NALAYATHIRAN PROJECT UNDER NAAN MUDHALVAN SCHEME

BY THE INITIATIVE OF THE

GOVERNMENT OF TAMILNADU

SUBMITTED BY

COLLEGE NAME	: C. Abdul Hakeem College of Engineering and technology
TEAM ID	: PNT2022TMID39680
PROJECT DOMAIN	: CLOUD APPLICATION DEVELOPMENT
PROJECT NAME	: PERSONAL EXPENSE TRACKER

TEAM LEADER	: SUDHARSAN K	(510619205040)
TEAM MEMBERS	: TAMIL SELVAN R	(510619205044)
	PRAVEEN KUMAR S	(510619205025)
	SURENDAR S	(510619205041)

INTRODUCTION

PROJECT OVERVIEW :-

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 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.

PURPOSE :-

Personal Expense Tracker Application is designed for the people who are unaware of their daily expenses. It is planned to solve the issues like poor financial management, unnecessary expenditure, no proper savings, etc. People will be aware of their daily expenses which will lead to the proper savings. We can reduce the unwanted daily expenses, save money in a better manner, daily expenses can be tracked and it will alert us when we are in a shortage of money. It reduces man power for accounting and financing the money, human errors can also be eradicated.

LITERATURE SURVEY

PERSONAL EXPENSE TRACKER APPLICATION

I. INTRODUCTION

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 named as Expense to store and calculate these data. Expense is an application that runs on Android smartphones. 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.

From the beginning of human civilization, people have exchanged their fortune with each other for buying or selling goods. It has become a crucial and unchangeable part of our daily life since then. Most of us have a fixed income and we get it in a timely basis (i.e. daily, monthly, yearly etc.). Moreover, everyone follows a strict budget of expense. Generally, the budget is assembled as per category. The categories are distinct, for example, food, entertainment, transportation, education, healthcare, clothing etc. However, the budget of expense is restricted to the income. For that reason, we need to track our expense so that it doesn‟t exceed our budget. In old days, people used to track their expense manually i.e., using pen and paper system which takes a lot of effort and time. Nowadays, the availability of electronic devices like smartphones, computers have made our life a lot easier and faster. We can use computers to track our daily expense by using the online and offline software available. But the computer is not accessible all the time. The smart solution to the problem is to use smartphones.

II. INTERFACE DESIGN

Our application is designed by following the Google Design Pattern rules . So that the users can use it with ease. The interface is simple and user-friendly. The contents of our application are divided into four major sections. Those are Debit, Credit, Balance, and History. The debits and credits are shown in different lists. The last entry is always on top of the list and the previous one is just below the last entry and so on. It allows users to access their latest entries first. Also, there is a search option for searching debit or credit entries. There is a floating action button on debit and credit interface which is used to add new records. Therefore, new debit or credit entries can simply be added by tapping the floating add button. In the balance section, there are different pie charts showing the proportion of debits and credits . The history section is represented by a calendar view where debit and credit entries of a particular date can be seen by tapping on a date.

III. PROPOSED SYSTEM

Our proposed solution is a smart assistant for the users to track daily expense. The users simply need to scan receipts of any kind with their smartphone camera and our application is smart enough to detect the necessary words and amount from the receipt. The data of the expenditure is then saved on the database category wise. Our application structure has been divided into two major parts. One is debit and another is credit. All the expenditures are included in the debit part and the incomes are included in the credit part. The incomes are calculated automatically from the messages received in the inbox. There are some additional features available in our system. The users can set a total budget for the whole month as well as for individual categories. The system will notify the users if they attempt to exceed the set budget. There is a pie chart (see figure 5) representation to summarize the total expense. A calendar view (see figure 6) is also available to show the expense histories of the users. Our approach solves many issues and limitations of the currently available expense tracking systems in the market. It saves a lot of time and efforts of the users as the major processing is automatically done by the application.

IV. SYSTEM OVERVIEW

Expense is divided into four major sections. Those are Debit, Credit, Balance, and history. This system works as a one tap solution for tracking everyday expense. It also preserves yearly and monthly records. For the availability of the records, users can check their histories to keep track of their expense so that they do not exceed t

V. Credit:

The credit information can be stored using the mechanism . When users open the credit interface, the system reads all the messages from the user‟s messaging inbox. If the messages are from the bank about any transaction of credit, the app reads it and saves necessary information from the messages (SMS). There is also another option for saving credit. The users can manually give the input of credit in the app. After successfully saving the input, they can view the data on the list view. Figure 4 shows the credit input technique.

VI. Balance:

In balance interface, users can see two different types of pie charts, which are yearly and monthly balance. The pie charts represent the total estimation from all the categories (e.g., food, entertainment, transportation, education, healthcare, clothing, bank, groceries). Yearly balance includes the credit and debit information of a specific year. Monthly balance includes all the credit and debit information of 6 months.

VII. CONCLUSION

In today‟s world, time is the most valuable asset because people lack ample of it. People are obsessed with completing tasks in lesser time and our system is an approach serving this purpose. Expense can manage daily expense much faster than any other traditional app in the market which takes manual input. Our system proves to be most effective for the people aged 40 and over and an efficient solution comparing to any of the traditional applications. Nowadays, the world is leaning towards the one tap solution and our system is one of a kind. After all, automation is the way of future and Expense can be a step towards it. The application still has a lot of aspects that need to be improved. The app performs poorly if the input image from the receipt includes a lot of noises. The app is unable to detect the region of interest automatically. On that note, the user has to set the region of the receipt to have a better result. Again, the performance of the character recognition from the receipt declines in low lights. Therefore, this system triggers few research scopes which can be a starting point for the improvement of the proposed approach.

VIII. REFERENCE

[1] "44% of World Population will Own Smartphones in 2017." [Online]. Available: https://www.strategyanalytics.com/strategyanalytics/blogs/smart-phones/2016/12/21/44-of- world-populationwill-own-smartphones-in-2017#.Wq1BcehuY2y.

[2] "Daily Expense 3 - Apps on Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=mic.app.gastosdiarios.

[3] "Monefy - Money Manager - Apps on Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.monefy.app.lite.

[4] "Money Lover: Budget Planner, Expense Tracker - Apps on Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.bookmark.money .

[5] "AndroMoney (Expense Track) - Apps on Google Play." [Online].

[6] https://ieeexplore.ieee.org/

PROBLEM STATEMENT

DOMAIN : CLOUD APPLICATION DEVELOPMENT

PROJECT : PERSONAL EXPENSE TRACKER APPLICATION

WHO DOES THE PROBLEM AFFECT?

People who are unaware of their daily expenses

WHAT IS THE ISSUE?

***** Poor Financial Management

* Unnecessary Expenditure

* Not Proper Savings

What is the Impact of the Issue?
People will be unaware of their daily expenses which can lead to unnecessary expenditure and not proper savings can be done.

What would happen if we didn't solve the problem?
* Financial crisis may occur

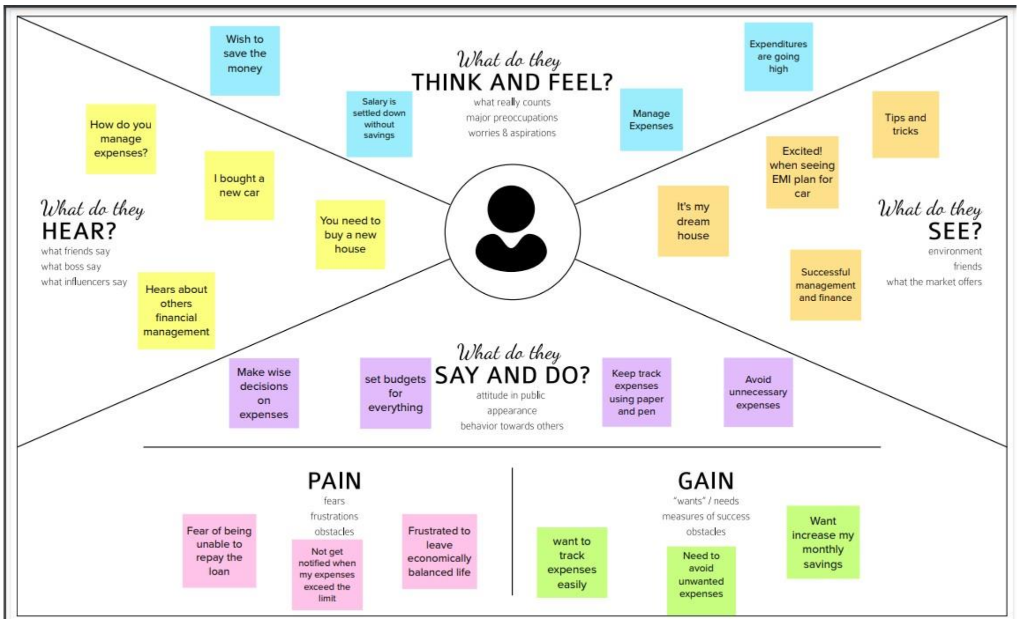
What would happen when it is fixed?
we can reduce unwanted daily expenses, we can save money in a better manner, daily expenses can be tracked, alerts when we are shortage of money.

WHY IT IS IMPORTANT THAT WE FIX THE PROBLEM?

Proper savings can be done and we can reduce the daily expenses and the application gives proper analysis of our expenditure and income. It reduces man power for accounting and financing the money, human errors can be eradicated.

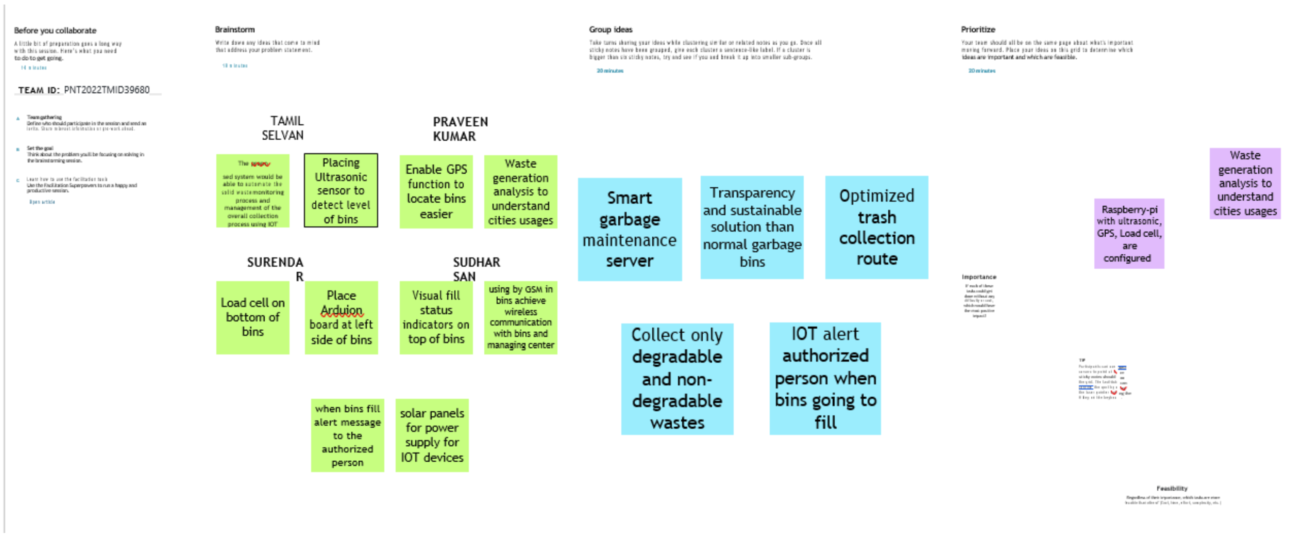
IDEATION AND PROPOSED SOLUTION

EMPATHY MAP LINK :


What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Wish to save the money
Salary is settled down without savings
Manage Expenses
Expenditures are going high
What do they HEAR?
what friends say
what boss say
what influencers say
How do you manage expenses?
I bought a new car
You need to buy a new house
Hears about others financial management
What do they SEE?
environment
friends
what the market offers
Tips and tricks
Excited! when seeing EMI plan for car
It's my dream house
Successful management and finance
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
Make wise decisions on expenses
set budgets for everything
Keep track expenses using paper and pen
Avoid unnecessary expenses
PAIN
fears
frustrations
obstacles
Fear of being unable to repay the loan
Not get notified when my expenses exceed the limit
Frustrated to leave economically balanced life
GAIN
"wants" / needs
measures of success
obstacles
want to track expenses easily
Need to avoid unwanted expenses
Want increase my monthly savings

IDEATION AND BRAINSTROMING

BRAINSTROM :


Before you collaborate
A little bit of preparation goes a long way with this session. Here's what you need to do to get going.
TEAM ID: PNT2022TMID39680
Team gathering
Define who should participate in the session and send an invite.
Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.
Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.
Open article
Brainstorm
Write down any ideas that come to mind that address your problem statement.
TAMIL SELVAN
The proposed system would be able to automate the solid waste monitoring process and management of the overall collection process using IOT
Placing Ultrasonic sensor to detect level of bins
PRAVEEN KUMAR
Enable GPS function to locate bins easier
Waste generation analysis to understand cities usages
SURENDAR
Load cell on bottom of bins
Place Arduion board at left side of bins
SUDHARSAN
Visual fill status indicators on top of bins
using by GSM in bins achieve wireless communication with bins and managing center
when bins fill alert message to the authorized person
solar panels for power supply for IOT devices
Group ideas
Smart garbage maintenance server
Transparency and sustainable solution than normal garbage bins
Optimized trash collection route
Collect only degradable and non-degradable wastes
IOT alert authorized person when bins going to fill
Prioritize
Importance
Feasibility
Raspberry-pi with ultrasonic, GPS, Load cell, are configured
Waste generation analysis to understand cities usages

PROBLEM SOLUTION FIT :

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 2 constant overload to rely on the daily entry of the expenditure and total estimation till the end of

PROPOSED SOLUTION

Problem Statement (Problem to be solved) :

People who are unaware of their daily expenses

Idea / Solution description :

Proper savings can be done and we can reduce the daily expenses and the application gives proper analyses of our expenditure and income. It reduces man power for accounting and financing the money, human errors can be eradicated.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	People who are unaware of their daily expenses.
2.	Idea / Solution description	Proper savings can be done and we can reduce the daily expenses and the application gives proper analyses of our expenditure and income. It reduces man power for accounting and financing the money, human errors can be eradicated.
3.	Novelty / Uniqueness	*Expense alerts *Graphical representation *Categories each of your transactions
4.	Social Impact / Customer Satisfaction	*User Friendly Interface *Flexibility of adding day to day expenses
5.	Business Model (Revenue Model)	Achieve your business goals with a tailored web app that perfectly fits your business
6.	Scalability of the Solution	*Access Anywhere Anytime *Accurate Tracking *History of expenses

REQUIREMENT ANALYSIS

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Access / Login	Login through e-mail ID & Password
FR-4	User Data	Adding user expenses through input field & categories
FR-5	User Alert	Alerting user through registered e-mail ID

Non-functional Requirements:

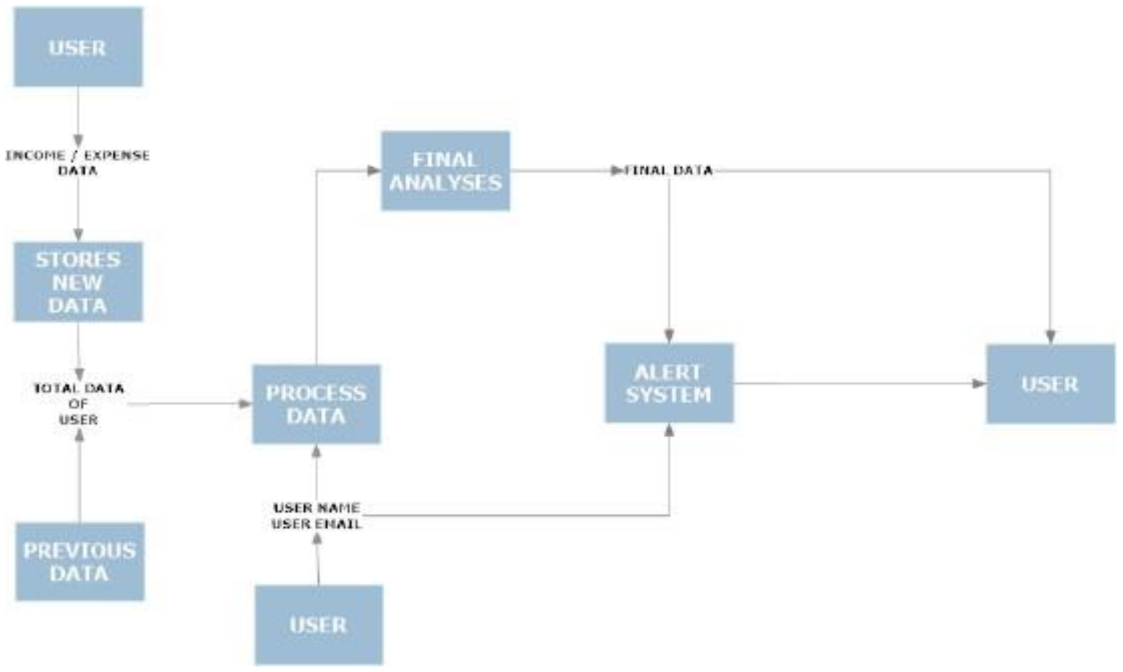
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	User friendly interface
NFR-2	**Security**	Strong security system
NFR-3	**Reliability**	Highly reliable for the old age people to track the expenses
NFR-4	**Performance**	Low data usage, instant email alerts while exceeding limits.
NFR-5	**Availability**	Available for all platforms (Mobile User, Web User)
NFR-6	**Scalability**	Access Anywhere Anytime

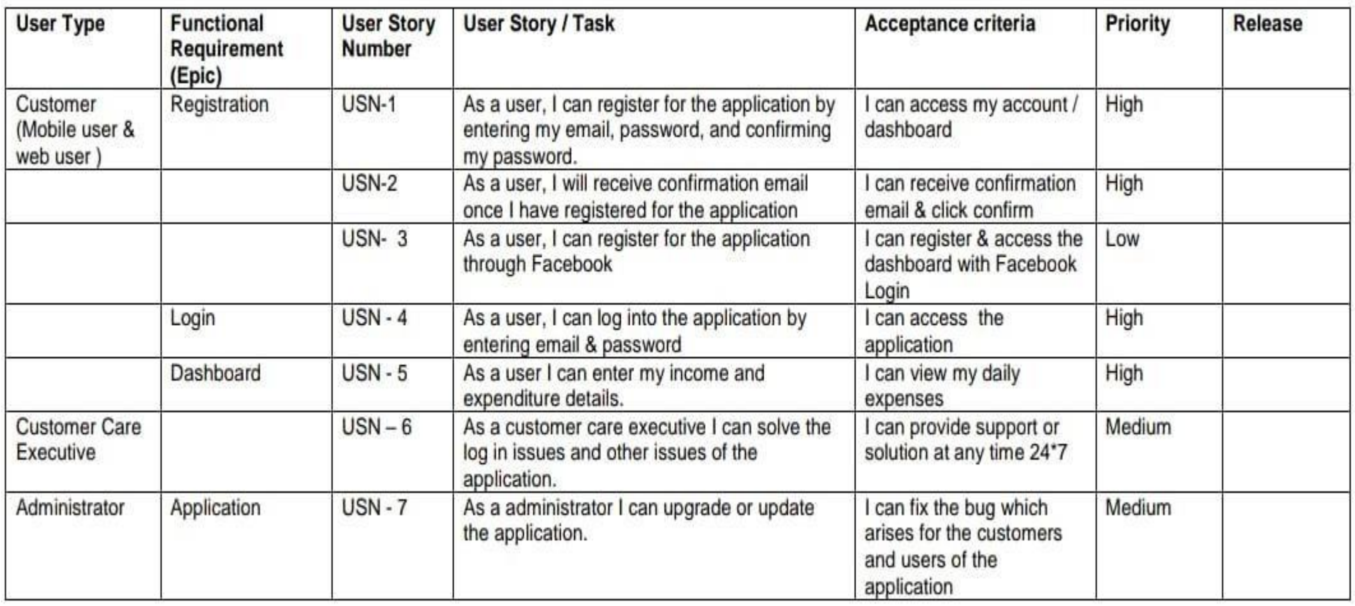
PROJECT DESIGN

DATA FLOW DIAGRAMS :

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



USER STORIES :



User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user & web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	
		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	
		USN- 3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	
	Login	USN - 4	As a user, I can log into the application by entering 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 daily expenses	High	
Customer Care Executive		USN – 6	As a customer care executive I can solve the log in issues and other issues of the application.	I can provide support or solution at any time 24*7	Medium	
Administrator	Application	USN - 7	As a administrator I can upgrade or update the application.	I can fix the bug which arises for the customers and users of the application	Medium	

TECHNOLOGY ARCHITECTURE :

Technical Architecture:

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

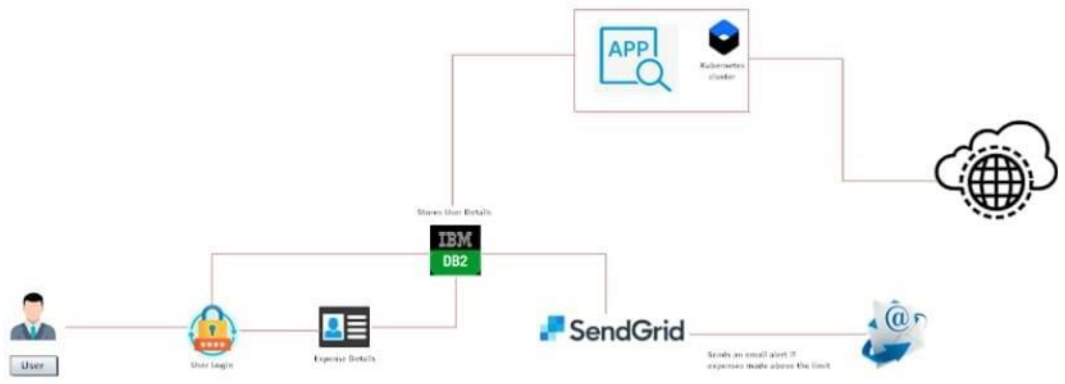


Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Flask	Python
2.	Security Implementations	SSL Certificate	SHA-256, Encryptions
3.	Scalable Architecture	3 – tier Architecture, Micro-services	Kubernetes with Docker
4.	Availability	load balancers, distributed servers	IBM Cloud
5.	Performance	15 connections at a time, 200Mb of Data Storage	IBM Db2

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Web UI	HTML, CSS, JavaScript
2.	Application Logic-1	Backend Process – Web page route, Db Connection	Python Flask
3.	Application Logic-2	Docker containers	Docker
4.	Cloud Database	Database Service on Cloud	IBM DB2
5.	File Storage	File storage requirements	IBM Object Storage
6.	External API-1	Sending email alert	Sendgrid
7.	Infrastructure (Server / Cloud)	Application Deployment on Cloud Cloud Server Configuration : 1vCPU, 1 Cluster, 25 GB Object Storage	IBM Kubernetes Cluster

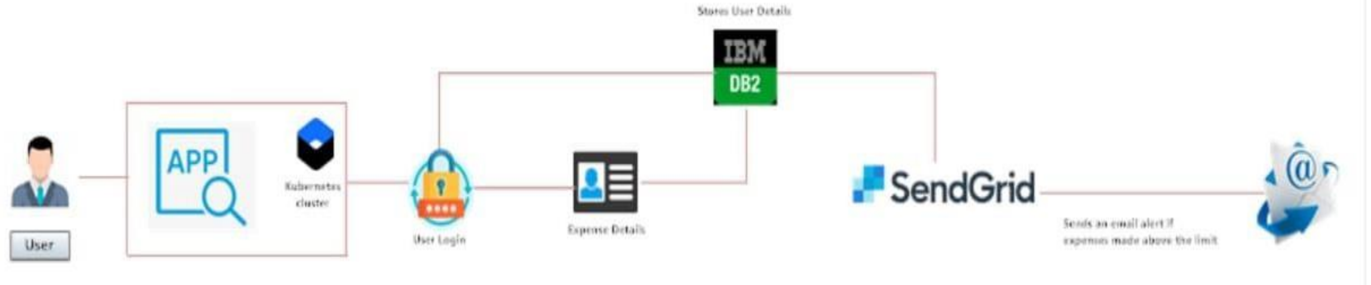
SOLUTION ARCHITECTURE :

Solution Architecture:

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

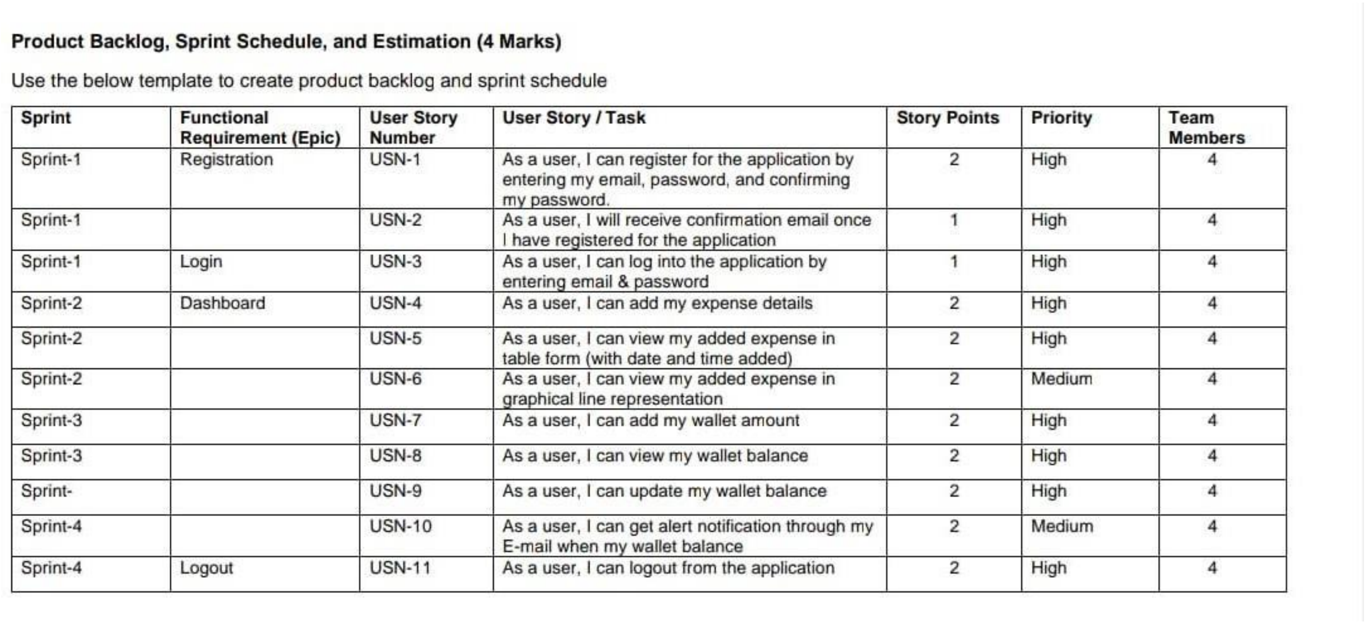
- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution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

SOLUTION ARCHITECTURE DIAGRAM



PROJECT PLANNING AND SCHEDULING

SPRINT PLANNING AND ESTIMATION :

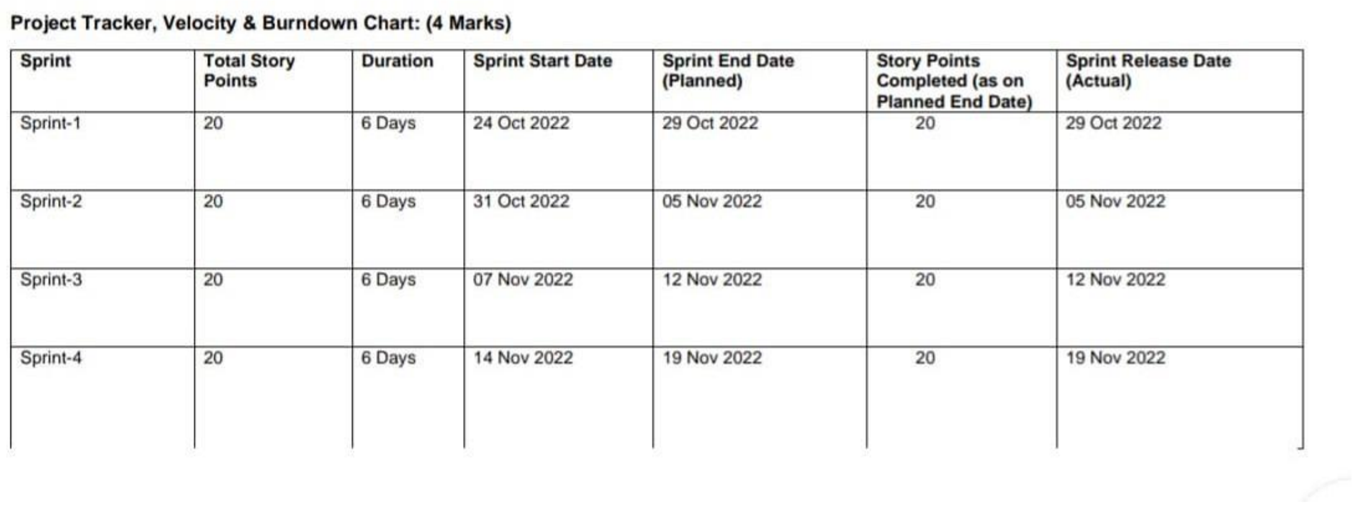


Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint 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.	2	High	4
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	4
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	4
Sprint-2	Dashboard	USN-4	As a user, I can add my expense details	2	High	4
Sprint-2		USN-5	As a user, I can view my added expense in table form (with date and time added)	2	High	4
Sprint-2		USN-6	As a user, I can view my added expense in graphical line representation	2	Medium	4
Sprint-3		USN-7	As a user, I can add my wallet amount	2	High	4
Sprint-3		USN-8	As a user, I can view my wallet balance	2	High	4
Sprint-		USN-9	As a user, I can update my wallet balance	2	High	4
Sprint-4		USN-10	As a user, I can get alert notification through my E-mail when my wallet balance	2	Medium	4
Sprint-4	Logout	USN-11	As a user, I can logout from the application	2	High	4

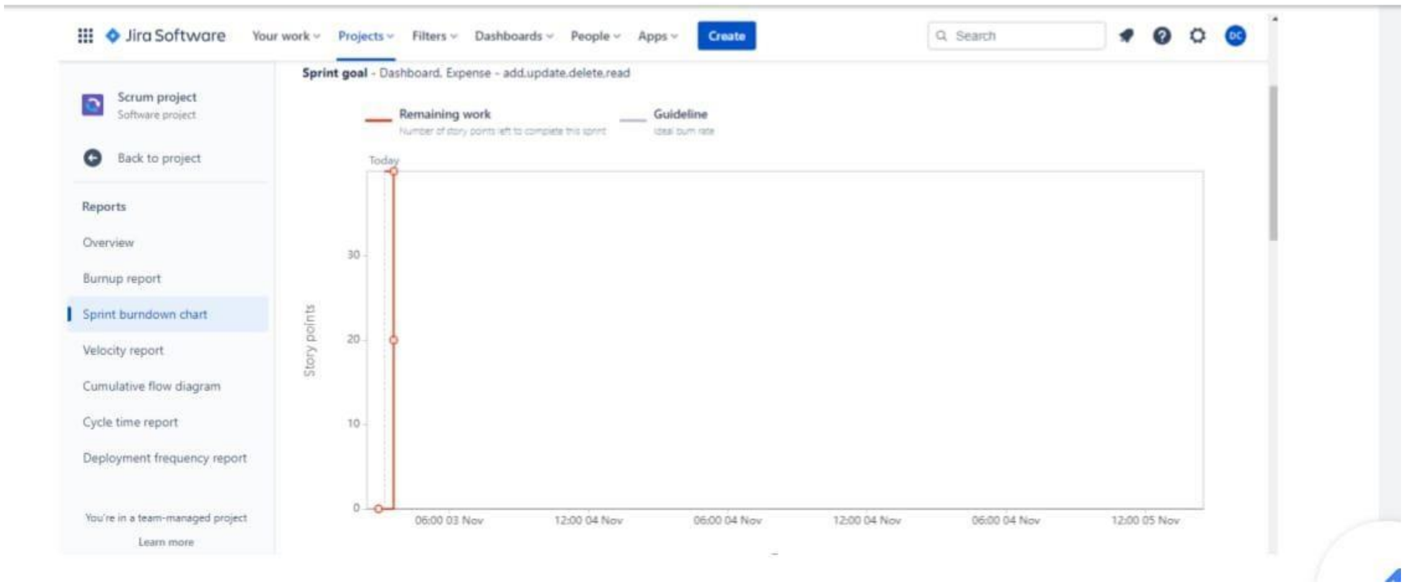
SPRINT DELIVERY SCHEDULE :



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

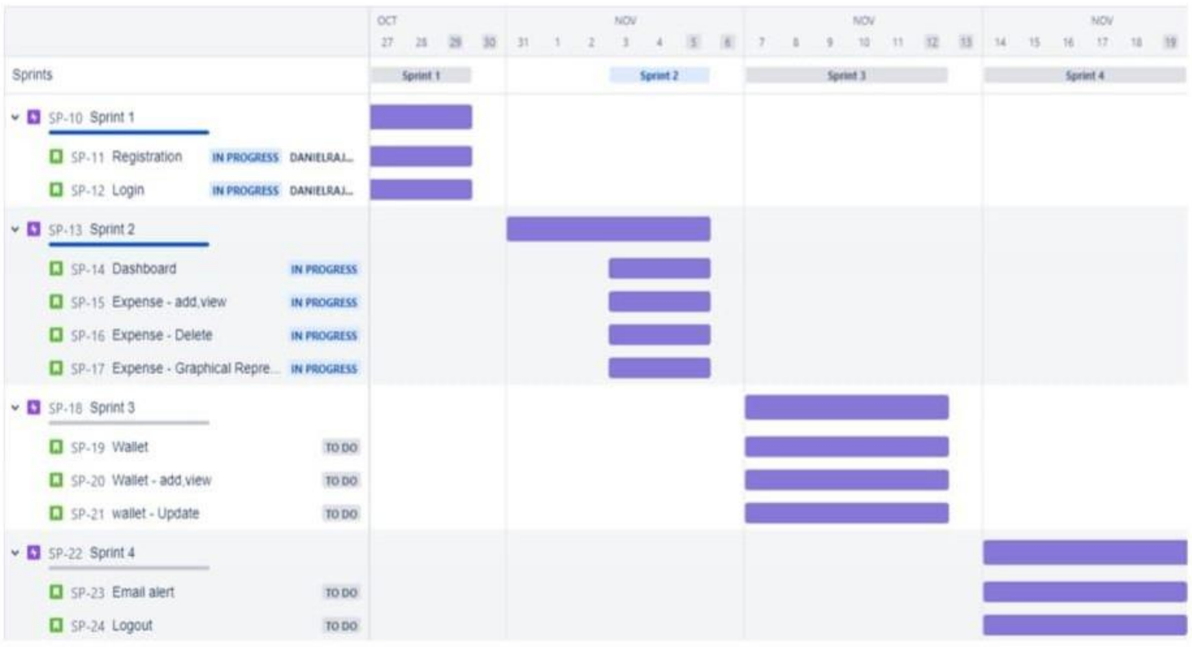
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

REPORTS FROM JIRA :



Sprint goal - Dashboard, Expense - add,update,delete,read

Roadmap:



Sprints		
SP-10 Sprint 1		
SP-11 Registration	IN PROGRESS	DANIELRAJ...
SP-12 Login	IN PROGRESS	DANIELRAJ...
SP-13 Sprint 2		
SP-14 Dashboard	IN PROGRESS	
SP-15 Expense - add,view	IN PROGRESS	
SP-16 Expense - Delete	IN PROGRESS	
SP-17 Expense - Graphical Repre...	IN PROGRESS	
SP-18 Sprint 3		
SP-19 Wallet	TO DO	
SP-20 Wallet - add,view	TO DO	
SP-21 wallet - Update	TO DO	
SP-22 Sprint 4		
SP-23 Email alert	TO DO	
SP-24 Logout	TO DO	

CODING & SOLUTIONING

CODE LINK : Github Link

FEATURES :

- User can able to signup with their mail ID and can create an account
- User will receive an E-mail on successful registration
- User can Login with their credentials and can access the dashboard
- User can now add money to the wallet linked to their account
- User can add expenses with the detail / category
- User can delete their expense if they have added accidentially
- User can able to see their expenses in table form with complete detail and time added
- User can see the total expense amount below the table
- As the user add the expense amount the app automatically deducts the expense amount from their wallet
- If the amount goes below certain range the app automatically sends alert E-mail to the user‟s email ID with wallet remaining balance amount
- User can update their wallet amount anytime
- User can see their expense details in graphical form
- User gets Popup notification for every change (Login, Signup, adding balance, adding / deleting expense, updating balance)
- The application can run in any devices such as Mobile, desktop, tablet etc…. (Responsive Design)

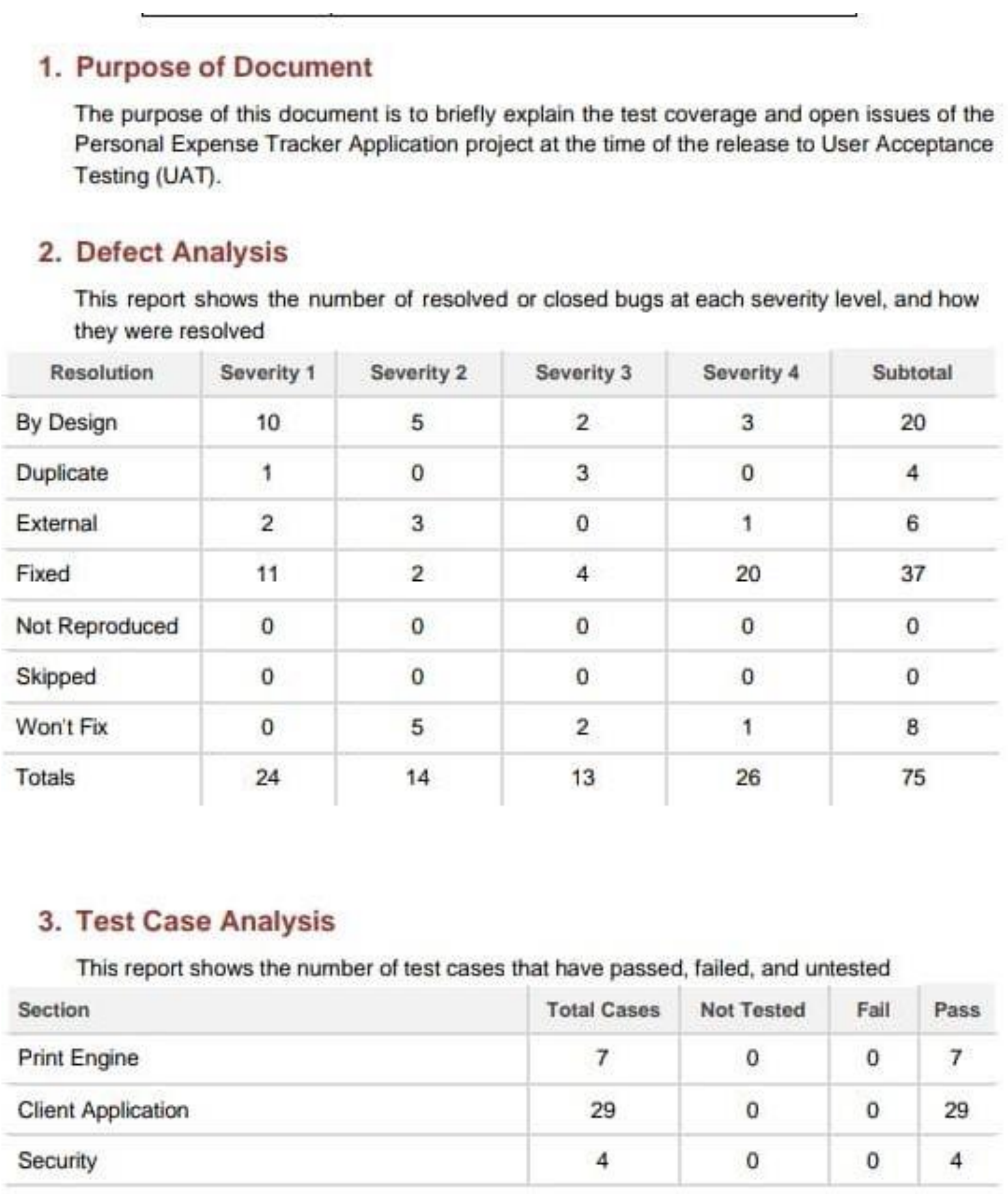
- The application is deployed in IBM Cloud server, Database used – IBM DB2
- Application has Load balancing feature (nGinx) – As many users can use the application without any reduction in speed
- User can able to logout of their application after completing their work

TESTING

1. Test cases

Link: Test Case

USER ACCEPTANCE TESTING REPORT



1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Personal Expense Tracker Application project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

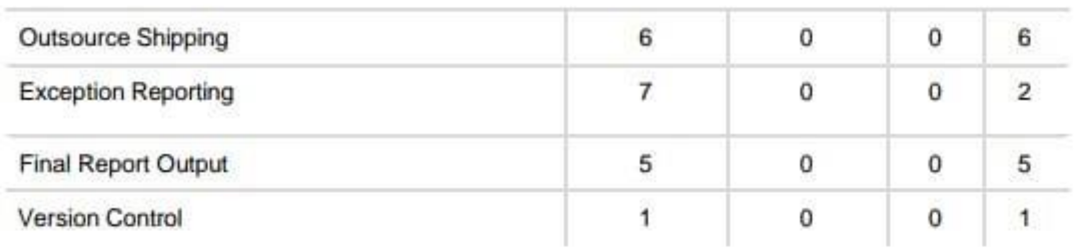
This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	5	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	75

3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	29	0	0	29
Security	4	0	0	4



Outsource Shipping	6	0	0	6
Exception Reporting	7	0	0	2
Final Report Output	5	0	0	5
Version Control	1	0	0	1

RESULTS

Performance Metrics : Link

ADVANTAGES & DISADVANTAGES

ADVANTAGES :-

1. Achieve your business goals with a tailored mobile app that perfectly fits your business.
2. Scale-up at the pace your business is growing.
3. Deliver an outstanding customer experience through additional control over the app.
4. Control the security of your business and customer data.
5. Open direct marketing channels with no extra costs with methods such as push notifications.
6. Boost the productivity of all the processes within the organization.
7. Increase efficiency and customer satisfaction with an app aligned to their needs.
8. Seamlessly integrate with existing infrastructure.
9. Ability to provide valuable insights.
10. Optimize sales processes to generate more revenue through enhanced data collection.

DISADVANTAGES :-

1. Determining the right process.
2. Feeling constrained.
3. Spending more than necessary.
4. Finding the time for it.
5. Making the right decisions.
6. Impacting how employees feel.
7. Overlooking important factors.

CONCLUSION

As a conclusion, our project Personal Expense Tracker is a very good upcoming application to solve the financial problems and other related problems. It is an user friendly application. The customers will be successful in using this application and save more money without spending it lavishly. The users of this application will manage the accounts effectively and keep a track on the investments and expenses to get a detailed report on it for further advancements

FUTURE SCOPE

The Expense Tracker app **tracks all the expenses and helps the user to manage his/her expenses so that the user is the path of financial stability**. The Tracking of expenses is categorised by week, month and year, it helps to see the more expenses made.These apps analyze your SMSs and track your spends. So every time you spend on your debit card, credit card through any wallet, the expenses gets recorded automatically.

Let us assume that you spend Rs 500 at a coffee shop or Rs 2,000 at a petrol pump. The app will read the SMS with the amount deducted and the merchant name and add it to your list of expenses. It also tracks your ATM withdrawals. You can enter cash expenses manually on the app.

Not only can you track your expenses, you can categorise them too. You can select from pre-defined categories like bills, groceries, shopping, eating out, fuel and so on and also adds custom categories. Every spending is grouped into categories. When you play your electricity bills, it gets grouped under „bills,‟ when you go out for dining, the expenses gets grouped under „Eating out‟ and so on. So you know exactly how much of money you are spending under each of these categories.

APPENDIX

SOURCE CODE:

```
from flask import Flask, render_template, url_for, request, redirect, session,
flash, abort
from flask_login import LoginManager
from flask_login import login_required, current_user, login_user,
logout_user, UserMixin
from werkzeug.security import generate_password_hash,
check_password_hash
import ibm_db
from mailjet_rest import Client
api_key = '7ff055208301386a41dd34e5d95953c3'
api_secret = '0302e0c78d439b8d97dc2c74e0d1644d'
mailjet = Client(auth=(api_key, api_secret), version='v3.1')

# Ibm Db2

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=824dfd4d-99de-
440d-9991-
629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=301
19;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=
fpq67161;PWD=3IOG7aiAt5sF2eBq", '', '')

# App

app = Flask(__name__)
app.config['SECRET_KEY'] = '310819106018'


login_manager = LoginManager()
login_manager.login_view = 'login'
login_manager.init_app(app)


class User(UserMixin):
    def __init__(self, user_json):
        self.user_json = user_json

    def get_id(self):
        object_id = self.user_json.get('PERSONID')
        return str(object_id)


@login_manager.user_loader
def load_user(user_id):
    sql = "SELECT * FROM login WHERE personid=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, user_id)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    return User(account)


@app.route('/')
def index():
    return render_template('index.html', index='active',
success=request.args.get('success'), danger=request.args.get('danger'))


@app.route('/login')
def login():
    return render_template('login.html', login='active',
danger=request.args.get('danger'), success=request.args.get('success'))


@app.route('/login', methods=['POST', 'GET'])
def login_rec():
    if request.method == 'POST':

        email = request.form['email']
        password = request.form['password']
        remember = True if request.form.get('remember') else False

        sql = "SELECT * FROM login WHERE email=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)

        if not account:
            return redirect(url_for('signup', danger="You do not have an
registered account so, please register and login"))
        else:
            if not check_password_hash(account['PASSWORD'], password):
                return redirect(url_for('login', danger="You've entered a wrong
password"))
            else:
                userdetails = User(account)
                login_user(userdetails, remember=remember)
                return redirect(url_for('dashboard', success='Login Successfull'))


@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('index', success="Logout successfull"))


@app.route('/signup')
def signup():
    return render_template('signup.html', signup='active',
danger=request.args.get('danger'))


@app.route('/signup', methods=['POST', 'GET'])
def addrec():
    if request.method == 'POST':

        firstname = request.form['firstname']
        lastname = request.form['lastname']
        email = request.form['email']
        password = request.form['password']
        re_password = request.form['re-password']

        sql = "SELECT * FROM login WHERE email=?"
        prep_stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(prep_stmt, 1, email)
        ibm_db.execute(prep_stmt)
        account = ibm_db.fetch_assoc(prep_stmt)

        if account:
            return redirect(url_for('login', danger="You already have an account
so, please login with your credentials"))

        elif (password != re_password):
            return redirect(url_for('signup', danger="Your password doesn't
match"))

        else:
            insert_sql = "INSERT INTO
login(firstname,lastname,email,password) VALUES (?,?,?,?)"
            prep = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep, 1, firstname)
            ibm_db.bind_param(prep, 2, lastname)
            ibm_db.bind_param(prep, 3, email)
            ibm_db.bind_param(prep, 4, generate_password_hash(
                password, method='sha256'))
            ibm_db.execute(prep)

            data = {
                'Messages': [
                    {
                        "From": {
                            "Email": "310819106018@smartinternz.com",
                            "Name": "Expense Tracker"
                        },
                        "To": [
                            {
                                "Email": email,
                                "Name": firstname
                            }
                        ],
                        "TemplateID": 4331789,
                        "TemplateLanguage": True,
                        "Subject": "Welcome",
                        "Variables": {
                            "name": firstname
                        }
                    }
                ]
            }
            mailjet.send.create(data=data)

            return redirect(url_for('login', success="Registration Successfull"))


@app.route('/dashboard')
@login_required
def dashboard():

    # Expense Details SQL
    expensedetails = []
    sql = "SELECT AMOUNT,DETAILS,CHAR(DATE(DANDT),USA) AS
DATEADDED, CHAR(TIME(DANDT),USA) AS TIMEADDED FROM
USERDATA WHERE USERID = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, current_user.user_json['PERSONID'])
    ibm_db.execute(stmt)
    details = ibm_db.fetch_assoc(stmt)
    while details != False:
        expensedetails.append(details)
        details = ibm_db.fetch_assoc(stmt)

    label = [row['DATEADDED'] for row in expensedetails]
    amountlabel = [row['AMOUNT'] for row in expensedetails]

    # Totalexpense SQL
    sql2 = "SELECT SUM(AMOUNT) AS TOTALVAL FROM USERDATA
WHERE USERID = ?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2, 1, current_user.user_json['PERSONID'])
    ibm_db.execute(stmt2)
    totalexpense = ibm_db.fetch_assoc(stmt2)
    if totalexpense['TOTALVAL'] is None:
        totalexpense['TOTALVAL'] = 0
```

```
   # walletbalance SQL
   sql3 = "SELECT SUM(WALLETAMOUNT) AS TOTALVAL FROM
WALLET WHERE WALLETID = ?"
   stmt3 = ibm_db.prepare(conn, sql3)
   ibm_db.bind_param(stmt3, 1, current_user.user_json['PERSONID'])
   ibm_db.execute(stmt3)
   walletbalance = ibm_db.fetch_assoc(stmt3)
   if walletbalance['TOTALVAL'] is None:
      walletbalance['TOTALVAL'] = 0
      availablebalance = 0
   else:
      availablebalance = int(
         walletbalance['TOTALVAL']) - int(totalexpense['TOTALVAL'])
   if (availablebalance <= 50):
      flash("Your balance is too low!!!")
   elif (availablebalance > 50 and availablebalance <= 200):
      flash("Your balance is getting low so take care of your expenses...!!!")

   return render_template('dashboard.html', dashboard='active',
name=current_user.user_json['FIRSTNAME'],
success=request.args.get('success'), danger=request.args.get('danger'),
expensedetails=expensedetails, totalexpense=totalexpense['TOTALVAL'],
walletbalance=availablebalance, label=label, amountlabel=amountlabel)


@app.route('/addexpense/<balance>', methods=['POST'])
@login_required
def addexpense(balance):
   amount = request.form['amount']
   detail = request.form['details']

   if (int(amount) == 0):
      return redirect(url_for('dashboard', danger="Please enter some
amount"))

   else:
      sql = "INSERT INTO USERDATA(USERID,AMOUNT,DETAILS)
VALUES(?,?,?)"
      stmt = ibm_db.prepare(conn, sql)
```

```
        ibm_db.bind_param(stmt, 1, current_user.user_json['PERSONID'])
        ibm_db.bind_param(stmt, 2, amount)
        ibm_db.bind_param(stmt, 3, detail)
        ibm_db.execute(stmt)

        if (int(balance) <= 100):
            data = {
                'Messages': [
                    {
                        "From": {
                            "Email": "310819106018@smartinternz.com",
                            "Name": "Expense Tracker"
                        },
                        "To": [
                            {
                                "Email": current_user.user_json['EMAIL'],
                                "Name": current_user.user_json['FIRSTNAME']
                            }
                        ],
                        "TemplateID": 4332120,
                        "TemplateLanguage": True,
                        "Subject": "Low Balance !!",
                        "Variables": {
                            "amount": balance
                        }
                    }
                ]
            }
            mailjet.send.create(data=data)

        return redirect(url_for('dashboard', success="Expense added 
successfully"))


@app.route('/addmoney', methods=['POST'])
@login_required
def addmoney():
    amount = request.form['walletamount']

    if (int(amount) == 0):
        return redirect(url_for('dashboard', danger="Please enter some
amount"))

    else:
        sql = "INSERT INTO WALLET(WALLETID,WALLETAMOUNT)
VALUES(?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, current_user.user_json['PERSONID'])
        ibm_db.bind_param(stmt, 2, amount)
        ibm_db.execute(stmt)

        return redirect(url_for('dashboard', success="Money added
successfully"))



# Delete
@app.route('/deleteexpense/<val>/<amount>')
@login_required
def deleteexpense(val, amount):

    sql = "DELETE USERDATA WHERE USERID=? AND
CHAR(TIME(DANDT),USA)= ? AND AMOUNT=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, current_user.user_json['PERSONID'])
    ibm_db.bind_param(stmt, 2, val)
    ibm_db.bind_param(stmt, 3, amount)
    ibm_db.execute(stmt)

    return redirect(url_for('dashboard', success="Deleted Successfully"))



@app.errorhandler(500)
def page_not_found():
    return redirect(url_for('index', danger="oops!!! error occured, try
again")), 500



@app.errorhandler(404)
def not_found():
    return redirect(url_for('index', danger="oops!!! error occured, try
again")), 404


if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000)
```

GITHUB LINK : https: //github.com/IBM-EPBL/IBM-Project-18479-1659685956