A PROJECT REPORT ON

PERSONAL EXPENSE TRACKER APPLICATION

Domain : Cloud App Development

Team ID : PNT2022TMID23325

College Name : Velammal Engineering College

Department : Computer Science Engineering

Team Members:

1. Jegadeesh SSRA (Team Leader) (113219031060)
2. Elavarasan E (113219031040)
3. Godwin Rithish M (113219031043)
4. Behorin J (113219031025)

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

A Personal Expense Tracker Application is a particular form of digital diary that aids in keeping track of all of our cash transitions and moreover offers daily, weekly, monthly, and yearly reports on all financial activities.

User receives alerts to keep track of income and expenses that can system for tracking the application. All data is kept in offline mode for easy access at any time and from any location. The Daily Expense Tracker's user interface is incredibly straightforward and appealing, making it simple to grasp and the finest approach to record our financial data.

1.1. PROJECT OVERVIEW

Simply put, personal finance includes all of the financial decisions and actions that a finance software facilitates by assisting you in effectively managing your finances. A personal finance software will not only assist you with accounting and budgeting, but it will also provide you with valuable advice on money management.

Users of personal finance applications will be prompted to enter their costs, after which their wallet balance will be updated and displayed to them. Users can also receive a graphical analysis of their expenses. They can choose to establish a cap on how much can be used in that month, and if the cap is surpassed, the user will receive an email alert.

1.2. PURPOSE

When you keep track of your spending, you can make sure your money is being utilised wisely and you will know where it goes. You can learn why you're in debt and how you got there by keeping track of your spending. You can then use this information to create a debt relief plan that works for you.

You may plan for both short-term and long-term expenses by using a budget to make sure you're not spending more than you're earning. It's a simple, practical solution for folks with all types of income and expenses to maintain order in their finances.

CHAPTER 2

LITERATURE SURVEY

A literature review is a piece of academic writing that places the academic literature on a particular topic in perspective to show knowledge of and understanding of it. This chapter shows the different techniques that have been implemented.

2.1. EXISTING PROBLEM

The expense tracker existing system does not offer the user portable device management level, is only used on desktop software, and is therefore impossible to update anywhere expenses are done and is unable to update the location of the expense details disrupting that the proposed system provides. The user's daily, weekly, and monthly spending must be maintained in Excel sheets and CSV files at the moment. The ability to conveniently keep track of one's everyday costs does not now have a fully comprehensive answer.To do this, one must maintain a journal in a diary or computer system, and all calculations must be made by the user, which might occasionally result in errors that cause losses. Due to imperfect data maintenance, the current system is not user friendly. The sole negative where the rest are absent from this endeavor is that there will be no reminder to stay a human on a specified date. This project won't have any information because it doesn't remind people to do anything each month, which has some drawbacks. However, it can be used to calculate income and expenses, so we suggest a new project to solve this issue.

2.2 REFERENCE

[1] Expense Tracker ATIYA KAZI, PRAPHULLA S. KHERADE, RAJ S. VILANKAR, PARAG M. SAWANT May 2021

[2] Intelligent Online Budget Tracker Girish Bekaroo and Sameer Sunhaloo Proceedings of the 2007 Computer Science and 2007

[3] Online Income and Expense Tracker S. Chandini, T. Poojitha, D. Ranjith, V.J. Mohammed Akram, M.S. Vani, V. Rajyalakshmi Mar 2019

[4] Family Expense Manager Application Rajaprabha M N 2017

[5] A Novel Expense Tracker using Statistical Analysis Muskaan Sharma, Ayush Bansal, Dr. Raju Ranjan, Shivam Sethi June 2021

[6] Expense Tracker Hrithik Gupta, Anant Prakash Singh, Navneet Kumar and J. Angelin Blessy December 2020

[7] Expense Manager: An Expense Tracking Application using Image Processing Nupur Sawarkar, Pranay Yenagandula, Devang Shetye, Prof. Shruti Agrawal April 2022

[8] D2D Expense Tracker Application Anjali Kumar, Utkarsh Ra, Aman Kumar 2021

[9] Daily Expense Tracker Mobile Application Nuura Najati Binti Mustafa 2021

[10] Daily Expense Tracker Shivam Mehra, Prabhat Parashar 2021

2.3 PROBLEM STATEMENT DEFINITION

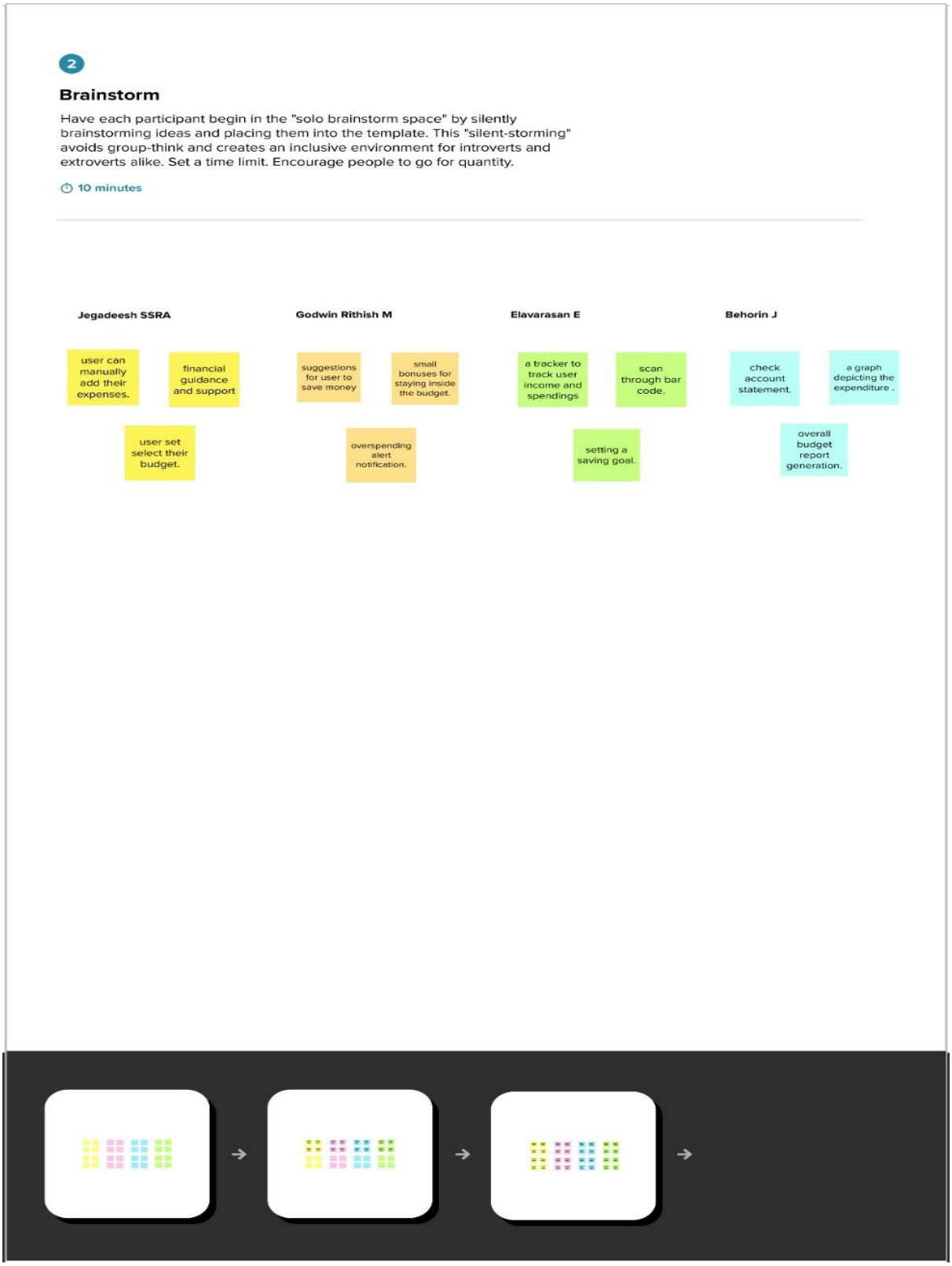
In our daily life money is the most important portion and without it we cannot last one day on earth but if we keep on track all financial data then we can overcome this problem. Most of the people cannot track their expenses and income one way they face the money crisis and depression. This situation motivates us to make an android app to track all financial activities. Using the Personal Expense Tracker Application user can be tracking expenses day to day and making life tension free.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

3.2. IDEATION & BRAINSTORMING



3.3. PROPOSED SOLUTION

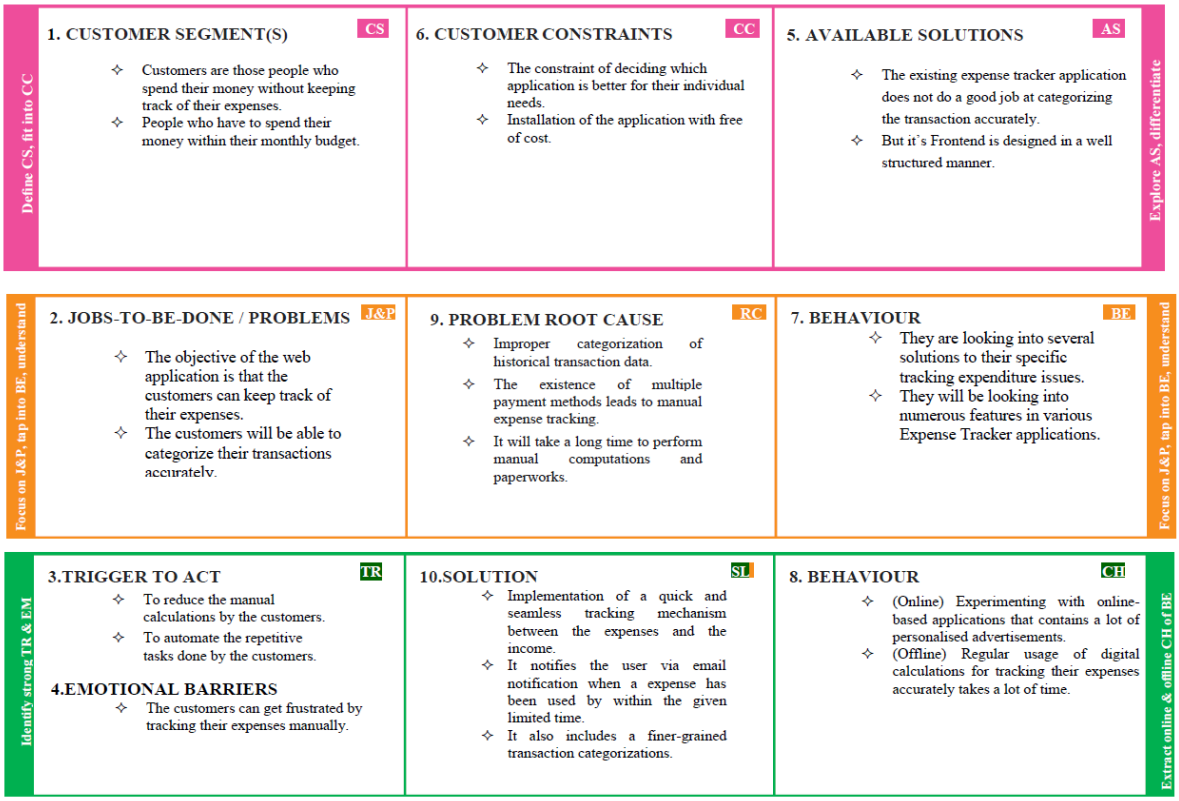
S.NO.	Parameter	Description
1.	Problem Statement	In a paper-based expense tracker system it is difficult to track our monthly expenses manually. In a paper-based expense tracker system it is difficult to track our monthly expenses manually. The paper-based expense records may get lost in case of fire accidents, floods etc.
2.	Scalability of the Solution	This application can handle large numbers of users and data with high performance and security. This application can adapt for both large-scale and small-scale purposes. Easily available in all kinds of devices.
3.	Idea / Solution description	Daily expense management system which is specially designed for non-salaried and salaried personnel for keeping track of their daily expenditure in an easy and effective way through a computerized system which tends to eliminate manual paperwork. 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. 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.
4.	Novelty / Uniqueness	The user gets notified when their expense exceeds the limit and also it reminds the user when they forgot to make an entry. Tracking

		expenses through SMS. Data analytics on expenses. Future expense prediction

5.	Social Impact / Customer Satisfaction	The application should be able to generate reports of their spending and notify users if they have exceeded their budget. It is designed to be dynamic to produce the prediction. It also provides users' personal information, their income as well as their expenses. This application can create awareness among common people about finance and stuff. This application also helps users to be financially responsible. It Reduces time rather than entering details manually.
6.	Business Model (Revenue Model)	This Application is provided for free of cost. But It will have some advertisements. In premium version there is no advertisement and contains some additional features.

3.4. PROBLEM SOLUTION FIT

Project Title: PERSONAL EXPENSE TRACKER Project Design Phase -I - Solution Fit Team ID: PNT2022TMID23325



CHAPTER 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 Registration	Registration through application registration through Gmail
FR-2	User Confirmation	Confirmation via email confirmation via OTP
FR-3	User monthly expense tentative data	Data to be registered in the app
FR-4	User monthly income data	Data to be registered in the app
FR-5	Alert/Notification	Alert through email alert via SMS

FR-6	User Budget Plan	Planning and Tracking of user expense and budget limit

4.2 NON FUNCTIONAL REQUIREMENTS

Following are the non functional requirements of the Proposed solution

FR NO.	Non Functional Requirement	Description
NFR-1	Usability	Effectiveness, efficiency and overall satisfaction of the user
NFR-2	Security	Authentication,authorisation and encryption of the application
NFR-3	Reliability	Probability of failure free operations in a specified environment for a specified time
NFR-4	Performance	How the application is functioning and how responsive the application is to the end users

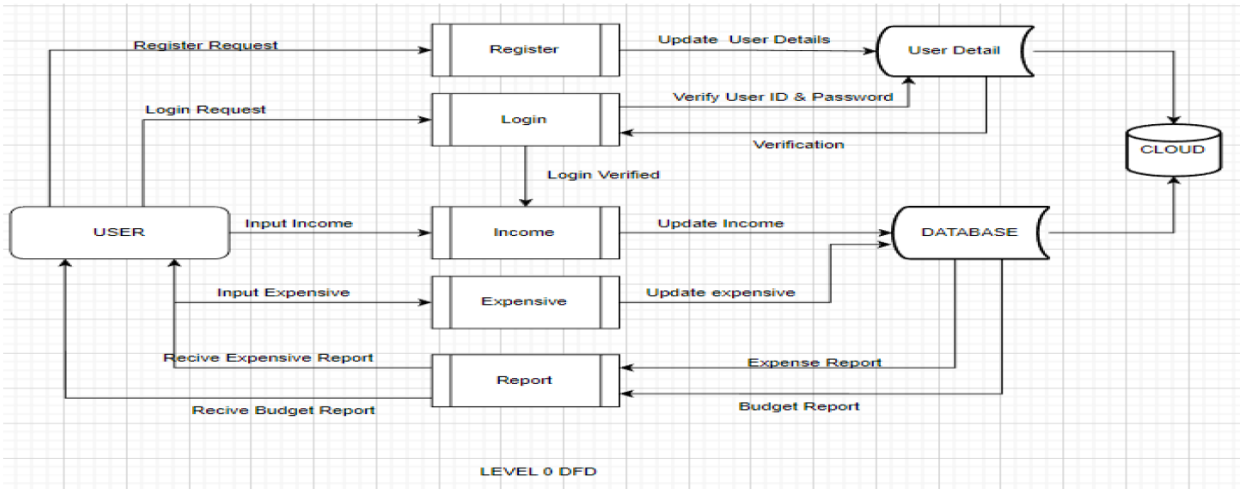
NFR-5	Availability	Without near 100% availability,application reliability and the user satisfaction will affect the solution
NFR-6	Scalability	Capacity of the application to handle growth,especially in handling more users

CHAPTER 5

PRODUCT DESIGN

5.1 DATA FLOW DIAGRAMS

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.



LEVEL 0 DFD

5.2 TECHNICAL ARCHITECTURE

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

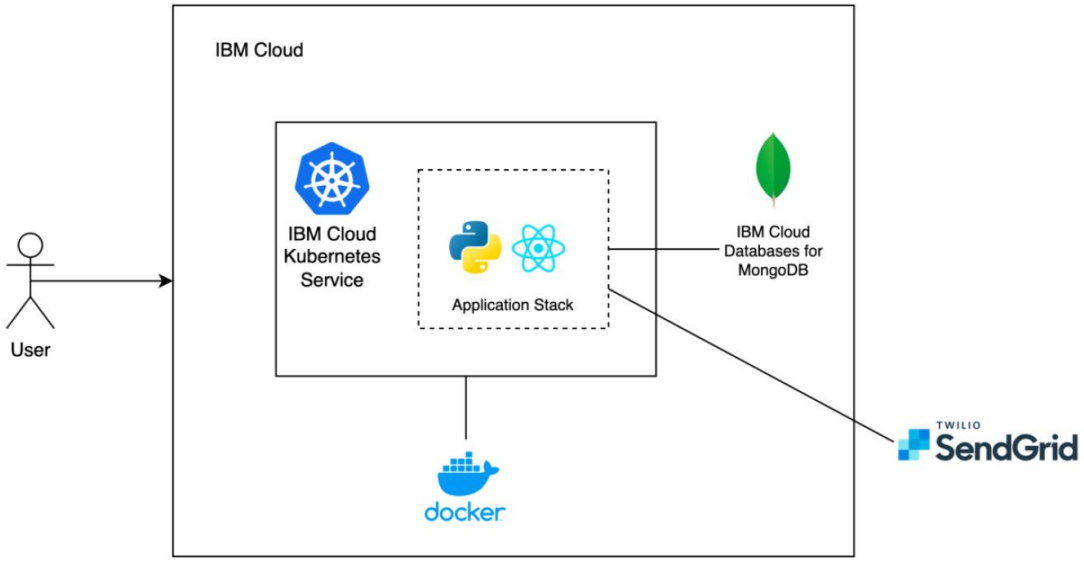


Figure 1: Solution Architecture of Expense Tracker App

5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobileuser & 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 an administrator I can upgrade or update the application.	I can fix the bug which arises for the customers and users of the application	Medium	

CHAPTER 6

PROJECT PLANNING & 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.	3	High	Aiswarya. S Danujaa. R
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	3	High	Aiswarya. S Danujaa. R
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	5	High	Aiswarya. S Danujaa. R

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	Aiswarya. S Danujaa. R
Sprint-1		USN-5	Once logged In, Keep me logged for few hours to avoid repeated login if the page is refreshed	4	Medium	Aiswarya. S Danujaa. R
Sprint-2	Expense	USN-6	Add total income for the month and Allow for edit option	6	High	Sharmitha. S Sneha ganesh
Sprint-2		USN-7	Split the total income based on usage like entertainment, food, shopping etc.	2	Low	Sharmitha. S Sneha ganesh
Sprint-2		USN-8	Add the day to day expense.	6	High	Sharmitha. S Sneha ganesh
Sprint-2		USN-9	Display the user added expense	6	High	Sharmitha. S Sneha ganesh

Sprint-3		USN-10	Filter the expense data based on criteria	6	Medium	Aiswarya. S Danujaa. R

Sprint-3	Charts	USN-11	As a user I can display it in graphs		Low	Aiswarya.S Danujaa. R
Sprint-3	Alerts	USN-12	As a user I create custom alert for the balance		High	Aiswarya. S Danujaa. R
Sprint-4	Deployment	USN-13	As a user I should able to access it anywhere in the net		High	Sharmitha. S Sneha ganesh

6.2 SPRINT DELIVERY SCHEDULE

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	6	6 Days	15 Oct 2022	30 Oct 2022	6	30 Oct 2022
Sprint-2	2	6 Days	30 Oct 2022	05 Nov 2022	2	05 Nov 2022
Sprint-3	7	6 Days	05 Nov 2022	12 Nov2022	7	12 Nov 2022
Sprint-4	5	6 Days	12 Nov2022	19 Nov2022	5	19 ov 2022

CHAPTER 7

7.1 FEATURES

1. Track revenues & expenses

This is an essential feature and the primary function of an expense management app is evaluating revenues and studying the expenses. It is simple to import transactions from your mobile wallets, credit card, bank without risking your details. This can give you a clear picture about the earnings and expenses. You can evaluate if you are making more expenses and where you need to cut.

2. Arrange tax deductions

While filing taxes, simply upload your documents to the app. The app will arrange revenues and expenses into various tax categories. Expense management apps will organize the business spendings into the right tax categories.

3. Oversee vendors & contractors

You can include information of vendors and contractors in the app and delegate them categories. Moreover, check all the payments made to these vendors, such as who, how much, and when.

4. Safe access

Since you cannot tackle all the accounting yourself, you can give your app safe access to your books. Moreover, you can also provide special access to some functions of the expense tracking software to mitigate errors.

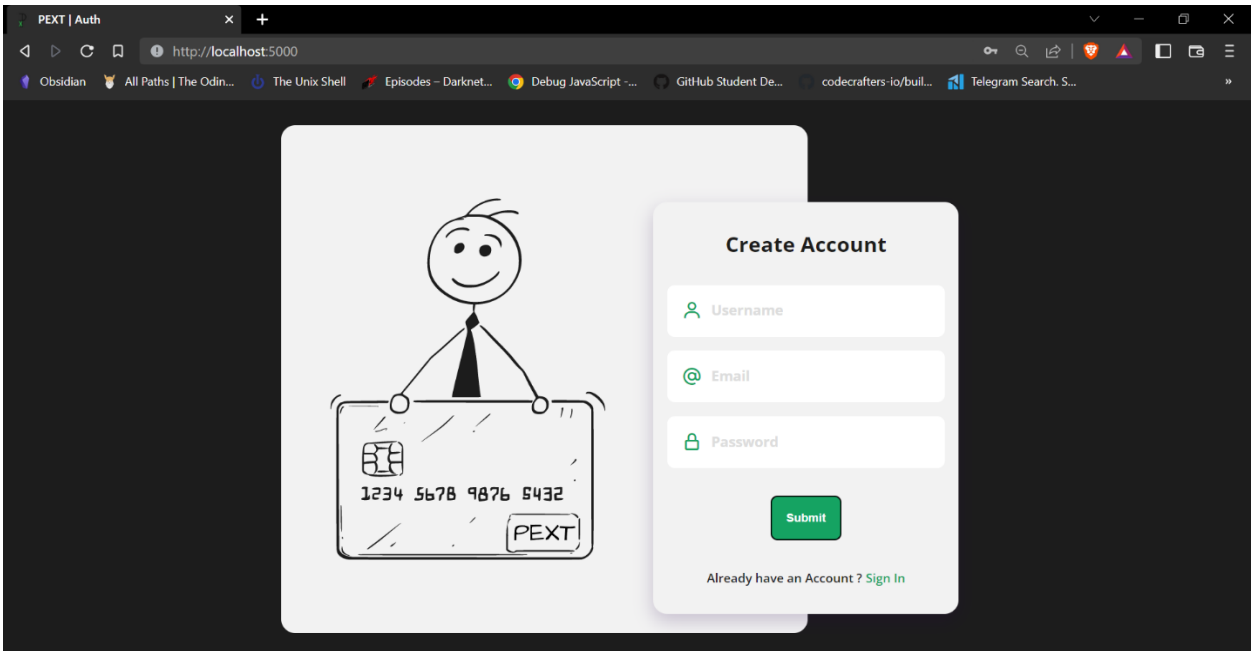
The expense management app helps in delegating work to particular users and boost your team's efficiency.

CHAPTER 8

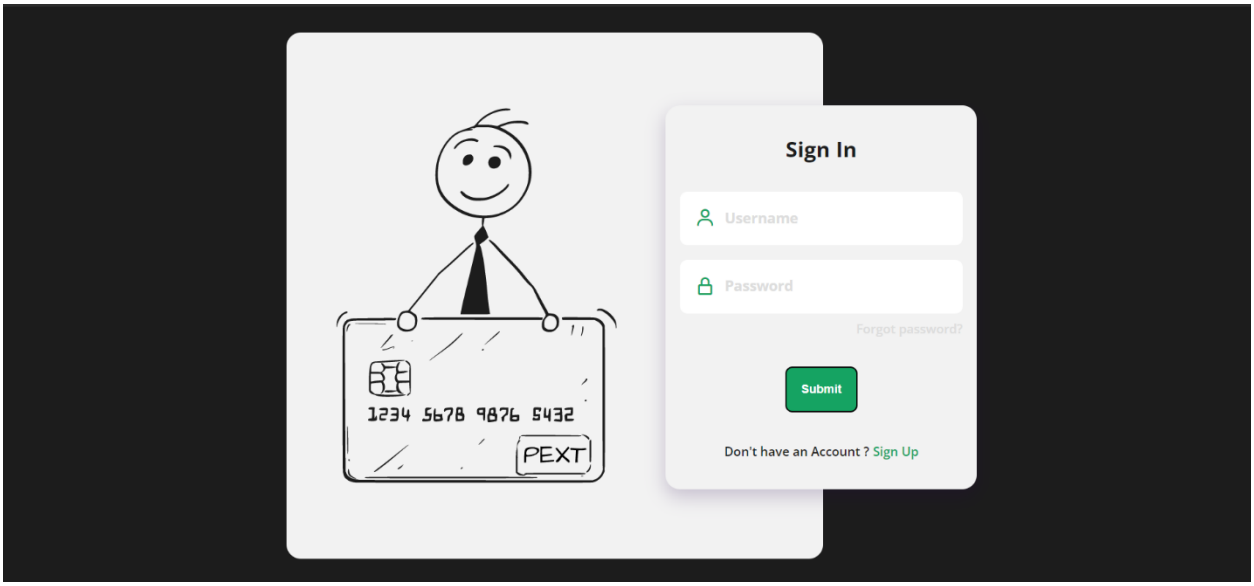
TESTING

8.1 TEST CASES

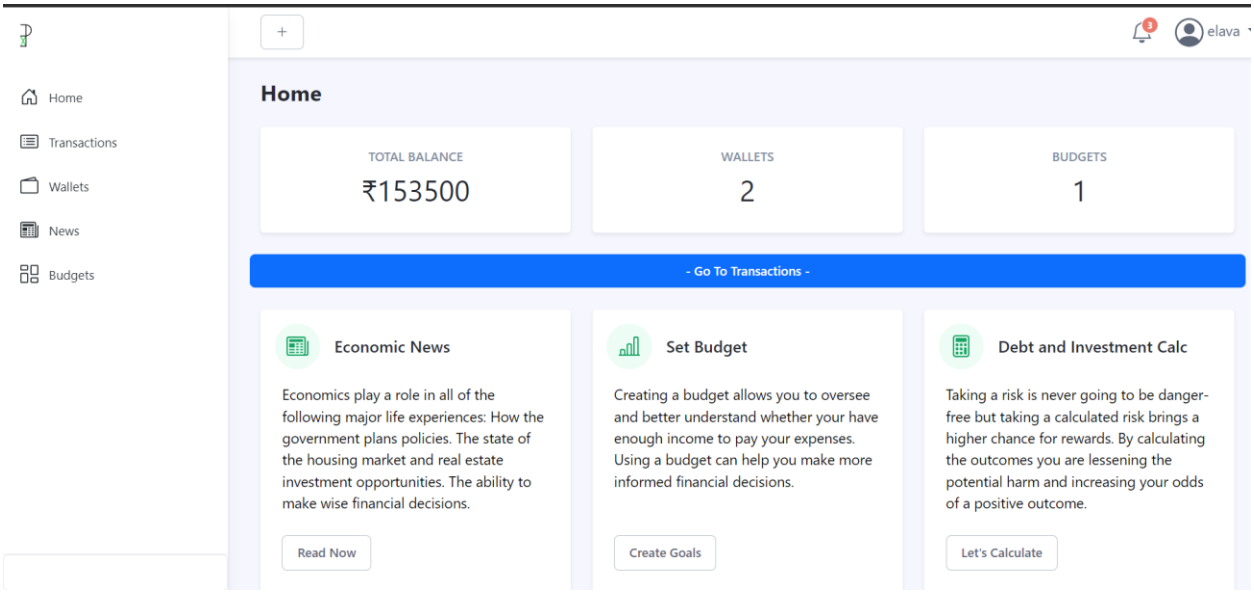
REGISTRATION PAGE



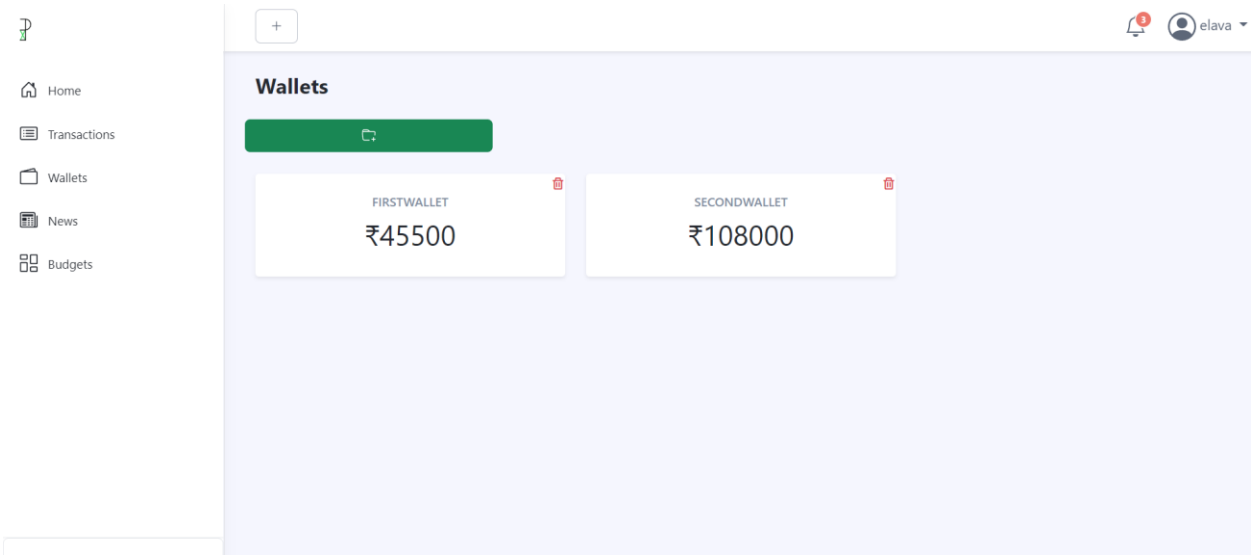
LOGIN



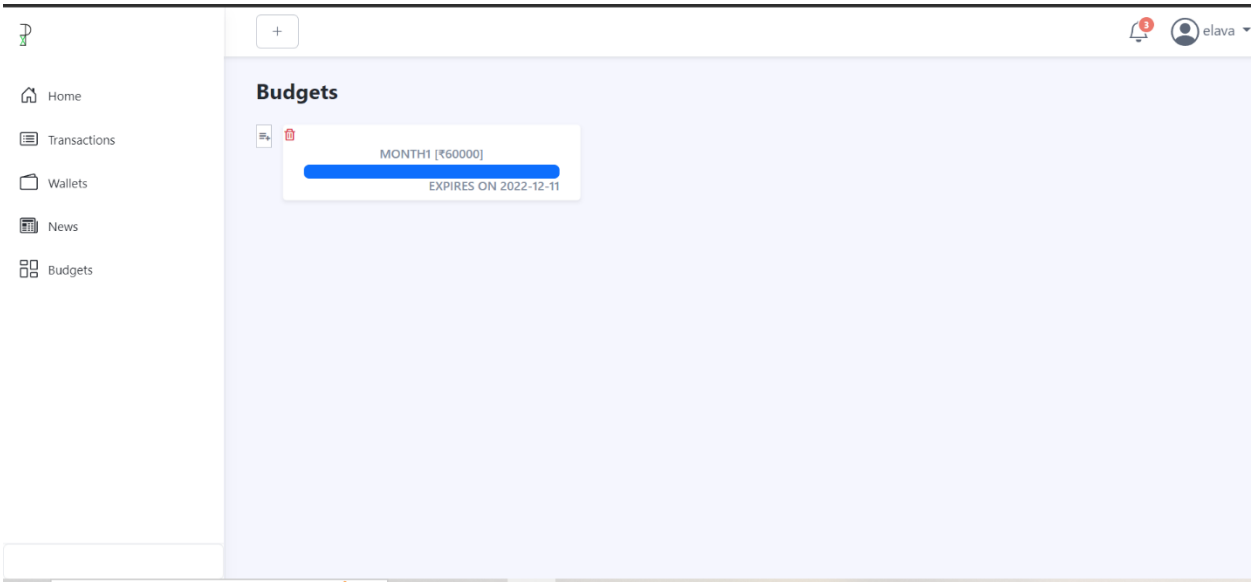
HOME PAGE



WALLET



BUDGETS



CHAPTER 9

RESULTS

9.1 PERFORMANCE METRICS

An application can be a very powerful tool for businesses if once the app becomes a success.

However, the success of an app is measured through numbers, metrics, and analytics.

Developing an app takes quite a lot, so once you've dedicated much time, money, and effort to the process, it's mandatory to measure mobile app performance.

CHAPTER 10

ADVANTAGES & DISADVANTAGES

10.1 ADVANTAGES

- Improved customer service
- Cloud-based solution
- Order Fulfillment
- Harness Customer Loyalty and Retention

10.2 DISADVANTAGES

- System Clash
- Reduced Physical Audits
- No solution to improve or eliminate bottlenecks in the service cycle

CHAPTER 11

CONCLUSION

Taking proper care of our record is crucial in every business, no matter how big or little, we must understand. We must educate ourselves about the idea of effective inventory management and its applications because we can see that managers do not fully grasp it.

A company's inventory management system is one of the reasons for its failure. Many customs to combat failure are present, and we can start from this point. Modern technologies can support us in managing and keeping an eye on inventory. We may learn, put new ideas into practice, and assess our company.

CHAPTER 12

FUTURE SCOPE

1) It will have a variety of record-keeping choices (such as food, travel expenses, salary, etc.).

2) It will continue to give updates about our daily spending automatically.

3) Despite being in a haste to make money in today's hectic and expensive world, we eventually gave up. As we naively waste money on unnecessary items and titles. We came over with the intention of following our profit.

4) The user can specify their own expense categories here, such as those for food, clothing, rent, and bills, where they must input the money that has been spent.

CHAPTER 13

APPENDIX

13.1. SOURCE CODE

app.py

```
from flask import Flask, render_template, redirect, send_file, session
from flask_session import Session

from routes.users import userBp
from routes.dash import dashBp
from routes.crud import crudBp

from dotenv import load_dotenv

import os

load_dotenv()
app = Flask(__name__ )
app.secret_key = os.environ["APP_SECRET_KEY"]
app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"

Session(app)

app.register_blueprint(userBp)
```

```
app.register_blueprint(dashBp)
app.register_blueprint(crudBp)

@app.route("/")
def index():
    if session.get("active") == None:
        return render_template("index.html")
    else:
        return redirect("/dash")

@app.route("/logout")
def logout():
    session.clear()
    return redirect("/")

@app.route('/favicon.ico')
def favicon():
    return send_file("static/img/logo.svg")

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

DB.py

```
from flask import session
```

```
import ibm_db
import hashlib
import datetime
import os

class Db:
    def __init__(self) -> None:
        host = os.environ["DBHOST"]
        uid = os.environ["DBUID"]
        pwd = os.environ["DBPWD"]
        ssl = os.environ["DBSSLCERT"]
        db = os.environ["DB"]
        port = os.environ["DBPORT"]
        self.conn =
ibm_db.connect(f"DATABASE={db};HOSTNAME={host};PORT={port};SECU
RITY=SSL;SSLServerCertificate={ssl};UID={uid};PWD={pwd};", "", "" )

    def generateId(self) -> str:
        return hashlib.md5("{}{}".format(
                session["active"],
                datetime.datetime.now().strftime('%m%d%Y%H%M%S%f')
                ).encode()).hexdigest()

    def execute(self, query: str) -> bool:
        try:
            ibm_db.exec_immediate(self.conn, query)
```

```
            return True
        except:
            print("SQLSTATE = {}".format(ibm_db.stmt_error()))
            return False

    def get(self, table_name: str, condition: str, columns : str = "*") -> tuple:
        try:
            query = f"SELECT {columns} FROM {table_name} WHERE
{condition}"
            print(query)
            stmt = ibm_db.exec_immediate(self.conn, query)
            return ibm_db.fetch_tuple(stmt)
        except:
            print("SQLSTATE = {}".format(ibm_db.stmt_error()))
            return ()

    def getall(self, table_name: str, condition: str, columns : str = "*") -> list:
        query = f"SELECT {columns} FROM {table_name} WHERE {condition}"
        print(query)
        stmt = ibm_db.exec_immediate(self.conn, query)
        data = []
        while True:
            temp = ibm_db.fetch_tuple(stmt)
            if temp != False:
                data.append(temp)
            else:
```

```
            break
        return data

    def delete(self, table_name:str, condition: str) -> bool:
        query = f"DELETE FROM {table_name} WHERE {condition}"
        print(query)
        return self.execute(query)

    def insert(self, table_name: str, values: list) -> bool:
        try:
            valuestup = ','.join("'{0}'".format(x) for x in values)
            query = f'INSERT INTO {table_name} VALUES ({valuestup})'
            print(query)
            return self.execute(query)
        except Exception as e:
            print(e)
            return False
```

USER,py

```
from werkzeug.security import generate_password_hash, check_password_hash
from flask import session
from modules.db import Db
```

```
class User(Db):
    def __init__(self) -> None:
        super().__init__()

    def register(self, form) -> bool:
        return self.insert(
            "users", [
                form['username'],
                form['email'],
                generate_password_hash(form['password'])
            ]
        )

    def login(self, form):
        try:
            data = self.get("users", f"username='{form['username']}'")
            if len(data) != 0:
                if check_password_hash(data[2], form["password"]):
                    session["active"] = data[0]
                    return True
                else:
                    return False
            else:
                return False
        except Exception as e:
```

```
            print(e)
            return False
```

WALLET.py

```
from flask import session
from modules.db import Db

class Wallet(Db):
    def __init__(self) -> None:
        """
            CREATE TABLE wallets (
                wid VARCHAR(32) UNIQUE NOT NULL,
                wname VARCHAR(30) NOT NULL,
                wamount INTEGER NOT NULL,
                username VARCHAR(30) NOT NULL
            )
        """
        super().__init__()

    def addWallet(self, form) -> bool:
        self.insert(
            "wallets", [
                self.generateId(),
                form["wname"],
```

```
            form["wamount"],
            session["active"]
        ]
    )
    return True

def deleteWallet(self,form) -> bool:
    if self.delete(
            "transactions", f"wallet='{form['wid']}'"
        ) and self.delete(
            "budgets", f"bwallet='{form['wid']}'"
        ):
        return self.delete(
            "wallets", f"wid='{form['wid']}'"
        )
    else:
        return False

def getWallet(self, wid) -> bool:
    return {
        "status" : "success",
        "data": self.get(
            "wallets", f"wid='{wid}'"
        )
    }
```

```
    def getWallets(self) -> dict:
        return {
            "user" : session["active"],
            "data" : self.getall(
                "wallets", "username='{}'".format(session['active'])
            )
        }

    def aboutWallets(self) -> dict:
        wallets = self.getall("wallets", f"username='{session['active']}'")
        bal = 0
        for wallet in wallets:
            bal += wallet[2]
        return {
            "total" : bal,
            "count" : len(wallets)
        }
```

TRANSACTION.py

```
from flask import session
from modules.db import Db

class Transaction(Db):
```

```
    def __init__(self) -> None:
        """
            CREATE TABLE transactions (
                tid VARCHAR(32) UNIQUE NOT NULL,
                tdate VARCHAR(12) NOT NULL,
                descp VARCHAR(1000) NOT NULL,
                wallet VARCHAR(32) NOT NULL,
                ttype VARCHAR(7) NOT NULL,
                category VARCHAR(20) NOT NULL,
                amount INTEGER NOT NULL,
                attachment VARCHAR(100),
                username VARCHAR(30) NOT NULL
            )
        """
        super().__init__()


    def addTransaction(self, form) -> bool:
        date, descp, wallet, ttype, category, famount, attachment = form["date"], form["descp"], form["wallet"], form["ttype"], form["category"], form["amount"], form["attachment"]


        if attachment == '':
            attachment = "None"


        data = self.get("wallets", f"wid='{wallet}'")
        if ttype == "income":
```

```
            amount = data[2] + int(famount)
        elif ttype == "expense":
            amount = data[2] - int(famount)
            if amount < 0:
                return False

        if self.execute(f"UPDATE wallets SET wamount={amount} WHERE
wid='{data[0]}'"):
            return self.insert(
                "transactions", [
                    self.generateId(),
                    date, descp, wallet, ttype, category, famount, attachment,
                    session["active"]
                ]
            )
        else:
            return False


    def getTransactions(self) -> dict:
        return {
            "user" : session["active"],
            "data" : self.getall(
                "transactions", "username='{}'".format(session['active'])
            )
        }
```

```
    def deleteTransaction(self, form) -> bool:
        transaction = self.get(
            "transactions", f"tid='{form['tid']}'"
        )
        wallet = self.get(
            "wallets", f"wid='{transaction[3]}'"
        )
        if transaction[4] == "income":
            amount = wallet[2] - transaction[6]
        else:
            amount = wallet[2] + transaction[6]


        if self.execute(
            f"UPDATE wallets SET wamount={amount} WHERE
wid='{transaction[3]}'"
        ):
            return self.delete(
                "transactions" , f"tid='{form['tid']}'"
            )
        else:
            return False
```

BUDGET.py

```
from flask import session
from modules.db import Db
```

```
class Budget(Db):
    def __init__(self) -> None:
        """
            CREATE TABLE budgets (
                bid VARCHAR(32) UNIQUE NOT NULL,
                bname VARCHAR(30) NOT NULL,
                bdate VARCHAR(12) NOT NULL,
                bamount INTEGER NOT NULL,
                bwallet VARCHAR(32) NOT NULL,
                username VARCHAR(30) NOT NULL
            )
        """
        super().__init__()

    def addBudget(self, form) -> bool:
        print(form)
        return self.insert(
            "budgets", [
                self.generateId(),
                form["name"],
                form["date"],
                form["amount"],
                form["wallet"],
                session["active"]
```

```
            ]
        )

    def deleteBudget(self,form) -> bool:
        self.delete(
            "budgets", f"bid='{form['bid']}'"
        )
        return True

    def getBudgets(self) -> dict:
        return {
            "user" : session["active"],
            "data" : self.getall(
                "budgets", "username='{}'".format(session['active'])
            )
        }
```

Github & Project Demo Link

GITHUB REPO :

https://github.com/IBM-EPBL/IBM-Project-2058-1658424816

PROJECT DEMO LINK:

http://169.51.195.172:32351/

PROJECT DEMONSTRATION VIDEO LINK:

https://drive.google.com/file/d/13v3aydynQ5ZNWwjRVnOkWMCdJHcI8l2L/view?usp=share_link