IBM – NALAIYA THIRAN PROJECT

PERSONAL EXPENSE TRACKER APPLICATION

INDUSTRY MENTOR : KUSHBOO

FACULTY MENTOR : SUGANYA A

TEAM ID : PNT2022TMID44963

TEAM MEMBERS :

TEAM LEAD : KALAIYARASAN V – 811219104007

TEAM MEMBER : SARANYA P – 811219104020

TEAM MEMBER : RAVINIYA K – 811219104018

TEAM MEMBER : NANCY N - 811219104014

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 wellbeing.

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 :

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.

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, realtime 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.

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

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.

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 coworkers 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.

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-financemanagement-a pps-in-india
2. https://www.factmr.com/report/personal-finance-mobile-appmarke t
3. https://www.moneytap.com/blog/best-money-managementapps/
4. https://relevant.software/blog/personal-finance-app-like-mint/

2.3 Problem Statement Definition

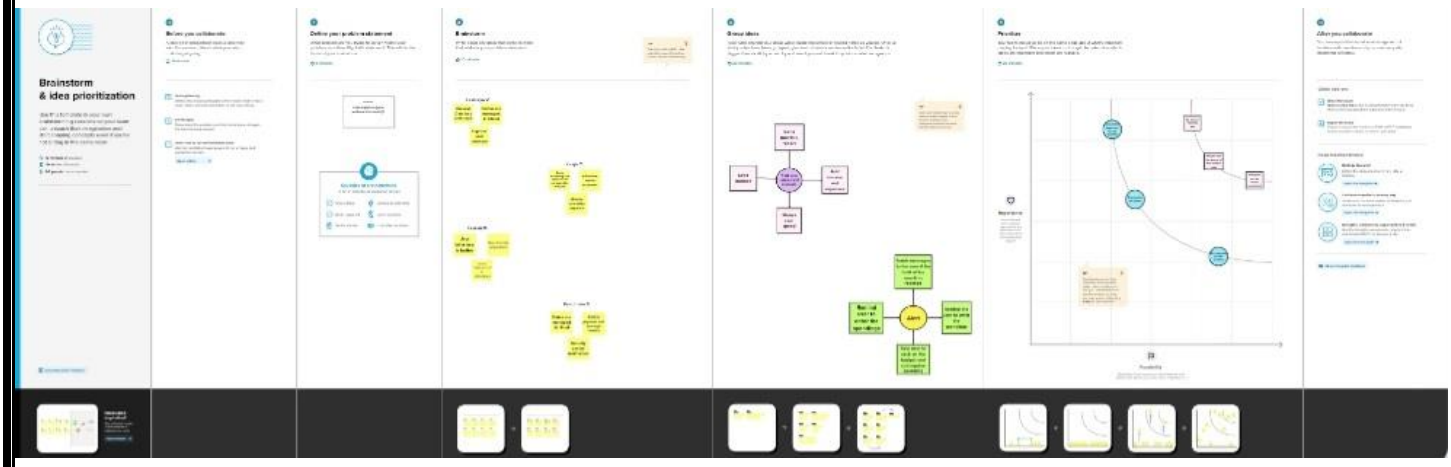
- Modern education does not focus on finance management. This is primarily due to lack of resources and the Indian value system on giving money to children. Failing to teach this valuable knowledge had left many Indians to recklessly spend their income and fall into vicious cycles of EMI and debt.

- 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

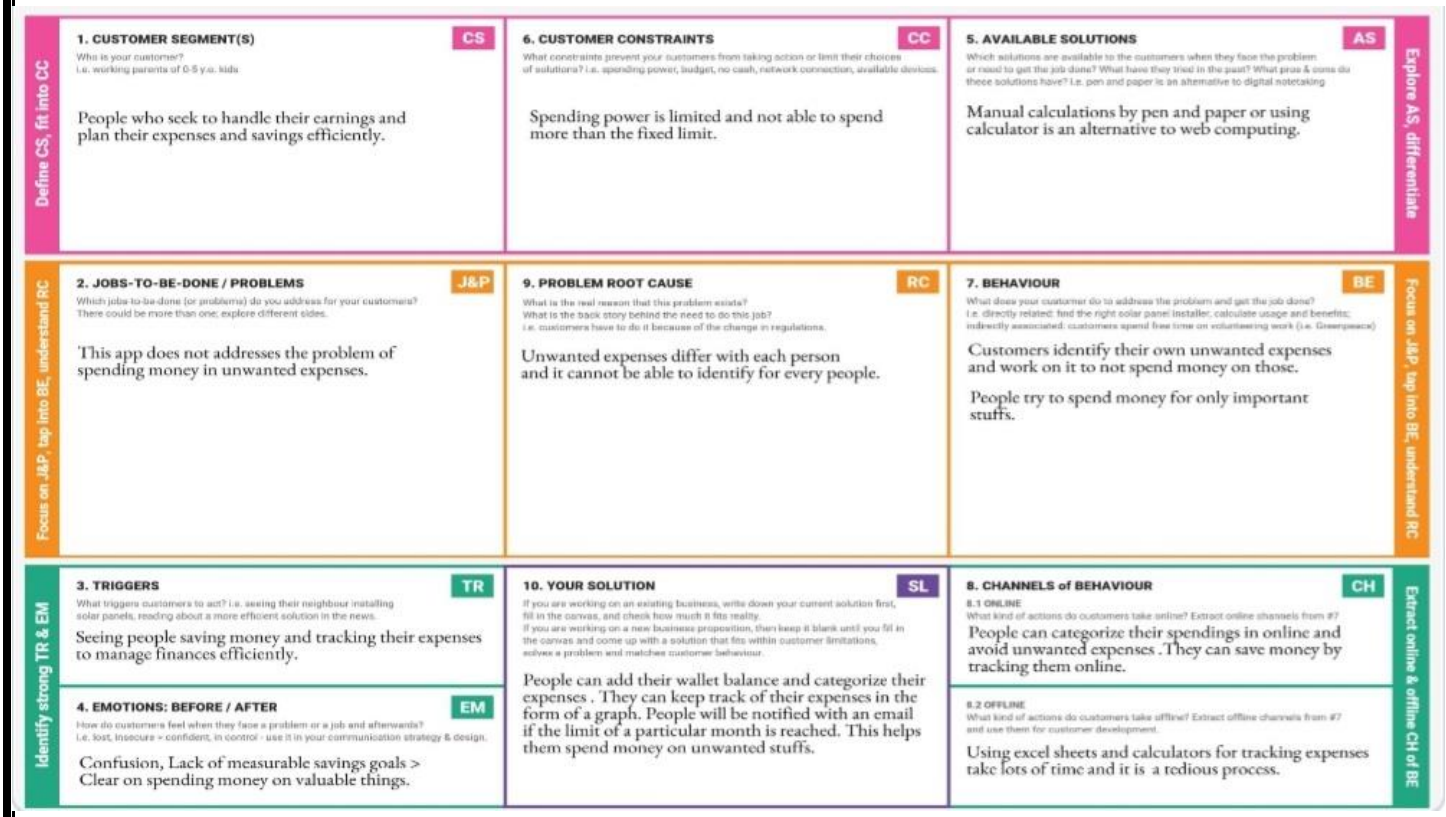


3.3 Proposed Solution

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

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	In today's financial world, money management is the necessary part of life. The proper balance between income and expense is a must for a comfortable livelihood. But in the absence of proper management of money, we left with no savings at all. Generally paper based expense tracking system is used to track the expenses. But the problem is that data might get lost due to flood, accidents etc..
2.	Idea / Solution description	Personal expense tracker app allow users to track their daily expenses so that they do not exceed their budget. This app will not only keep a check on user's expenses but also cut down the unnecessary expenses. This app helps to maintain the financial stability of users.
3.	Novelty / Uniqueness	It displays the expenditure on certain categories in a graphical format. It keeps track of the user's savings and expenditure and notifies to limit the expenditure. It saves time.
4.	Social Impact / Customer Satisfaction	It is used to reduce the unnecessary expenses made by the user. The user will be able to prioritize the spending. It lets you to regulate spending impulses and eliminate worthless spending, thereby avoiding debt. At every point, you will be aware about how much money you are left with.
5.	Business Model (Revenue Model)	It is a good business model. The user can be suggested the right product to buy based on their budget. It provides finance related advertisements. This app is provided for free of cost.
6.	Scalability of the Solution	It provides backup and recovery of data. It can handle large number of users with high performance and data security. Users can track this application anywhere and anytime. This app provides better user interface. Multiple users can access this application and it reduces human errors.

3.1 Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login through valid username and password
FR-4	Dashboard	User can add the expense and evaluate them by using the provided options. It displays the summary of expenses.
FR-5	Budget Limit	User can be able to set the budget limit for a week or a month. By setting the budget limit, the user can increase the savings by reducing the unwanted expenses.
FR-6	Expense Tracker	It helps to track the expenses by setting the budget limit. It makes the user to categorize the expenses.
FR-7	Report generation	It displays the graphical representation of users expense in the form of pie-chart ,bar graph etc..
FR-8	Alert	Alert the user through mail when the expense exceeds the budget. It gives the notification of every transactions.

4.2 Non-Functional requirements

Non-functional Requirements:

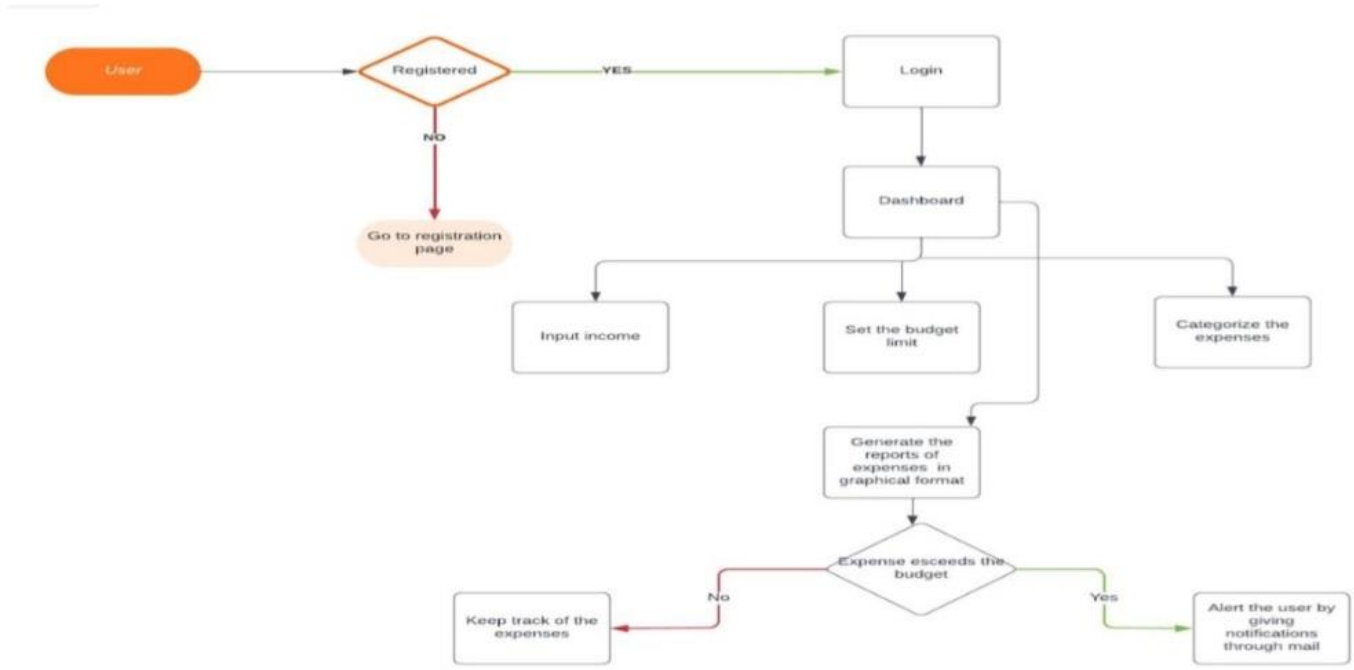
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	It helps the users to maintain the accurate track of expenses and savings. It is user friendly. It helps the users to cut down the unwanted expenses.
NFR-2	**Security**	Customers set their own account in this app using their email which is secured by a password. It might prevent from cyber crime activities.
NFR-3	**Reliability**	Each data record is stored in a effective database scheme with high security. There is no possibility of data loss.
NFR-4	**Performance**	It gives users the option to add or delete the categories of expenses and to track the money flow. The performance of this application is considered to be very good because of its lightweight database design.
NFR-5	**Availability**	This application is available for all the time. Users can access this app from any where and any time. It provides portability for the users.
NFR-6	**Scalability**	It has the ability to handle large number of users with high performance and data security.

5. PROJECT DESIGN

5.1 Data Flow Diagrams

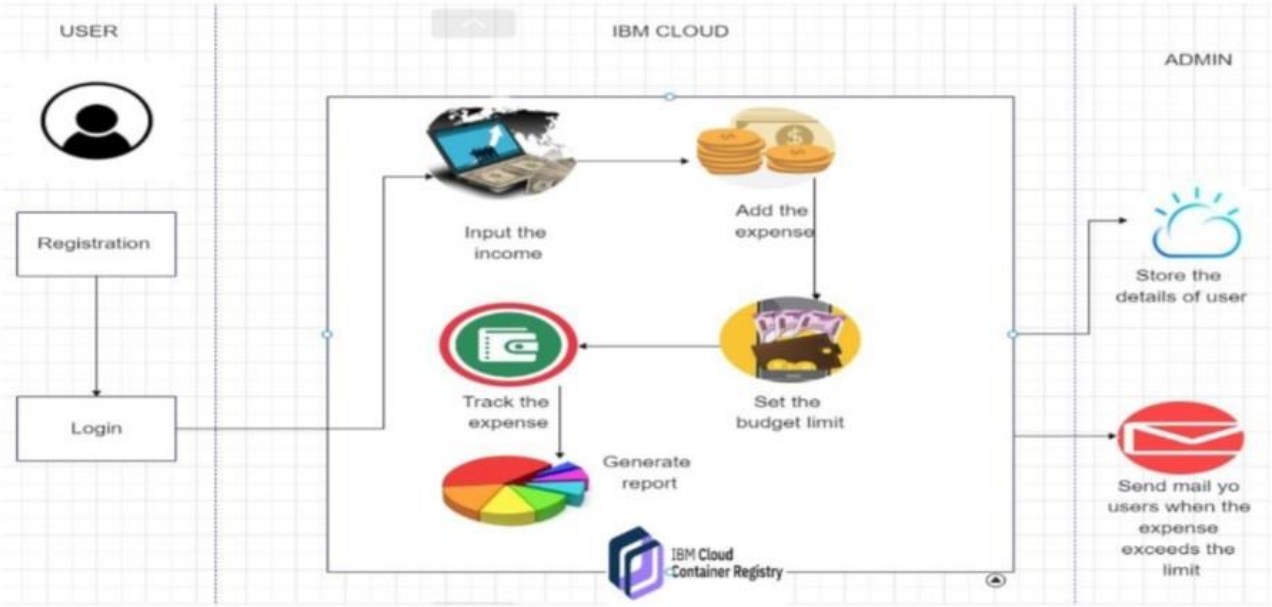
Data Flow Diagrams:



5.2 Solution & Technical Architecture

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.
	Login	USN-2	As a user, I can log into the application by entering email & password
	Add	USN -3	As a user , I can add in new expenses.
	Remove	USN – 4	As a user , I can remove previously added expenses.
	View	USN - 5	As a user , I can view my expenses in the form of graphs and get insights.
	Get alert message	USN - 6	As a user , I will get alert messages if I exceed my target amount.
Administrator	Add / remove user	USN – 7	As admin , I can add or remove user details on db2 manually.
		USN - 8	As admin , I can add or remove user details on sendgrid.

6. PROJECT PLANNING

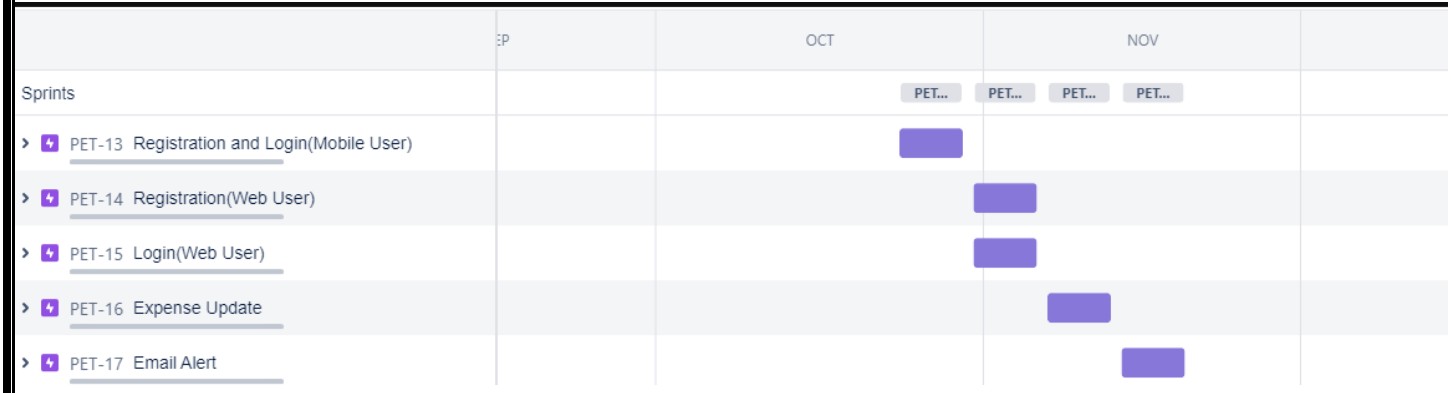
6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Kalaiyarasan V
		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Raviniya k
	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Saranya P
	Dashboard	USN-4	After login, it redirects the user to the dashboard	'2	High	Nancy N

Sprint 2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Saranya P
	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Kalaiyarasan V
	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Raviniya K
		USN-4	Making dashboard interactive with JS	2	High	Kalaiyarasan V
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Nancy N
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user's query	1	Medium	Saranya P
	SendGrid	USN-3	Using SendGrid to send mail to the user about their expenses	1	Low	Raviniya K
		USN-4	Integrating both frontend and backend	2	High	Nancy N
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	Kalaiyarasan V, Raviniya K
	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Saranya P, Nancy N
	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Saranya P, Kalaiyarasan V

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'] = 'b0aebb68-94fa-46ec-a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud'
app.config['port'] = '31249'
```

```
app.config['protocol'] = 'tcpip'
app.config['uid'] = 'xgl34124'
app.config['pwd'] = 'lkP1B5zjTXYPKZUK'
app.config['security'] = 'SSL'
try:
    mysql = DB2(app)

    conn_str='database=bludb;hostname=b0aebb68-94fa-46ec-a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;port=
31249;protocol=tcpip;\
        uid=xgl34124;pwd=lkP1B5zjTXYPKZUK;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 = ? and
password = ?"
        stmt = ibm_db.prepare(ibm_db_conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        result = ibm_db.execute(stmt)
        print(result)
```

```
        account = ibm_db.fetch_row(stmt)
        print(account)

        param = "SELECT * FROM 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[4])

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



 #delete---the--data

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

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

```
#UPDATE---DATA

@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    param = "SELECT * FROM Expense WHERE  id = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["USERID"])
        temp.append(dictionary["DATE"])
        temp.append(dictionary["EXPENSENAME"])
        temp.append(dictionary["AMOUNT"])
        temp.append(dictionary["PAYMODE"])
        temp.append(dictionary["CATEGORY"])
        row.append(temp)
        print(temp)
        dictionary = ibm_db.fetch_assoc(res)

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




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

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

```
        category = request.form['category']
        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 id = ?"
        stmt = ibm_db.prepare(ibm_db_conn, sql)
        ibm_db.bind_param(stmt, 1, p4)
        ibm_db.bind_param(stmt, 2, expensename)
        ibm_db.bind_param(stmt, 3, amount)
        ibm_db.bind_param(stmt, 4, paymode)
        ibm_db.bind_param(stmt, 5, category)
        ibm_db.bind_param(stmt, 6, id)
        ibm_db.execute(stmt)

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


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

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

```
        sql = "INSERT INTO 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 id, limit FROM limit WHERE userid = " +
str(session['id']) + " ORDER BY id DESC LIMIT 1"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    s = "/-"
    while dictionary != False:
        temp = []
        temp.append(dictionary["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[4])
    if x[6] == "food":
        t_food += int(x[4])

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

    elif x[6] == "business":
        t_business  += int(x[4])
```

```
        elif x[6] == "rent":
            t_rent  += int(x[4])

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

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

    print(total)

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



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


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

```
    #   print(texpense)

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

        while dictionary1 != False:
            temp = []
            temp.append(dictionary1["DT"])
            temp.append(dictionary1["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[4])
    if x[6] == "food":
        t_food += int(x[4])

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

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

    elif x[6] == "EMI":
        t_EMI  += int(x[4])
```

```
        elif x[6] == "other":
            t_other  += int(x[4])

    print(total)

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



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

 @app.route("/year")
 def year():
    #   cursor = mysql.connection.cursor()
    #   cursor.execute('SELECT MONTH(date), SUM(amount) FROM
expenses WHERE userid= %s AND YEAR(DATE(date))= YEAR(now())
GROUP BY MONTH(date) ORDER BY MONTH(date)
',(str(session['id'])))
    #   texpense = cursor.fetchall()
    #   print(texpense)

    param1 = "SELECT MONTH(date) as mn, SUM(amount) as tot
FROM 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)


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

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

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

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

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

print(total)

print(t_food)
print(t_entertainment)
```

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



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

  #log-out

  @app.route('/logout')

  def logout():
     session.pop('loggedin', None)
     session.pop('id', None)
     session.pop('username', None)
     session.pop('email', None)
     return render_template('home.html')


  app.run(debug=True)
```

7.2 Feature 2

Home.html

```
<html lang="en">

<head>
```

```
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initialscale=1.0">


  <link rel="stylesheet" href="..\static\css\home.css">
  <title>My Website</title>
</head>

<body>
  <!-- Header -->
  <section id="header">
    <div class="header container">
      <div class="nav-bar">
        <div class="brand">
          <a href="#hero">
            <h1><span>B</span>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"
dataafter="Service">Services</a></li>          <li><a href="#about"
data-after="About">About</a></li>
            <li><a href="#contact" dataafter="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>
  <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>
    </div>
  </section>
  <!-- End Footer -->
  <script src="..\static\js\home.js"></script>
</body>
</html>
```

8. 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 budgetmaking 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.

9. CONCLUSION & FUTURE SCOPE

9.1 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.

9.2 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

10 . 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.

GitHub & Project Demo Link

https://github.com/IBM-EPBL/IBM-Project-48757-1660812706