PERSONAL EXPENSE TRACKER

PROJECT BASED LEARNING

Submitted By

HARINI K B (910619104025)

DEEPA REKHA S (910619104014)

JASMINE PRIYA E (910619104032)

AATHI SRI T G (910619104002)

Of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

K.L.N. COLLEGE OF ENGINEERING, POTTAPALAYAM

ANNA UNIVERSITY: CHENNAI 6200 025

NOVEMBER 2022

BONAFIDE CERTIFICATE

Certified that this project report **"PERSONAL EXPENSE TRACKER APPLICATION"** is the bonafide work of **"HARINI K B (910616104025)","AATHI SRI T G (910616104002)" , "JASMINE PRIYA E(910616104032)"** and **"DEEPA REKHA S"(910619104014)** who carried out the project under my supervision.

SIGNATURE

Dr.S.MIRUNA JOE AMALI

HEAD OF THE DEPARTMENT

Computer science and engineering,

K.L.N. College of Engineering,

Pottapalayam,

Sivagangai-630 612.

SIGNATURE

Mr.D.Prabhu

ASSISTANT PROFESSOR

Computer science and engineering,

K.L.N. College of Engineering,

Pottapalayam,

Sivagangai-630 612

Submitted for the project viva-voce
conducted on 22-09-2020

EVALUATOR

MENTOR

ABSTRACT

Expense Tracker is used to maintain and manage data of daily expenditure in a more precise way it can give profound knowledge of their expenses. User can choose the kind of spending they wanted to do, even the amount etc. and all these details is going to be saved by the internal database storge. In this system user can actually have a knowledge about their expenditure on their daily basis, weekly as well as monthly basis. This systematic way of storing your information related to your expenses would help you to keep a track of your expenditure and further you do not have to do the manual stuff. Some statistical analysis has to be done to be able to give users correct information on their expenses and help them spend better. This helps the society to prevent the issues like bankruptcy and save time from manual calculations. User can provide his/her income to calculate the total expense per day and the results will be stored for each individual user. People when usually go for trips with friends, can use this tracker to maintain their expense. This project will provide a lot of benefits to the users with the help of which they will be surely able to keep track of each penny. It is time to stop using paper and excel sheets to keep track of your digital as well as cash payments. Using paper is not easy to manage. It is common to delete files accidentally or misplace files. This expense tracker provides a complete digital solution to this problem. Excel sheets do very little to help in tracking expenses. Furthermore, they don't have the advanced functionality of preparing graphical visuals automatically. Not only it will save the time of the people but also it will assure error-free calculations. The user just has to enter the income and expenditures and everything else will be performed by the system.

TABLE OF CONTENTS

10. ADVANTAGES & DISADVANTAGES
11. CONCLUSION
12. FUTURE SCOPE
13. APPENDIX
Source Code
GitHub & Project Demo Link

1.INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 Project Overview:

In today's world financial well-being is the dream of every person and managing and keeping track of their expenses play a crucial role in this goal. If a person is not able to manage his expenses, he/she is likely to end up in a crisis. Money management is a necessary part of life. A proper balance between income and expense is a must for a comfortable livelihood. There must be some savings that can be used at a later point of life when needed. But in the absence of proper management of money, we left with no savings at all. Some people note down every single expense which is a good practice. Expense Tracker to efficiently manage house-old budget. Our system will allow user to keep track of their expenses. Some statistical analysis has to be done to be able to give users correct information on their expenses and help them spend better. This helps the society to help them from issues like bankruptcy and save time from manual calculations. For using such application, a user needs to provide his/her total income or the amount he/she is spending per day and each user details or information are going to be stored in a unique way. Every user is required to register on the system to create a record unique to the user. This expense tracker uses statistical analysis which are going to keep a track of your expenses and would even give you results accordingly. If you exceed the monthly target, it notifies you through email. Tracking application will generate a report accordingly on monthly basis and would generate a statistical analysis of your expenses in a more sorted and easier to understand way. Today, people don't have to worry as there are numerous applications and techniques using which they can manage their expenses. Also called expense manager, an expense tracker is software that facilitates keeping a record of an individual's money inflow and outflow. Most of the people in the world live on a static revenue, and they discover that towards the end of the month they don't even have enough money to meet their essentials. Though this problem can be due to low salary, most of the time it is because of improper money management skills. Using an expense tracker can help you keep track of how much money you spend every day and on what. At the end of the month, you will have a clear picture of where your money is going. This is one among the simplest ways to urge your expenses in check.

1.2 Purpose:

In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management. Personal finance applications will ask users to add their expenses and based on their expense's 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.

2. LITERATURE SURVEY

CHAPTER 2
LITERATURE SURVEY

2.1 Existing Problem:

(R N Rajprabha, 2017) created an android version of family budget manager with later evolved in PDA and tablet features. **(Ravi Sharma, 2017)** stated users sometimes feels uncomfortable in sharing their personal information with an app and he suggested security and usability are two major concerns. Even the advanced UI needs to maintain retention. **Researchers of Mother Terresa university, Andhra Pradesh (2019)** also stated an online income and budget tracker in a website mode but that project used csv mode to store data but that project had a drawback in its existing model as it can't handle the data efficiently in addition to that it wasn't user-friendly and an unpopulated data project. All these researches above suggest some of the modern way of dealing with expense tracking. Many of the researches like these actually represents the evolution in ideas with time "evolution is not a necessity it's more like change in thinking and time" in which we analyze estimate and evaluate the things according to new requirements. But still the kind of technology used in it is kind of projects were used in previous days there are certain android apps as well still they too also have different consequences as well as drawbacks in itself. And I also feel like these should be way easier to handle to a desktop device. As sometimes android apps will provide in accurate results if the information is incorrect and many of the times, we almost got forget to enter details too and most them don't even provide notification for that as well. Based on the literature review.
This study shows the evolution as well as the comparison from selected researches according to the adopted knowledge in it. This paper suggests some effective changes that are still needed and why the transition is necessarycsv mode to store data but that project had a drawback in its existing model as it can't handle the data efficiently in addition to that it wasn't user-friendly and an unpopulated data project. All these researches above suggest some of the modern way of dealing with expense tracking. Many of the researches like these actually represents the evolution in ideas with time "evolution is not a necessity it's more like change in thinking and time" in which we analyze estimate and evaluate the things according to new requirements. But still the kind of technology used in it is kind of projects were used in previous days there are certain android apps as well still they too also have different consequences as well as drawbacks in itself. And I also feel like these should be way easier to handle to a desktop device. As sometimes android apps will provide in accurate results if the information is incorrect and many of the times, we almost got forget to enter details too and most them don't even provide notification for that as well. Based on the literature review. This study shows the evolution as well as the comparison from selected researches according to the adopted knowledge in it. This paper suggests some effective changes that are still needed and why the transition is necessary.

A. Intelligent Online Budget Tracker

The development of this application has been conducted in a stepwise manner using the well-defined methodology, RUP, customized according to the requirements of the system. Most of the goals set at the start of the development phase have been met. Security problems like web security or network security have also been treated in the design and

development of the system, thus increasing the reliability of the system. Quality management issues have also been handled satisfactorily.

B. Online Income and Expense Tracker

This project is work more efficient than the other income and expense tracker. The project successfully avoids the manual calculation for calculating the income and expense per month. The modules are developed efficiently and also in an attractive manner.

C. Family Expense Manager Application

As the result, the user can make use of this application in his/her daily life. After being used it can be a part of daily life to update and view daily expenses and family expenses. This helps to keep track of expenses & manage it for the user as they are busy in their daily routine, they are not able to keep track of their incomes & expenses.

D. Personalized Expense Managing Assistant Using Android

Some of the features are like enabling users to register to the application using an existing email or social network account, it will synchronize the user's profile information to the application. Apart from this, the application can be used to gather samples of data related to user's expenses with consents and use those sample data as parameters to assess patterns of spending. Using some data mining techniques expenses can be classified and can be used in market analysis and planning

E. Mobiwik Expense Tracking Application

Mobikwik came up with a new feature in their app called Expense Manager. With this feature, you can track and manage your expenditures(expenses), savings, reminders and bill payments. This is a personal budget management app that tracks your expenditures and income and gives you recommendations to make you economically strong. The main idea of developing this feature for giving users a clear picture that how much they are spending and where they are spending and when. We remind them to pay their utilities and card bills before the due date by using the same platform in just one tap, instead of going any other way. Also serving them by giving saving tips for their good future investment.

References

S. NO	Title, Publication and Authors	Techniques & Mechanisms	Parameter Analysis	Tools	Findings
1	Spending Tracker: A Smart Approach to Track Daily Expense, Vol.12 No.6 (2021), UDAY PRATAP SINGH, AAKASH KUMAR GUPTA, DR B.BALAMURUGAN	YNAB and Penny AI	YNAB is an amount trackerthat automatically tracks our expenses through our bank account or credit card	Java (Apache Netbeans 11.3) and My SQL Workbench.	Have multi-language features. The main feature of this app is that you can track your expense by the mentioning date, month and year.
2	A Novel Expense Tracker using Statistical Analysis, June 2021| IJIRT | Volume 8 Issue 1 | ISSN: 2349-6002, MUSKAAN SHARMA , AYUSH BANSAL, DR. RAJU RANJAN , SHIVAM SETHI4	Statistical Analysis	In which using excel accounting team designed a Cost Allocation tool1 in which a spreadsheet is used to allocatethe product category both by site and the cooperation and a Cost allocation tool 2 which is a developed to further integrate and allocate cost to identify which manufactur	Excel,CAT tool, CSS and xml technologies.	Keep tracking daily expenses and budgeting; B) Save money for necessary expenses which in return will help toplan the future investments.

			er is profitable touse.		
3	Expense Tracker, MAY 2021 | IRE Journals | Volume 4Issue 11 | ISSN: 2456-8880, ATIYA KAZI , PRAPHULLA S. KHERADE , RAJ S.VILANKAR3, PARAG M. SAWANT,	Digital record System	Generates a monthly reportof the expensesin PDF format.	Angular 8 for front end and SQL Lite for backend.	Users are provided with three options for data entry namely Income, Expense and Wish List. The remainder isset if the type future expense.

4	A Review on Budget Estimator Android Application International Research Journal of Engineering and Technology (IRJET) NAMITA JAGTAP, PRIYANKA JOSHI, ADITYA KAMBLE	Google places AP, Haversine Algorithm	Do not required any GPU support to run this application, the major advantage is when we develop any application using angular ionic framework then it has capability to run on all platforms like android, Ios, windows.	Angular ionic framework.	Firebase Authentication to sign in a user by sending an SMS message to the user's phone. Send a verification code to the user's phone and verify it.
s5	Expense Tracker International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) Volume 9, Issue 4, September 2020. PROF MIRIAM THOMAS, LEKSHMI P , AND DR. MAHALEKSHMI T	Least Square Algorithm	The least squares method is a statistical procedure to find the best fit for a set of data points by minimizing the sum of the offsets or residuals of points from the plotted curve. Least squares regression is used to predict the behaviour of dependent variables	Net Beans for Java, PHP, HTML5 and JavaScript, Dreamweaver, MySQL(XAMPP)	In this application, there are 3 logins such as admin, manager and staff. Admin has the privilege to add, edit, delete manager, add, edit, delete staff, and to get all custom reports. For Manager, the privileges are to add type of expense, verify expense, add type of income, verify income and generate reports. For staff, the

					privileges are to add and edit expense, income and calculations, and send for verifications.
6	Student expense tracking application IJARIIE-ISSN(O)-2395-4396, Vol-8 Issue-2 2022. SAUMYA DUBEY, PRAGYA DUBEY, RIGVED RISHABH KUMAR	Digital record System and Statistical Analysis	This android application can be used on all android devices above android version 5.0.	Android studio for the front end and Firebase for the backend	The application size is less than 20 MB. It doesn't need any high-end hardware specifications.
7	Expense Tracker Application March 2021| IJIRT |Volume 7 Issue 10 | ISSN: 2349-6002 VELMURUGAN.R 1 , MRS.P.USHA	View Analytics	It also have added a special feature which will distributes your expenses in different categories suitable for the user. An expense history will also be provided in application.	Java, Xml, MySQL	The proposed system should provide different categories for the user to select from and they can enter the amount and mode of payment. This system should be able to analyze the information, provide analytics on which category did the user spent most of their money

8	Expense Tracker : ASmart Approach to Track Everyday Expense Easy Chair Preprint December 25, 2020 HRITHIK GUPTA, ANANT PRAKASH SINGH, NAVNEET KUMAR AND J. ANGELIN BLESSY	YNAB and Penny AI, The waterfall model	Have developed the necessary system to workwithout internet. We need a database, desktop, application anduser to use this system.	Java (Apache Netbeans 11.3) and My SQL Workbench 8.0CE	Expense Tracker application havefive entities: expense, user, backup, notification, transaction.
9	D2D Expense Tracker Application	YNAB and Penny Techni ques	D2D applications provide day byday updates tothe user regarding their expenses.	Android Studio, SQ Lite, Java JDK, Eclipse	Provide billing tothe user regardingtheir transactions and Expesnes.

2.3 Problem statement definition

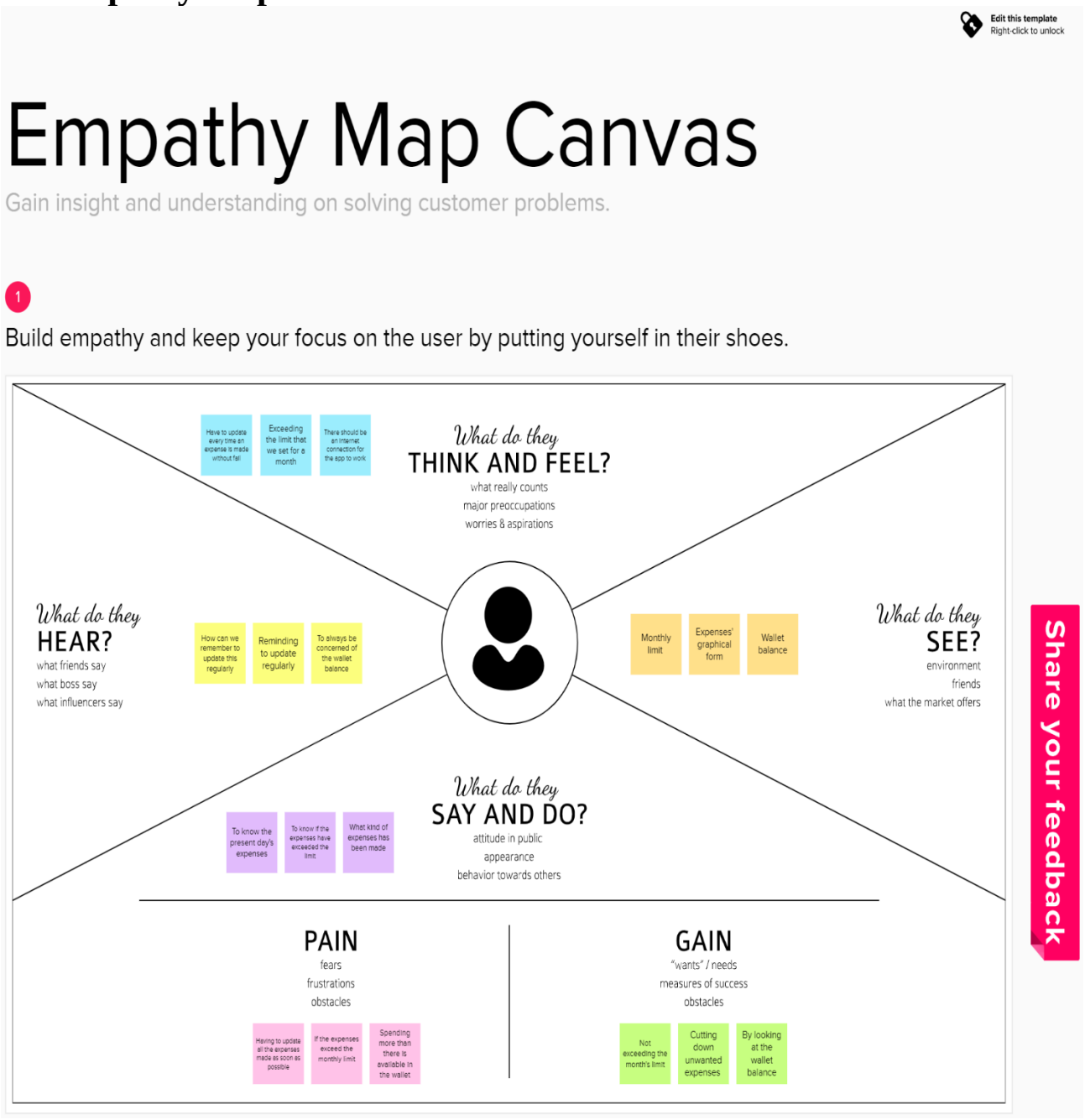
Sugan tried budgeting and failed miserably, using an expense tracker can solve your budget planning problems. Expense tracking isn't just about saving all of your receipts or writing down every cent you spend for the rest of eternity. It's actually much simpler than that. Tracking expenses is the difference between creating a budget that works, and one that doesn't. Whether you're about to start budgeting for the first time, or have been living on a budget for years, knowing where your money goes is the one thing that is guaranteed to make your budget work.

- Tracking expenses will build a budget that works.
- Monitor spending to make sure all monthly expenses are covered.
- Even with a successful budget, check-in to ensure spending plan is up to date.

3.IDEATION AND PROPOSED SOLUTION

CHAPTER 3
IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas:

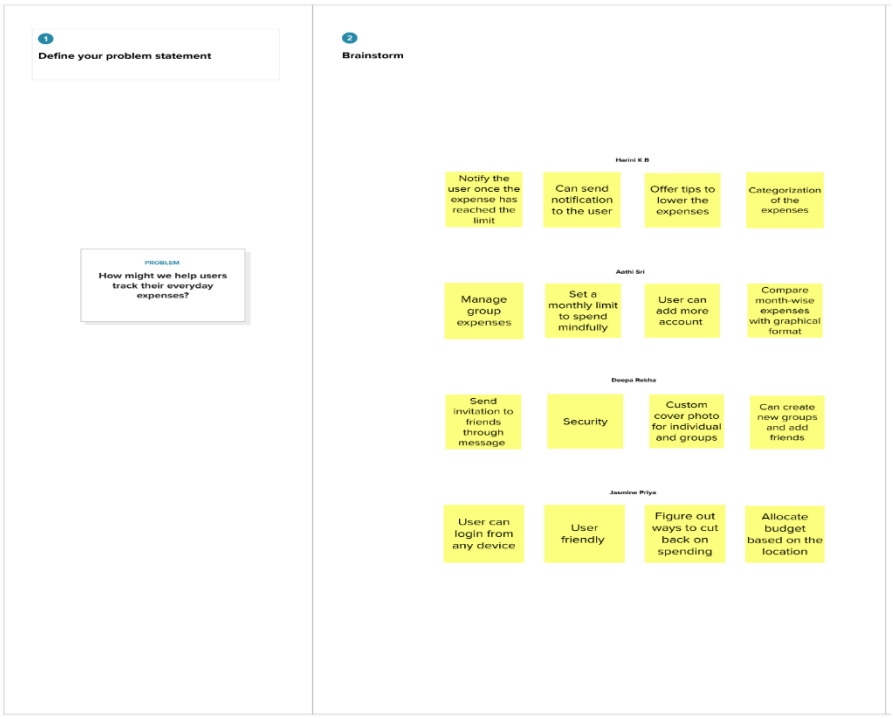


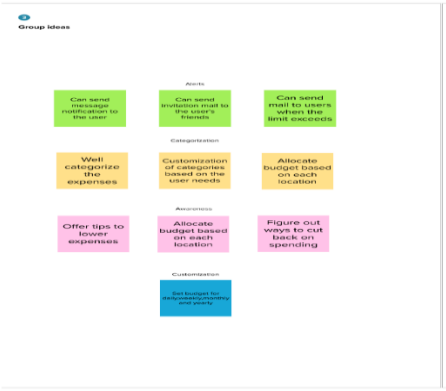
3.2 Ideation and Brainstorming:

IDEATION PHASE

BRAINSTORM AND IDEA PRIORITIZATION TABLE

DATE	19/09/2022
TEAM ID	PNT2022TMID11417
PROJECT NAME	Expense tracker application
MAXIMUM MARKS	2 MARKS





3.3 Proposed solution:

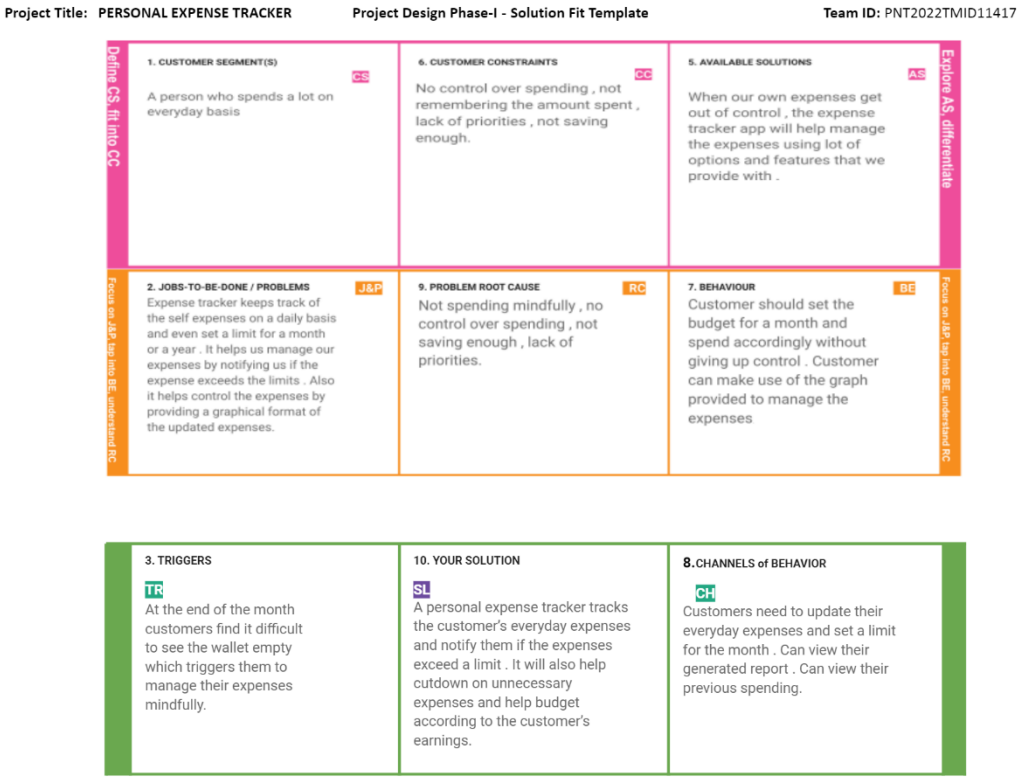
Project team shall fill the following information in proposed solution template.

S.No	Parameter	Description
1	Problem Statement (Problem to besolved)	Money management is a necessary part of life. A proper balance between income and expense is a must for a comfortable livelihood. There must be some savings that can be used at a later point of life when needed. But in the absence of proper management of money, we left with no savings at all. Some people note down every single expense which is a good practice.Expense Tracker to efficiently manage house-old budget. Our system will allow user to keep track of their expenses.

2	Idea / Solution description	To create personal expense tracker application which manage their income and expenditure details and gives better ideas to the user to reduce unnecessary expense and budgeting ideas. It will alert the user when he/she exceeds the budget limit. User can analyse his expense details and make a report of it.
3	Novelty / Uniqueness	This personal expense tracker Application has features that enables the user to have an option to set a limit for the amount to be used. If the limit is exceeded the user will be notified with an Email and SMS alert.
4	SocialImpact / customer Satisfaction	User can track their income and expense details. Itnotifies the user when budget limit is exceeded so that user can be aware in spending money
5	Business Model (Revenue Model)	This application can be made with free and premium If a user needs a ad-free access.

		environment they should have a premium subscription. So through this advertisement free subscription-based revenue model will earn moneywhich can be useful for the maintenance of the application.
6	Scalability of the Solution	This application has a better storage capacity to maintain large amount of data. This project is highly feasible and can update our income, expense and budget details whenever the user needs to change.

3.4 Problem solution fit:



4. EMOTIONS: BEFORE / AFTER EM Before: Spending mindlessly , a higher chance of going into debt . After: More financial security , stress free control.		

4. REQUIREMENT ANALYSIS

4.1 Functional requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Login	Registration through Gmail
FR-2	User Confirmation	Confirmation via Gmail
FR-3	User Password	Confirmation via comparing database details
FR-4	User Details for Wallet	Registration through Register Form
FR-5	User Details Confirmation	Confirmation via password
FR-6	Wallet update	Add income in the wallet
FR-7	Email Alert	Alert Message through Gmail if wallet amount exceeds

Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	User can tracker their expense in graphical manner Which helps to understand their expenditure in easy manner.
NFR-2	**Security**	We only store the information needed to save user from the trouble of syncing or updating financial information manually. Application also has a security feature that lets users set a password to access their account.
NFR-3	**Reliability**	The database update process can rollback to all related details in case of problem arise in updating
NFR-4	**Performance**	The application can perform well and fast
NFR-5	**Availability**	Wallet alert mail will be send to user when they exceed their limit. User can set limit to some specific categories like Education, Pharmacy, Groceries etc.

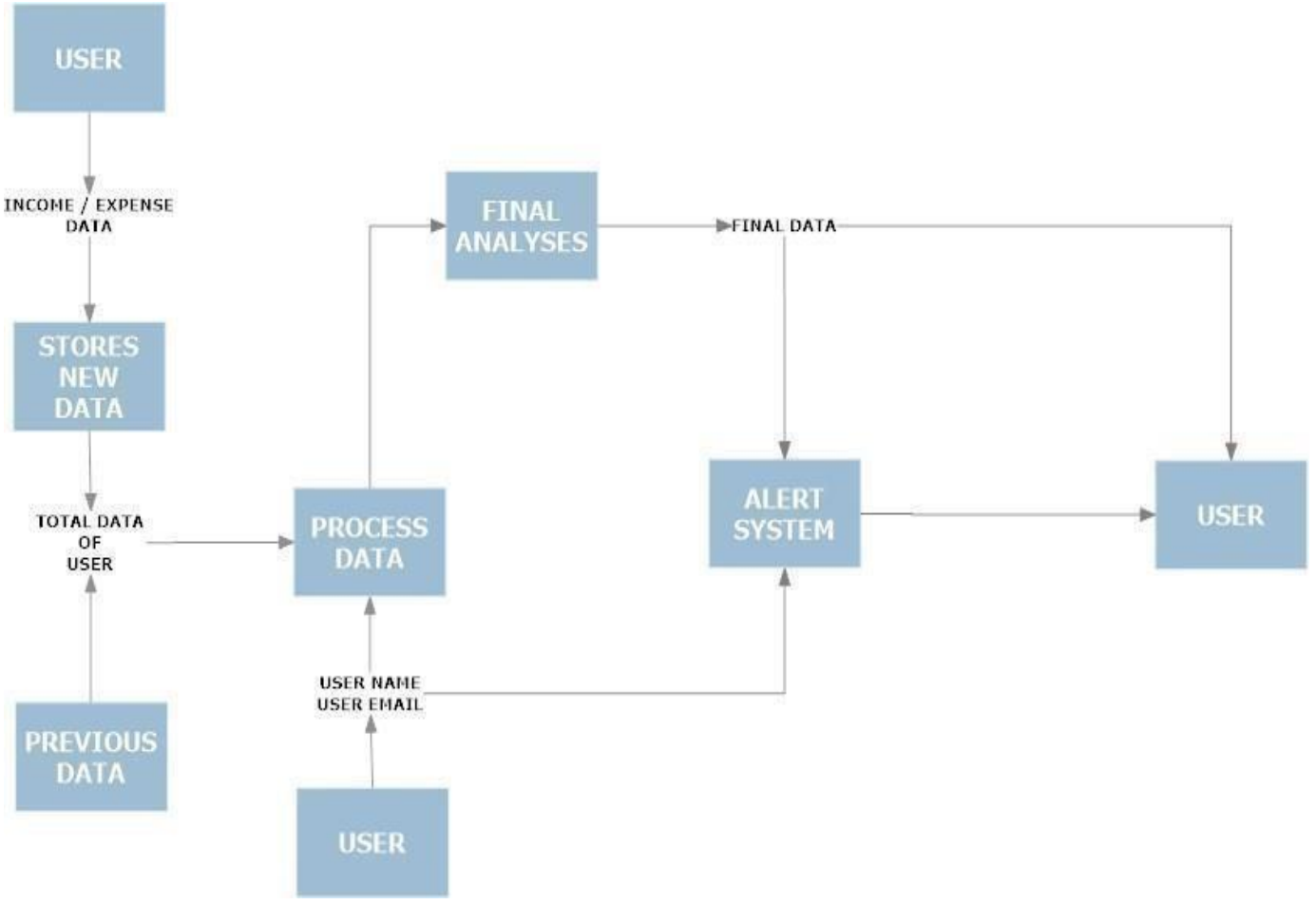
NFR 6	**Scalability**	This application can able to with stand many number of users

5.PROJECT DESIGN

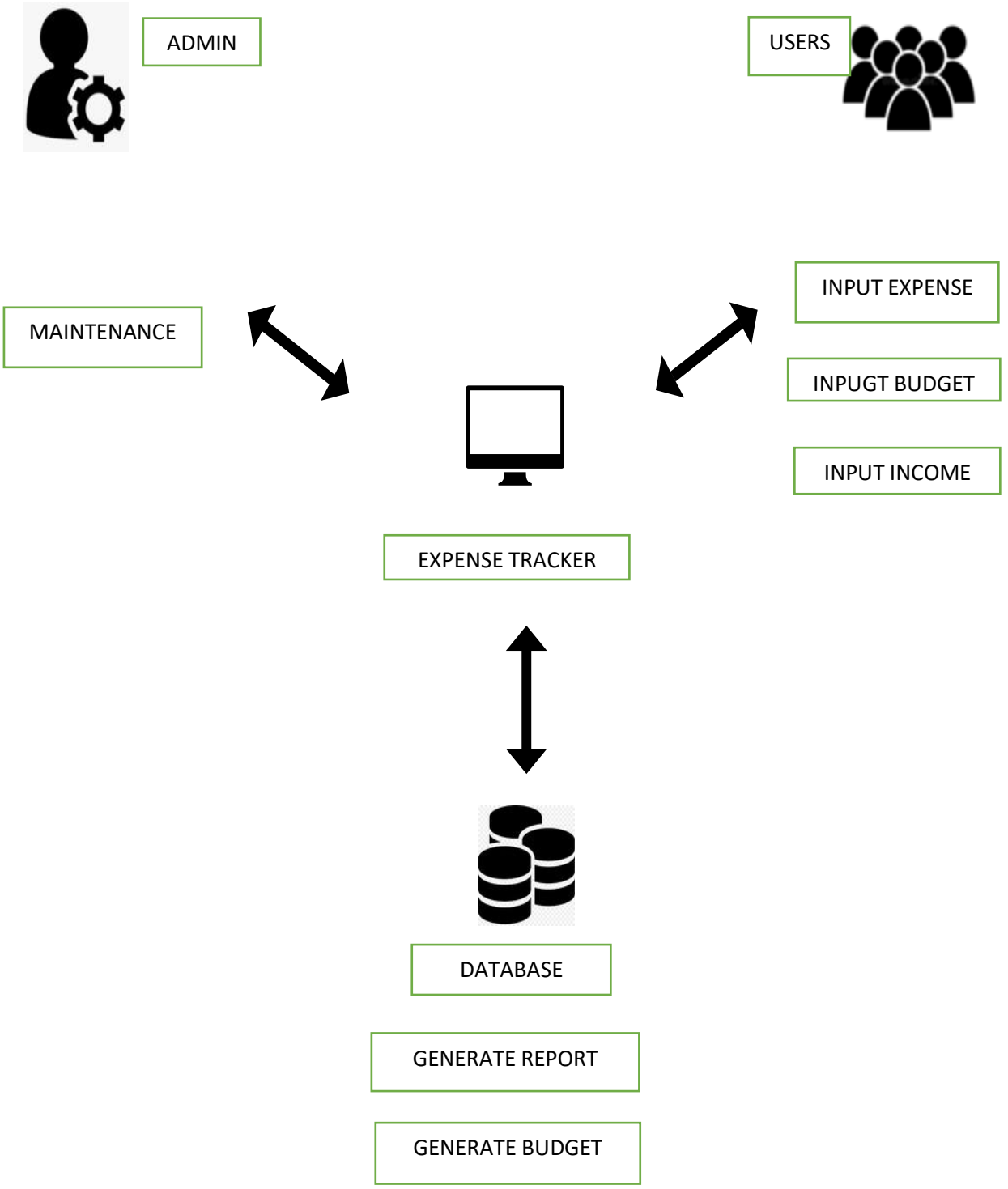
5.1 Data flow diagrams

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 rightamount of the system requirement graphically. It shows how data enters and leaves the system, whatchanges the information, and where data is stored



5.2 solution and technical architecture:



5.3 User stories:

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

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user &webuser)	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	
		USN-2	As a user, I will receive confirmation email after registeration on.	I can receive confirmationemail& click confirm	High	
		USN- 3	As a user, I can register for the application through Facebook	I can register & access the dashboard with FacebookLogin	Low	
	Login	USN - 4	As a user, I can log into the application by email & password	I can access the application	High	
	Dashboard	USN - 5	As a user I can enter my income and expenditure details.	I can view my monthly/weekly/daily expenses	High	
Customer Care Executive		USN – 6	As a customer care executive, I can solve the issues and other issues of the application.	I can provide support or solution at any time 24*7	Medium	

Administrator	Application	USN - 7	As an administrator I canupgrade or updatethe application.	I can fix the bug which arises for the customersand users of the application	Medium	

6.PROJECT PLANNING AND SCHEDULING

6.1 Sprint planning and scheduling:

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	Harini
		USN-2	As a user, I will receive confirmation email once I have registered for the application	2	High	Deeparekha
	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Aathi Sri

	Dashboard	USN-4	Logging in takes to the dashboard for the logged user.	`1	High	Jasmine Priya
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	Deeparekha
	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Aathi Sri
	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Jasmine Priya
		USN-4	Making dashboard interactive with JS	2	High	Harini
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Aathi Sri
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user's query	1	Medium	Jasmine Priya
	SendGrid	USN-3	Using SendGrid to send mail to the user about their expenses	1	Low	Harini
		USN-4	Integrating both frontend and backend	2	High	Deeparekha

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	Jasmine Priya
	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Harini
	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Deeparekha
	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Aathi Sri

6.2 Sprint delivery schedule

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.	3	High	Harini Aathi Sri
Sprint-1		USN-2	As a user, I will receive confirmation	3	High	Harini Aathi Sri

			email once I have registered for the application			
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	5	High	Harini Aathi Sri
Sprint-1	Dashboard & Logout	USN-4	As a user, once I logged in I can access all the features of the web app and Logout once I completed all the work.	5	High	Harini Aathi Sri
Sprint-1		USN-5	Once logged In, Keep me logged for few hours to avoid repeated login if the page is refreshed	4	Medium	Harini Aathi Sri
Sprint-2	Expense	USN-6	Add total income for the month and Allow for edit option	6	High	Deeparekha Jasmine Priya
Sprint-2		USN-7	Split the total income based on usage like entertainment, food, shopping etc.	2	Low	Deeparekha Jasmine Priya
Sprint-2		USN-8	Add the day to day expense.	6	High	Deeparekha Jasmine Priya
Sprint-2		USN-9	Display the user added expense	6	High	Deeparekha Jasmine Priya
Sprint-3		USN-10	Filter the expense data based on criteria	6	Medium	Harini Aathi Sri

Charts	USN-11	As a user I can display it in	4	Low	Harini Aathi Sri

		graphs			
Alerts	USN-12	As a user I create custom alert for the balance	10	High	Harini Aathi Sri
Deployment	USN-13	As a user I should able to access it anywhere in the net	20	High	Deeparekha Jasmine Priya

Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date (Planned)**	**Story Points Completed (as on Planned End Date)**	**Sprint Release Date (Actual)**
Sprint-1	20	3 Days	8 Nov2022	9 Nov 2022	20	9 Nov 2022
Sprint-2	20	3 Days	9 Nov 2022	10 Nov 2022	20	10 Nov 2022
Sprint-3	20	4 Days	10 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	12 Nov 2022	13 Nov 2022	20	13 Nov 2022

Velocity:

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let’s calculate the team’s average velocity (AV) per iteration unit (story points per day).

7.CODING AND SOLUTIONING

SAMPLE CODING

APP.PY

```
from
flaskimpo
rt Flask,
render_te
mplate,
request,
redirect,
session
,url_for
        import ibm_db
        import re
        import sendemail


        app = Flask(__name__)


        hostname = 'b0aebb68-94fa-46ec-a1fc-
        1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;'
        uid = 'zxj87173'
        pwd = 'Wt0IJeWtwN6irxqB'
        driver = "{IBM DB2 ODBC DRIVER}"
        db_name = 'Bludb'
        port = '31249'
        protocol = 'TCPIP'
        cert = "certi.crt"
        dsn = (
            "DATABASE ={0};"
            "HOSTNAME ={1};"
            "PORT ={2};"
            "UID ={3};"
            "SECURITY=SSL;"
            "PROTOCOL={4};"
            "PWD ={6};"
        ).format(db_name, hostname, port, uid, protocol, cert, pwd)
        connection = ibm_db.connect(dsn, "", "")
        app.secret_key = 'a'



        #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():
    global user_email
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        query = "SELECT * FROM register WHERE email=?;"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            query = "INSERT INTO register values(?,?,?);"
            stmt = ibm_db.prepare(connection, query)
            ibm_db.bind_param(stmt, 1, username)
```

```
            ibm_db.bind_param(stmt, 2, email)
            ibm_db.bind_param(stmt, 3, password)
            ibm_db.execute(stmt)
            session['loggedin'] = True
            session['id'] = email
            user_email = email
            session['email'] = email
            session['username'] = username


            msg = 'You have successfully registered ! Proceed Login Process'
            return render_template('login.html', msg = msg)
    else:
        msg = 'PLEASE FILL OUT OF THE FORM'
        return render_template('register.html', msg=msg)



 #LOGIN--PAGE

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

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

    if request.method == 'POST' :
        email = request.form['email']
        password = request.form['password']
        sql = "SELECT * FROM register WHERE email =? AND password=?;"
        stmt = ibm_db.prepare(connection, sql)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print (account)

        if account:
            session['loggedin'] = True
            session['id'] = account['EMAIL']
            user_email=  account['EMAIL']
            session['email']=account['EMAIL']
            session['username'] = account['USERNAME']

            return redirect('/home')
```

```
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)


#CHANGE FORGOT PASSWORD


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

@app.route("/forgotpw", methods =['GET', 'POST'])
def forgotpw():
    msg = ''
    if request.method == 'POST' :
        email = request.form['email']
        password = request.form['password']
        query = "SELECT * FROM register WHERE email=?;"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            query = "UPDATE register SET password = ? WHERE email = ?;"
            stmt = ibm_db.prepare(connection, query)
            ibm_db.bind_param(stmt, 1, password)
            ibm_db.bind_param(stmt, 2, email)
            ibm_db.execute(stmt)
            msg = 'Successfully changed your password ! Proceed Login Process'
            return render_template('login.html', msg = msg)
    else:
        msg = 'PLEASE FILL OUT THE CORRECT DETAILS'
        return render_template('forgot.html', msg=msg)




#ADDING----DATA



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

```
@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():
    global user_email
    que = "SELECT * FROM expenses where id = ? ORDER BY 'dates' DESC"
    stm = ibm_db.prepare(connection, que)
    ibm_db.bind_param(stm, 1, session['email'])
    ibm_db.execute(stm)
    dictionary=ibm_db.fetch_assoc(stm)
    expense=[]
    while dictionary != False:

exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],d
ictionary["PAYMODE"],dictionary["CATEGORY"])
        expense.append(exp)
        dictionary = ibm_db.fetch_assoc(stm)
    i=len(expense)+1
    idx=str(i)
    dates = request.form['date']
    expensename = request.form['expensename']
    amount = request.form['amount']
    paymode = request.form['paymode']
    category = request.form['category']
    query = "INSERT INTO expenses VALUES (?,?,?,?,?,?,?);"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, session['email'])
    ibm_db.bind_param(stmt, 2, dates)
    ibm_db.bind_param(stmt, 3, expensename)
    ibm_db.bind_param(stmt, 4, amount)
    ibm_db.bind_param(stmt, 5, paymode)
    ibm_db.bind_param(stmt, 6, category)
    ibm_db.bind_param(stmt, 7, idx)
    ibm_db.execute(stmt)
    print(dates + " " + expensename + " " + amount + " " + paymode + " " + category)

    return redirect("/display")
```

```
#DISPLAY---graph


@app.route("/display")
```

```
def display():
    query = "SELECT * FROM expenses where id = ? ;"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, session['email'])
    ibm_db.execute(stmt)
    dictionary=ibm_db.fetch_assoc(stmt)
    rexpense=[]
    while dictionary != False:

exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],d
ictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"])
        rexpense.append(exp)
        dictionary = ibm_db.fetch_assoc(stmt)
    que = "SELECT MONTH(dates) as DATES, SUM(amount) as AMOUNT FROM expenses WHERE
id=? AND YEAR(dates)= YEAR(now()) GROUP BY MONTH(dates);"
    stm = ibm_db.prepare(connection, que)
    ibm_db.bind_param(stm, 1,session['email'])
    ibm_db.execute(stm)
    dictionary=ibm_db.fetch_assoc(stm)
    texpense=[]
    while dictionary != False:
        exp=(dictionary["DATES"],dictionary["AMOUNT"])
        texpense.append(exp)
        dictionary = ibm_db.fetch_assoc(stm)
    print(texpense)

    quer = "SELECT * FROM expenses WHERE id = ? AND YEAR(dates)= YEAR(now());"
    st = ibm_db.prepare(connection, quer)
    ibm_db.bind_param(st, 1,session['email'])
    ibm_db.execute(st)
    dictionary=ibm_db.fetch_assoc(st)
    expense=[]
    while dictionary != False:

exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],d
ictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"])
        expense.append(exp)
        dictionary = ibm_db.fetch_assoc(st)

    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[3]
        if x[5] == "food":
            t_food += x[3]

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

        elif x[5] == "business":
            t_business  += x[3]
        elif x[5] == "rent":
            t_rent  += x[3]

        elif x[5] == "EMI":
            t_EMI  += x[3]

        elif x[5] == "other":
            t_other  += x[3]

    print(total)

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

    qur = "SELECT * FROM expenses WHERE id = ? AND MONTH(dates)= MONTH(now());"
    stt = ibm_db.prepare(connection, qur)
    ibm_db.bind_param(stt, 1, session['email'])
    ibm_db.execute(stt)
    dictionary=ibm_db.fetch_assoc(stt)
    lexpense=[]
    while dictionary != False:

exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],d
ictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"])
        lexpense.append(exp)
        dictionary = ibm_db.fetch_assoc(stt)


    ttotal=0
    to_food=0
    to_entertainment=0
    to_business=0
    to_rent=0
    to_EMI=0
```

```
to_other=0


for x in lexpense:
    ttotal += x[3]
    if x[5] == "food":
       to_food += x[3]

    elif x[5] == "entertainment":
       to_entertainment  += x[3]

    elif x[5] == "business":
       to_business  += x[3]
    elif x[5] == "rent":
       to_rent  += x[3]

    elif x[5] == "EMI":
       to_EMI  += x[3]

    elif x[5] == "other":
       to_other  += x[3]

print(ttotal)




qy = "SELECT max(IDX) as IDX FROM limits where id=?;"
smt = ibm_db.prepare(connection, qy)
ibm_db.bind_param(smt, 1, session['email'])
ibm_db.execute(smt)
dictionary = ibm_db.fetch_assoc(smt)
uexpense=[]
while dictionary != False:
   exp=(dictionary["IDX"])
   uexpense.append(exp)
   dictionary = ibm_db.fetch_assoc(smt)
k=uexpense[0]
qu = "SELECT NUMBER FROM limits where id=? and idx=?"
sm = ibm_db.prepare(connection, qu)
ibm_db.bind_param(sm, 1, session['email'])
ibm_db.bind_param(sm, 2, k)
ibm_db.execute(sm)
dictionary = ibm_db.fetch_assoc(sm)
fexpense=[]
while dictionary != False:
   exp=(dictionary["NUMBER"])
   fexpense.append(exp)
```

```
        dictionary = ibm_db.fetch_assoc(stmt)

    if len(fexpense) <= 0:
        print("Enter the limit First")
    else:
        if ttotal > fexpense[0]:
            m=sendemail.sendgridmail(session["email"])
            print(m)
        else: print("Error")
    return render_template("display.html",rexpense=rexpense, 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 )



#delete---the--data


@app.route('/delete/<idx>', methods = ['POST', 'GET' ])
def delete(idx):
    query = "DELETE FROM expenses WHERE id=? and idx=?;"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, session["email"])
    ibm_db.bind_param(stmt, 2, idx)
    ibm_db.execute(stmt)
    print('deleted successfully')
    return render_template("display.html")



#UPDATE---DATA


@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    query = "SELECT * FROM expenses WHERE id=? and idx=?;"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, session['email'])
    ibm_db.bind_param(stmt, 2, id)
    ibm_db.execute(stmt)
    dictionary=ibm_db.fetch_assoc(stmt)
    expense=[]
    while dictionary != False:

exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],d
ictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"])
```

```
        expense.append(exp)
        dictionary = ibm_db.fetch_assoc(stmt)
    print(expense)
    return render_template('edit.html', expenses = expense[0])
```

```
@app.route('/update/<id>', methods = ['POST'])
def update(id):
  if request.method == 'POST' :
      dates = request.form['date']
      expensename = request.form['expensename']
      amount = request.form['amount']
      paymode = request.form['paymode']
      category = request.form['category']
      query = "UPDATE expenses SET dates = ? , expensename = ? , amount = ?, paymode = ?, category
= ? WHERE id = ? and idx=?;"
      stmt = ibm_db.prepare(connection, query)
      ibm_db.bind_param(stmt, 1, dates)
      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, session['email'])
      ibm_db.bind_param(stmt, 7, id)
      ibm_db.execute(stmt)

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

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

```
@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
    que = "SELECT * FROM limits where id = ? ;"
    stm = ibm_db.prepare(connection, que)
    ibm_db.bind_param(stm, 1, session['email'])
    ibm_db.execute(stm)
    if request.method == "POST":
        dictionary=ibm_db.fetch_assoc(stm)
        expense=[]
        while dictionary != False:
            exp=(dictionary['ID'],dictionary['NUMBER'],dictionary['IDX'])
            expense.append(exp)
            dictionary = ibm_db.fetch_assoc(stm)
        i=len(expense)+1
        idx=str(i)
        number= request.form['number']
        query = "INSERT INTO limits VALUES(?,?,?)"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, session['email'])
        ibm_db.bind_param(stmt, 2, number)
        ibm_db.bind_param(stmt, 3, idx)
        ibm_db.execute(stmt)
        return redirect('/limitn')


@app.route("/limitn")
def limitn():
    query = "SELECT max(IDX) as IDX FROM limits where id=?;"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, session['email'])
    ibm_db.execute(stmt)
    dictionary = ibm_db.fetch_assoc(stmt)
    expense=[]
    while dictionary != False:
        exp=(dictionary["IDX"])
        expense.append(exp)
        dictionary = ibm_db.fetch_assoc(stmt)
    k=expense[0]
    que = "SELECT NUMBER FROM limits where id=? and idx=?"
    stmt = ibm_db.prepare(connection, que)
    ibm_db.bind_param(stmt, 1, session['email'])
    ibm_db.bind_param(stmt, 2, k)
    ibm_db.execute(stmt)
    dictionary = ibm_db.fetch_assoc(stmt)
    texpense=[]
    while dictionary != False:
        exp=(dictionary["NUMBER"])
```

```
        texpense.append(exp)
        dictionary = ibm_db.fetch_assoc(stmt)
    s=texpense[0]
    return render_template("limit.html" , y= s)



#REPORT


@app.route("/today")
def today():
    query = "SELECT dates, amount FROM expenses  WHERE id = ? AND DATE(dates) =
DATE(NOW()); "
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, str(session['email']))
    ibm_db.execute(stmt)
    dictionary=ibm_db.fetch_assoc(stmt)
    texpense=[]
    while dictionary != False:
      exp=(dictionary["DATES"],dictionary["AMOUNT"])
      texpense.append(exp)
      dictionary = ibm_db.fetch_assoc(stmt)
    print(texpense)

    query = "SELECT * FROM expenses WHERE id = ? AND DATE(dates) = DATE(NOW())"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, session['email'])
    ibm_db.execute(stmt)
    dictionary=ibm_db.fetch_assoc(stmt)
    expense=[]
    while dictionary != False:
        exp=(dictionary["AMOUNT"],dictionary["PAYMODE"],dictionary["CATEGORY"])
        expense.append(exp)
        dictionary = ibm_db.fetch_assoc(stmt)

    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[0]
       if x[2] == "food":
```

```
            t_food += x[0]

        elif x[2] == "entertainment":
            t_entertainment  += x[0]

        elif x[2] == "business":
            t_business  += x[0]
        elif x[2] == "rent":
            t_rent  += x[0]

        elif x[2] == "EMI":
            t_EMI  += x[0]

        elif x[2] == "other":
            t_other  += x[0]

    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():
    query = "SELECT dates, SUM(amount) as AMOUNT FROM expenses WHERE id= ? AND
MONTH(dates)= MONTH(now()) GROUP BY dates ORDER BY dates;"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, str(session['email']))
    ibm_db.execute(stmt)
    dictionary=ibm_db.fetch_assoc(stmt)
    texpense=[]
    while dictionary != False:
     exp=(dictionary["DATES"],dictionary["AMOUNT"])
     texpense.append(exp)
```

```
  dictionary = ibm_db.fetch_assoc(stmt)
print(texpense)

query = "SELECT * FROM expenses WHERE id = ? AND MONTH(dates)= MONTH(now());"
stmt = ibm_db.prepare(connection, query)
ibm_db.bind_param(stmt, 1, session['email'])
ibm_db.execute(stmt)
dictionary=ibm_db.fetch_assoc(stmt)
expense=[]
while dictionary != False:

exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],d
ictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"])
  expense.append(exp)
  dictionary = ibm_db.fetch_assoc(stmt)

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[3]
   if x[5] == "food":
      t_food += x[3]

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

   elif x[5] == "business":
      t_business  += x[3]
   elif x[5] == "rent":
      t_rent  += x[3]

   elif x[5] == "EMI":
      t_EMI  += x[3]

   elif x[5] == "other":
      t_other  += x[3]

print(total)

print(t_food)
print(t_entertainment)
```

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




    return render_template("month.html", texpense = 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():
    query = "SELECT MONTH(dates) as DATES, SUM(amount) as AMOUNT FROM expenses
WHERE id=? AND YEAR(dates)= YEAR(now()) GROUP BY MONTH(dates);"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1,session['email'])
    ibm_db.execute(stmt)
    dictionary=ibm_db.fetch_assoc(stmt)
    texpense=[]
    while dictionary != False:
     exp=(dictionary["DATES"],dictionary["AMOUNT"])
     texpense.append(exp)
     dictionary = ibm_db.fetch_assoc(stmt)
    print(texpense)

    query = "SELECT * FROM expenses WHERE id = ? AND YEAR(dates)= YEAR(now());"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1,session['email'])
    ibm_db.execute(stmt)
    dictionary=ibm_db.fetch_assoc(stmt)
    expense=[]
    while dictionary != False:

exp=(dictionary["ID"],dictionary["DATES"],dictionary["EXPENSENAME"],dictionary["AMOUNT"],d
ictionary["PAYMODE"],dictionary["CATEGORY"],dictionary["IDX"])
      expense.append(exp)
      dictionary = ibm_db.fetch_assoc(stmt)

    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[3]
        if x[5] == "food":
            t_food += x[3]

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

        elif x[5] == "business":
            t_business  += x[3]
        elif x[5] == "rent":
            t_rent  += x[3]

        elif x[5] == "EMI":
            t_EMI  += x[3]

        elif x[5] == "other":
            t_other  += x[3]

    print(total)

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




    return render_template("year.html", texpense = 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)
    return render_template('home.html')




if __name__ == "__main__":
    app.run(debug=True)
```

8.CONCLUSION