PERSONAL EXPENSE TRACKER APPLICATION

Team ID: PNT2022TMID03162

A PROJECT REPORT

Submitted by

SAHANAJ ANEEZ M(19EUEC122)

ROHAN RAM R B(19EUEC118)

RUPESH(19EUEC119)

SAI HARIHARAN K(19EUEC123)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution, Affiliated to Anna University Chennai - 600 025)

NOVEMBER 2022

ABSTRACT

.

Tracking regular expense is a key factor to maintain a budget. People often track expense using pen and paper method or take notes in a mobile phone or a computer. These processes of storing expense require further computations and processing for these data to be used as a trackable record. In this work, we are proposing an automated system to store and calculate these data. Personal expense tracker is an android based application. This application allows the user to maintain a computerized diary.Personal expense tracker application which will keep a track of Expenses of a user on a dayto-day basis. This application keeps a record of your expenses and also will give you a category wise distribution of your expenses. With the help of this application user can track their daily/weekly/monthly expenses. This application will also have a feature which will help you stay on budget because you know your expenses. We also have added a special feature which will distributes your expenses in different categories suitable for the user. By using this application, users can save their expense by simply scanning the bills or receipt copies. This application extracts the textual information from the receipts and saves the amount and description for further processing. It also monitors user's income by tracking the received SMS's from the user's saving accounts. By calculating income and expense it produces the user's balance in monthly and yearly basis. Overall, this is a smart automated solution for tracking expense.

1. INTRODUCTION

1.1 PROJECT OVERVIEW :

When it comes to tracking expenses, you can make your system as simple as collecting receipts and organizing them once a month. You might get a little more information from other expense tracking systems (listing them in a spreadsheet, using money management software or even choosing an online application), but all methods have one thing in common: you have to get in the habit of thinking about your expenses. It's very easy to misplace a receipt or forget about any cash you spent. You may even think that a cup of coffee or a trip to the vending machine isn't worth tracking — although those little expenses can add up amazingly fast. There are all sorts of opportunities to throw a kick into your plan to track expenses. You have to get in the habit of doing so, to reduce those lapses, and make sure that the data you're basing financial decisions on is solid.This project will request the clients to add their expenses and in view of their costs ,wallet status will be refreshed which will be noticeable to the client.

- The user interacts with the application.
- Application will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user.
- Also, users can get an analysis of their expenditure in graphical forms.
- They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert
- Setting up Application Environment
- Create Flask project

Work with IBM Cloud CLI, Docker CLI, Sendgrid

Implementation of Web Application

- Create UI to Interact with the application

Connect IBM DB2 with Python

Integration of Sendgrid Service with Python

- Deployment of Cloud Application
- Containerize the application

Upload Image in IBM container directory

Deploy on Kubernetes Cluster

1.2 PURPOSE :

- Help the people to track their expenses.
- Alert users when they exceed the limit of their budget.
- A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about financial management.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

- Lack of visual analytics for visual data.
- Paper based tracking of expenses is very difficult to maintain.
- Most of the applications are used only for personal use.
- Effective financial management requires the proper tracking of income and expenses.
- Efforts has to be made to include each and every transactions into the

input field.

- 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.

2.2 REFERENCES

[1]. Palestinian Ministry of Education and Higher Education. Palestinian Higher Education Statistics.

[2]. Accreditation and Quality Assurance Committee (AQAC) in Palestine. General Report of Information Technology and Engineering Higher Education in Palestine. Accreditation and Quality Assurance Commission (AQAC). Ramallah, Palestine: Palestinian Ministry of Education and Higher Education; 2007 Apr.

[3]. Engineering Association of Palestine. Current Engineering Statistics Book. Ramallah; 2005.

[4]. Prados J, Peterson G, Lattuca L. Quality Assurance of Engineering Education Through Accreditation: The Impact of Engineering Criteria 2000 and Its Global Influence. Journal of Engineering Education. 2005 Jan; 94(1):165–84.

[5]. Chen JW, Yen M. Engineering Accreditation: A Foundation for Continuing Quality

Improvement. 2005 Mar 1–5; Tainan. Exploring Innovation in Education and Research.

[6]. Homma H. Accreditation System in Indonesia. JSME news. 2004 May; 15(1)

[7]. Oberst B, Jones R. International Trends in Engineering Accreditation and Quality Assurance. World Expertise L.L.C.

[8]. Google, 2015. Google developers. [Online] Available at: https://developers.google.com/maps/docum entation/android/intro#accessibility.

[9] APPLE, App states and multitasking. Apple, http://developer.apple.com/library/IOs/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/ManagingYourApplicationsFlow/ManagingYourApplicationsFlow.html#//apple_ref/doc/uid/TP40007072-CH4-SW1, accessed October 2012, n.d.

[10] THE OMNI GROUP, OmniGraffle for Mac. The Omni Group, http://www.omnigroup.com/products/omnigraffle/, accessed November 2012, n.d.

[11] D. MARK, J. NUTTING AND J. LAMARCHE, Beginning iPhone 4 Development: Exploring the iOS SDK, Apress, New York, New York, 2011.

[12] APPLE, Tab bar controllers. Apple, http://developer.apple.com/library/ios/#documentation/WindowsViews/Conceptual/ViewControllerCatalog/Chapters/TabBarControllers.html, accessed October 2012, n.d.

2.3 PROBLEM STATEMENT DEFINITION

Many organizations have their own system to record their income and expenses, which they feel is the main key point of their business progress. It is good habit for a person to record daily expenses and earning but due to unawareness and lack of proper applications to suit their privacy, lacking decision making capacity people are using traditional note keeping methods to do so. Due to lack of a complete tracking

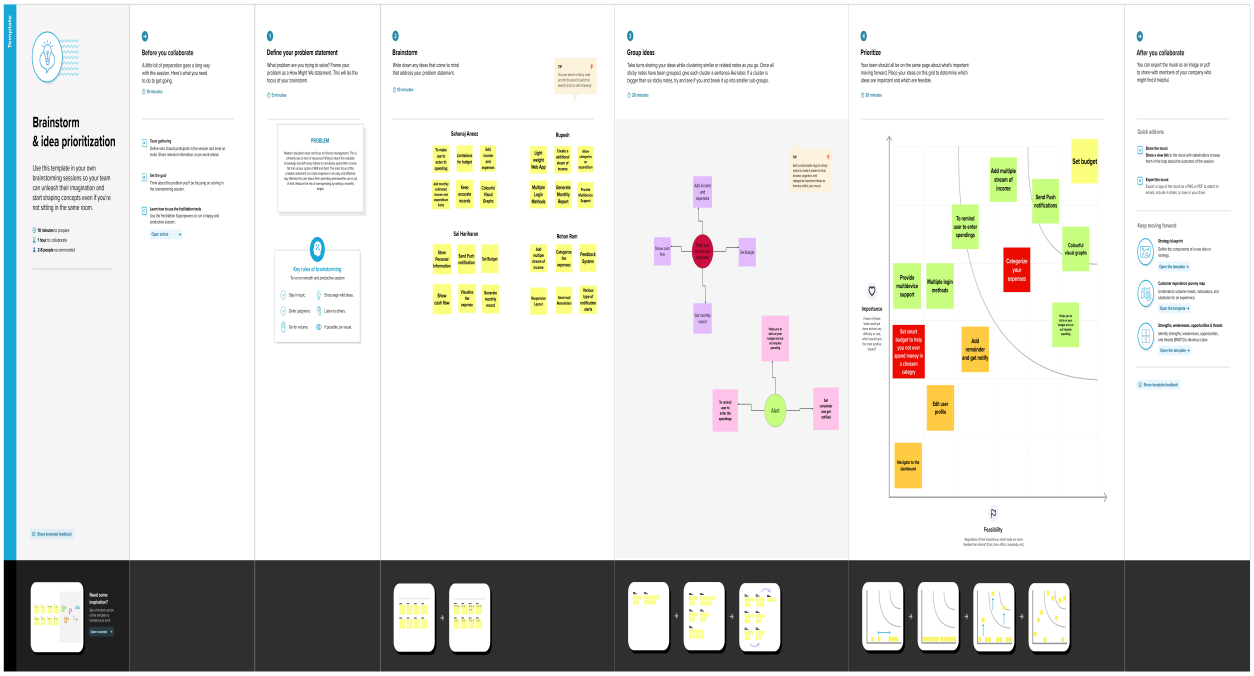
system, there is a constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.

The paper based expense tracker system does not provide the user portability , existing system only used on paper based records so unable to update anywhere expenses done and unable to update the location of the expense details disruptive that the proposed system.

3. IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

3.2 IDEATION AND BRAINSTORMING



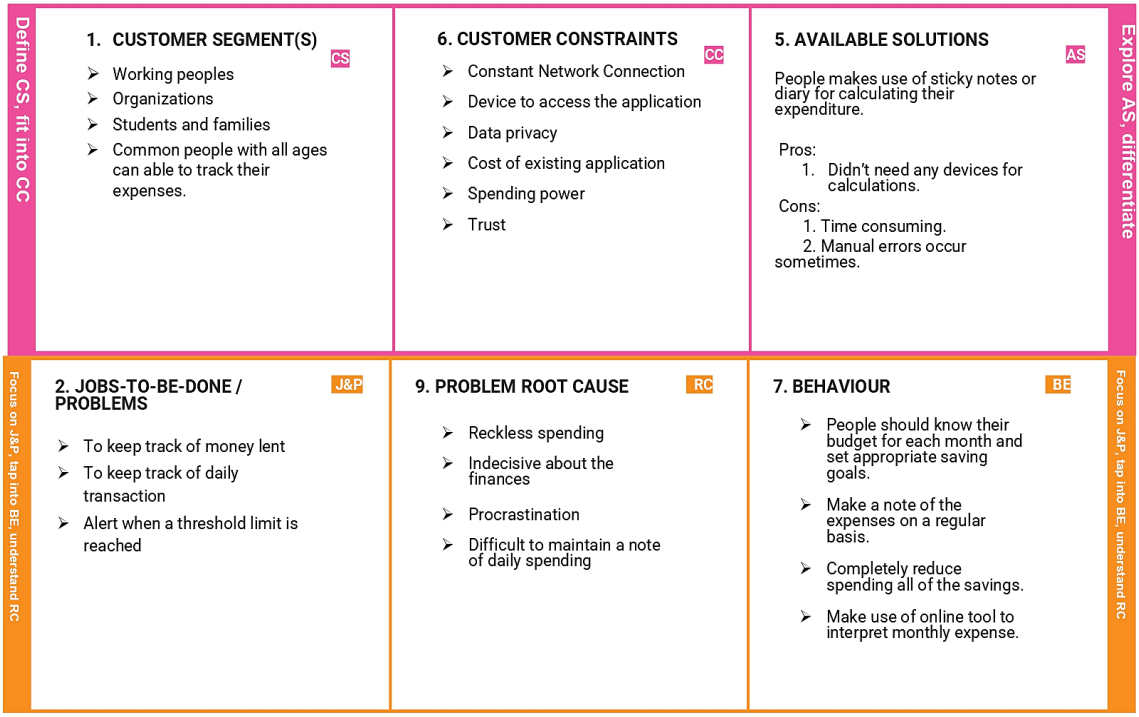
3.3 PROPOSED SOLUTION

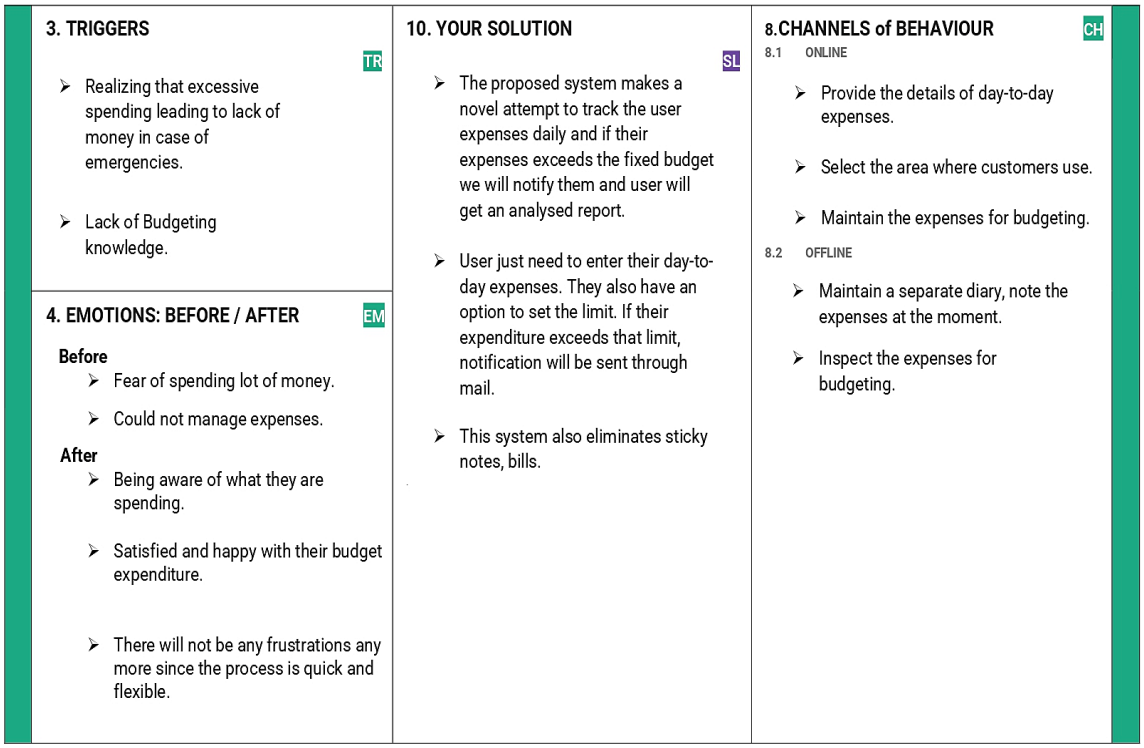
S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Now a days we can't remember where our money goes.And we didn'tHave consciousness in our spending. Therefore, we decided to find an easier way to get rid of this problem.
2.	Idea / Solution description	This app makesyour life easierby helping youto manage your finances efficiently. This personal expense app will not only help you with budgeting and accounting but also give you helpful insights about financial management.
3.	Novelty / Uniqueness	The status generated with visual graphs are more effective than log books.It also helpsin using technology to gain better insights from

		patterns.
4.	Social Impact / Customer Satisfaction	Betterfinancial knowledge is gained. Gamified approach can be used to give self satisfaction. Reduced chances of bad debt in future.
5.	Business Model(Revenue Model)	Wecan provide the application in a subscription based.
6.	Scalability of the Solution	This application can handle large number of users simultaneously. There will be no lagging.

3.4 PROBLEM SOLUTION FIT

Project Title: Personal Expense Tracker Application | **Project Design Phase-I - Solution Fit** | **Team ID:** PNT2022TMID03162





4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

FR No.	Functional Requirement (Epic)	Sub Requirement (Story/ Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration throughLinkedIN
FR-2	User Confirmation	Confirmation via EmailConfirmation via OTP
FR-3	User Login	Login throughEmail/Username and Password

FR-4	Dashboard panel	Shows the summary of expenses/incomes basedon Category, vendorand the paymentsource.
FR-5	Add bill	Add the bill, expenses in add bill category. These features enables customize category of billpayments.
FR-6	Expense Tracker	Check your account statements. Categorize your expenses. Buildyour budget. Trackyour Expense.
FR-7	User Monthly/Weekly plan	This feature can enablethe user to set budgetlimit of a week/month.
FR-8	Expense Planner	Calculate yournet income. Spending limits for eachcategory of expenses.
FR-9	Report Generation	Graphical representation of report can be generated.
FR-10	Alert	Alert messageof excessed budgetlimit through mail.Alert message of every loginof the account. Notifications of every transactions.

4.2 NON FUNCTIONAL REQUIREMENTS

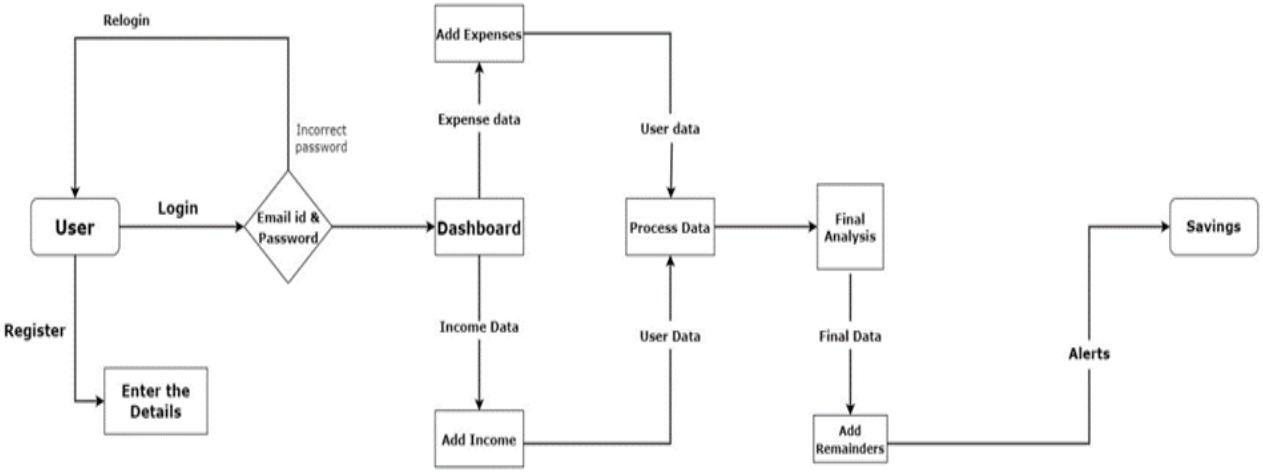
FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	To handleearnings and planexpenses and savings efficiently. Easy to use. Helps to keep an accurate record and track of their income and expenses easily.

NFR-2	**Security**	We save the password in the encrypted form so it will add more secure to the application user. Your login user name and passwords are stored securelyina separate database using multilayered hardwareand software encryption
NFR-3	**Reliability**	Each datarecord is storedon a well-built efficient database schema. There is no risk of dataloss.
NFR-4	**Performance**	Keeping a daily recordof your expenses by tracking receipts, invoices and other outgoing expenses improves the financial health of your budget. The system's throughput is boosted because to the lightweight database support.
NFR-5	**Availability**	The application should be 100% available to user. User can ableto access theapplication with thehelp of the internet through the web browser.
NFR-6	**Scalability**	The ability to appropriately handleincreases the demand. Highly scalable because of Kubernetes (Automation of deployment and scalability).

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

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

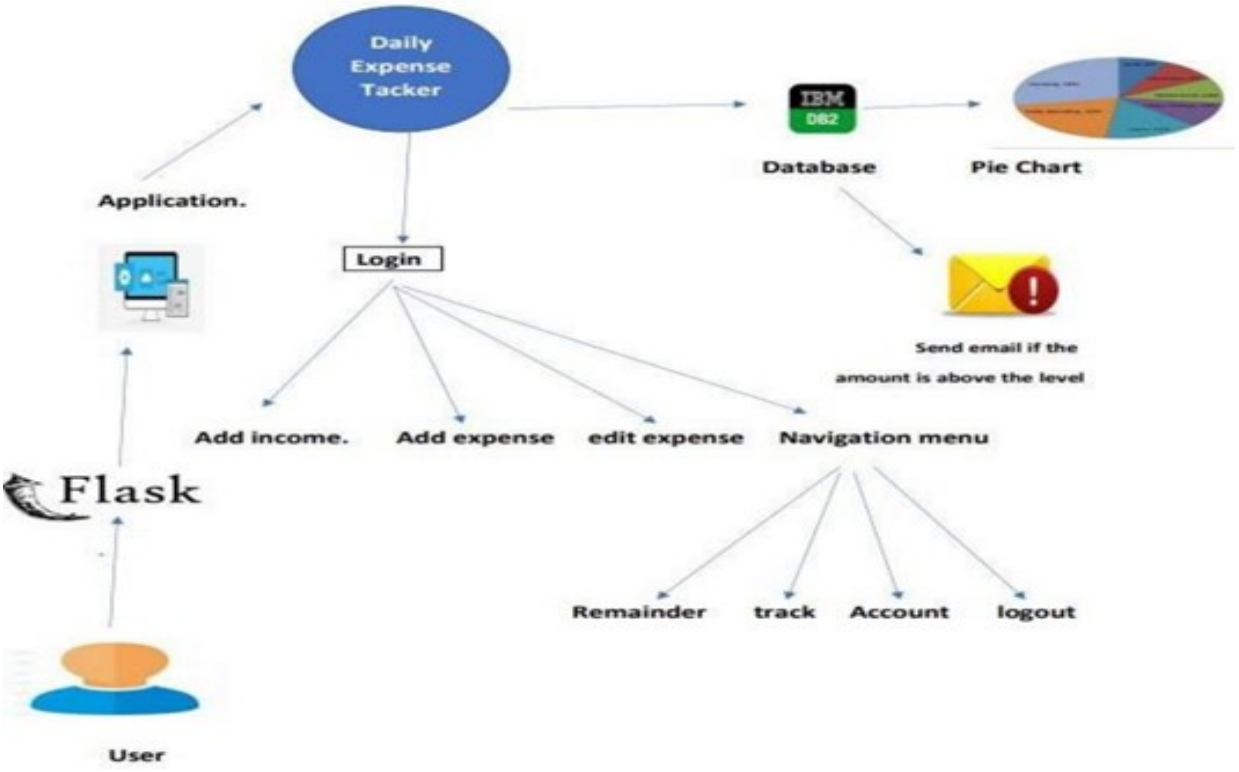


5.2 SOLUTION AND TECHNICAL ARCHITECTURE

SOLUTION ARCHITECTURE

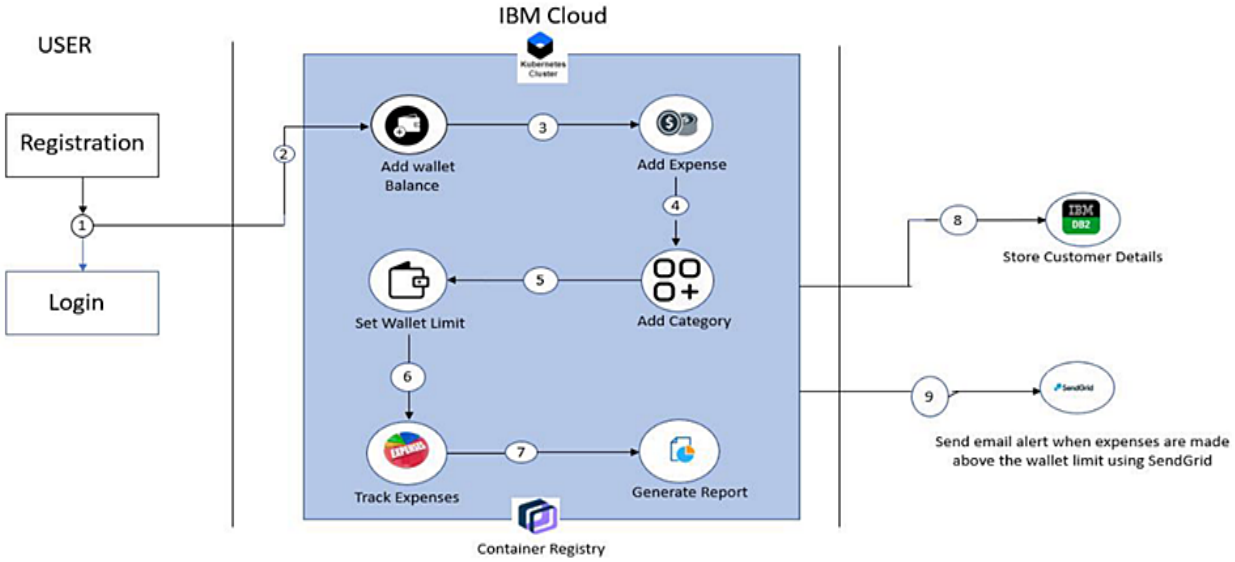
Solution architecture is a complex process – with many sub-processes – that bridgesthe gap between business problems and technology solutions. Its goals are to:

1. Find the best tech solution to solve existing business problems.
2. Describe the structure, characteristics, behavior, and other aspectsof the software to project stakeholders.
3. Define features, development phases, and solution requirements.
4. Provide specifications according to which the solution is defined, managed, and delivered.



Architecture of the Personal Expense Tracker Application

TECHNICAL ARCHITECTURE



5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can view my daily expenses	High	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can login into application using correct register email id and password	High	Sprint-1
	Dashboard	USN-6	As a user I can see the daily expenses and expenditure details		High	Sprint-1
		USN-7	As a user, I can see my expenses for my clear understand purpose	I can see my expenses in graphical representations	High	Sprint-2

Customer (Web user)		USN-8	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-2
Customer Care Executive		USN-9	As a customer care executive, I can solve the issues of the application immediately	I can provide customer support of working hours through phone call and chat	Medium	Sprint-4
Administrator		USN-10	As an Administrator, I can update or upgrade the server or application	I can fix the bug of server side of data manage, crush	Medium	Sprint-3

6. PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

Sprint	Functional Requirement(Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Registration	USN-1	As a user, I can register for the application by entering my email, password and confirming my password.	2	High	Sahanaj Aneez, Rohan Ram R B
		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Rupesh, Sai Hariharan K
	Login	USN-3	As a user, I can log into the application by entering email& password	1	High	Sahanaj Aneez, Rohan Ram R B
	Dashboard	USN-4	Logging in takes to the dashboard for the logged user.	`2	High	Rupesh, Sai Hariharan K
Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only						

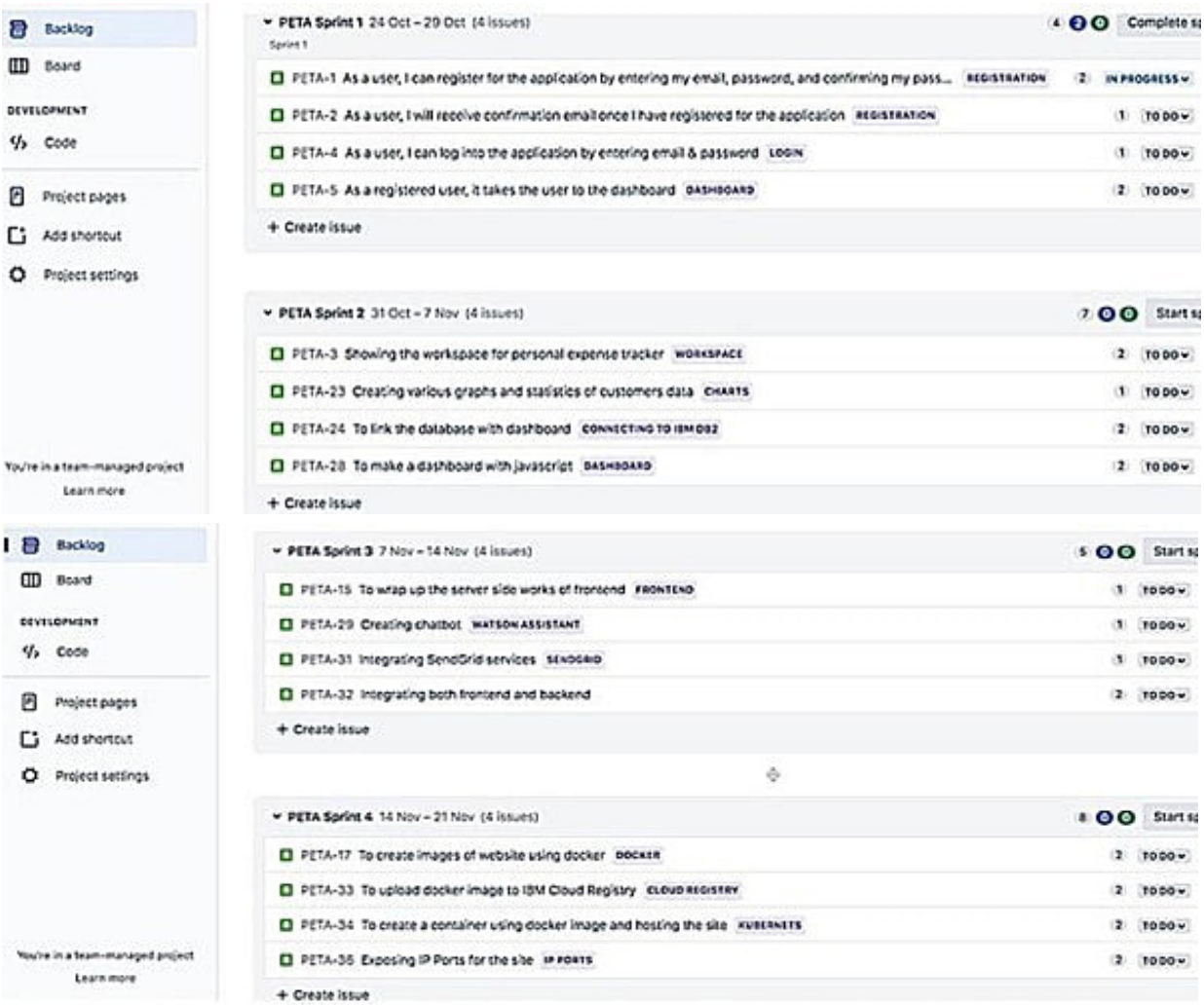
Sprint 2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Sahanaj Aneez, Rohan Ram R B
	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Rupesh, Sai Hariharan K
	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Sahanaj Aneez, Rohan Ram R B
		USN-4	Making dashboard interactive with JS	2	High	Rupesh, Sai Hariharan K
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Sahanaj Aneez, Rohan Ram R B
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifyinguser's query	1	Medium	Rupesh, Sai Hariharan K
	SendGrid	USN-3	Using SendGrid to send mail to the user about theirexpenses	1	Low	Sahanaj Aneez, Rohan Ram R B
		USN-4	Integrating both frontend and backend	2	High	Rupesh, Sai Hariharan K
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	Sahanaj Aneez, Rohan Ram R B
	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Rupesh, Sai Hariharan K
	Kubernetes	USN-3	Create container using the docker image and hosting thesite	2	High	Sahanaj Aneez, Rohan Ram R B
	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Sahanaj Aneez, Rohan Ram R B Rupesh, Sai Hariharan K

6.2 SPRINT DELIVERY SCHEDULE

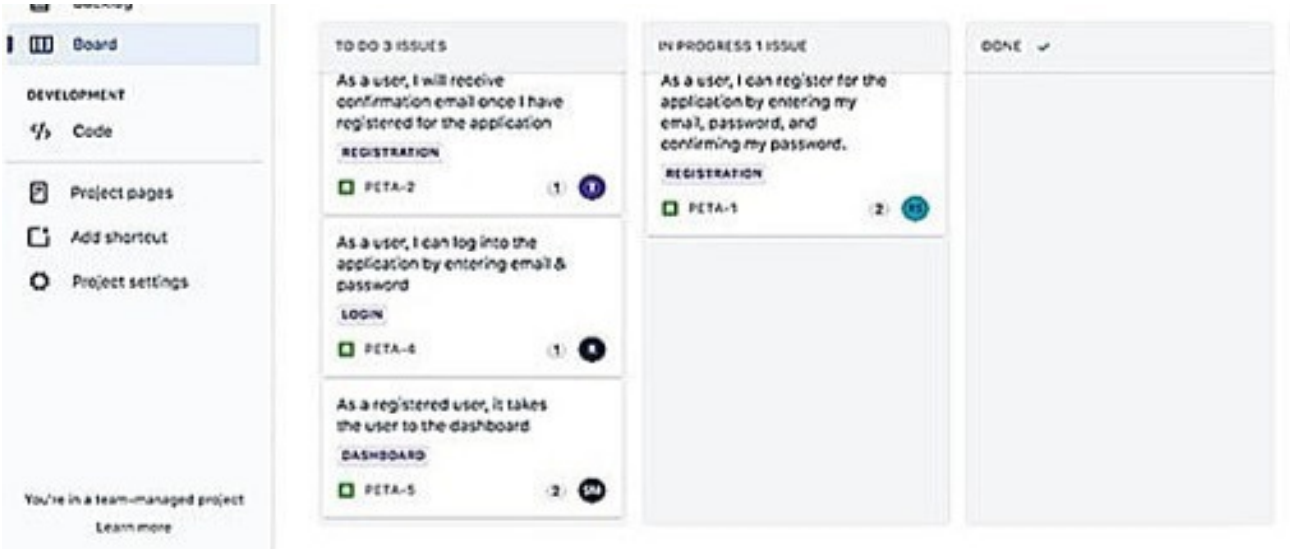
Sprint	Total Story Points	Duration	Sprint StartDate	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint ReleaseDate (Actual)
Sprint-1	20	6 Days	23 Oct 2022	28 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	30 Oct 2022	04 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	06 Nov 2022	11 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	13 Nov 2022	18 Nov 2022	20	19 Nov 2022

6.3 REPORTS FROM JIRA

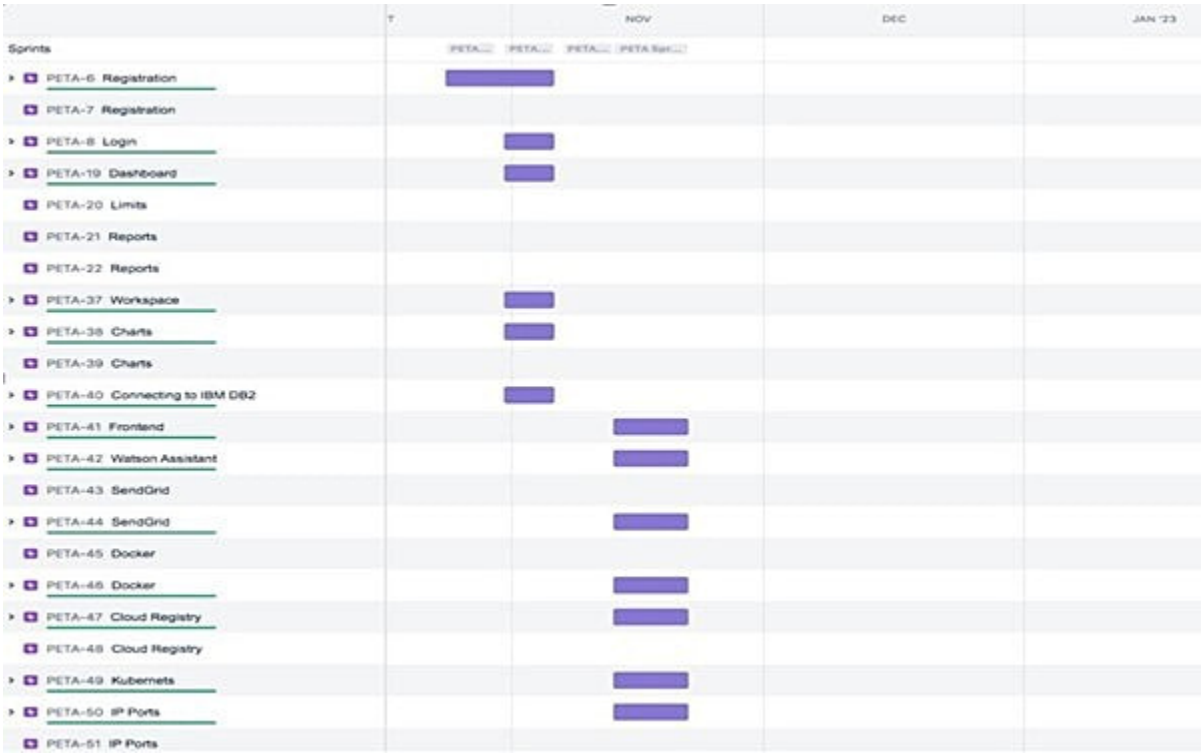
BACKLOG



BOARD



ROADMAP



7. CODING AND SOLUTION

7.1 FEATURES

Feature 1: Add Expense

Feature 2: Update expense

Feature 3: Delete Expense

Feature 4: Set Limit

Feature 5: Send Alert Emails to users

Other Features: Track your expenses anywhere, anytime. Seamlessly manage your money and budget without any financial paperwork. Just click and submit your invoices and expenditures. Access, submit, and approve invoices irrespective of time and location. Avoid data loss by scanning your tickets and bills and saving in the app. Approval of bills and expenditures in real-time and get notified instantly. Quick settlement of claims and reduced human errors with an automated and streamlined billing process.

7.2 CODES

app.py:

```
# -*- coding: utf-8 -*-
"""
Spyder Editor
This is a temporary script file.
"""

from flask import Flask, render_template, request, redirect, session
# from flask_mysqldb import MySQL
# import MySQLdb.cursors
import re

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

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


app = Flask(__name__)

app.secret_key = 'a'

# app.config['MYSQL_HOST'] = 'remotemysql.com'
# app.config['MYSQL_USER'] = 'D2DxDUPBii'
# app.config['MYSQL_PASSWORD'] = 'r8XBO4GsMz'
# app.config['MYSQL_DB'] = 'D2DxDUPBii'
"""
dsn_hostname = "3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"
dsn_uid = "sbb93800"
dsn_pwd = "wobsVLm6ccFxcNLe"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "31498"
dsn_protocol = "tcpip"
dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
    "PORT={3};"
    "PROTOCOL={4};"
    "UID={5};"
    "PWD={6};"
).format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol, dsn_uid,
dsn_pwd)
"""
# app.config['DB2_DRIVER'] = '{IBM DB2 ODBC DRIVER}'
app.config['database'] = 'bludb'
app.config['hostname'] = '3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'
```

```
app.config['port'] = '31498'
app.config['protocol'] = 'tcpip'
app.config['uid'] = 'sbb93800'
app.config['pwd'] = 'wobsVLm6ccFxcNLe'
app.config['security'] = 'SSL'
try:
   mysql = DB2(app)

   conn_str='database=bludb;hostname=3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=31498;protoco
l=tcpip;\
         uid=sbb93800;pwd=wobsVLm6ccFxcNLe;security=SSL'
   ibm_db_conn = ibm_db.connect(conn_str,'','')

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

# app.config['']

# mysql = MySQL(app)


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

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



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

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



@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    print("Break point1")
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

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

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

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

        print("Break point5")
        sql = "SELECT * FROM register WHERE username = ?"
        stmt = ibm_db.prepare(ibm_db_conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        result = ibm_db.execute(stmt)
```

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

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

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

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

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

# print(dictionary["username"])
print("break point 6")
if account:
    msg = 'Username already exists !'
elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
    msg = 'Invalid email address !'
elif not re.match(r'[A-Za-z0-9]+', username):
    msg = 'name must contain only characters and numbers !'
else:
    sql2 = "INSERT INTO register (username, email,password) VALUES (?, ?, ?)"
    stmt2 = ibm_db.prepare(ibm_db_conn, sql2)
    ibm_db.bind_param(stmt2, 1, username)
    ibm_db.bind_param(stmt2, 2, email)
    ibm_db.bind_param(stmt2, 3, password)
    ibm_db.execute(stmt2)
```

```
        # cursor.execute('INSERT INTO register VALUES (NULL, % s, % s, % s)', (username,
email,password))
        # mysql.connection.commit()
        msg = 'You have successfully registered !'
    return render_template('signup.html', msg = msg)




 #LOGIN--PAGE

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

@app.route('/login',methods =['GET', 'POST'])
def login():
    global userid
    msg = ''


    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        # cursor = mysql.connection.cursor()
        # cursor.execute('SELECT * FROM register WHERE username = % s AND password
= % s', (username, password ),)
        # account = cursor.fetchone()
        # print (account)

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

```
        print(account)

        param = "SELECT * FROM register WHERE username = '' + "\'" + username + "\'" + "
and password = '' + "\'" + password + "\'"
        res = ibm_db.exec_immediate(ibm_db_conn, param)
        dictionary = ibm_db.fetch_assoc(res)

        # sendmail("hello sakthi","sivasakthisairam@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)

#ADDING----DATA



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



@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():

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

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

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

    sql = "INSERT INTO expenses (userid, date, expensename, amount, paymode,
category) VALUES (?, ?, ?, ?, ?, ?)"
    stmt = ibm_db.prepare(ibm_db_conn, sql)
    ibm_db.bind_param(stmt, 1, session['id'])
    ibm_db.bind_param(stmt, 2, p4)
    ibm_db.bind_param(stmt, 3, expensename)
    ibm_db.bind_param(stmt, 4, amount)
    ibm_db.bind_param(stmt, 5, paymode)
    ibm_db.bind_param(stmt, 6, category)
    ibm_db.execute(stmt)

    print("Expenses added")

    # email part

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

```
        temp.append(dictionary["ID"])
        temp.append(dictionary["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 += x[4]

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

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

    return redirect("/display")
```

```
#DISPLAY---graph

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

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

    param = "SELECT * FROM expenses WHERE userid = " + str(session['id']) + " ORDER
BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["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):
    #  cursor = mysql.connection.cursor()
    #  cursor.execute('DELETE FROM expenses WHERE  id = {0}'.format(id))
    #  mysql.connection.commit()

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

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


#UPDATE---DATA

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

    param = "SELECT * FROM expenses WHERE  id = " + id
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    row = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["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']

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

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

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

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

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

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

      sql = "INSERT INTO limits (userid, limitss) VALUES (?, ?)"
      stmt = ibm_db.prepare(ibm_db_conn, sql)
      ibm_db.bind_param(stmt, 1, session['id'])
      ibm_db.bind_param(stmt, 2, number)
      ibm_db.execute(stmt)

      return redirect('/limitn')


@app.route("/limitn")
def limitn():
  # cursor = mysql.connection.cursor()
```

```
    # cursor.execute('SELECT limitss FROM `limits` ORDER BY `limits`.`id` DESC LIMIT 1')
    # x= cursor.fetchone()
    # s = x[0]

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

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

#REPORT

@app.route("/today")
def today():
    #   cursor = mysql.connection.cursor()
    #   cursor.execute('SELECT TIME(date)   , amount FROM expenses  WHERE userid =
%s AND DATE(date) = DATE(NOW()) ',(str(session['id'])))
    #   texpense = cursor.fetchall()
    #   print(texpense)

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

      while dictionary1 != False:
          temp = []
```

```
        temp.append(dictionary1["TN"])
        temp.append(dictionary1["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 expenses WHERE userid = " + str(session['id']) + " AND
DATE(date) = DATE(current timestamp) ORDER BY date DESC"
     res = ibm_db.exec_immediate(ibm_db_conn, param)
     dictionary = ibm_db.fetch_assoc(res)
     expense = []
     while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["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 += x[4]
        if x[6] == "food":
            t_food += x[4]

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

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

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

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

    print(total)

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



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

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

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

    while dictionary1 != False:
        temp = []
        temp.append(dictionary1["DT"])
        temp.append(dictionary1["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 expenses WHERE userid = " + str(session['id']) + " AND
MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current
timestamp) ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
```

```
dictionary = ibm_db.fetch_assoc(res)
expense = []
while dictionary != False:
    temp = []
    temp.append(dictionary["ID"])
    temp.append(dictionary["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 += x[4]
    if x[6] == "food":
        t_food += x[4]

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

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

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

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

    print(total)

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



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

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

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

```
    dictionary1 = ibm_db.fetch_assoc(res1)
    texpense = []

    while dictionary1 != False:
        temp = []
        temp.append(dictionary1["MN"])
        temp.append(dictionary1["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 expenses WHERE userid = " + str(session['id']) + " AND
YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"
    res = ibm_db.exec_immediate(ibm_db_conn, param)
    dictionary = ibm_db.fetch_assoc(res)
    expense = []
    while dictionary != False:
        temp = []
        temp.append(dictionary["ID"])
        temp.append(dictionary["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 += x[4]
    if x[6] == "food":
        t_food += x[4]

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

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

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

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

print(total)

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

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

#log-out

@app.route('/logout')

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


port = os.getenv('VCAP_APP_PORT', '8080')
if __name__ == "__main__":
   app.secret_key = os.urandom(12)
   app.run(debug=True, host='0.0.0.0', port=port)
```

deployment.yaml:

```
apiVersion: apps/v1
 kind: Deployment
 metadata:
  name: sakthi-flask-node-deployment
 spec:
  replicas: 1
  selector:
    matchLabels:
      app: flasknode
  template:
```

```
    metadata:
      labels:
        app: flasknode
    spec:
      containers:
      - name: flasknode
        image: icr.io/sakthi_expense_tracker2/flask-template2
        imagePullPolicy: Always
        ports:
        - containerPort: 5000
```

flask-service.yaml:

```
apiVersion: v1
kind: Service
metadata:
  name: flask-app-service
spec:
  selector:
    app: flask-app
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 5000
  type: LoadBalancer
```

manifest.yml:

```
applications:
 - name: Python Flask App IBCMR 2022-10-19
   random-route: true
   memory: 512M
   disk_quota: 1.5G
```

sendemail.py:

```
import smtplib
import sendgrid as sg
import os
from sendgrid.helpers.mail import Mail, Email, To, Content
SUBJECT = "expense tracker"
s = smtplib.SMTP('smtp.gmail.com', 587)

def sendmail(TEXT,email):
    print("sorry we cant process your candidature")
    s = smtplib.SMTP('smtp.gmail.com', 587)
    s.starttls()
    # s.login("il.tproduct8080@gmail.com", "oms@1Ram")
    s.login("tproduct8080@gmail.com", "lxixbmpnexbkiemh")
    message  = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)
    # s.sendmail("il.tproduct8080@gmail.com", email, message)
    s.sendmail("il.tproduct8080@gmail.com", email, message)
    s.quit()
def sendgridmail(user,TEXT):

    # from_email = Email("shridhartp24@gmail.com")
    from_email = Email("tproduct8080@gmail.com")
    to_email = To(user)
    subject = "Sending with SendGrid is Fun"
    content = Content("text/plain",TEXT)
    mail = Mail(from_email, to_email, subject, content)

    # Get a JSON-ready representation of the Mail object
    mail_json = mail.get()
    # Send an HTTP POST request to /mail/send
    response = sg.client.mail.send.post(request_body=mail_json)
    print(response.status_code)
    print(response.headers)
```

7.3 DATABASE SCHEMA

Tables :

1.Admin:

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,username VARCHAR(32) NOT NULL, email VARCHAR(32) NOT NULL,password VARCHAR(32) NOT NULL

2.Expense:

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
userid INT NOT NULL, date TIMESTAMP(12) NOT NULL,expensename VARCHAR(32) NOT NULL,
amount VARCHAR(32) NOT NULL,
paymode VARCHAR(32) NOT NULL,
category VARCHAR(32) NOT NULL

3.LIMIT:

id INT NOT NULL GENERATED ALWAYS AS IDENTITY,userid VARCHAR(32) NOT NULL,
limit VARCHAR(32) NOT NULL

8. TESTING

8.1 TEST CASES

S. NO	Test Cases	Result
1	Verify the user is able to see the Sign up page when the user clicks the signup button in navigation bar	Positive
2	Verify the UI elements in the Sign up page	Positive
3	Verify the user is able to register into the application by providing valid details	Positive

4	Verify the user is able to see the sign in page when the user clicks the sign in button in navigation bar	Positive
5	Verify the UI elements in the Sign in page	Positive
6	Verify the user is able to login into the application by providing valid details	Positive
7	Verify the user is able to see the add expenses page when the user clicks the add expenses navigation bar	Positive
8	Verify the UI elements in the add expenses page	Positive
9	Verify the user is able to add expenses by providing valid details	Positive
10	Verify the user is able to see the home button in the add expenses page	Positive
11	Verify whether the expenses added can be deleted from the cloud	Positive
12	Verify whether the expenses added can be edited	Positive
13	Verify whether the date, month and year are valid while user entering the expenses	Positive
14	Verify whether the data types entered by the user in the add expenses page are float or integers	Positive
15	Verify whether expenses added previously still exist	Positive
16	Verify whether the expenses amount entered fits between in the range of integers or float datatypes	Positive

8.2 USER ACCEPTANCE TESTING

1. Defect Analysis

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

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

2. Test Case Analysis

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

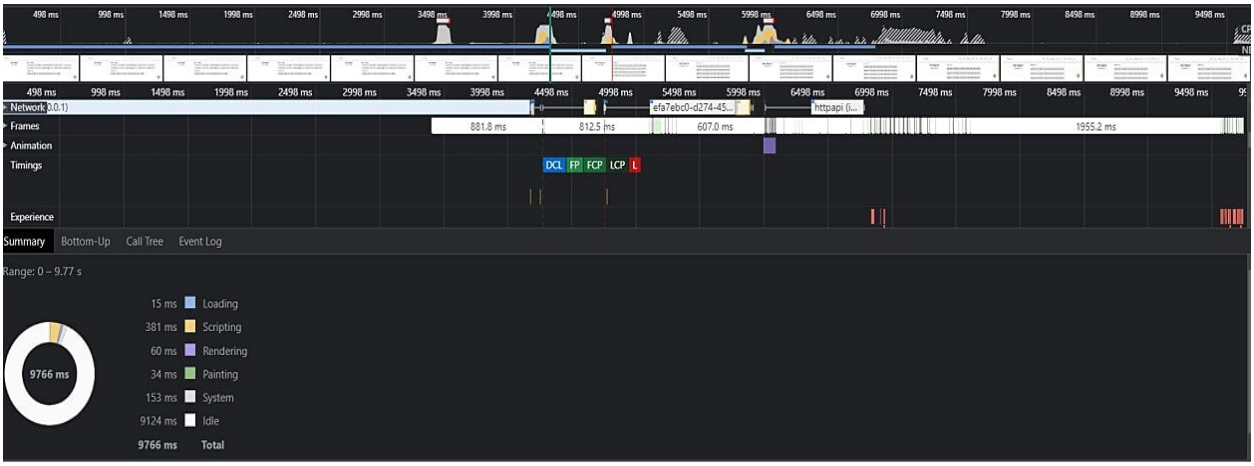
Section	Total Cases	Not Tested	Fail	Pass
Interface	7	0	0	7
Login	20	0	0	20
Logout	2	0	0	2
Limit	3	0	0	3
Signup	8	0	0	8
Final Report Output	4	0	0	4

9. RESULTS

9.1 PERFORMANCE METRICES

The performance metrics include page speed, time to title, bounce rate, time to start render, time to interact, DNS lookup speed, Requests per second, error rate, time to first byte/last byte and conversion rate. The performance metrics of our application are efficient and are given below for reference.

Performance Metrices



10. ADVANTAGES & DISADVANTAGES :

ADVANTAGES :

- In this day and age, when expenses are going through the roof, it has become crucial that you learn to make your money work for you so that you can create a nest egg for the future.
- Has various components of updating and viewing users expenditure
- User can track his expenses by choosing a day and using various filtering Options to study expenses

- Visualization using pie chart with percentage view shows graphical Representation .
- This approach effectively keeps away from the manual figuring for trying not to ascertain the pay and cost each month.
- 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
- With a daily expense manager, you will be able to allocate money to different priorities and this will also help you cut down on unnecessary spending. As a result, you will be able to save and be able to keep worry at bay.

- A daily money tracker helps you budget your money so that you use it wisely. If you find that every month your expenses are more than what you earn, it is time to put your house in order and get a money manager app that keeps track of your money without any problem.

DISADVANTAGES :

- Lack of visual analytics for expense data
- Lack of support for splitting group expenses
- Suitable for only Personal use.
- Errors are another common problem expense reports drafted with Excel. As is the case with all tasks performed manually and based on paper, it is extremely

likely that business travellers who report expenses make involuntary mistakes: numbers entered wrongly, duplicated expenses or expenses relating to a previous settlement period, misapplication of the expense policy, etc.

- Frequent tracking of cash spending can allow one to catch and correct errors so that the budget plan is still able to be adhered to despite the mistake.
- Another con that may occur when spending is being tracked is an error, but this may also be able to be changed into a pro if the person does regular tracking.

11. CONCLUSION

From this project, we are able to manage and keep tracking the dailyexpenses as well as income. **Personal** 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.By using this application, users can save their expense by simply scanning the bills or receipt copies. This application extracts the textual information from the receipts and saves the amount and description for further processing. It also monitors user's income by tracking the received SMS's from the user's saving accounts. By calculating income and expense it produces the user's balancein monthly and yearly basis. Overall, this is a smart automatedsolutionfor tracking expense.

12. FUTURE SCOPE :

- **Mobility :**

Businesses are becoming increasingly global and employees are more mobile

than ever. According to the Certify Expense Management Trends Report 2018, 47 percent of organizations felt that mobile applications and accessibility were a critical capability of expense reporting software.

Expense management software will begin to respond to this change in the manner that people work by facilitating mobility. Employees will be able to submit reports and managers can approve the claims from a smartphone. Also, mobile applications will become more intuitive and responsive, encouraging greater adoption.

- **Travel booking :**

According to a 2018 study by Tripactions, 90 percent of respondents believe that travel is important for growth. However, 50 percent of employees don't use the corporate travel solution offered by their organization owing to difficulties and the time taken to book a trip. They prefer using consumer channels.Using multiple external vendors and disjointed channels to book travel can become tedious and lead to a lot of last-minute chaos. Also, the expenses have to be recorded accurately in the expense management system. It's imperative to integrate the same into expense management software.

- **Integrated system :**

In 2019, expense management software will begin to consolidate currently disparate actions into a seamless system with credit cards, bank accounts, accounting, payroll, CRM, and more in one place. All these systems share a lot of common information, and it makes perfect sense to connect all these to ensure data integrity.

When integrated with travel applications and travel management company (TMC) solutions, transactions can be entered directly into the system without the need for manual input. Corporate cards can also be linked to the software, which makes it easier

to reconcile credit card statements with expense reports

- **Artificial intelligence :**

As machine learning and artificial intelligence (AI) evolve, they will add to the sophistication of expense management software. There will be improved ability to assign expense general ledger codes based on submissions and smarter categorization of expense types. Rudimentary versions of this exist in a few tools that use rule-based categorization.AI-powered automation is also expected to minimize fraudulent expense reports. It can monitor, track, and approve expenses, and compliance-related issues will immediately be flagged and eliminated.

13. APPENDIX

Source Code Github Link: https://github.com/IBM-EPBL/IBM-Project-23271-1659875386

Project Demo Link:
https://drive.google.com/drive/folders/1Iamil0jHX6m9PEsP7ZDlQDprWFBkKJPa?usp=sharing