● Save time by using intelligent payee and category suggestions.

● Transfer funds between Envelopes and Accounts with ease.

● Match the budget period to the real-life situation.

● Analyse spending with the Spending by Envelope Report.

● Use the Income vs. Spending Report to keep track of your cash flow.

● Export transactions to CSV

● Carryover any unused funds to the next month as a reward for your incredible self-control.

● Plan your finances ahead of time to stay on track with your budget.

3.Real byte Money Manager App

You can use the budget planner and spending tracker to keep track of your personal and business financial transactions, review financial data on a daily/weekly/monthly basis, and manage your assets.

Key Features of the App:

● System of double-entry bookkeeping

● Management of budgets and expenses ● Management of credit and debit cards ● Get access to statistics immediately.

● Bookmarking feature

● Backup/restore function

4.Money - Budget Manager and Expense Tracker App

Money tracks the user's expenses and compares them to the monthly income and the budget planner. Money's money manager app keeps your monthly budget in top shape. As a result, it could also serve as the best expense tracker app.

Key Features of the App:

- With the intuitive and simple-to-use interface, you can quickly add new records.
- Maintain a multi-currency track.
- Backup and export personal finance data with a single click.
- Protect your data with passcode protection.
- View your spending distribution on a simple chart, or get detailed information from the records list.
- Use a budget tracker to save money.
- Use your own Google Drive or Dropbox account to safely synchronise.
- Take control of recurring payments.
- Create multiple accounts.
- Use the built-in calculator to crunch numbers.

5.Wallet - Money, Budget, Finance & Expense Tracker App

Wallet can automatically track your daily expenses by syncing your bank account, view weekly expense reports, plan your shopping expenses, and share specific features with your loved ones. You can manage your money with a wallet from anywhere and at any time.

Key Features of the App:

- Transactions are automatically and securely synced, then intelligently categorised and budgeted.
- Simple graphs and financial overviews provide actionable insights into the state of your finances, including accounts, credit and debit cards, debts, and cash.
- Arrange your bills and keep track of their due dates.
- Examine upcoming payments and how they will affect your cash flow.

- Selected accounts can be shared with family, friends, or co-workers who need to work together on a budget. Everyone is welcome to contribute from any platform, including Android, iPhone, and the Web.

- Other features include support for multiple currencies, automatic cloud sync, receipt and warranty tracking, categories and templates, geo-mapping transactions, hash-tagging, shopping lists, exports to CSV/XLS/PDF, debt management, PIN security, standing orders, notifications, reports, and more.

6.Walnut - All Indian Banks Money Manager App

Walnut automates and secures the tracking of your monthly expenses. You can stay within your budget, pay your bills on time, and save more money each month by using the Walnut app. They also provide personal loans.

Key Features of the App:

- Keep a close eye on your credit card balances.

- Use BHIM UPI to send money.
- Locate ATMs that accept cash near you in real-time.

- Export your information and create expense reports (in PDF & CSV format).

- Verify the balance of your bank account.

- Keep track of train, cab, movie, and event reservations, among other things.

- Search and share information about places you visit with friends and social networks.

- Report your bank, card, or any other interesting messages directly from the app.

2.2 References

1. https://moneyview.in/insights/best-personal-finance-management-a ppsin-india

2. https://www.factmr.com/report/personal-finance-mobile-app-market

3. https://www.moneytap.com/blog/best-money-management-apps/

4. https://relevant.software/blog/personal-finance-app-like-mint/

5. https://www.onmanorama.com/lifestyle/news/2022/01/11/financial-l iteracy-trend-among-todays-youth-investment.html

6. https://www.livemint.com/money/personal-finance/96-indian-parent sfeel-their-children-lack-financial-know-how-survey-1166133611085

5.html

7. https://economictimes.indiatimes.com/small-biz/money/importance-offinancial-literacy-amongst-youngsters/articleshow/85655134.cms

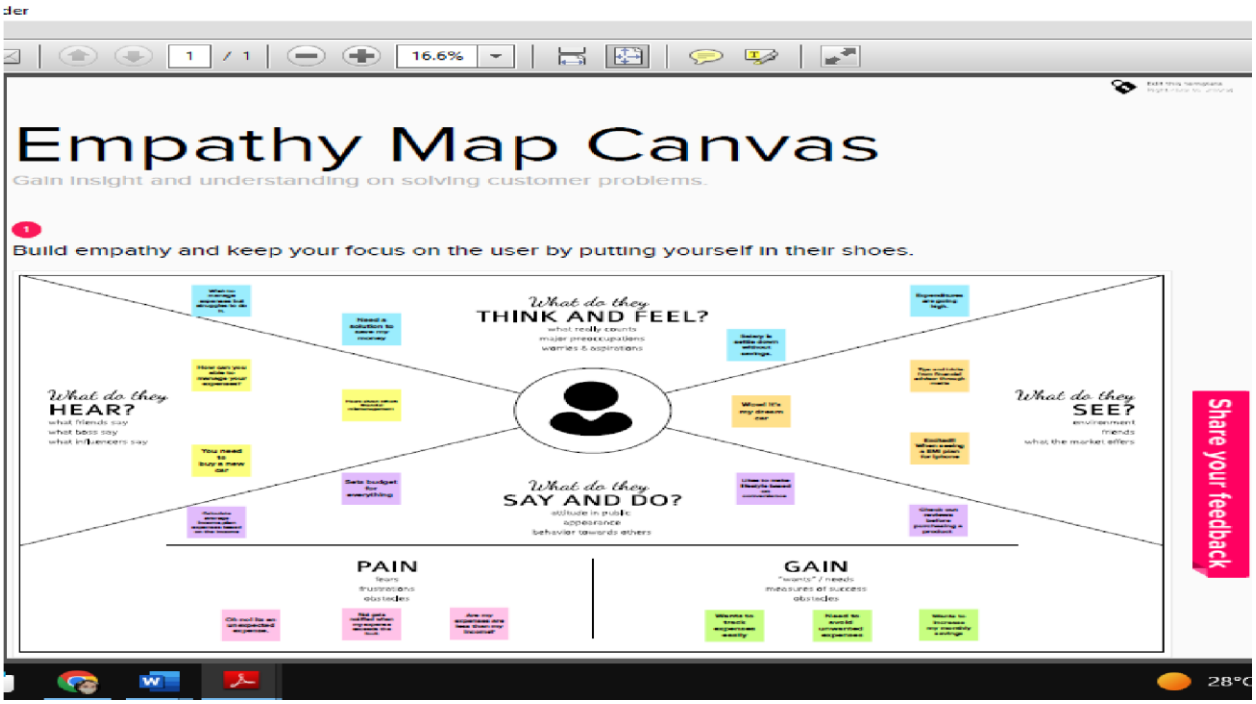
2.3 Problem Statement Definition

- Modern education does not focus on finance management. This is primarily due to lack of resources and the Indian value system on giving money to children. Failing to teach this valuable knowledge had left many Indians to recklessly spend their income and fall into vicious cycles of EMI and debt.

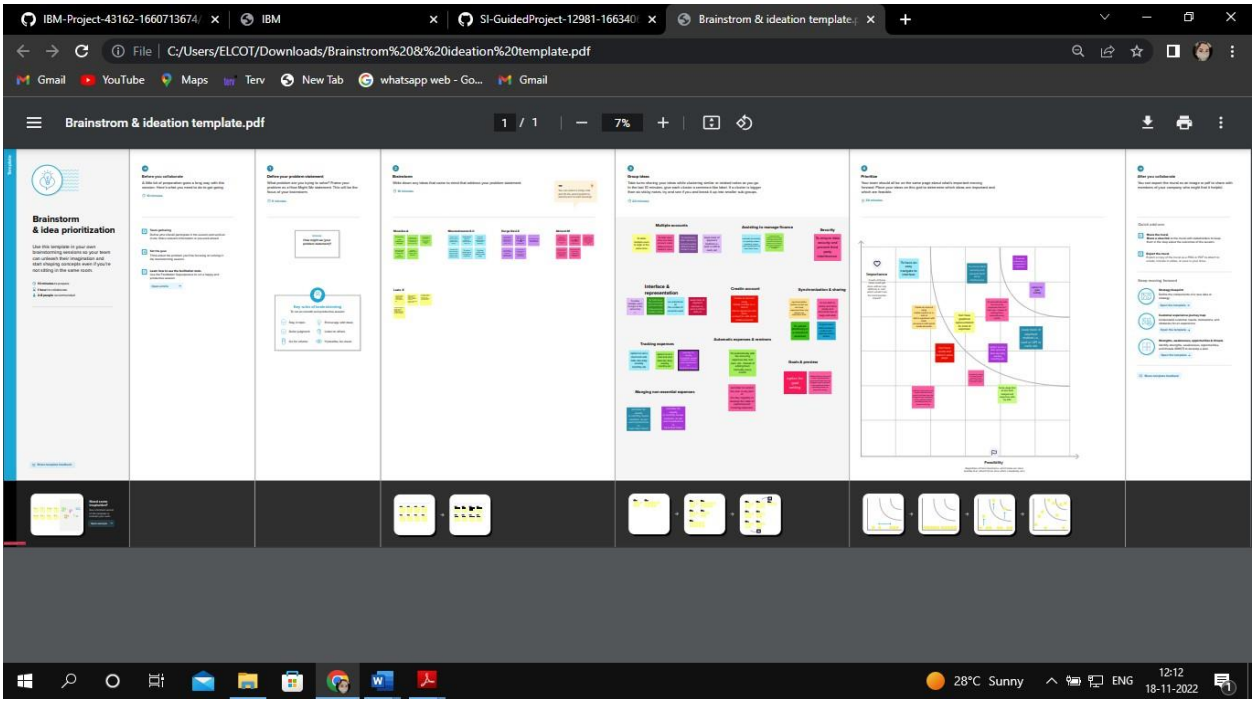
- Many of them are just a month's salary away from bankruptcy. This issue is tackled by providing a web application for where people can plan their monthly expenses into categories, set alerts and get visual insights from their spending patterns.

3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

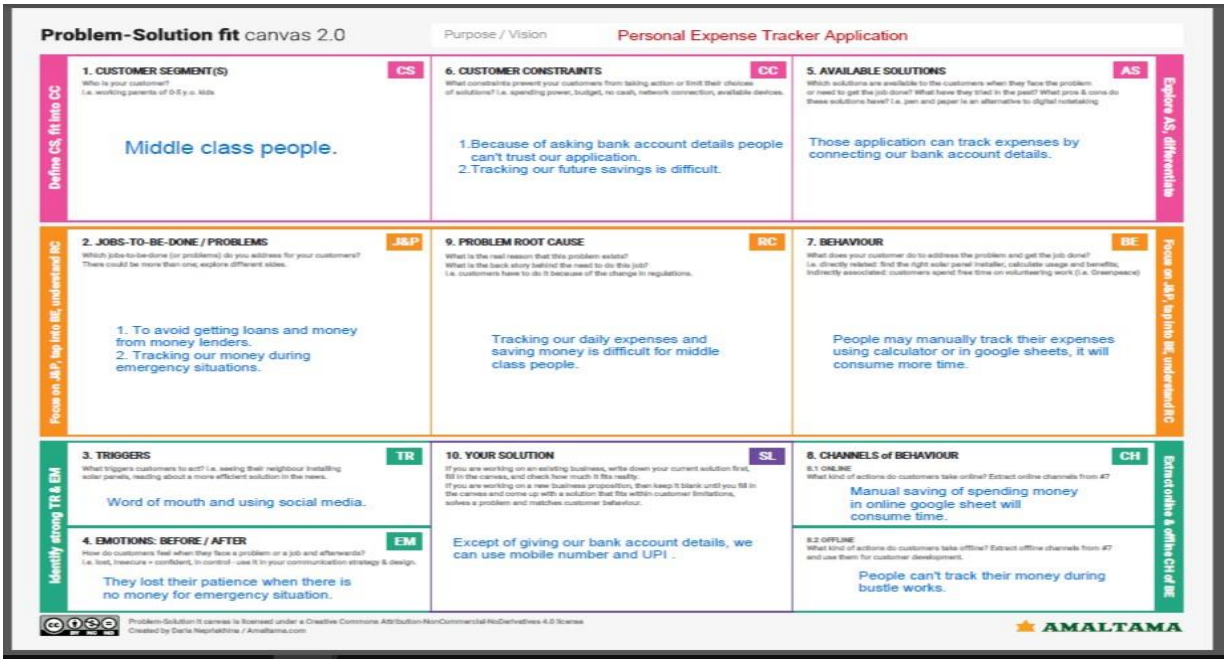


3.3 Proposed Solution

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

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	At the end of the month we start to have money crisis. Lack of proper planning of our income. Person has to keep a log in a diary or in a computer. All the calculations need to be done by the user. Overload to rely on the daily entry of the expenditure.
2.	Idea / Solution description	An expense tracker app allows you to monitor and categorize your expenses across different bank and investment accounts and credit cards. Some of these apps also offer budgeting tools, credit monitoring, mileage tracking, receipt keeping, and advice to grow your net worth.
3.	Novelty / Uniqueness	Expense tracker apps help you collect and classify your purchases so that you can identify areas that might be trimmed. Or, in the case of building net worth, places where you can allocate more money, such as savings. You might track expenses for a while just to get an idea of where your money's going, or it might be a stepping stone toward making and following a budget.
4.	Social Impact / Customer Satisfaction	Make wise decision Manage your expenses Budget planning can be done Makes report and give accurate survey.
5.	Business Model (Revenue Model)	Cost Effective one.
6.	Scalability of the Solution	Secure and safe to use Improves financial security Improves money management

3.1 Problem Solution fit



4. REQUIREMENT ANALYSIS

FunctionalRequirements:

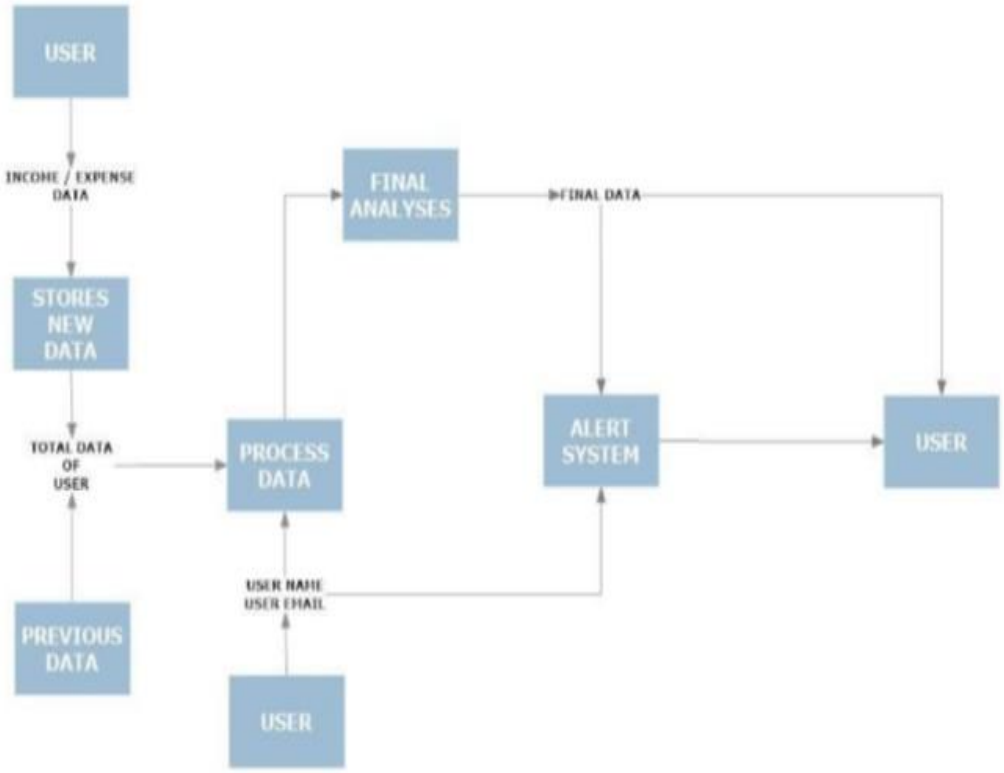
FRNo.	Functional Requirement(Epic)	SubRequirement(Story/Sub-Task)
FR-1	User Registration	Form for collecting details
FR-2	Login	Enter username and password
FR-3	Calendar	Personal expense tracker application must allow user to add the data to their expenses.
FR-4	Expense Tracker	This application should graphically represent the expense in the form of report.
FR-5	Report generation	Graphical representation of report must be generated.
FR-6	Category	This application shall allow users to add categories of their expenses.

Non-functional Requirements:

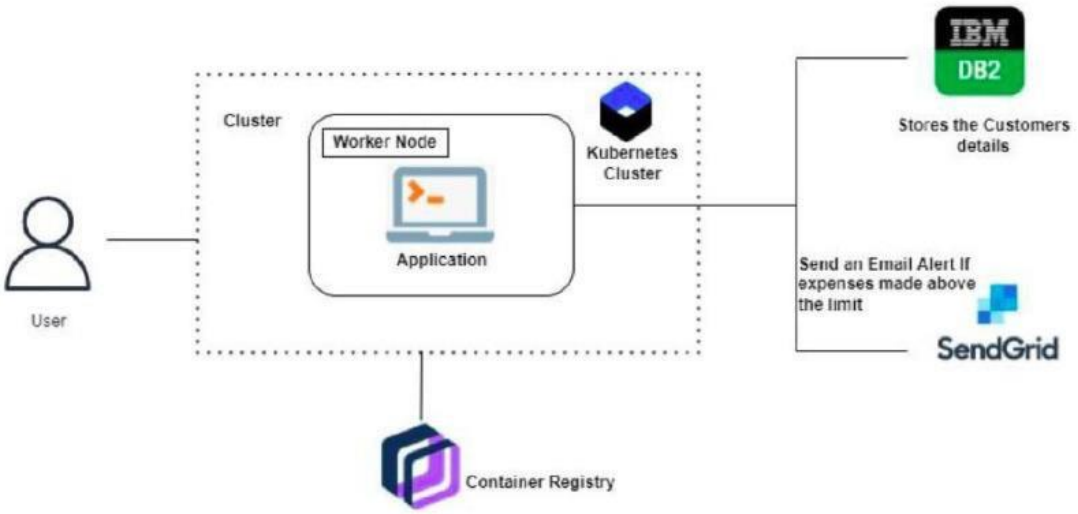
FRNo.	Non-Functional Requirement	Description
NFR-1	**Usability**	Helps to keep an accurate record of your income and expenses.
NFR-2	**Security**	Budget tracking apps are considered very safe from those who commit cyber crimes.
NFR-3	**Reliability**	Each data record is stored on a well built efficient database schema. There is no risk of data loss.
NFR-4	**Performance**	The types of expense are categories along with an option.Throughput of the system is increased due to light weight database support.
NFR-5	**Availability**	The application must have a 100% up-time.
NFR-6	**Scalability**	The ability to appropriately handle increasing demands.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories

Table-1: Components & Technologies:

S.No.	Component	Description	Technology
1.	User Interface	The user can Interact with the application with use of Chatbot	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	The application contains the sign in/sign up where the user will login into the main dashboard	Java / Python
3.	Application Logic-2	Dashboard contains the fields like Add income, Add Expenses, Save Money	IBM Watson STT service
4.	Application Logic-3	The user will get the expense report in the graph form and also get alerts if the expense limit exceeds	IBM Watson Assistant,SendGrid
5.	Database	The Income and Expense data are stored in the MySQL database	MySQL, NoSQL, etc.
6.	Cloud Database	With use of Database Service on Cloud, the User data are stored in a well secured Manner	IBM DB2, IBM Cloudant etc.
7.	File Storage	IBM Block Storage used to store the Financial data of the user	IBM Block Storage or Other Storage Service or Local Filesystem

Table-2: Application Characteristics:

S.No.	Characteristics	Description	Technology
1.	Open-Source Frameworks	Flask Framework in Python is used to implement this Application	Python-Flask
2.	Security Implementations	This Application Provides high security to the user Financial data. It can be done by using the Container Registry in IBM cloud	Container Registry, Kubernetes Cluster
3.	Scalable Architecture	Expense Tracker is a life time access supplication. It's demand will increase when the user's income are high	Container Registry, Kubernetes Cluster
4.	Availability	This application will be available to the user at any part of time	Container Registry, Kubernetes Cluster
5.	Performance	The performance will be high because there will be no network traffics in the application	Kubernetes Cluster

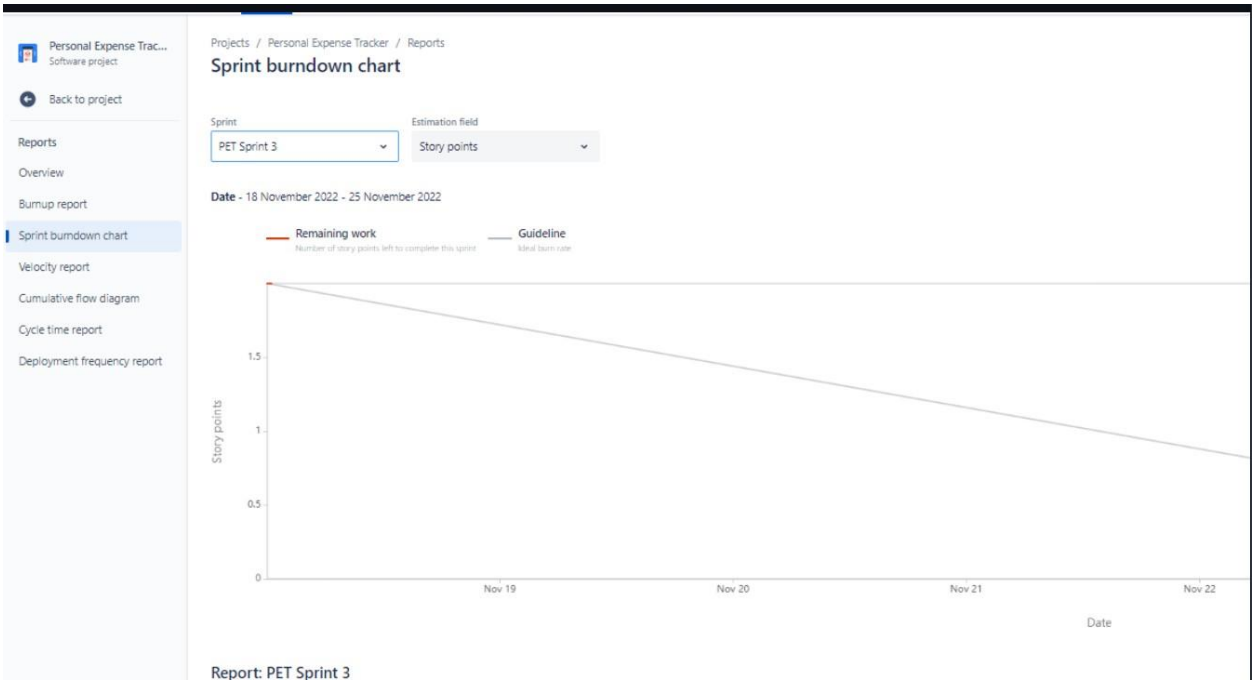
6. PROJECT PLANNING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points
Sprint - 1	Registration	USN -1	As a user , I can register for the application by entering my email , new password and confirming the same password.	2
		USN -2	As a user , I will receive confirmation email once I have registered for the application.	1
	Login	USN -3	As a user , I can log into the application by entering email and password / Google OAuth.	2
	Dashboard	USN -4	Logging in takes the user to their dashboard.	1
Sprint - 2		USN -5	As a user ,I will update my salary at the start of each month.	1
		USN -6	As a user , I will set a target/limit to keep track of my expenditure.	1
	Workspace	USN -7	Workplace for personal expense tracking	1
	Charts	USN -8	Graphs to show weekly and everyday expenditure	2
		USN -9	As a user , I can export raw data as csv file.	1

6.2 Reports from JIRA

Burnt Down Chart



7.CODING & SOLUTIONING

7.1 FEATURE 1

from flask import Flask, render_template, request, redirect, session
import re import sendgrid from flask_db2 import DB2 import ibm_db
import ibm_db_dbi import os

from sendemail import sendmail

app = Flask(__name__)

app.secret_key = 'a'

app.config['database'] = 'bludb'
app.config['hostname'] = '21fecfd8-47b7-4937-840d-

```
d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'
app.config['port'] = '31864' app.config['protocol'] = 'tcpip' app.config['uid']
= 'qlw76991' app.config['pwd'] = 'kjIvjWN4s8YibDmp'
app.config['security'] = 'SSL' try:
   mysql = DB2(app)

   conn_str='database=bludb;hostname=21fecfd8-47b7-4937-
840dd791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=3
1864;protocol=tcpip;\
         uid=qlw76991;pwd=kjIvjWN4s8YibDmp;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())

#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():
   if request.method == "POST":
```

```
        user_name = request.form['username']
        email = request.form['email']
        pass_word = request.form['password']
        query = "INSERT INTO Admin (username,email,password) values
(?,?,?)"
        insert_stmt = ibm_db.prepare(ibm_db_conn, query)
        ibm_db.bind_param(insert_stmt, 1, user_name)
        ibm_db.bind_param(insert_stmt, 2, email)
        ibm_db.bind_param(insert_stmt, 3, pass_word)
        ibm_db.execute(insert_stmt)
        msg = 'Account Created Successfully'
        return render_template("signup.html", msg=msg)

@app.route("/signin",methods=['post','get']) def
signin():
    if request.method=="post":
        return render_template("login.html")
    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 Admin WHERE username = ? = ?"
        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 Admin 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@gmail.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)

@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']

    print(date)
    p1 = date[0:10]
    p2 = date[11:13]
    p3 = date[14:]
    p4 = p1 + "-" + p2 + "." + p3 + ".00"
    print(p4)
    sql = "INSERT INTO Expense (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 Expense WHERE 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["USERID"])
      temp.append(dictionary["DATE"])
      temp.append(dictionary["EXPENSENAME"])
      temp.append(dictionary["AMOUNT"])
      temp.append(dictionary["PAYMODE"])
      temp.append(dictionary["CATEGORY"])
      expense.append(temp)
      print(temp)
      dictionary = ibm_db.fetch_assoc(res)

   total=0
   for x in expense:
      total += int(x[3])


   param = "SELECT userid, limit FROM limit WHERE userid = " + str(session['id'])
   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["LIMIT"])
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[len(temp)-1]

    if total > 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'])

    return redirect("/display")

    #DISPLAY---graph

@app.route("/display")
def display():
    print(session["username"],session['id'])
    param = "SELECT * FROM Expense 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["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
```

```
        temp.append(dictionary["CATEGORY"])
        expense.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

    return render_template('display.html' ,expense = expense)
@app.route('/delete/<string:id>', methods = ['POST', 'GET' ]) def
delete(id):
    param = "DELETE FROM Expense WHERE  userid = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)

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

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

      sql = "UPDATE Expense SET date = ? , expensename = ? , amount = ?, paymode
= ?, category = ? WHERE userid = ?"
      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 limit (userid, limit) 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 userid,limit FROM limit WHERE userid = " + str(session['id'])
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = "/-"
    while dictionary != False:
        temp = []
        temp.append(dictionary["LIMIT"])
        print(temp)
        row.append(temp)
        dictionary = ibm_db.fetch_assoc(res)
        s = temp[len(temp)-1]
    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 Expense 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["AMOUNT"])
       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 Expense 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["USERID"])
       temp.append(dictionary["DATE"])
```

```
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        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 += int(x[3])
        if x[5] == "food":
            t_food += int(x[3])

        elif x[5] == "entertainment":
            t_entertainment  += 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("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
    while dictionary1 != False:
       temp = []
       temp.append(dictionary1["DT"])
       temp.append(dictionary1["TOT"])
       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 Expense 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["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        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 += int(x[3])
        if x[5] == "food":
            t_food += int(x[3])

        elif x[5] == "entertainment":
            t_entertainment  += int(x[3])

        elif x[5] == "business":
            t_business  += int(x[3])
        elif x[5] == "rent":
            t_rent  += int(x[3])
        elif x[5] == "EMI":
            t_EMI  += int(x[3])
        elif x[5] == "other":
            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("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
Expense 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["TOT"])
        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 Expense 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["USERID"])
            temp.append(dictionary["DATE"])
            temp.append(dictionary["EXPENSENAME"])
            temp.append(dictionary["AMOUNT"])
            temp.append(dictionary["PAYMODE"])
            temp.append(dictionary["CATEGORY"])
            expense.append(temp)
            print(temp)
            dictionary = ibm_db.fetch_assoc(res)
        for x in expense:
            total += int(x[3])
            if x[5] == "food":
                t_food += int(x[3])

            elif x[5] == "entertainment":
                t_entertainment  += int(x[3])

            elif x[5] == "business":
                t_business  += int(x[3])
            elif x[5] == "rent":
```

```
            t_rent  += int(x[3])
        elif x[5] == "EMI":
            t_EMI  += int(x[3])

        elif x[5] == "other":
            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("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')


app.run(debug=True)
```

home.html

```
<!DOCTYPE html>
<html lang="en">

<head>

 <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>udget <span>T</span>racker</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 App <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>Budget Tracker provides a many services  to the customer and industries. Financial solutions to meet your needs whatever your money goals,there is a Budget solution to help you reach them </p>
</div>
<div class="service-bottom">
<div class="service-item">  <div class="icon"><img src="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>Budget Tracker 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 Budget Tracker application you can view your all the financial activities
</p>
</div>
</div>
```

```
  </div>
 </section>
 <!-- End Service 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>+91 234 123 1234</h2>
      <h2>+91 234 123 1234</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>info@gmail.com</h2>
      <h2>abcd@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>4th main-road,Bengaluru,Karnataka,India</h2>
        </div>
      </div>
    </div>
  </div>
 </section>
 <!-- End Contact Section -->

 <!-- Footer -->
 <section id="footer">
  <div class="footer container">
    <div class="brand">
      <h1><span>B</span>udget <span>T</span>racker</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
```

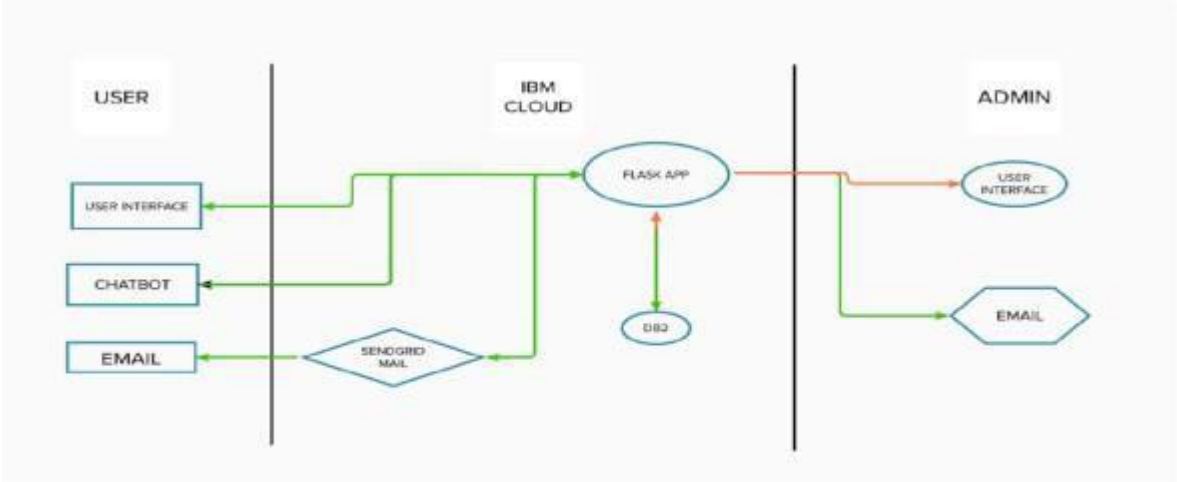
```
src="https://img.icons8.com/bubbles/100/000000/behance.png" /></a>
</div>
    </div>
    <p>Copyright © 2019-2023 GCE CSE girls . All rights reserved</p>    </div>
 </section>
 <!-- End Footer -->
 <script src="..\static\js\home.js"></script>
</body>

</html>
```

8 TESTING

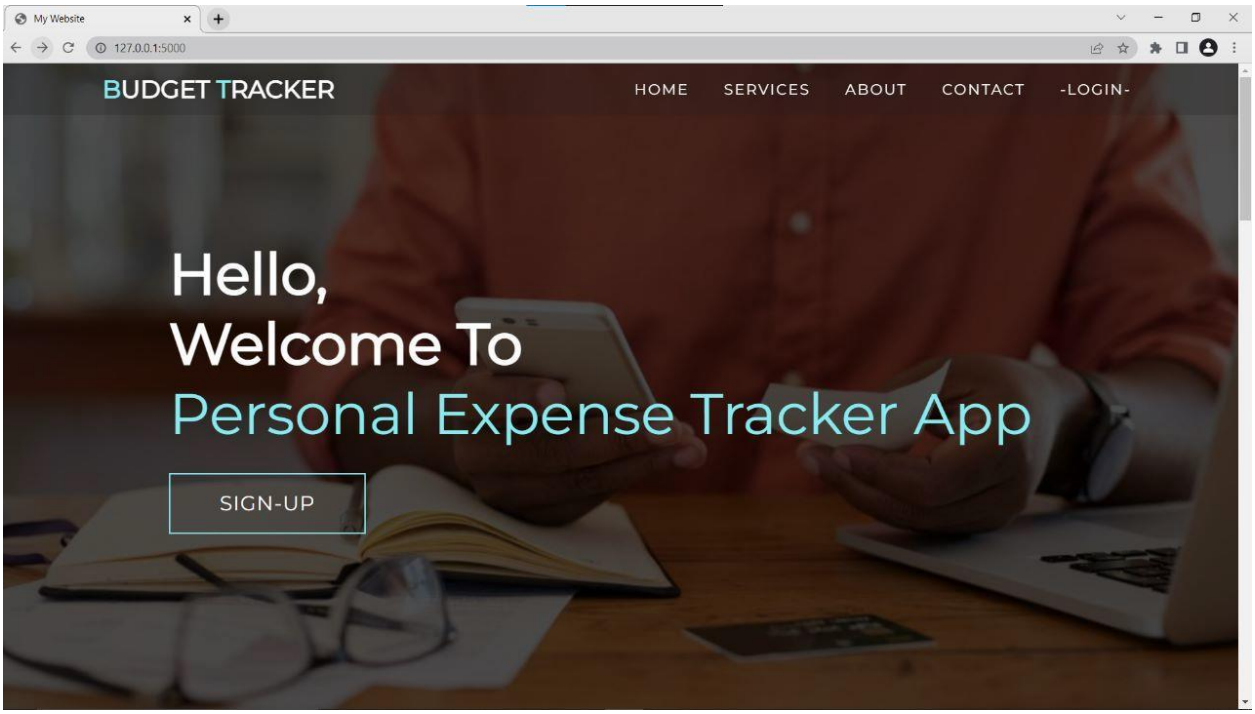
Black Box testing is the method that does not consider the internal structure, design, and product implementation to be tested. In other words, the tester does not know its internal functioning. The Black Box only evaluates the external behavior of the system. The inputs received by the system and the outputs of responses it produces are tested.

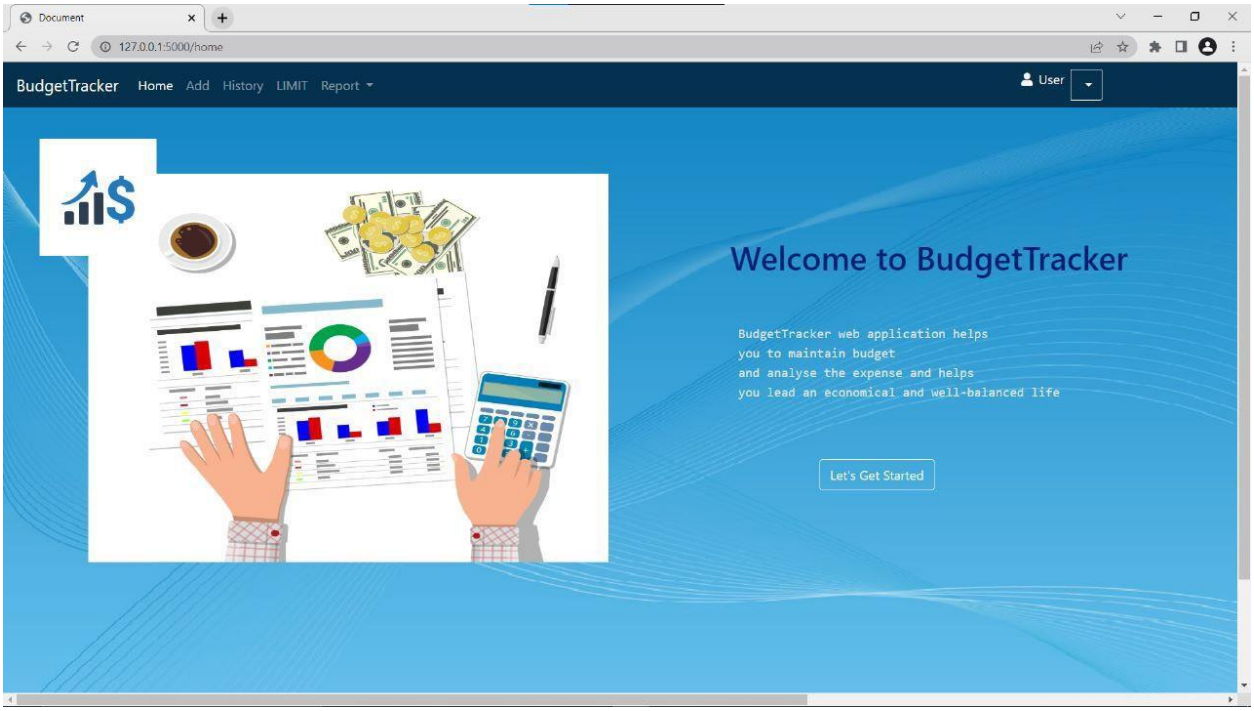
The White Box Test method is the one that looks at the code and stíuctuíe of the product to be tested and uses that knowledge to perform the tests. This method is used in the Unit Testing phase, although it can also occur in other stages such as Integration Tests. For the execution of this method, the tester or the person who will use this method must have extensive knowledge of the technology used to develop the program.

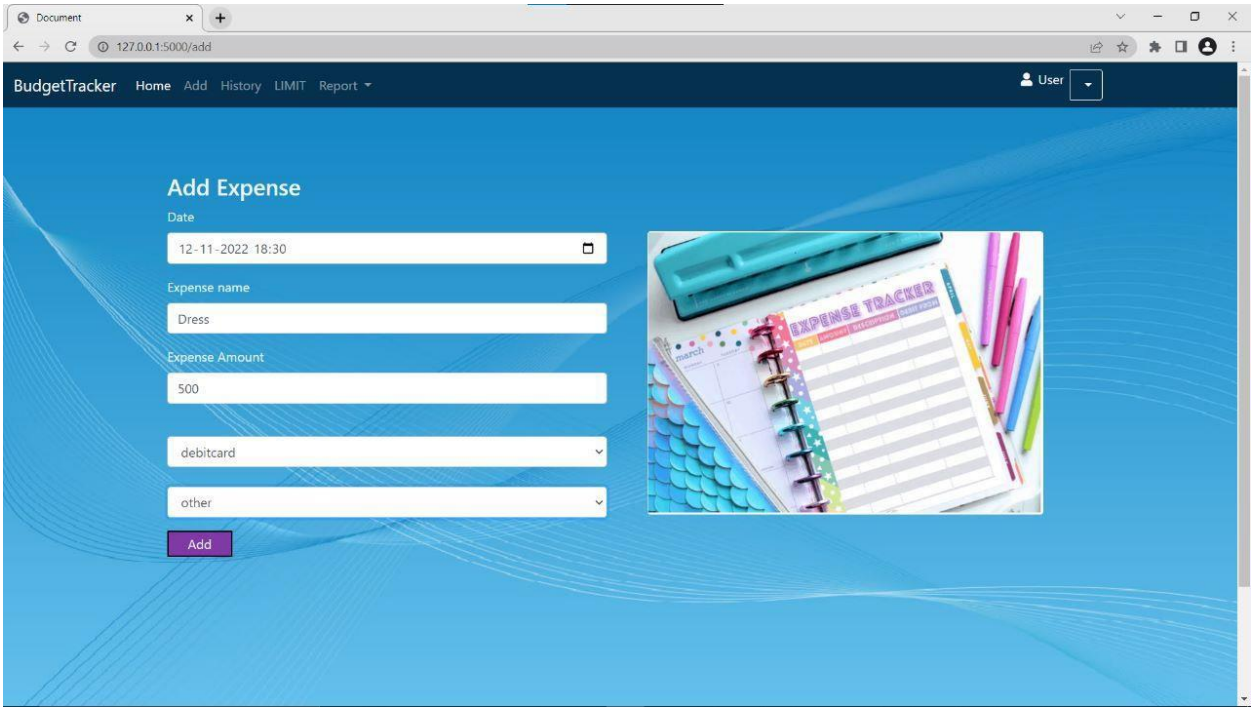
8.1 Test Cases

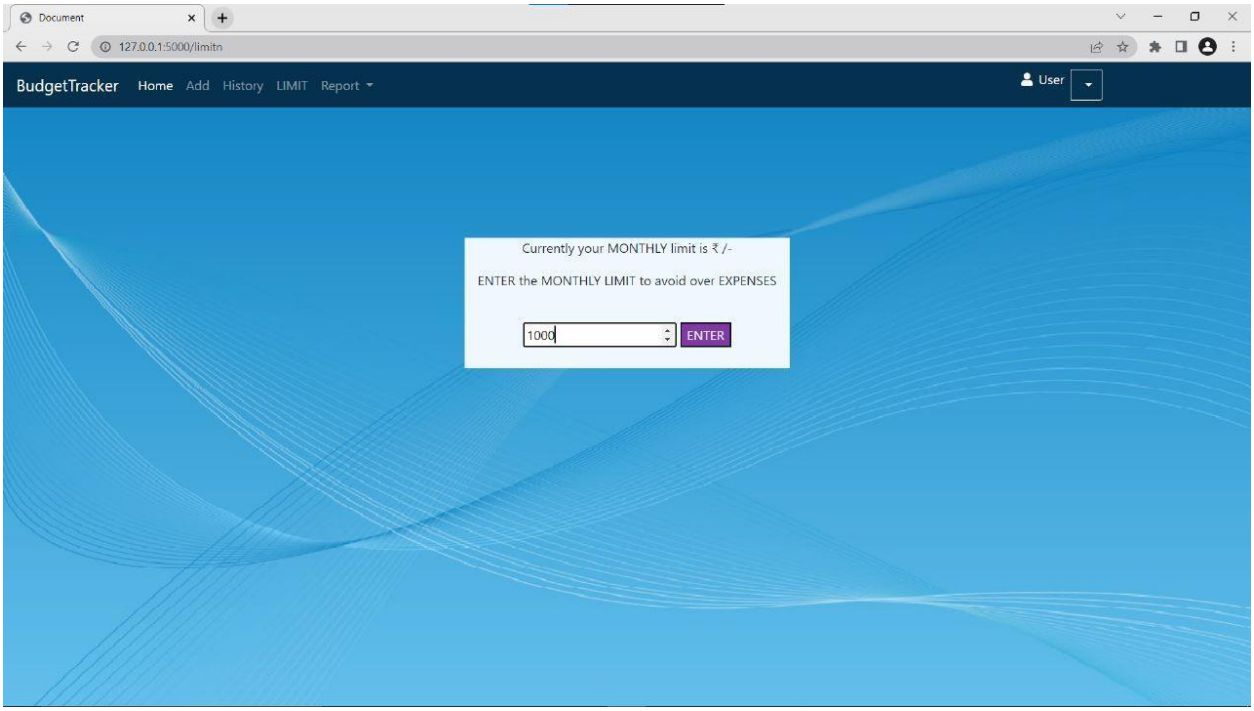


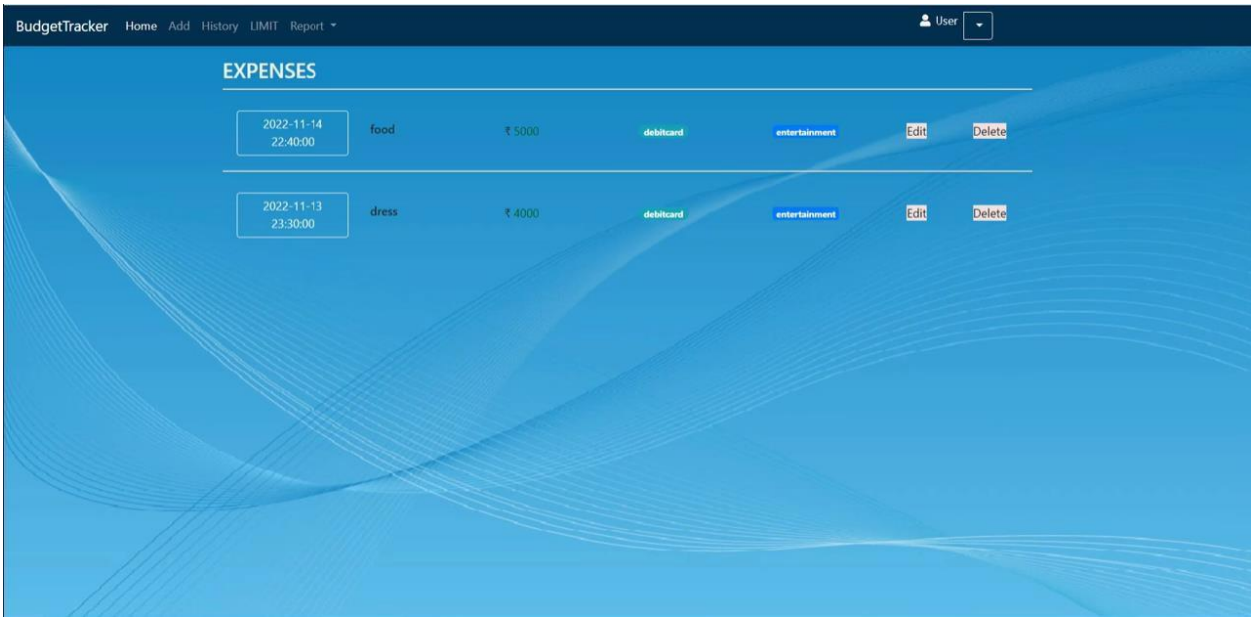
9 RESULT

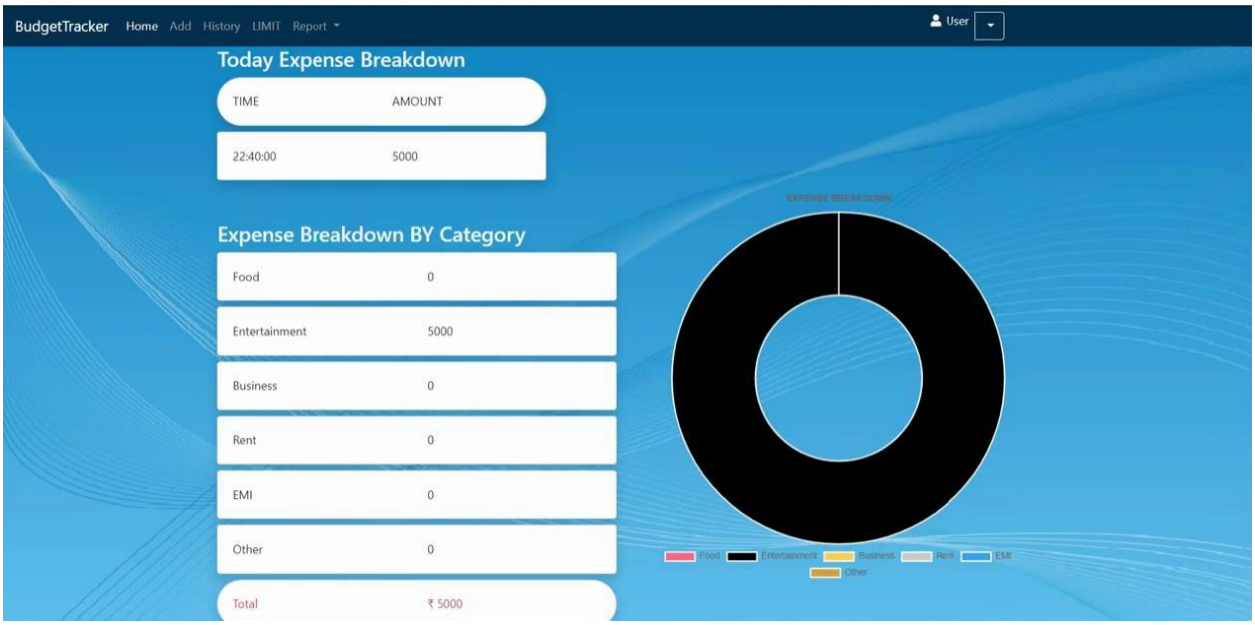



Document
127.0.0.1:5000/home
BudgetTracker
Home
Add
History
LIMIT
Report
User
Welcome to BudgetTracker
BudgetTracker web application helps
you to maintain budget
and analyse the expense and helps
you lead an economical and well-balanced life
Let's Get Started


Document
127.0.0.1:5000/add
BudgetTracker
Home
Add
History
LIMIT
Report
User
Add Expense
Date
12-11-2022 18:30
Expense name
Dress
Expense Amount
500
debitcard
other
Add
EXPENSE TRACKER


Document
127.0.0.1:5000/limitn
BudgetTracker
Home
Add
History
LIMIT
Report
User
Currently your MONTHLY limit is ₹ /-
ENTER the MONTHLY LIMIT to avoid over EXPENSES
1000
ENTER


BudgetTracker
Home
Add
History
LIMIT
Report
User
EXPENSES
2022-11-14 22:40:00
food
₹ 5000
debitcard
entertainment
Edit
Delete
2022-11-13 23:30:00
dress
₹ 4000
debitcard
entertainment
Edit
Delete



9.1Performance Metrics

The performance of a recommendation algorithm is evaluated by using some specific metrics that indicate the accuracy of the system. The type of metric used depends on the type of filtering technique. Root Mean Square Error (RMSE), Receiver Operating Characteristics (ROC), Area Under Cover (AUC), Precision, Recall and F1 score is generally used to evaluate the performance or accuracy of the recommendation algorithms.

Root-mean square error (RMSE). RMSE is widely used in evaluating and comparing the performance of a recommendation system model compared to other models. A lower RMSE value indicates higher performance by the recommendation model. RMSE, as mentioned by [69], can be as represented as follows:

$$RMSE=\sqrt{1/Np\Sigma u,i(pui-rui)^2} \quad (1)$$

where, Np is the total number of predictions, pui is the predicted rating that a user u will select an item i and rui is the real rating.

Precision. Precision can be defined as the fraction of correct recommendations or predictions (known as True Positive) to the total number of recommendations provided, which can be as represented as follows:

Precision=True Positive (TP)/True Positive(TP)+False Positive (FP)(2) It is also defined as the ratio of the number of relevant recommended items to the number of recommended items expressed as percentages.

Recall. Recall can be defined as the fraction of correct recommendations or predictions (known as True Positive) to the total number of correct relevant recommendations provided, which can be as represented as follows:

Recall=True Positive (TP)/True Positive(TP)+False Negative (FN) (3) It is also defined as the ratio of the number of relevant recommended items to the total number of relevant items expressed as percentages.

F1 Score. F1 score is an indicator of the accuracy of the model and Page no: 40 ranges from 0 to 1, where a value close to 1 represents higher recommendation or prediction accuracy. It represents precision and recall as a single metric and can be as represented as follows:

F1 score=2×Precision∗Recall/Precision+Recall (4)

Coverage. Coverage is used to measure the percentage of items which are recommended by the algorithm among all of the items.

Accuracy. Accuracy can be defined as the ratio of the number of total correct recommendations to the total recommendations provided, which can be as represented as follows:

Accuracy=TP+FN/TP+FN+TN+FP (5)

Intersection over union (IoU). It represents the accuracy of an object detector used on a specific dataset [70].

IoU=TP/TP+FN+FP (6)

ROC. ROC curve is used to conduct a comprehensive assessment of the algorithm's performance [57].

AUC. AUC measures the performance of recommendation and its baselines as well as the quality of the ranking based on pairwise comparisons [5].

Rank aware top-N metrics. The rank aware top-N recommendation metric finds some of the interesting and unknown items that are presumed to be most attractive to a user [71]. Mean reciprocal rank (MRR), mean average precision (MAP) and normalized discounted cumulative gain (NDCG) are three most popular rank aware metrics.

MRR. MRR is calculated as a mean of the reciprocal of the position or rank of first relevant recommendation [72,73]. MRR as mentioned by [72,73] can be expressed as follows:

MRR=1/Nu Σu∈Nu 1/Lnu[k]∈Ru (7) where u, Nu and Ru indicate specific user, total number of users and the set of items rated by the user, respectively. L indicates list of ranking length (n) for user (u) and k represents the position of the item found in the he lists L. MAP: MAP is calculated by determining the mean of average precision at the points where relevant products or items are found. MAP as mentioned by [73] can be expressed as follows.

MAP=1Nu|Ru|Σnk=1k(Lnu[k]∈Ru)Pu@k (8) Page no:
41

where Pu represents precision in selecting relevant item for the user.

NDCG: NDCG is calculated by determining the graded relevance and positional information of the recommended items, which can be expressed as follows [73].

NDCGu=Σnk=1G(u,n,k)D(k)/Σnk=1G∗(u,n,k)D(k) (9) where D (k) is a discounting function, G (u, n, k) is the gain obtained recommending an item found at k-th position from the list L and G* (u, n, k) is the gain related to k-th item in the ideal ranking of n size for u user.

10 ADVANTAGES & EXISTING SYSTEM

Advantages

1. The best organizations have a way of tracking and handling these reimbursements.This ideal practice guarantees that the expenses tracked are accurately and in a timely manner. From a company perspective,timely settlements of these expenses when tracked well will certainly boost employees' morale
2. Financially Aware and Improve Money Management tracking your expenditures ensures you achieve your project financial targets.How is that? By clearly understanding your project spending using project budget limits, you can aptly make the necessary changes to complete your project within time and budget.
3. Effective expense tracking and reporting to avoid conflict. As a project manager or business owner, you can set clear policies for the expense types and

reimbursement limits to avoid misunderstandings are about costs. Tracking the project expenses by asking team members to provide receipts is helpful to avoid conflict and maintain compliance also. An excellent reporting mechanism is extremely helpful to support the amount to be reimbursed to your team and also invoicing to your customer.

4. Helps anticipate the costs of similar projects When you formally track and report expenses, you have a permanent documentation which helps you correctly anticipate expenses for similar projects in the future. This is even more significant when it comes to budget-making process.

5. Tracking the amount of money spent on the projects is important to invoice customers and determine the cost & profitability analysis when your company is providing services to another company. On the other hand, expense tracking or internal project is important for cost and ROI calculation. Understanding how this money is being utilized across the project is such a significant issue. The consequence for not properly tracking and reporting project expenses may lead to a budgetary issues.

EXISTING SYSTEM

A) How it Actually Works

In existing, we need to maintain the Excel sheets, CSV etc. files for the user daily and monthly expenses. In existing,

there is no as such complete solution to keep a track of its daily expenditure easily. To do so a personas to keep a log in a diary or in a computer, also all the calculations needs to be done by the user which may sometimes results in errors leading to losses.

B) Drawbacks of the Existing System

There can be many disadvantages of using a manual accounting system. Accounting, for any business, can be a

complex undertaking. A manual accounting system requires you to understand the accounting process in a way that may be unnecessary with a computerized accounting system. This can be an advantage or a disadvantage, depending on the person doing the bookkeeping; often, a specially trained professional is needed to

ensure that accounting is done properly. Unraveling the complexity of your financial records by hand may be time consuming. Since it takes time to generate reports.

11 CONCLUSION & FUTURE SCOPE

11.1 CONCLUSION

In conclusion, developing a personal budget and tracking all expenses and spending is acrucial aspect of personal finances. Set aside a fixed amount in a savings account, they say youshould always have three months work of your living expenses in a savings account in case ofemergencies.

11.2 FUTURE SCOPE

The main objective of this project is support to the user to sustain all financial activitieslike digital automated dairy. This application helps the user to avoid unexpectedexpenses and bad financial situations.

➢Using this application, users can manage all financial data and track all expenseand income category wise.

➢Creating a category and recording all expenses and income under the category.

➢Enable the notification system user get notification daily at a specific time thatcan help the user insert expense and income.

➢Backup and Restore all information.

➢Report are generated in PDF format in category wise or time period.

12 APPENDIX

FRS can be defined as a means of feature matching between fashion products and users or consumers under specific matching criteria. Different research addressed apparel attributes such as the formulation of colors, clothing shapes, outfit or styles, patterns or prints and fabric structures or textures. Guan et al. studied these features using image recognition, product attribute extraction and feature encoding. Researchers have also considered user features such as facial features,

body shapes, personal choice or preference, locations and wearing occasions in predicting users' fashion interests. A well-defined user profile can differentiate a more personalized or customized recommendation system from a conventional system. Various research projects on apparel recommendation systems with personalized styling

guideline and intelligent recommendation engines have been conducted based on similarity recommendation and expert advisor recommendation systems.