PERSONAL EXPENSE TRACKER APPLICATION

IBM-Project-51228-1660976272

NALAIYA THIRAN PROJECT BASED LEARNING ONPROFESSIONAL READLINESS FOR INNOVATION, EMPLOYNMENT AND ENTERPRENEURSHIP

PROJECT REPORTBY

TEAM ID: PNT2022TMID47221

ROLL	NAME	REG No
Team Leader	MANIVASANTHAN R	822719104029
Team Member	ARAVINTH R	822719104004
Team Member	GIREESH R	822719104020
Team Member	MANIKANDAN M	822719104028

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE ENGINEERING

GOVERNMENT COLLEGE OF ENGINEERING, SENGIPATTI, THANJAVUR - 613404

AFFILIATED BY
ANNA UNIVERSITY

INDEX

1. INTRODUCTION

1.1 PROJECT OVERVIEW

Category: Cloud App Development

Team ID: PNT2022TMID47221

SKILLS REQUIRED:

HTML, JavaScript, IBM Cloud Object Storage, Python-Flask, Kubernetes, Docker, IBM DB2, IBM Container Registry

ABSTRACT:

Expense tracker is an android based application. This application allows the user to maintain a computerized diary. Expense tracker application which will keep a track of Expenses of a user on a day-to-day basis. This application keeps a record of your expenses and also will give you a category wise distribution of your expenses. With the help of this application user can track their daily/weekly/monthly expenses. This application will also have a feature which will help you stay on budget because you know your expenses. Expense tracker application will generate report at the end of month to show Expense via a graphical representation. We 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.

CUSTOMER SEGMENT:

The person who is busy and couldn't manage their expenses regularly and we will keep track of the expenses regularly and will notify them.

1.2 PURPOSE

Personal expense management plays a very important role. Without an expense tracker, one may miss out the ability to manage the finances wisely and effortlessly. So, the expenses may go through the roof. With a help of a proper expense tracker, one becomes aware of how, when and why the amount is being is spent. So, an expense tracker is a software or application that helps to keep an accurate record of your money inflow and outflow.

Many people in India live on a fixed income, and they find that towards the end of the month they don't have sufficient money to meet their needs. While this problem can arise due to low salary, invariably it is due to poor money management skills. Using a daily expense manager can help you keep track of how much you spend every day and on what. At the end of the month, you will have a clear picture where your money is going. This is one ofthe best ways to get your expenses under control and bring some semblance of order to your finances.

Today there are several expense manager applications in the market. Before you decide to go in for a money manager, it is important to decide the type you want.

2. LITERATURE SURVEY

2.1 Existing Problem:

The Expense Manager is a mobile application intended to run on android device namely smart phone. Expense Manager is designed to efficiently cater the needs of users by eliminating imparting costs and settling vows to friends. The application encourages corresponding users help in who owes who, and for what. Aim is use better approaches to help users and their companions to share expenses easily. This new application will let bunch users and their companions to have detailed view inside this application around individual costs. The app allows its users to add a remark to an expense, click on the expense name in any expense list. Bill posting will have space for comments and notes container with a "Post" catch underneath.

The Expense Manager has notification option to notify each time somebody adds a remark to an expense user is on, or user can withdraw to posted bill. The additional feature that we are going to add in this application that enable us to collect the sample data of users expenses and use this to study patterns of expenses in certain area or by specific kinds of spending for market analysis. These patterns can be derived using some data mining techniques such as clustering, classification and association. Some of the conventional methods used to tackle this problem in normal circumstances are like making use of a sticky note by normal users, Proficient people deal with this kind problems by using spreadsheet to record expenses and using a ledger to maintain large amounts data by especially by experts. As this how that it is variable methods used by different people. This makes using this data inconsistent. There are still problems in areas like there is no assurance for data consistency, there are chances of critical inputs can be missed and the manual errors may creep in.

The Data recorders are not always handy and it could be hectic process to have overall view of those expenses. We believe a handy design a handy mobile application which handles these problems. Such that app is capable of recording the expenses and giving comprehensive view with easy to use user interface and this app is intelligent enough to answer ‘Who owes who? And by how much?’ The mobile applications that are available in the market are very useful to the smartphone users and make their life easy. imparting costs and settling vows to friends. The application encourages corresponding users help in who owes who, and for what. Aim is use better approaches to help users and their in the field of expenditures of user. This idea serves as main objective of research project.

The research also includes syncing of the applications with some social networks and emails as well. This section of paper is very important and this will guide our team to successfully accomplish the goals set for research. Here, the research project methodology describes the steps and approaches to be fallowed to attain final product. As explained above our project is of splitting the expenses between the groups and also to efficiently manage the personal expenses as well. However, our projects will have additional features included as part of our research so that it makes our project unique in the market.

These features would make the project more efficient and very useful for our users. Apart from the benefits user gets and there is an important use of the system that enables us to use the data of the user with his prior permissions for the purpose of data mining for several other functionalities to be applied in market by analyzing user expenses. suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

2.2 References:

www.expenseeasy.com
www.easy2money.com
https://www.tendietracker.com
www.moneymanager.com

2.3 PROBLEM STATEMENT DEFINITION:

At the instant, there is no as such complete solution present easily or we should say free of cost which enables a person to keep a track of its daily expenditure easily. To do so a person has to keep a log in a diary or in a computer, also all the calculations needs to be done by the user which may sometimes results in errors leading to losses. Due to lack of a complete tracking system, there is a constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.

Defining the problem:

How to go to Expenses list page?	Click view expense page to see the expenses monthly basis.
How to add a new expense category?	Go to add expenses link then add the expenses details
How to do define categories?	Rename the categories types (Food, Medical and etc).
How to set up multiple people?	Tap Payers option. Then add the Payers in your account.
How to allocate budgets?	In budget tap you can able to allocate the budgets for your income.
What kind of expense are available?	There are many kind of expenses categories like Food, Travel, Shopping, Medical, Education and Other expenses.

3.IDEATION AND PROPOSED SOLUTION

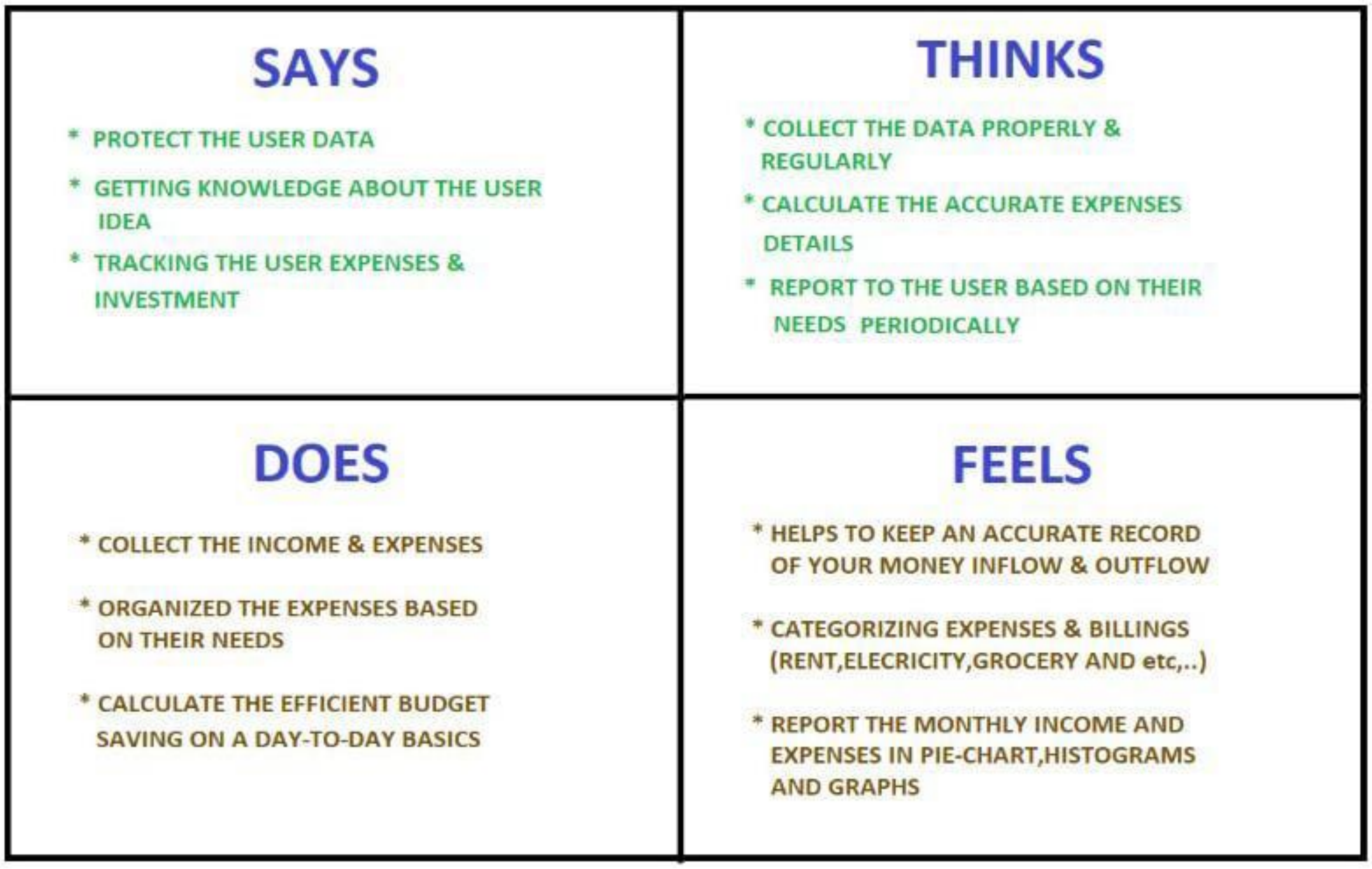
3.1 EMPATHY MAP CANVAS:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviour and attitudes. It is a useful tool to helps teams better understand their users. Creating an

effective solution requires understanding the true problem and the person who is experiencing it. The

exercise of creating the map helps participants consider things from the user's perspective along withhis

or her goals and challenges.



SAYS

* PROTECT THE USER DATA
* GETTING KNOWLEDGE ABOUT THE USER IDEA
* TRACKING THE USER EXPENSES & INVESTMENT

THINKS

* COLLECT THE DATA PROPERLY & REGULARLY
* CALCULATE THE ACCURATE EXPENSES DETAILS
* REPORT TO THE USER BASED ON THEIR NEEDS PERIODICALLY

DOES

* COLLECT THE INCOME & EXPENSES
* ORGANIZED THE EXPENSES BASED ON THEIR NEEDS
* CALCULATE THE EFFICIENT BUDGET SAVING ON A DAY-TO-DAY BASICS

FEELS

* HELPS TO KEEP AN ACCURATE RECORD OF YOUR MONEY INFLOW & OUTFLOW
* CATEGORIZING EXPENSES & BILLINGS (RENT,ELECRICITY,GROCERY AND etc,..)
* REPORT THE MONTHLY INCOME AND EXPENSES IN PIE-CHART,HISTOGRAMS AND GRAPHS

3.2 Ideation and Brainstorming:

Analysis the proper solution about the problem statement

1. Find the Problem
2. Analyses the problem deeply
3. Try to resolve problem
4. Find the better solution
5. Try the possible Testcase
6. Then deploy the Solution
7. Upgrade it frequently based on the needs.

3.3 Proposed Solution:

- Effective financial management requires the proper tracking of income and expenses.
- There are many software options to help you **track all of your spending.**
- Keeping track of your expenses will help you work **within your budget** and make **strategic investments** in your business.
- It helps who want to target their spending and track their expenses over time.

Rent, utilities, equipment, furniture, inventory, licenses, insurance, marketing, staff – small businesses incur **all sorts of expenses**. It’s essential that you keep careful track of all your spending to make sure you stay on budget.

3.3.1 PROBLEMS AND PAINS:

Personal finance applications will ask users to add their expenses and based on their expense 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.

3.4 SOLUTION:

The proposed system makes a novel attempt to track the user expenses daily and if their expenses exceeds the fixed budget we will notify them through mail and user will get an analyzed report. If the user spends large amount of money in a particular area continuously, we will notify them to reduce the spending in that particular area.

BEFORE Our Application

Fear of spending lot of money and couldn't manage their expenses.

AFTER Our Application

They can manage their expense

4.REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT:

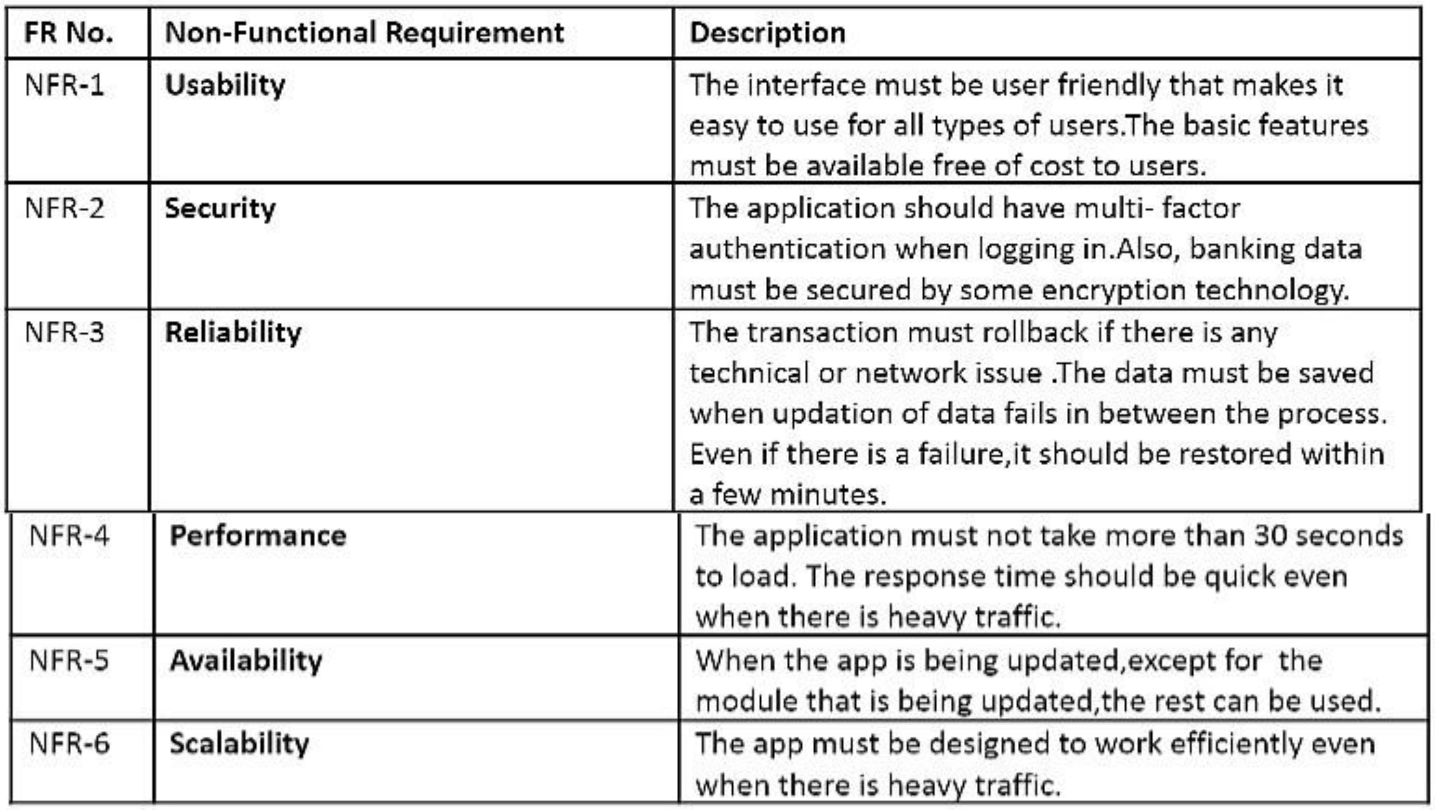
Following are the functional requirements of the proposed solution.

- ➔ IBM Cloud,
- ➔ HTML,
- ➔ JavaScript,
- ➔ IBM Cloud Object Storage,
- ➔ Python-Flask,
- ➔ Kubernetes,
- ➔ Docker,
- ➔ IBM DB2,
- ➔ IBM Container Registry

Functional Requirement	Details
Registration	Register via Application
User Confirmation	Confirmation via Email or Message
Login	Check the login credentials (ID & Password)
Add Information	Add the daily income and expenses through application form
Track Expense	Tracking the expenses based on their previous input data and monitor it regularly
Report	Report to the user when expenses cross the limit Report the through email Weekly once
Types of Income	Categories the income type (job, business, etc..)
Types of Expenses	Categories the Expenses type (food, medical, etc..)

4.2 NON-FUNCTIONAL REQUIREMENTS:

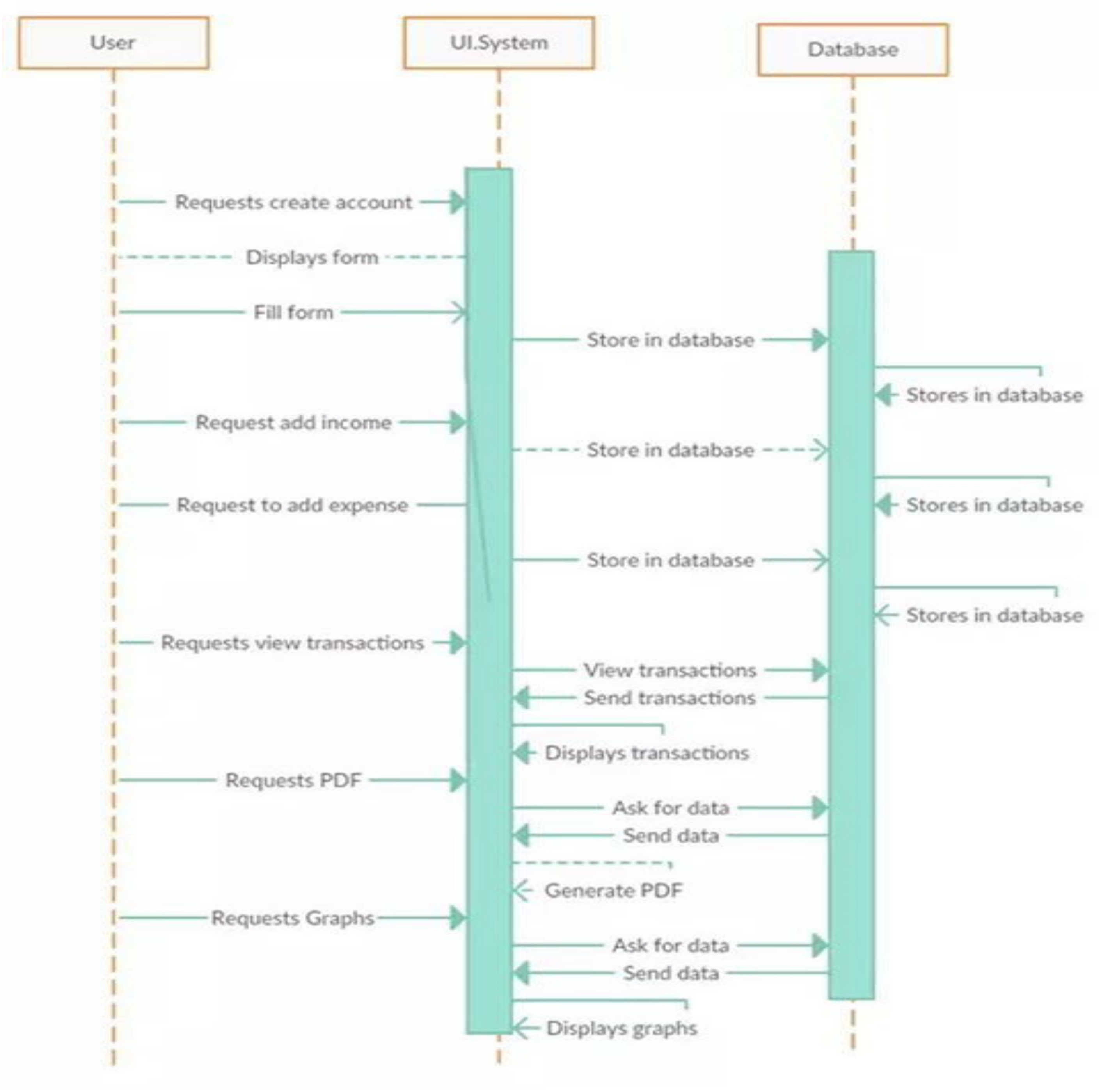
Following are the non-functional requirements of the proposed solution.



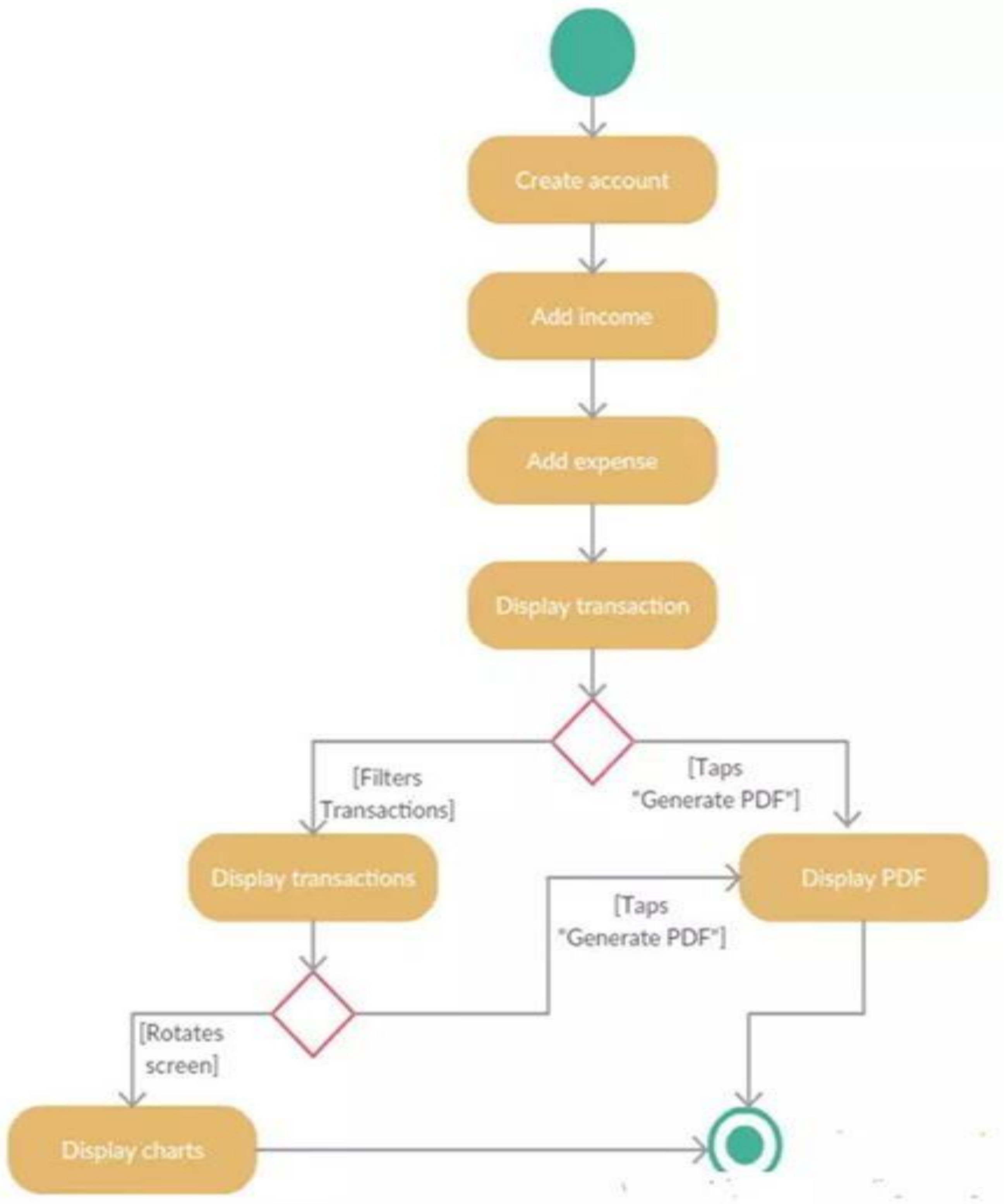
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The interface must be user friendly that makes it easy to use for all types of users.The basic features must be available free of cost to users.
NFR-2	**Security**	The application should have multi- factor authentication when logging in.Also, banking data must be secured by some encryption technology.
NFR-3	**Reliability**	The transaction must rollback if there is any technical or network issue .The data must be saved when updation of data fails in between the process. Even if there is a failure,it should be restored within a few minutes.
NFR-4	**Performance**	The application must not take more than 30 seconds to load. The response time should be quick even when there is heavy traffic.
NFR-5	**Availability**	When the app is being updated,except for the module that is being updated,the rest can be used.
NFR-6	**Scalability**	The app must be designed to work efficiently even when there is heavy traffic.

5.PROJECT DESIGN

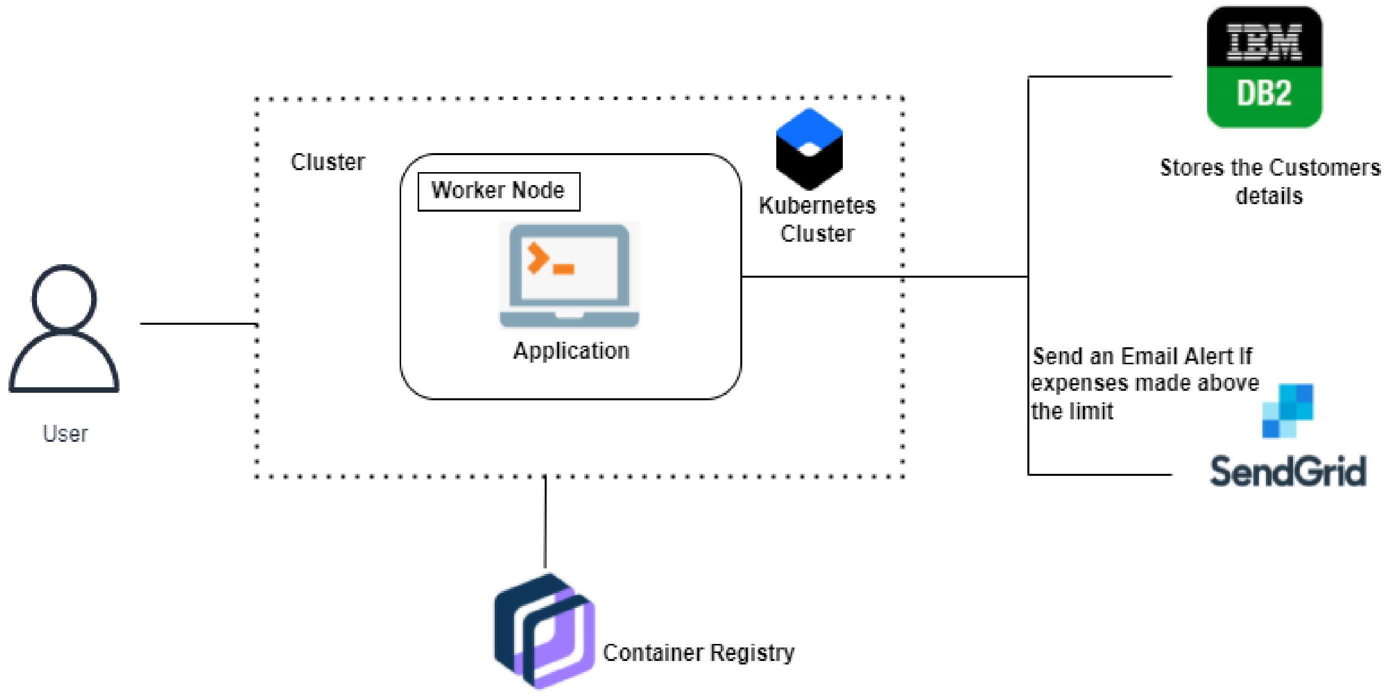
5.1 SEQUENCE DIAGRAM



5.2 ACTIVITY DIAGRAM



5.3 WORK FLOW DIAGRAM:



6.Project Planning

6.1 Milestone & Sprint:

6.1.1 Project Tracker, Velocity & Burndown Chart:

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

6.1.2 Average Velocity:

20 points/ 6 days = 3.34

6.1.3 Chart:

Graphical Representation

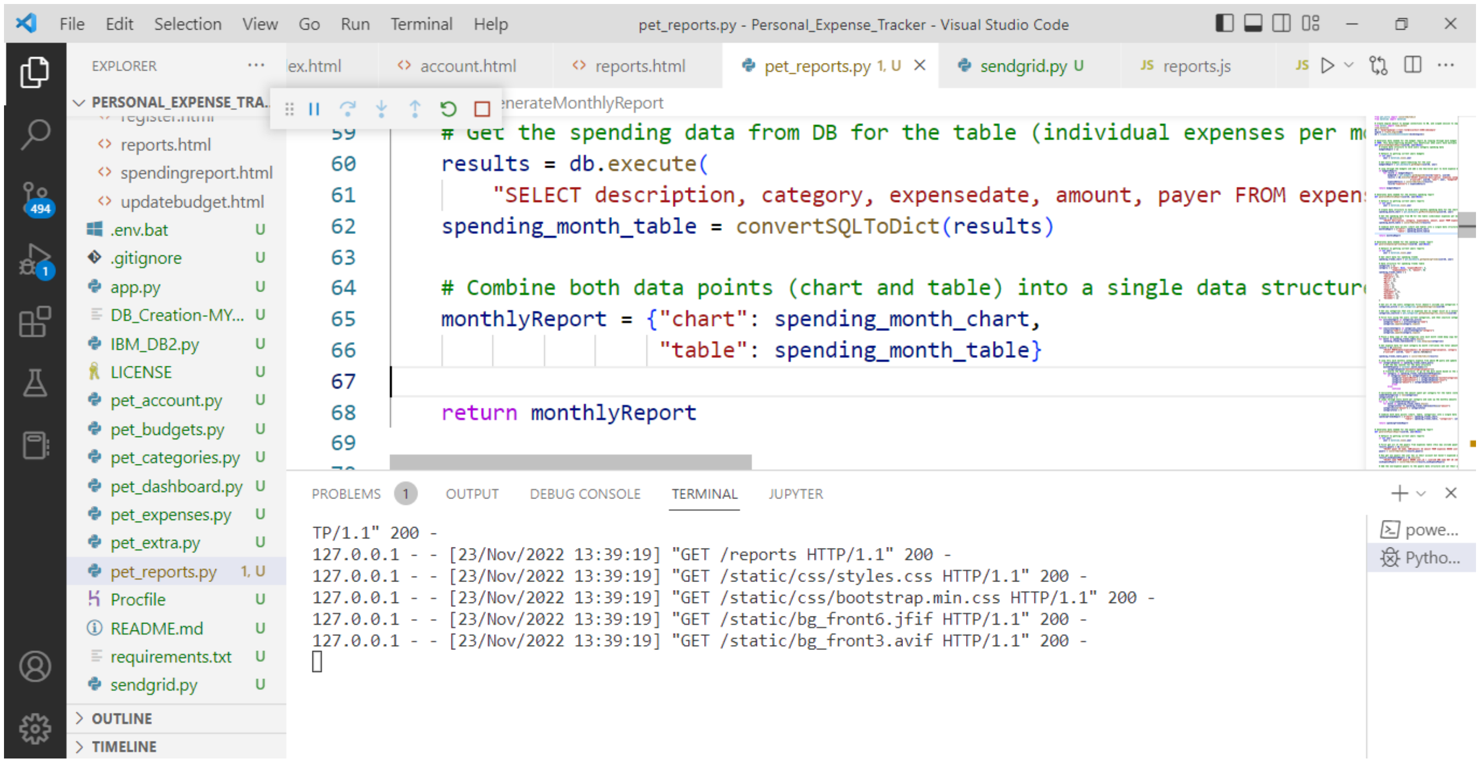
6.2 Sprint Delivery Plan:

Sprint	Functions	User Story / Task	Priority	Team Members
Sprint-1	Registration	Register details Username and Password	High	Mani Vasanthan, Aravinth
	Login	Login with corresponding details	High	Gireesh, Aravinth
Sprint-2	Design Budget	Allocate budget based on their salary	Medium	Mani Vaanthan, Gireesh
	Expense Details	Add, Update and Delete the expenses	Medium	Aravinth, Gireesh
Sprint-3	Connect to IBM DB2	Store and retrieve the data from IBM DB2	High	Mani Vasanthan
	Categories define	Define the categories of spending	low	Aravinth
Sprint-4	Pie-charts View	View in pictorial representation	low	Aravinth, Gireesh
	Generate Reports	Generate and email the reports	High	Aravinth, Mani Vasanthan, Gireesh

7.CODING & SOLUTIONING

7.1 Feature - 1

The Data are Available at in each of pages.

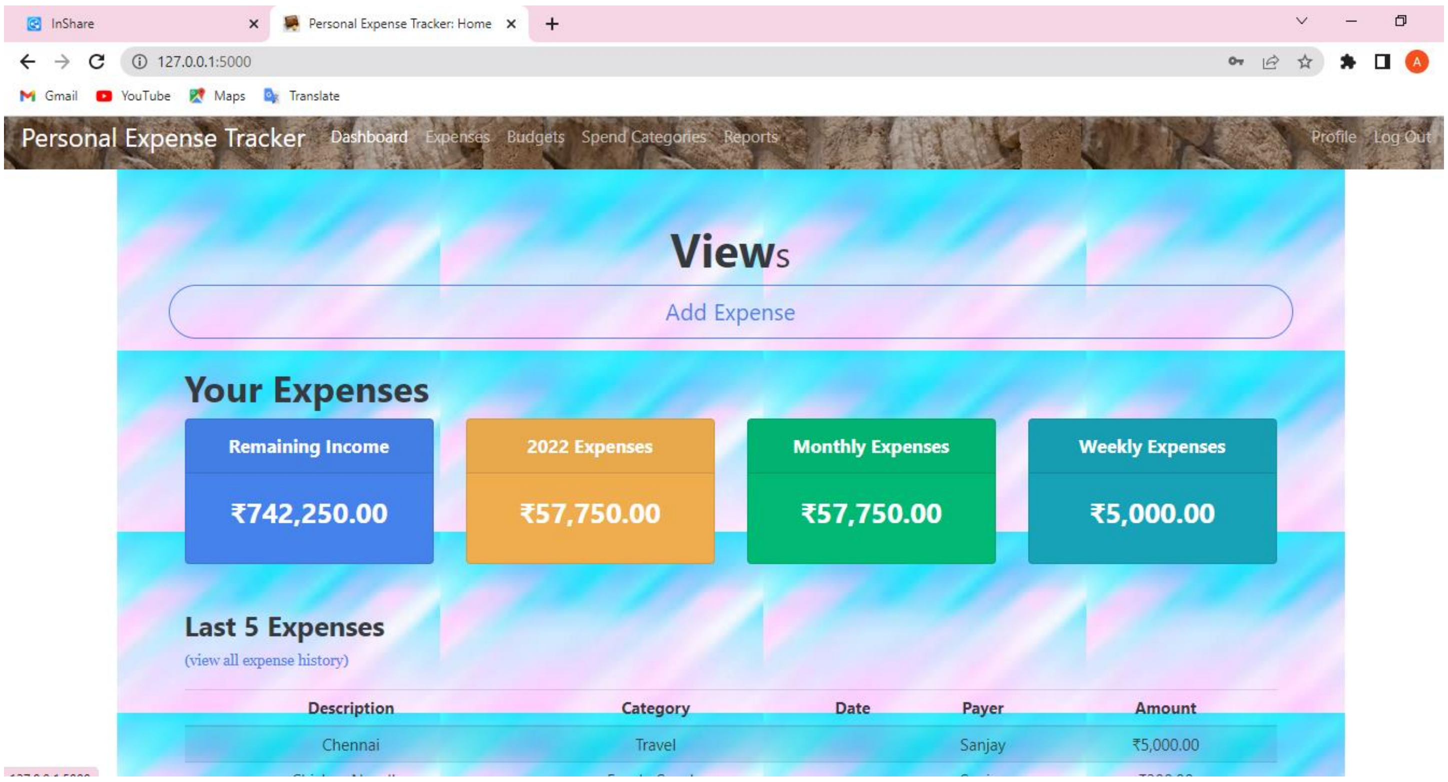


```python
    # Get the spending data from DB for the table (individual expenses per m
    results = db.execute(
        "SELECT description, category, expensedate, amount, payer FROM expens
    spending_month_table = convertSQLToDict(results)

    # Combine both data points (chart and table) into a single data structur
    monthlyReport = {"chart": spending_month_chart,
                     "table": spending_month_table}

    return monthlyReport
```

```
TP/1.1" 200 -
127.0.0.1 - - [23/Nov/2022 13:39:19] "GET /reports HTTP/1.1" 200 -
127.0.0.1 - - [23/Nov/2022 13:39:19] "GET /static/css/styles.css HTTP/1.1" 200 -
127.0.0.1 - - [23/Nov/2022 13:39:19] "GET /static/css/bootstrap.min.css HTTP/1.1" 200 -
127.0.0.1 - - [23/Nov/2022 13:39:19] "GET /static/bg_front6.jfif HTTP/1.1" 200 -
127.0.0.1 - - [23/Nov/2022 13:39:19] "GET /static/bg_front3.avif HTTP/1.1" 200 -
```

7.1.1 Dashboard

```python
import os
import calendar
import pet_budgets
from flask import request, session
from flask_session import Session
from sqlalchemy import create_engine
from sqlalchemy.orm import scoped_session, sessionmaker
from pet_extra import convertSQLToDict
from datetime import datetime

db = scoped_session(sessionmaker(bind=engine))
def getTotalSpend_Year(userID):
    totalSpendYear = convertSQLToDict(
        db.execute("SELECT SUM(amount) AS expenses_year FROM expenses WHERE user_id =
:usersID AND YEAR(date(expensedate)) = YEAR(CURRENT_DATE)",{"usersID": userID}).fetchall()
    )
    return totalSpendYear[0]['expenses_year']
def getTotalSpend_Month(userID):
    results = db.execute(
        "SELECT SUM(amount) AS expenses_month FROM expenses WHERE user_id = :usersID AND
YEAR(date(expensedate)) = YEAR(CURRENT_DATE) AND MONTH(date(expensedate)) =
MONTH(CURRENT_DATE)",
        {"usersID": userID}).fetchall()
    totalSpendMonth = convertSQLToDict(results)
    return totalSpendMonth[0]['expenses_month']

def getTotalSpend_Week(userID):
    results = db.execute(
        "SELECT SUM(amount) AS expenses_week FROM expenses WHERE user_id = :usersID AND
YEAR(date(expensedate)) = YEAR(CURRENT_DATE) AND WEEK(date(expensedate)) =
WEEK(CURRENT_DATE)",
        {"usersID": userID}).fetchall()
    totalSpendWeek = convertSQLToDict(results)
    return totalSpendWeek[0]['expenses_week']

def getLastFiveExpenses(userID):
    results = db.execute(
        "SELECT description, category, expenseDate, payer, amount FROM expenses WHERE
user_id = :usersID ORDER BY id DESC LIMIT 5", {"usersID": userID}).fetchall()
    lastFiveExpenses = convertSQLToDict(results)
    if lastFiveExpenses:
        return lastFiveExpenses
    else:
        return None

def getBudgets(userID, year=None):
    budgets = []
    budget = {"name": None, "amount": 0, "spent": 0, "remaining": 0}
    if not year:
        year = datetime.now().year
    budgets_query = pet_budgets.getBudgets(userID)
    if budgets_query and year in budgets_query:
        for record in budgets_query[year]:
            budgetID = record["id"]
            budget["name"] = record["name"]
            budget["amount"] = record["amount"]
            results = db.execute(
                "SELECT SUM(amount) AS spent FROM expenses WHERE user_id = :usersID AND
YEAR(date(expensedate)) = :year AND category IN (SELECT categories.name FROM
budgetcategories INNER JOIN categories on budgetcategories.category_id = categories.id
WHERE budgetcategories.budgets_id = :budgetID)",
                {"usersID": userID, "year": year, "budgetID": budgetID}).fetchall()
            budget_TotalSpent = convertSQLToDict(results)

            if (budget_TotalSpent[0]["spent"] == None):
                budget["spent"] = 0
            else:
                budget["spent"] = budget_TotalSpent[0]["spent"]

            if (budget["spent"] > budget["amount"]):
                budget["remaining"] = 0
            else:
                budget["remaining"] = budget["amount"] - budget["spent"]
            budgets.append(budget.copy())
        return budgets
```



Views

Your Expenses

Remaining Income	2022 Expenses	Monthly Expenses	Weekly Expenses
₹742,250.00	₹57,750.00	₹57,750.00	₹5,000.00

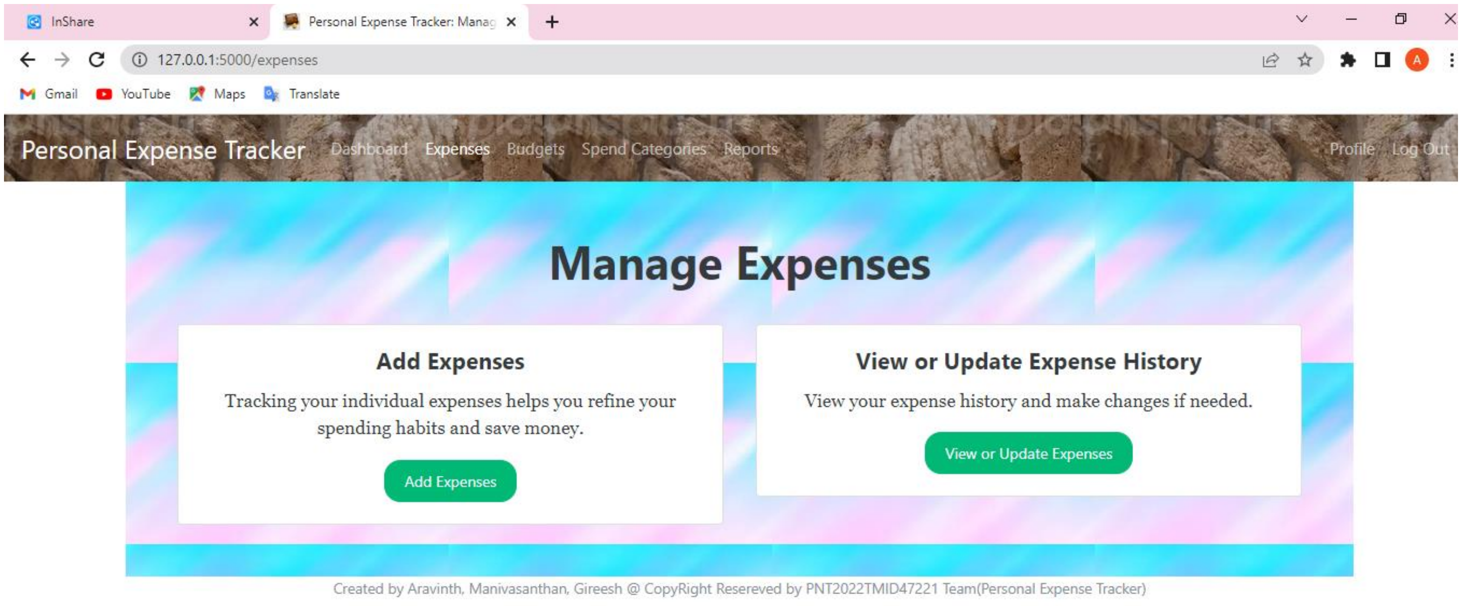
Last 5 Expenses

(view all expense history)

Description	Category	Date	Payer	Amount
Chennai	Travel		Sanjay	₹5,000.00

7.1.2 Expenses

```python
def addExpenses(formData, userID):
    expenses = []
    expense = {"description": None, "category": None,
               "date": None, "amount": None, "payer": None}
    if "." not in formData[0][0]:
        for key, value in formData:
            expense[key] = value.strip()
        expense["amount"] = float(expense["amount"])
        expenses.append(expense)
    else:
        counter = 0
        for key, value in formData:
            cleanKey = key.split(".")
            expense[cleanKey[0]] = value.strip()
            counter += 1

            if counter % 5 == 0:
                expense["amount"] = float(expense["amount"])
                expenses.append(expense.copy())
    for expense in expenses:
        now = datetime.now().strftime("%m/%d/%Y %H:%M:%S")
        db.execute("INSERT INTO expenses (description, category, expenseDate, amount,
payer, submitTime, user_id) VALUES (:description, :category, :expenseDate, :amount, :payer,
:submitTime, :usersID)",
                   {"description": expense["description"], "category": expense["category"],
"expenseDate": expense["date"], "amount": expense["amount"], "payer": expense["payer"],
"submitTime": now, "usersID": userID})
    db.commit()
    return expenses
def getHistory(userID):
    results = db.execute("SELECT description, category, expenseDate AS date, payer, amount,
submitTime FROM expenses WHERE user_id = :usersID ORDER BY id ASC",
                         {"usersID": userID}).fetchall()
    history = convertSQLToDict(results)
    return history
def getExpense(formData, userID):
    expense = {"description": None, "category": None,
               "date": None, "amount": None, "payer": None, "submitTime": None, "id": None}
    expense["description"] = formData.get("oldDescription").strip()
    expense["category"] = formData.get("oldCategory").strip()
    expense["date"] = formData.get("oldDate").strip()
    expense["amount"] = formData.get("oldAmount").strip()
    expense["payer"] = formData.get("oldPayer").strip()
    expense["submitTime"] = formData.get("submitTime").strip()
    expense["amount"] = float(expense["amount"].replace("₹", "").replace(",", ""))
    expenseID = db.execute("SELECT id FROM expenses WHERE user_id = :usersID AND
description = :oldDescription AND category = :oldCategory AND expenseDate = :oldDate AND
amount = :oldAmount AND payer = :oldPayer AND submitTime = :oldSubmitTime",
                           {"usersID": userID, "oldDescription": expense["description"],
"oldCategory": expense["category"], "oldDate": expense["date"], "oldAmount":
expense["amount"], "oldPayer": expense["payer"], "oldSubmitTime":
expense["submitTime"]}).fetchone()
    return expense
```



Manage Expenses

Add Expenses

Tracking your individual expenses helps you refine your spending habits and save money.

View or Update Expense History

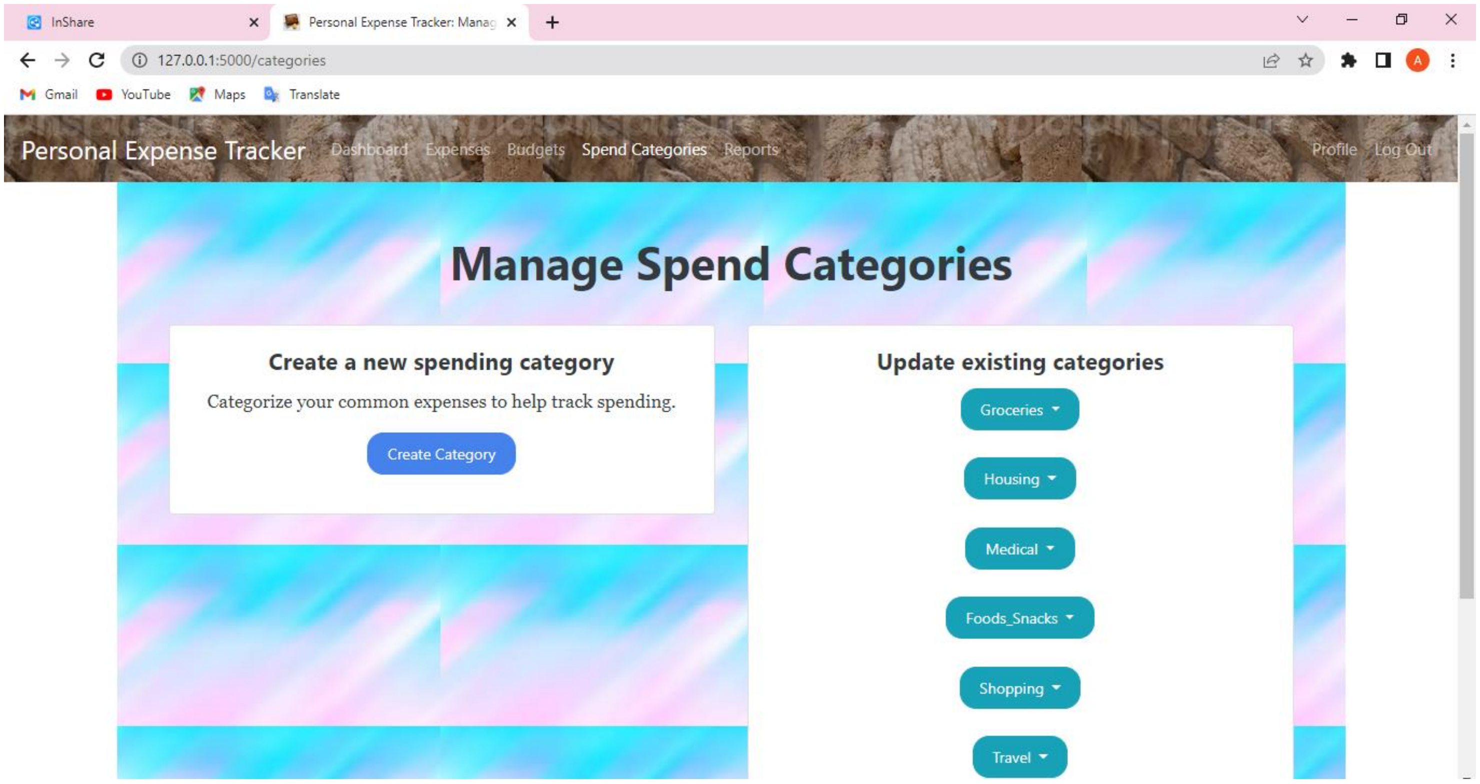
View your expense history and make changes if needed.

Created by Aravinth, Manivasanthan, Gireesh @ CopyRight Resereved by PNT2022TMID47221 Team(Personal Expense Tracker)

7.1.3 Spend Category

```
def getSpendCategories(userID):
    results = db.execute(
        "SELECT categories.name FROM usercategories INNER JOIN categories ON 
usercategories.category_id = categories.id WHERE usercategories.user_id = :usersID",
        {"usersID": userID}).fetchall()
    categories = convertSQLToDict(results)
    return categories
def getSpendCategories_Inactive(userID):
    results = db.execute(
        "SELECT category FROM expenses WHERE user_id = :usersID AND category NOT IN(SELECT 
categories.name FROM usercategories INNER JOIN categories ON categories.id = 
usercategories.category_id WHERE user_id = :usersID) GROUP BY category",
        {"usersID": userID}).fetchall()
    categories = convertSQLToDict(results)
    return categories
def getSpendCategoryLibrary():
    results = db.execute("SELECT id, name FROM categories").fetchall()
    return convertSQLToDict(results)
def getSpendCategoryName(categoryID):
    name = db.execute(
        "SELECT name FROM categories WHERE id = :categoryID", {"categoryID": 
categoryID}).fetchone()[0]
    return name
def getBudgetsSpendCategories(userID):
    results = db.execute("SELECT budgets.name AS budgetname, categories.id AS categoryid, 
categories.name AS categoryname FROM budgetcategories INNER JOIN budgets on 
budgetcategories.budgets_id = budgets.id INNER JOIN categories on 
budgetcategories.category_id = categories.id WHERE budgets.user_id = :usersID ORDER BY 
budgets.name, categories.name", {"usersID": userID}).fetchall()
    budgetsWithCategories = convertSQLToDict(results)
    return budgetsWithCategories
def getBudgetsFromSpendCategory(categoryID, userID):
    results = db.execute("SELECT budgets.id AS budgetid, budgets.name AS budgetname, 
categories.id AS categoryid, categories.name AS categoryname FROM budgetcategories INNER 
JOIN budgets on budgetcategories.budgets_id = budgets.id INNER JOIN categories on 
budgetcategories.category_id = categories.id WHERE budgets.user_id = :usersID AND 
budgetcategories.category_id = :categoryID ORDER BY budgets.name, categories.name", {
        "usersID": userID, "categoryID": categoryID}).fetchall()
    budgets = convertSQLToDict(results)
    return budgets
def updateSpendCategoriesInBudgets(budgets, oldCategoryID, newCategoryID):
    for budget in budgets:
        db.execute("UPDATE budgetcategories SET category_id = :newID WHERE budgets_id = 
:budgetID AND category_id = :oldID", {"newID": newCategoryID, "budgetID": 
budget["budgetid"], "oldID": oldCategoryID})
    db.commit()
def deleteSpendCategoriesInBudgets(budgets, categoryID):
    for budget in budgets:
        db.execute("DELETE FROM budgetcategories WHERE budgets_id = :budgetID AND 
category_id = :categoryID",{"budgetID": budget["budgetid"], "categoryID": categoryID})
    db.commit()
```

```
def generateSpendCategoriesWithBudgets(categories, categoryBudgets):
    categoriesWithBudgets = []
    for category in categories:
        categoryWithBudget = {"name": None, "budgets": []}
        categoryWithBudget["name"] = category["name"]
        for budget in categoryBudgets:

            if category["name"] == budget["categoryname"]:
                categoryWithBudget["budgets"].append(budget["budgetname"])
        categoriesWithBudgets.append(categoryWithBudget)
    return categoriesWithBudgets
```



Manage Spend Categories

Create a new spending category

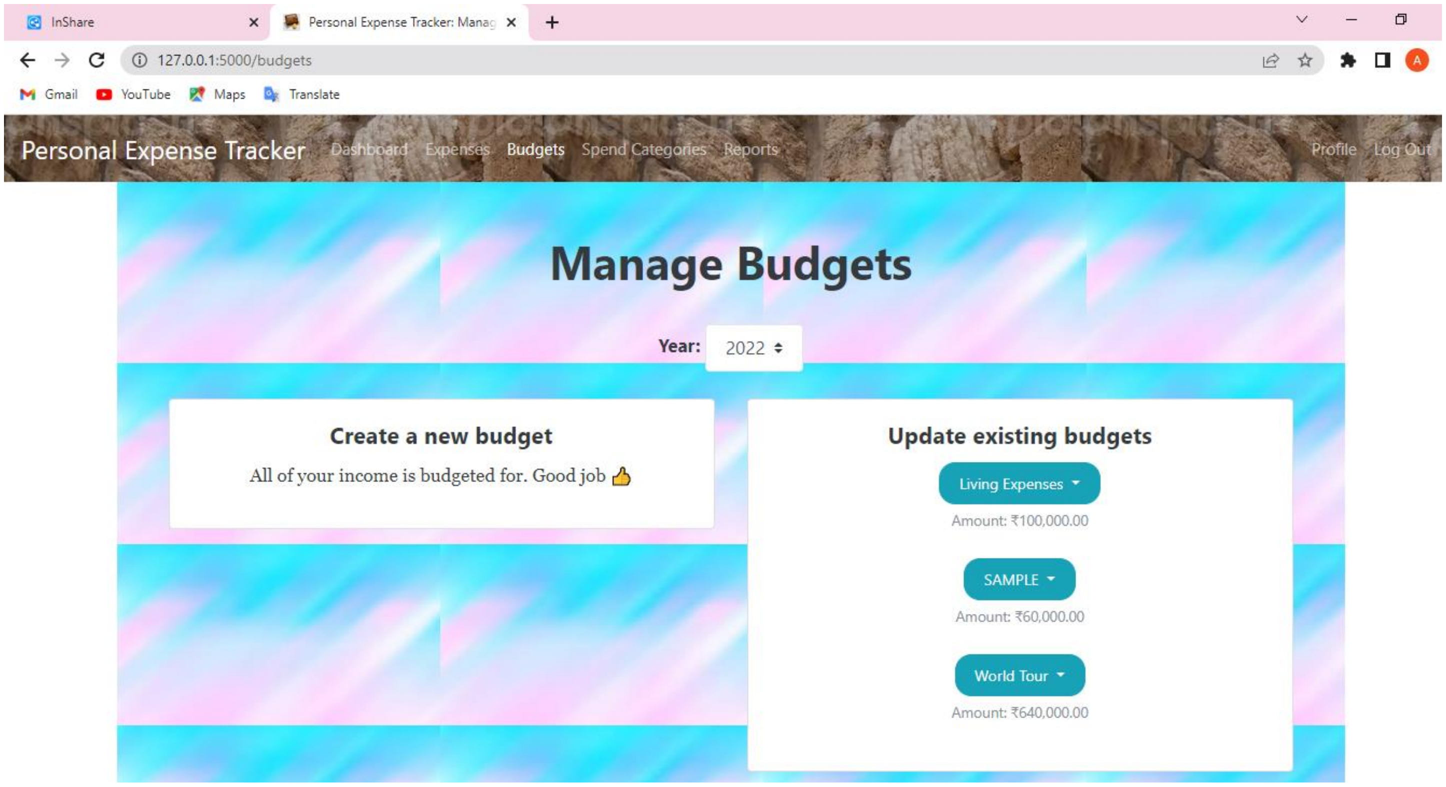
Categorize your common expenses to help track spending.

Update existing categories

7.1.4 Budgets

```
def getBudgets(userID):
    results = db.execute(
        "SELECT id, name, year, amount FROM budgets WHERE user_id = :usersID ORDER BY name
ASC", {"usersID": userID}).fetchall()
    budgets_query = convertSQLToDict(results)
    if budgets_query:
        budgets = {budget['year']: [] for budget in budgets_query}
        for budget in budgets_query:
            budgets[budget['year']].append({'amount': budget['amount'], 'id': budget['id'],
'name': budget['name']})
        return budgets
    return None
def getBudgetByID(budgetID, userID):
    budget = convertSQLToDict(db.execute(
        "SELECT name, amount, year, id FROM budgets WHERE user_id = :usersID AND id = 
:budgetID", {"usersID": userID, "budgetID": budgetID}).fetchall())
    return budget[0]

def getTotalBudgetedByYear(userID, year=None):
    if not year:
        year = datetime.now().year
    amount = db.execute( "SELECT SUM(amount) AS amount FROM budgets WHERE user_id = 
:usersID AND year = :year", {"usersID": userID, "year": year}).fetchone()[0]
    if amount is None:
        return 0
    else:
        return amount

def generateBudgetFromForm(formData):
    budget = {"name": None, "year": None, "amount": None, "categories": []}
    counter = 0
    for key, value in formData:
        counter += 1
        if counter <= 3:
            if key == "name":
                validBudgetName = re.search("^([a-zA-Z0-9_\s\-]*)$", value)
                if validBudgetName:
                    budget[key] = value.strip()
                else:
                    return {"apology": "Please enter a budget name without special 
characters except underscores, spaces, and hyphens"}
            elif key == "year":
                budgetYear = int(value)
                currentYear = datetime.now().year

                if 2020 <= budgetYear <= currentYear:
                    budget[key] = budgetYear
                else:
                    return {"apology": f"Please select a valid budget year: 2020 through 
{currentYear}"}
            else:
                amount = float(value.strip())
                budget[key] = amount
        else:
            if value == '':
                continue
            cleanKey = key.split(".")
            category = {"name": None, "percent": None}
            if cleanKey[0] == "categories":
                category["name"] = value.strip()
                percent = (int(formData[counter][1].strip()) / 100)
                category["percent"] = percent
                budget[cleanKey[0]].append(category)
            elif cleanKey[0] == "categoryPercent":
                pass
            else:
                return {"apology": "Only categories and their percentage of the overall 
budget are allowed to be stored"}
```

```
    return budget
```



Manage Budgets

Year: 2022

Create a new budget

All of your income is budgeted for. Good job 👍

Update existing budgets

Living Expenses

Amount: ₹100,000.00

SAMPLE

Amount: ₹60,000.00

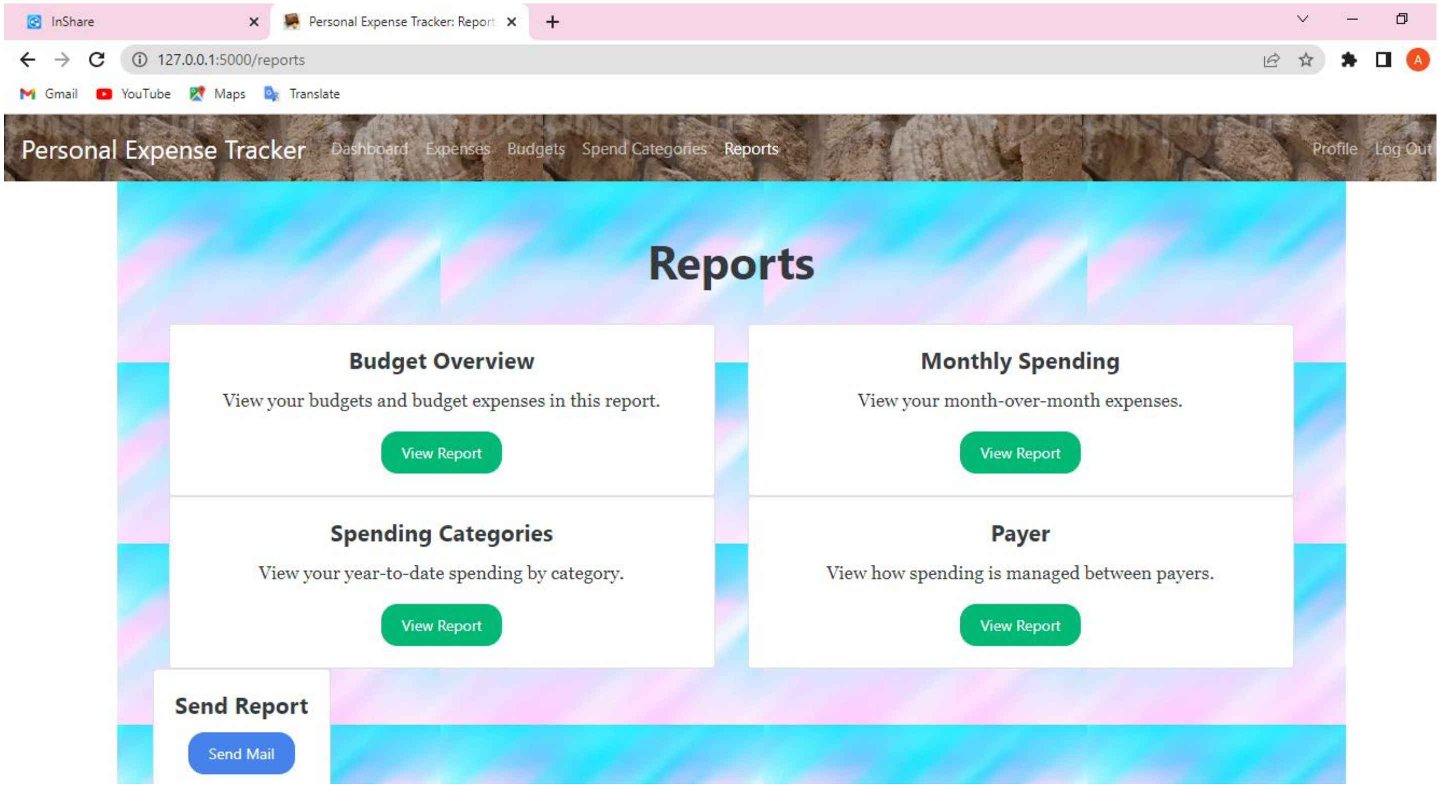
World Tour

Amount: ₹640,000.00

7.1.5 Reports

```
def generateBudgetsReport(userID, year=None):
    budgetsReport = []
    if not year:
        year = datetime.now().year
    budgetsReport = pet_dashboard.getBudgets(userID, year)
    if budgetsReport:
        for record in budgetsReport:
            budgetID = pet_budgets.getBudgetID(record["name"], userID)
            results = db.execute("SELECT expenses.description, expenses.category,
expenses.expenseDate, expenses.payer, expenses.amount FROM expenses WHERE user_id =
:usersID AND YEAR(date(expensedate)) = :year AND category IN (SELECT categories.name FROM
budgetcategories INNER JOIN categories on budgetcategories.category_id = categories.id
WHERE budgetcategories.budgets_id = :budgetID)",({ usersID": userID, "year": year,
"budgetID": budgetID}).fetchall()
            expenseDetails = convertSQLToDict(results)
            record["expenses"] = expenseDetails
    return budgetsReport
def generateMonthlyReport(userID, year=None):
    if not year:
        year = datetime.now().year
    spending_month_chart = pet_dashboard.getMonthlySpending(userID, year)
    results = db.execute(
        "SELECT description, category, expensedate, amount, payer FROM expenses WHERE
user_id = :usersID AND YEAR(date(expensedate)) = :year ORDER BY id ASC", {"usersID":
```

```
userID, "year": year}).fetchall()
    spending_month_table = convertSQLToDict(results)
    monthlyReport = {"chart": spending_month_chart,
                     "table": spending_month_table}
    return monthlyReport
```



Reports

Budget Overview

View your budgets and budget expenses in this report.

View Report

Monthly Spending

View your month-over-month expenses.

View Report

Spending Categories

View your year-to-date spending by category.

View Report

Payer

View how spending is managed between payers.

View Report

Send Report

Send Mail

7.2 Feature – 2

Data visualization on methods for expenditure are added. The pie chart is a pictorial representation of data that makes it possible to visualize the relationships between the parts andthe whole of a variable. The pie chart have been used to represent the weekly, monthly and payers expenses.

The recommended use for pie charts is two- dimensional, as three-dimensional use can be confusing. The dimensions form sectors of the measurement values. They can have one or twosizes and up to two measures.

The first dimension is used to define the angle of each sector that makes up the chart and the second dimension optionally determines the radius of each sector. Additionally, these plots are useful for comparing data over a fixed period since they do not show changes over time.

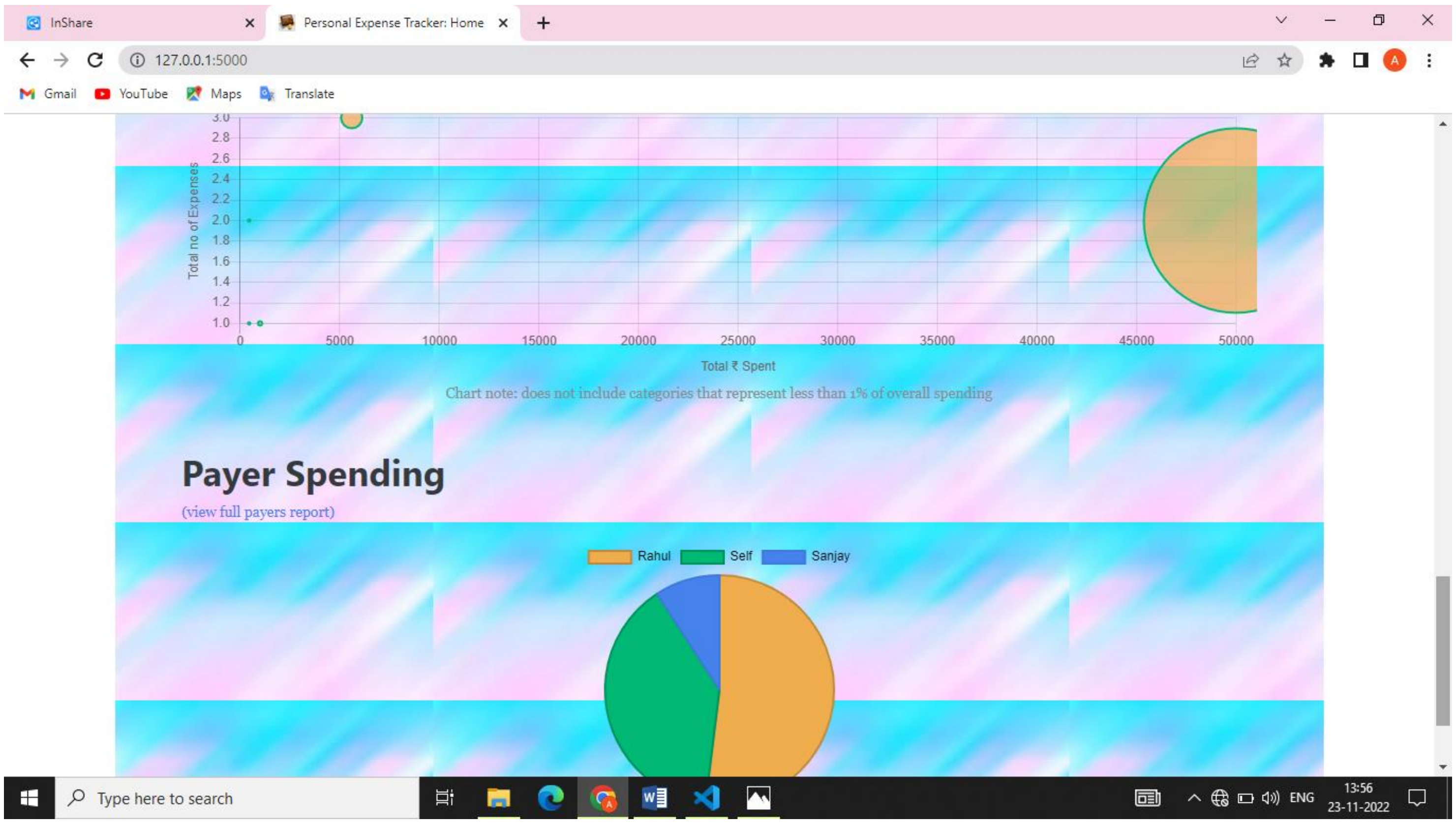


Chart note: does not include categories that represent less than 1% of overall spending

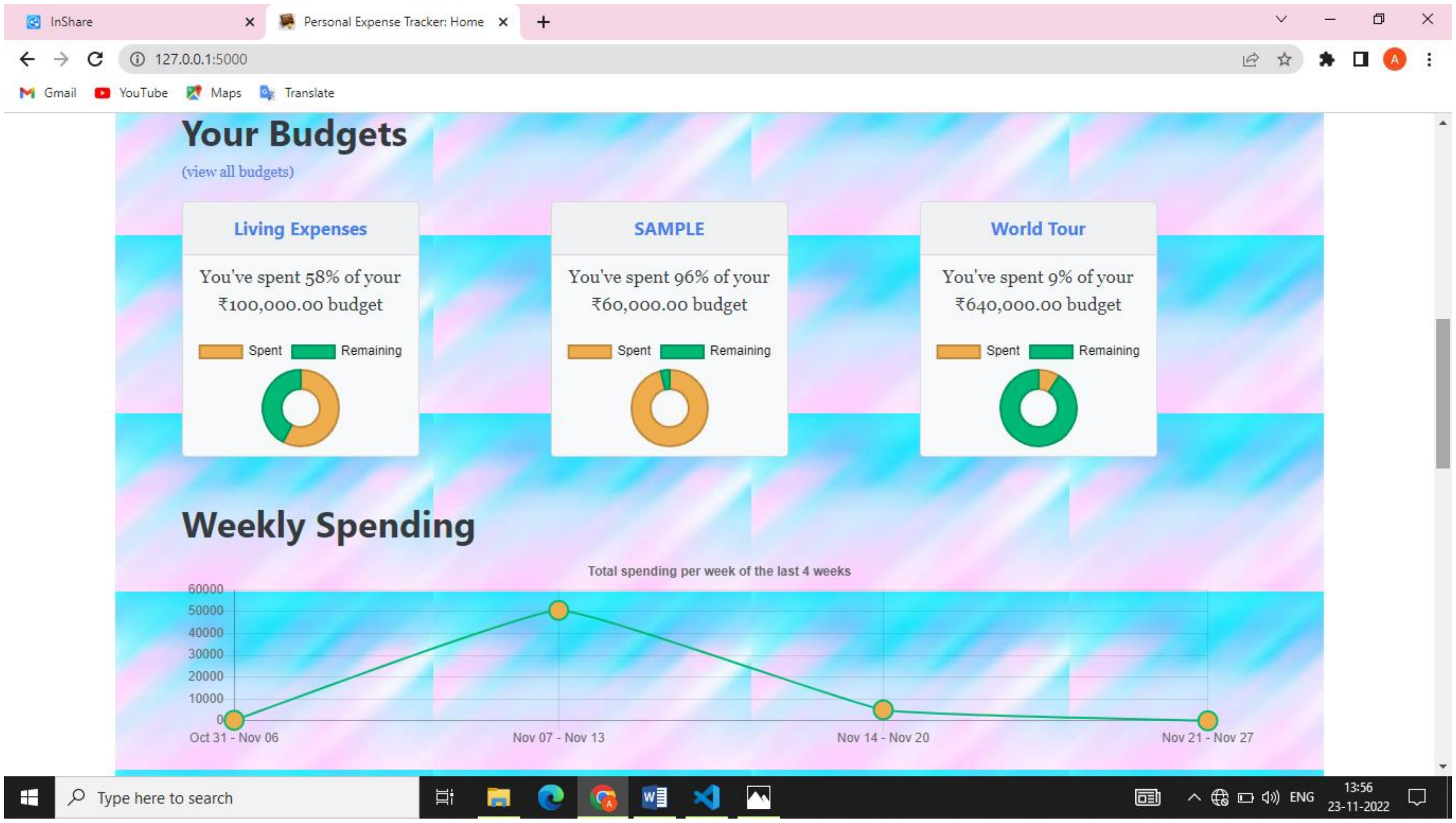
Payer Spending

(view full payers report)

```
function loadBudgetCharts(budgets) {
    if (budgets == null) {
        return;
    }
    else {
        let chartElement = []
        let budgetCharts = []
        for (i = 0; i < budgets.length; i++) {
            chartElement[i] = document.getElementById('budgetChart.' +
(i)).getContext('2d');
            budgetCharts[i] = new Chart(chartElement[i], {
                type: 'doughnut',
                data: {
                    labels: ['Spent', 'Remaining'],
                    datasets: [{
                        label: budgets[i].name,
                        data: [(Math.round(budgets[i].spent * 100) / 100),
(Math.round(budgets[i].remaining * 100) / 100)],
                        backgroundColor: [
                            'rgba(240, 173, 78, 1)',
                            'rgba(2, 184, 117, 1)'
                        ],
                        borderColor: [
                            'rgba(192, 138, 62, 1)',
                            'rgba(1, 147, 93, 1)'
                        ],
```

```
                        borderWidth: 2
```

```
                }]
            },
            options: {
                responsive: true,
                maintainAspectRatio: false,
                legend: {
                    labels: {
                        fontColor: 'black'
                              }
                        }
                    }
               });
          }
     }
}
```



Your Budgets

(view all budgets)

Living Expenses

You've spent 58% of your ₹100,000.00 budget

Spent Remaining

SAMPLE

You've spent 96% of your ₹60,000.00 budget

Spent Remaining

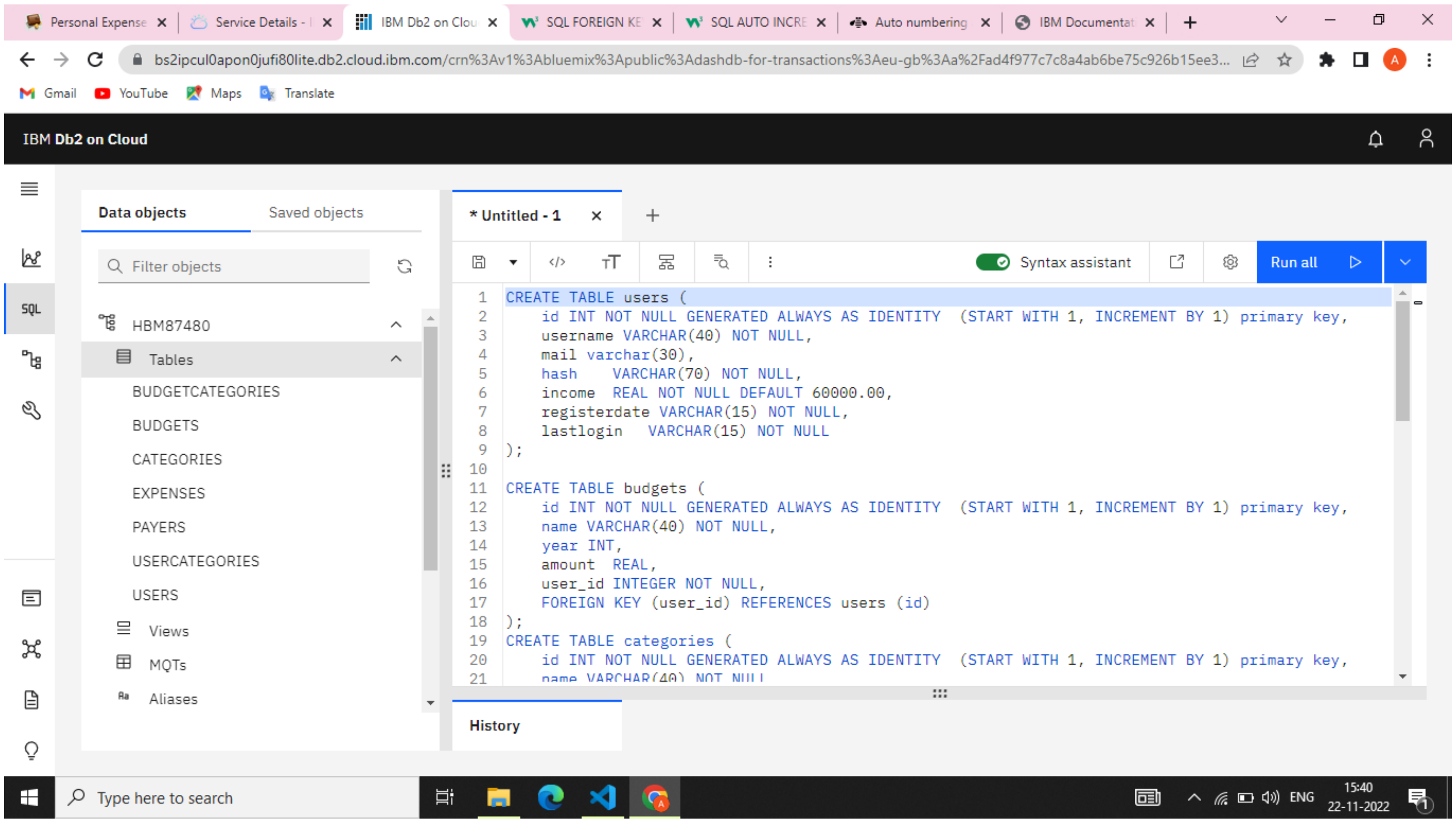
World Tour

You've spent 9% of your ₹640,000.00 budget

Spent Remaining

Weekly Spending

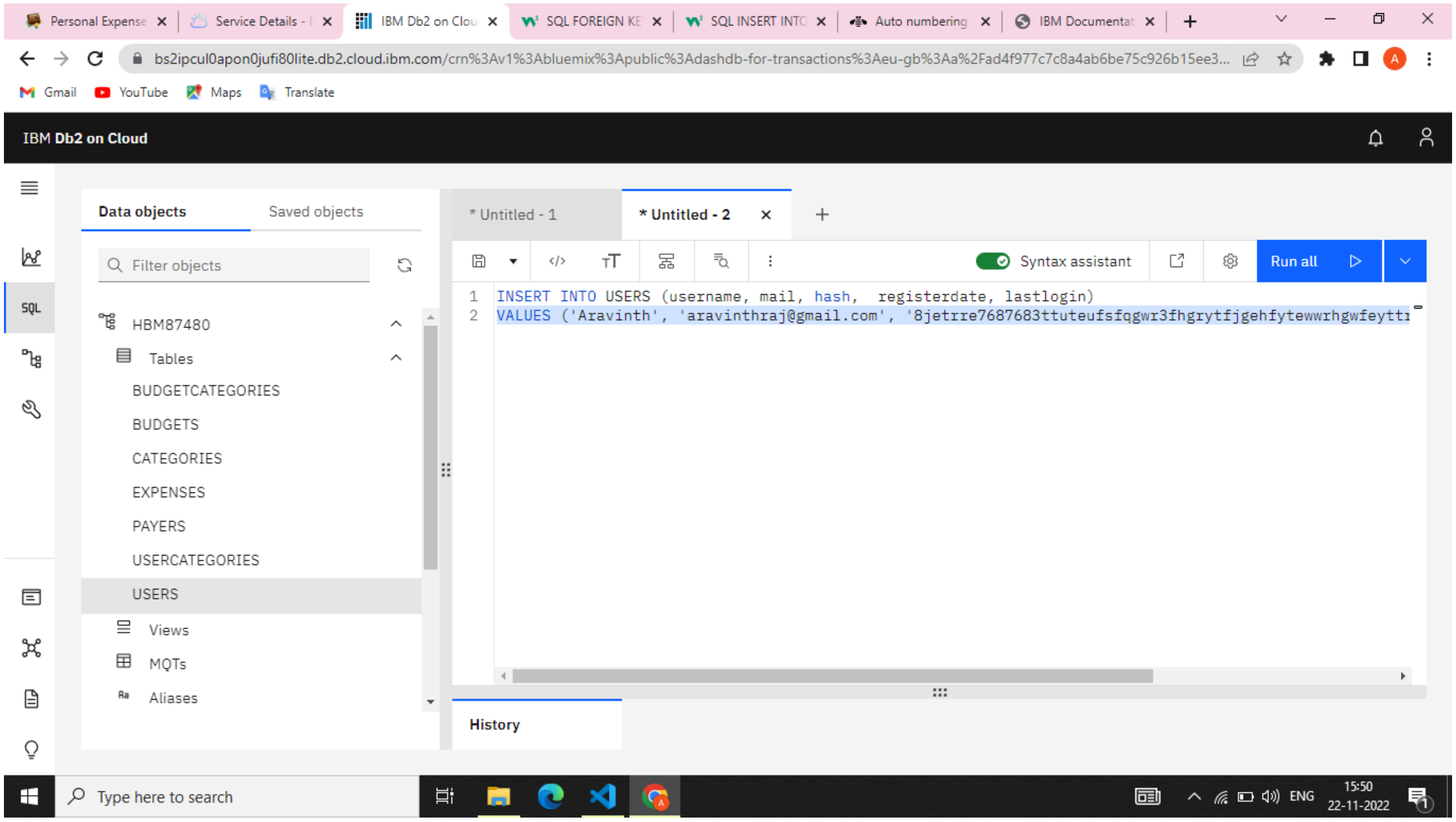
7.3 DATABASE IBM DB2:



```
CREATE TABLE users (
    id INT NOT NULL GENERATED ALWAYS AS IDENTITY  (START WITH 1, INCREMENT BY 1) primary key,
    username VARCHAR(40) NOT NULL,
    mail varchar(30),
    hash    VARCHAR(70) NOT NULL,
    income  REAL NOT NULL DEFAULT 60000.00,
    registerdate VARCHAR(15) NOT NULL,
    lastlogin   VARCHAR(15) NOT NULL
);

CREATE TABLE budgets (
    id INT NOT NULL GENERATED ALWAYS AS IDENTITY  (START WITH 1, INCREMENT BY 1) primary key,
    name VARCHAR(40) NOT NULL,
    year INT,
    amount  REAL,
    user_id INTEGER NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users (id)
);
CREATE TABLE categories (
    id INT NOT NULL GENERATED ALWAYS AS IDENTITY  (START WITH 1, INCREMENT BY 1) primary key,
    name VARCHAR(40) NOT NULL
```

```
use <database_name>;

CREATE TABLE IF NOT EXISTS users (
        id          INTEGER NOT NULL auto_increment PRIMARY KEY,
        username            TEXT NOT NULL,
        mail varchar2(30),
        hash        TEXT NOT NULL,
        income  REAL NOT NULL DEFAULT 60000.00,
        registerdate        TEXT NOT NULL,
        lastlogin           TEXT NOT NULL
);

CREATE TABLE IF NOT EXISTS budgets (
        id          INTEGER NOT NULL auto_increment PRIMARY KEY,
        name        TEXT NOT NULL,
   year INT,
        amount REAL,
        user_id INTEGER NOT NULL,
        CONSTRAINT budgets_user_id_fkey FOREIGN KEY (user_id)
                REFERENCES users (id) MATCH SIMPLE
                ON UPDATE NO ACTION ON DELETE NO ACTION
);
CREATE TABLE IF NOT EXISTS categories (
        id          INTEGER NOT NULL auto_increment PRIMARY KEY,
        name        TEXT NOT NULL
);
```

```
CREATE TABLE IF NOT EXISTS userCategories (
	category_id	INTEGER NOT NULL,
	user_id INTEGER NOT NULL,
	CONSTRAINT userCategories_category_id_fkey FOREIGN KEY (category_id)
		REFERENCES categories (id) MATCH SIMPLE
		ON UPDATE NO ACTION ON DELETE NO ACTION,
	CONSTRAINT userCategories_user_id_fkey FOREIGN KEY (user_id)
		REFERENCES users (id) MATCH SIMPLE
		ON UPDATE NO ACTION ON DELETE NO ACTION
);

CREATE TABLE IF NOT EXISTS budgetCategories (
	budgets_id	INTEGER NOT NULL,
	category_id	INTEGER NOT NULL,
	amount REAL NOT NULL DEFAULT 0,
	CONSTRAINT budgetCategories_budgets_id_fkey FOREIGN KEY (budgets_id)
		REFERENCES budgets(id) MATCH SIMPLE
		ON UPDATE NO ACTION ON DELETE NO ACTION,
	CONSTRAINT budgetCategories_category_id_fkey FOREIGN KEY (category_id)
		REFERENCES categories (id) MATCH SIMPLE
		ON UPDATE NO ACTION ON DELETE NO ACTION
);

CREATE TABLE IF NOT EXISTS payers (
	user_id INTEGER NOT NULL,
	name	TEXT NOT NULL,
	CONSTRAINT payers_user_id_fkey FOREIGN KEY (user_id)
		REFERENCES users (id) MATCH SIMPLE
		ON UPDATE NO ACTION ON DELETE NO ACTION
);

CREATE TABLE IF NOT EXISTS expenses (
	id	INTEGER NOT NULL auto_increment PRIMARY KEY,
	description	TEXT NOT NULL,
	category	TEXT NOT NULL,
	expensedate	TEXT NOT NULL,
	amount REAL NOT NULL,
	payer	TEXT NOT NULL,
	submittime	TEXT NOT NULL,
	user_id INTEGER,
	CONSTRAINT budgets_user_id_fkey1 FOREIGN KEY (user_id)
		REFERENCES users (id) MATCH SIMPLE
		ON UPDATE NO ACTION ON DELETE NO ACTION
);

INSERT INTO categories(name) VALUES ('Groceries');
INSERT INTO categories(name) VALUES ('Housing');
INSERT INTO categories(name) VALUES ('Medical');
INSERT INTO categories(name) VALUES ('Foods_Snacks');
INSERT INTO categories(name) VALUES ('Shopping');
INSERT INTO categories(name) VALUES ('Travel');
INSERT INTO categories(name) VALUES ('Entertainment');
INSERT INTO categories(name) VALUES ('Other');
```



```
INSERT INTO USERS (username, mail, hash,  registerdate, lastlogin)
VALUES ('Aravinth', 'aravinthraj@gmail.com', '8jetrre7687683ttuteufsfqgwr3fhgrytfjgehfytewwrhgwfeytti
```

8.TESTING

8.1 TEST CASES

TEST CASE ID	PURPOSE	TESTCASES	RESULT
TC1	Authentication	Password Check	Pass
TC2	Validate	Check Regular Expression for all	Pass
TC3	Add Income	Add incomes in wallet	Pass
TC4	Add expenses	Add Expenses then checks	Pass
TC5	Budgets	Allocate budgets	Pass
TC6	Categories Check	Define types categories you spend	Pass
TC7	Reports	Prepare Reports	Pass
C8	Mail	Send mail	Pass

8.2 USER ACCEPTANCE TESTING:

The testing done by the lot of my college students and Faculty. Out side of College the testing and website are published and reviews the positive feedback.

9.RESULTS

9.1 PERFORMANCE METRICS

● Tracking income and expenses: Monitoring the income and tracking all expenditures

● Transaction Reports: Generates reports

● Payments & Invoices: Accept and pay from credit cards, debit cards, net banking, mobile wallets, and bank transfers, and track the status of your invoices and bills in the mobile app itself. Also, the tracking app sends reminders for payments and automatically matches the payments with invoices.

● Reports: The expense tracking app generates and sends reports to give a detailed insight aboutprofits, losses, budgets, income, balance sheets, etc.,

● Access control: Increase your team productivity by providing access control to particular usersthrough custom permissions.

● Vendors and Contractors: Manage and track all the payments to the vendors and contractorsadded to the mobile app.

● Organizing Taxes: Import your documents to the expense tracking app, and it will streamlineyour income and expenses under the appropriate tax categories.

● Track Projects: Determine project profitability by tracking labor costs, payroll, expenses, etc., ofyour ongoing project.

●In-depth insights and analytics: Provides in-built tools to generate reports with easy-to- understand visuals and graphics to gain insights about the performance of your business.

● Recurrent Expenses: Rely on your budgeting app to track, streamline, and automate all therecurrent expenses and remind you

10.ADVANTAGES & DISADVANTAGES

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

11.CONCLUSION

From this project, we are able to manage and keep tracking the daily expenses as wellas income. While making this project, we gained a lot of experience of working as a team. We discovered various predicted and unpredicted problems and we enjoyed a lot solving them as a team. We adopted things like video tutorials, text tutorials, internet and learning materials to makeour project complete.

BEFORE Our Application

Fear of spending lot of money and couldn't manage their expenses.

AFTER Our Application

They can manage their expense

12.FUTURE

The project assists well to record the income and expenses in general. However, this project hassome limitations:

- This application does not provide higher decision capability.
- The application is unable to maintain the backup of data once it is uninstalled.

To further enhance the capability of this application, we recommend the following features to be incorporated into the system:

- Mobile apps
- Multiple language interface.
- Provide backup and recovery of data.
- Better user interface for user.

13.APPENDIX

SOURCE CODE GITHUB LINK:

https://github.com/karthiaravinth/IBM-Project-51228-1660976272