# Personal Expense Tracker Application

## Project Report

| | |
|---|---|
| ]Team ID | PNT2022TMID04980 |
| Project Name | Project – Personal Expense Tracker Application |
| Team Members | **KARTHICKNAVEEN KISHORE NISHANTH MOHAMED AFRITH KABILAN** |

## CONTENTS

# 1.INTRODUCTION

## 1.1Project overview

Mobile applications are top in user convenience and have over passed the web applications in terms of popularity and usability. There are various mobile applications that provide solutions to manage personal and group expense but not many of them provide a comprehensive view of both cases. In this paper, we develop a mobile application developed for the android platformthat keeps record of user personal expenses, his/her contributionin group expenditures, top investment options, view of the current stock market, read authenticated financial news and grabthe best ongoing offers in the market in popular categories. Withour application can manage their expenses and decide on their budget more effectively.

## 1.2Purpose

It also known as expense manager and money manager, an expense tracker is a software or application that helps to keep anaccurate record of your money inflow and outflow. Many peoplein India live on a fixed income, and they find that towards the end of the month they don't have sufficient money to meet theirneeds.

# 2.LITERATURE SURVEY

## 2.1Existing Problem

The problem of current generation population is that they can’t remember where all of the money they earned have gone andultimately have to live while sustaining the little money they have left for their essential needs. In this time there is no such perfect solution which helps a person to track their daily expenditure easilyand efficiently and notify them about the money shortage they have. For doing so have to maintain long ledgers or computer logs to maintain such data and the calculation is done manually by the user, which may generate error leading to losses.
Not having a complete tracking.

## 2.2Reference

•https://nevonprojects.com/daily-expense-tracker-system/

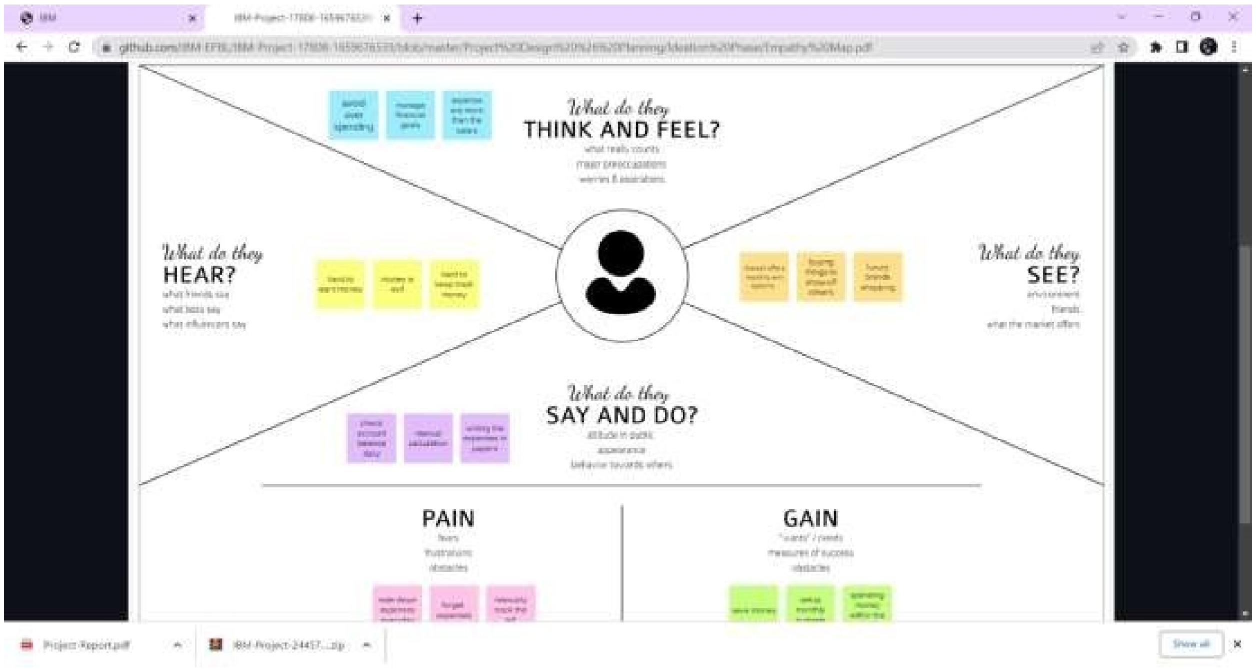
•https://data-flair.training/blogs/expense-tracker-python/
•https://phpgurukul.com/daily-expense-tracker-using-php-and-mysql/
•https://ijarsct.co.in/Paper391.pdf
•https://kandi.openweaver.com/?landingpage=python_all_projects&utm_sour
ce=google&utm_med
ium=cpc&utm_campaign=promo_kandi_ie&utm_content=kandi_ie_search
&utm_term=python_d
evs&gclid=Cj0KCQiAgribBhDkARIsAASA5bukrZgbI9UZxzpoyf0PofB1mZNxzcokUP-
3TchpYMclHTYFYiqP8aAmmwEALw_wcB

## 2.3Problem Statement Definition

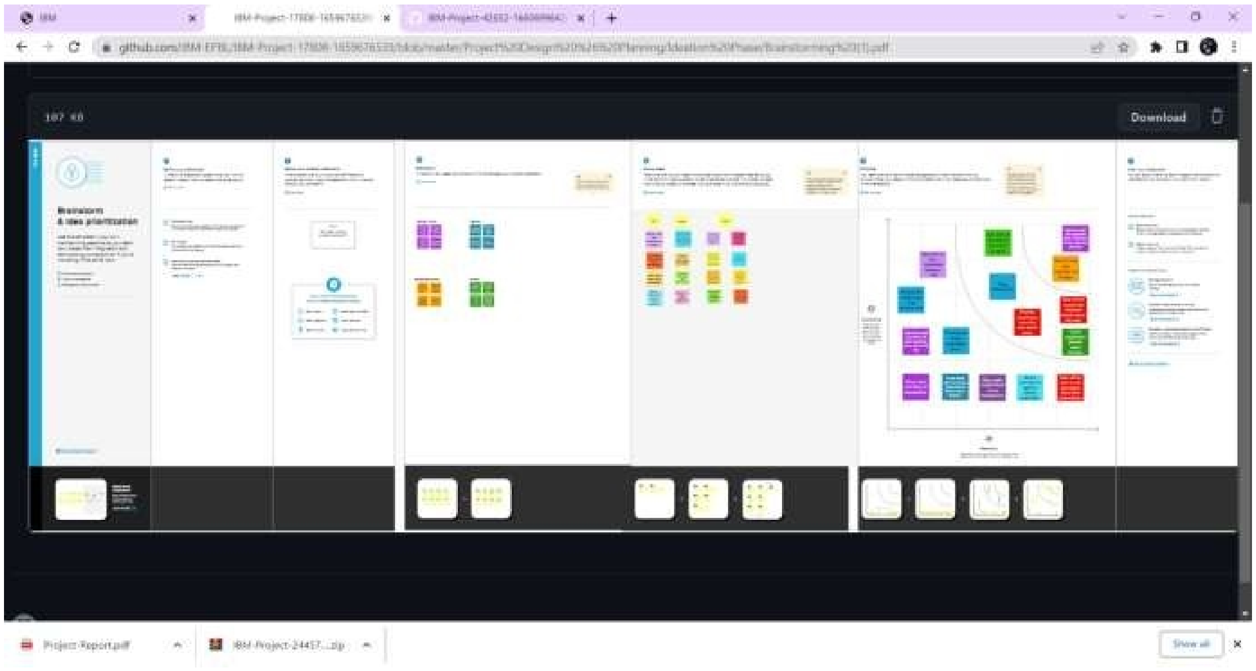
This Expense Tracker is a web application that facilitates the users to keep trackand manage their personal as well as business expenses. This application helpsthe users to keep a digital diary. It will keep track of a user's income and expenses on a daily basis. The user will be able to add his/her expenditures instantly and can review them anywhere and anytime with the help of the internet. He/she can easily import transactions from his/her mobile wallets without risking his/her information and efficiently protecting his/her privacy.
This expense tracker provides a complete digital solution to this problem. Excelsheets do very little to help in tracking Furthermore, they don't have the advanced functionality of preparing graphical visuals automatically. Not only
it will save the time of the people but also it will assure error free calculations. The user just has to enter the income and expenditures and everything else will be performed by the system.
Keywords: Expense Tracker, budget, planning,savings, graphical visualization of expenditure.

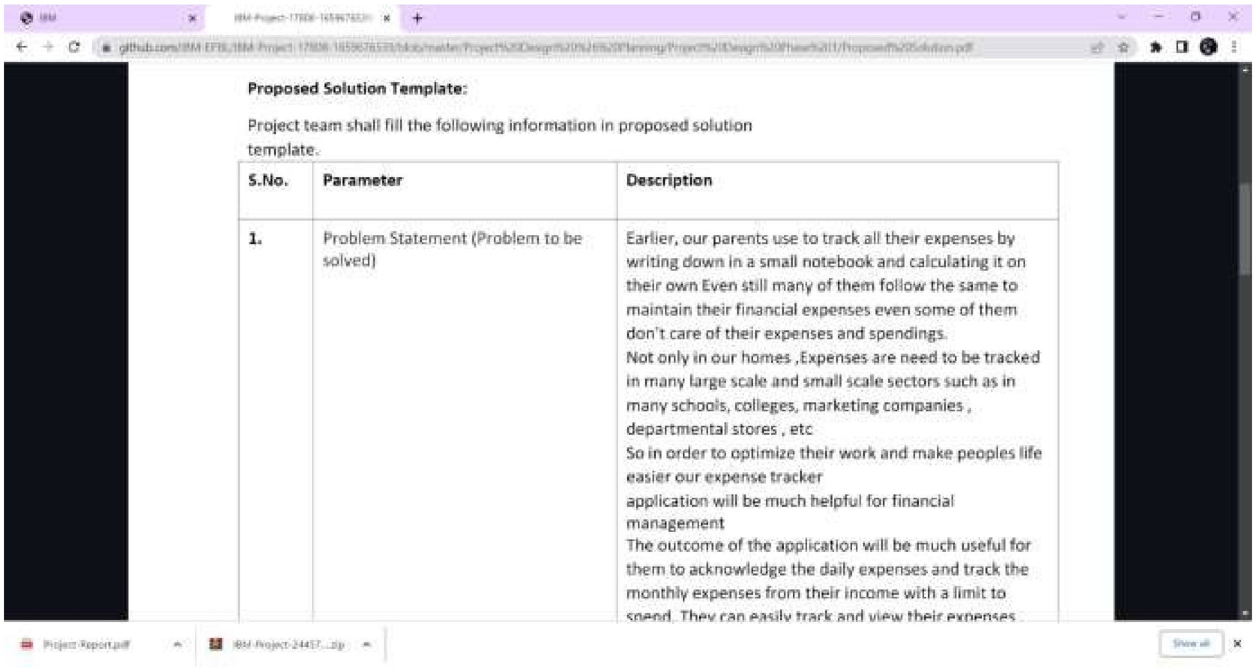
# 3.IDEATION & PROPOSED SOLUTION

## 3.1Empathy Map canvas



## 3.2Ideation & Brainstorming


IBM
IBM-Project-17808-1659676531
IBM-Project-42652-1660...
github.com/IBM-EPBL/IBM-Project-17808-1659676533/blob/master/Project%20Design%20%26%20Planning/Ideation%20Phase/Brainstorming%20(1).pdf
107 KB
Download
Brainstorm & idea prioritization
Project-Report.pdf
IBM-Project-24457....zip
Show all

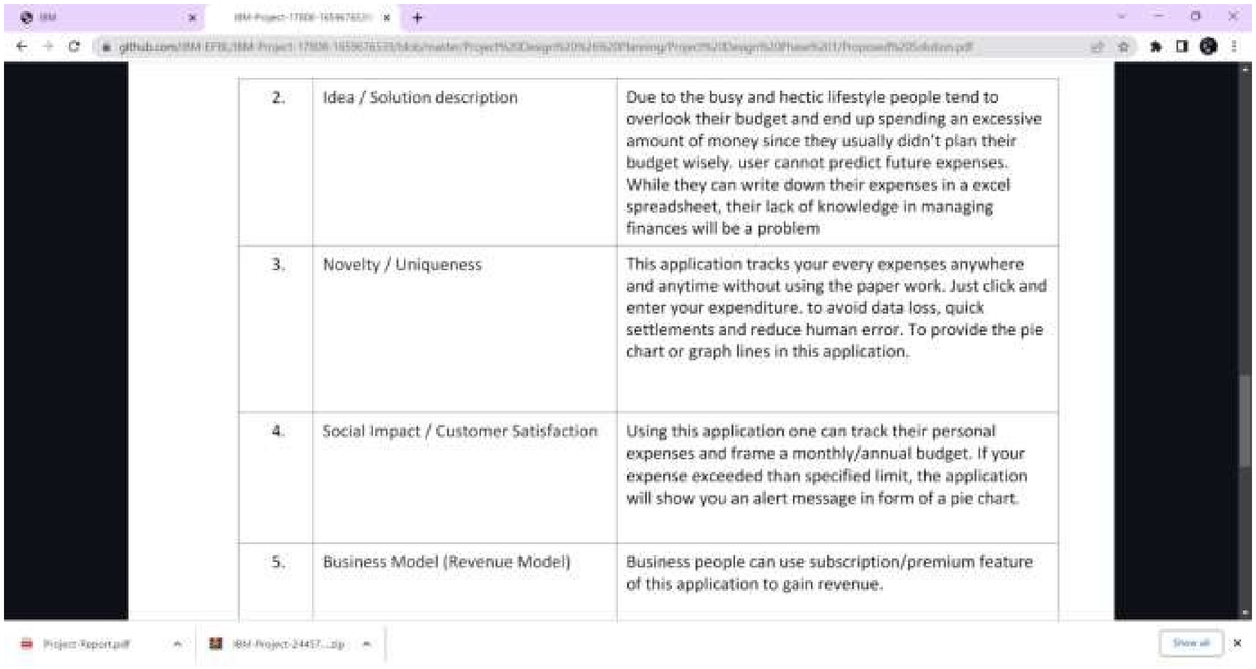


**Proposed Solution Template:**

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

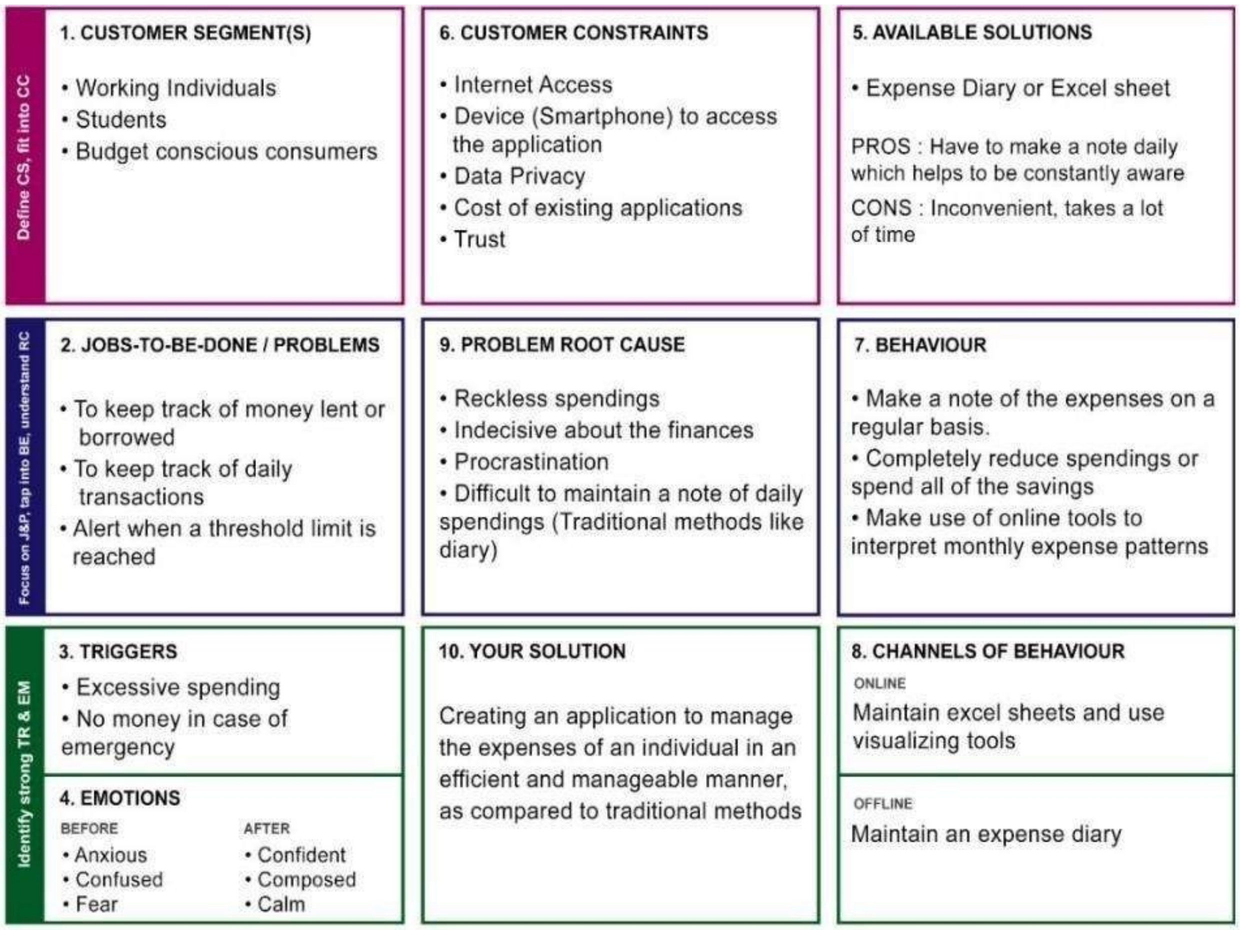
| S.No. | Parameter | Description |
|---|---|---|
| 1. | Problem Statement (Problem to be solved) | Earlier, our parents use to track all their expenses by writing down in a small notebook and calculating it on their own Even still many of them follow the same to maintain their financial expenses even some of them don't care of their expenses and spendings. Not only in our homes ,Expenses are need to be tracked in many large scale and small scale sectors such as in many schools, colleges, marketing companies , departmental stores , etc So in order to optimize their work and make peoples life easier our expense tracker application will be much helpful for financial management The outcome of the application will be much useful for them to acknowledge the daily expenses and track the monthly expenses from their income with a limit to spend. They can easily track and view their expenses |

## Proposed Solution



| | | |
|---|---|---|
| 2. | Idea / Solution description | Due to the busy and hectic lifestyle people tend to overlook their budget and end up spending an excessive amount of money since they usually didn't plan their budget wisely. user cannot predict future expenses. While they can write down their expenses in a excel spreadsheet, their lack of knowledge in managing finances will be a problem |
| 3. | Novelty / Uniqueness | This application tracks your every expenses anywhere and anytime without using the paper work. Just click and enter your expenditure. to avoid data loss, quick settlements and reduce human error. To provide the pie chart or graph lines in this application. |
| 4. | Social Impact / Customer Satisfaction | Using this application one can track their personal expenses and frame a monthly/annual budget. If your expense exceeded than specified limit, the application will show you an alert message in form of a pie chart. |
| 5. | Business Model (Revenue Model) | Business people can use subscription/premium feature of this application to gain revenue. |

## 3.4Proposed Solution Fit



Define CS, fit into CC

**1. CUSTOMER SEGMENT(S)**

- Working Individuals
- Students
- Budget conscious consumers

**6. CUSTOMER CONSTRAINTS**

- Internet Access
- Device (Smartphone) to access the application
- Data Privacy
- Cost of existing applications
- Trust

**5. AVAILABLE SOLUTIONS**

- Expense Diary or Excel sheet

PROS : Have to make a note daily which helps to be constantly aware

CONS : Inconvenient, takes a lot of time

Focus on J&P, tap into BE, understand RC

**2. JOBS-TO-BE-DONE / PROBLEMS**

- To keep track of money lent or borrowed
- To keep track of daily transactions
- Alert when a threshold limit is reached

**9. PROBLEM ROOT CAUSE**

- Reckless spendings
- Indecisive about the finances
- Procrastination
- Difficult to maintain a note of daily spendings (Traditional methods like diary)

**7. BEHAVIOUR**

- Make a note of the expenses on a regular basis.
- Completely reduce spendings or spend all of the savings
- Make use of online tools to interpret monthly expense patterns

Identify strong TR & EM

**3. TRIGGERS**

- Excessive spending
- No money in case of emergency

**4. EMOTIONS**

| BEFORE | AFTER |
|---|---|
| • Anxious | • Confident |
| • Confused | • Composed |
| • Fear | • Calm |

**10. YOUR SOLUTION**

Creating an application to manage the expenses of an individual in an efficient and manageable manner, as compared to traditional methods

**8. CHANNELS OF BEHAVIOUR**

ONLINE

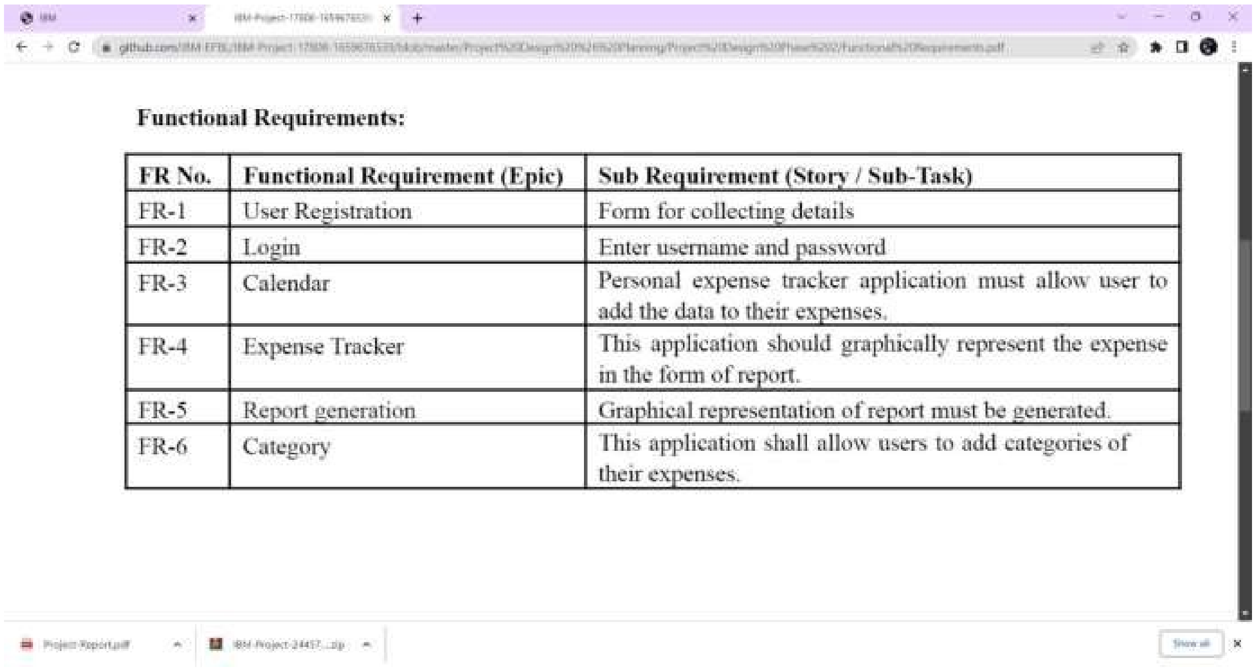
Maintain excel sheets and use visualizing tools

OFFLINE

Maintain an expense diary

# 4.REQUIREMENT ANALYSIS

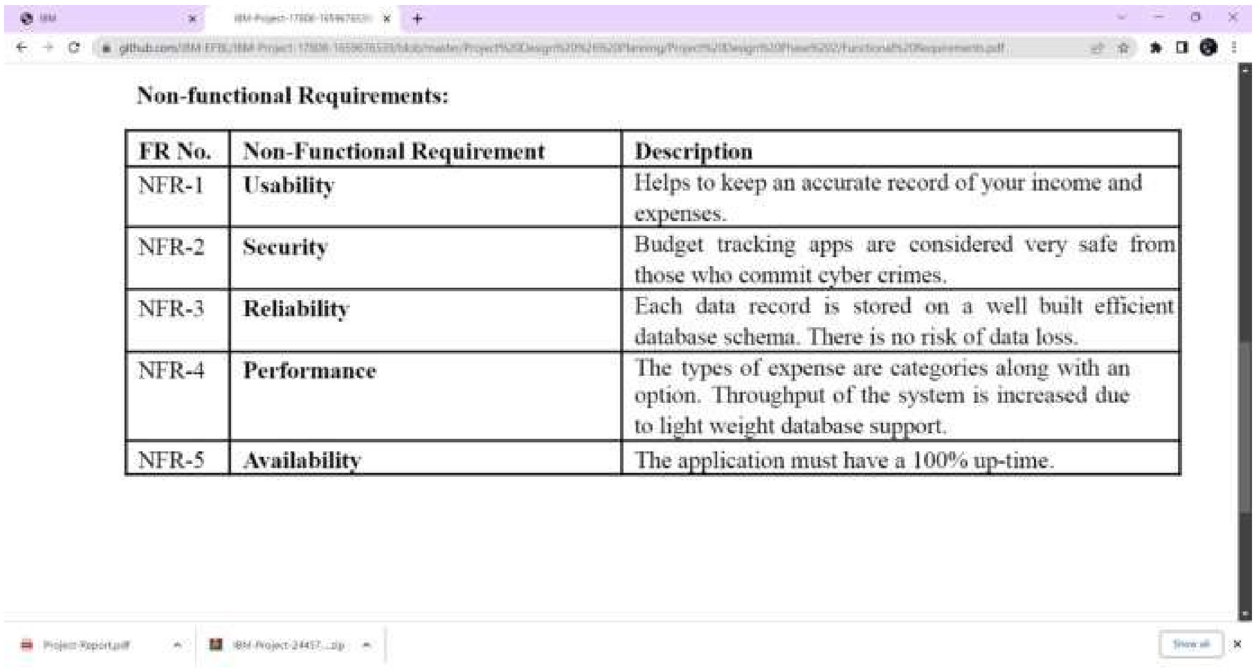
## 4.1Functional requirement



**Functional Requirements:**

| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
|---|---|---|
| FR-1 | User Registration | Form for collecting details |
| FR-2 | Login | Enter username and password |
| FR-3 | Calendar | Personal expense tracker application must allow user to add the data to their expenses. |
| FR-4 | Expense Tracker | This application should graphically represent the expense in the form of report. |
| FR-5 | Report generation | Graphical representation of report must be generated. |
| FR-6 | Category | This application shall allow users to add categories of their expenses. |

## 4.2Non-Functional requirement



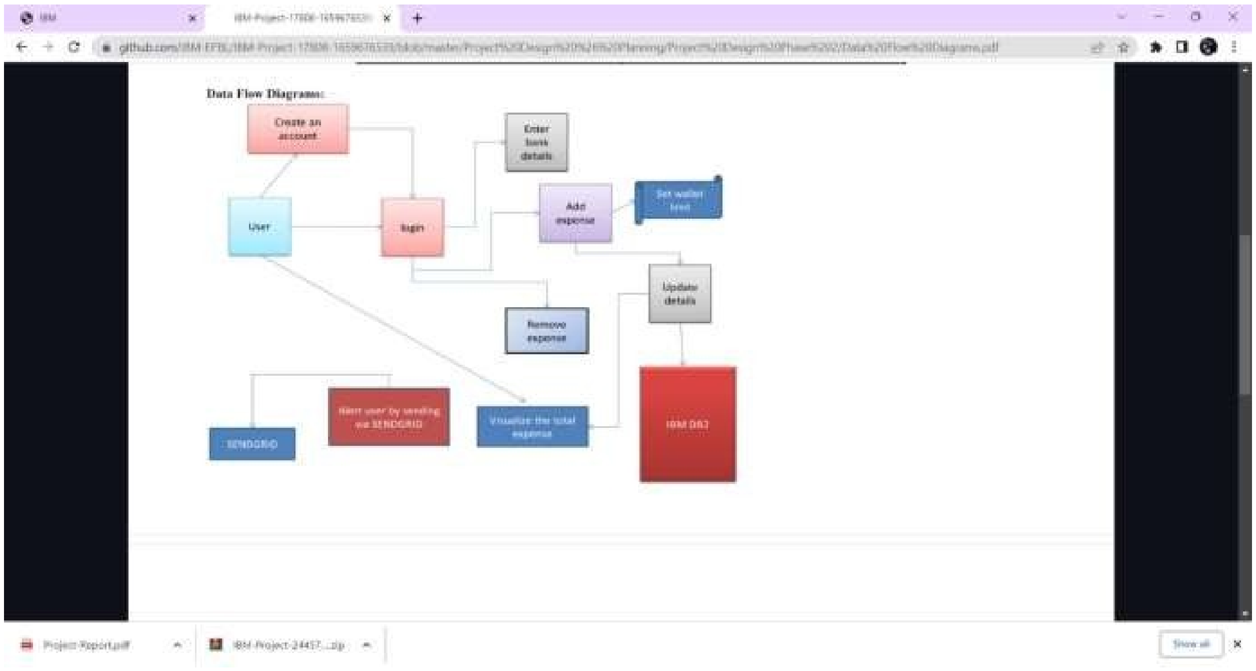
**Non-functional Requirements:**

| FR No. | Non-Functional Requirement | Description |
|---|---|---|
| NFR-1 | **Usability** | Helps to keep an accurate record of your income and expenses. |
| NFR-2 | **Security** | Budget tracking apps are considered very safe from those who commit cyber crimes. |
| NFR-3 | **Reliability** | Each data record is stored on a well built efficient database schema. There is no risk of data loss. |
| NFR-4 | **Performance** | The types of expense are categories along with an option. Throughput of the system is increased due to light weight database support. |
| NFR-5 | **Availability** | The application must have a 100% up-time. |

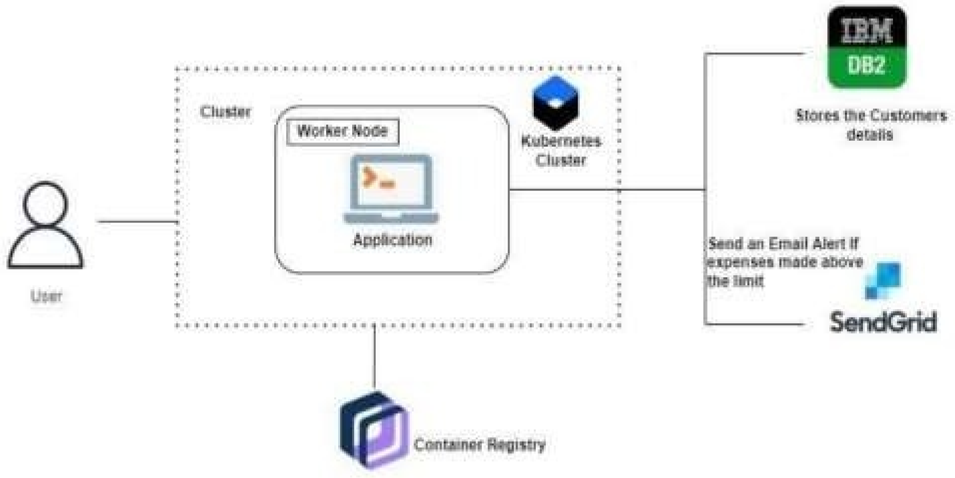
# 5.PROJECT DESIGN

## 5.1Data Flow Diagrams

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



## 5.2Solution & Technical Architecture



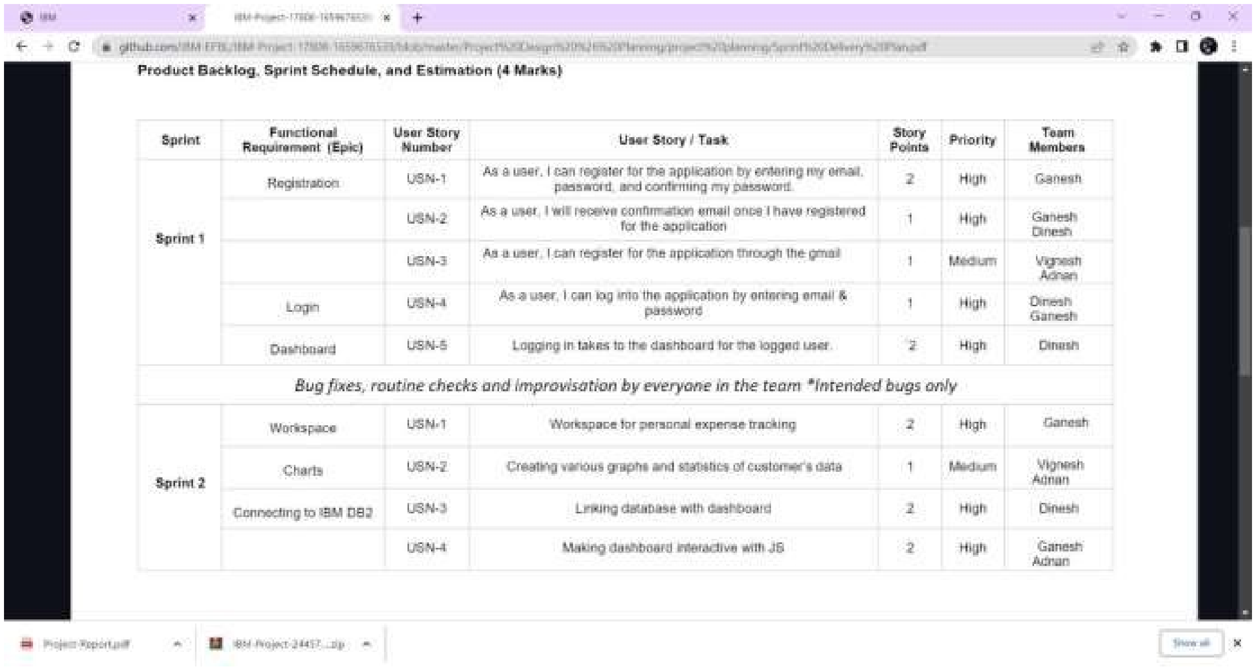
## 5.3User Stories

| User Type | Functional Requir ement (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,and password, andconfirming my password. | I can access my account/dash board | High | Sprint-1 |
|  |  | USN-2 | As a user, I will receivea confirmation email once I have registeredfor the application | I can receivea confirmation email& click confirm | High | Sprint-1 |
|  |  | USN-3 | As a user, I can register for the application through Facebook | I can register & accessthe dashboard with Facebook Login | Low | Sprint-2 |
|  |  | USN-4 | As a user, I can register for the application through aGoogle account. | I can register &access the dashboard witha Google Account login. | Medium | Sprint-1 |
|  | Login | USN-5 | As a user, I can log into the applicationby entering my email & password | I can access theapplicati on. | High | Sprint-1 |
|  | Dashboard | USN-6 | As a user, I can see the expenditure detailsandthe daily expense. | I can view the daily expenses and addthe | High | Sprint-1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | expense details. | | |
| CustomerCare Executive | | USN-7 | As a customer care executive,Icansolve the problem thatcustomers face. | I can provide support to customers at anytime 24*7. | Medium | Sprint-1 |
| Administrator | Application | USN-8 | As an administrator, Ican upgrade orupdate the application. | I can fix any bugs raised by customersand upgradethe application. | Medium | Sprint-1 |

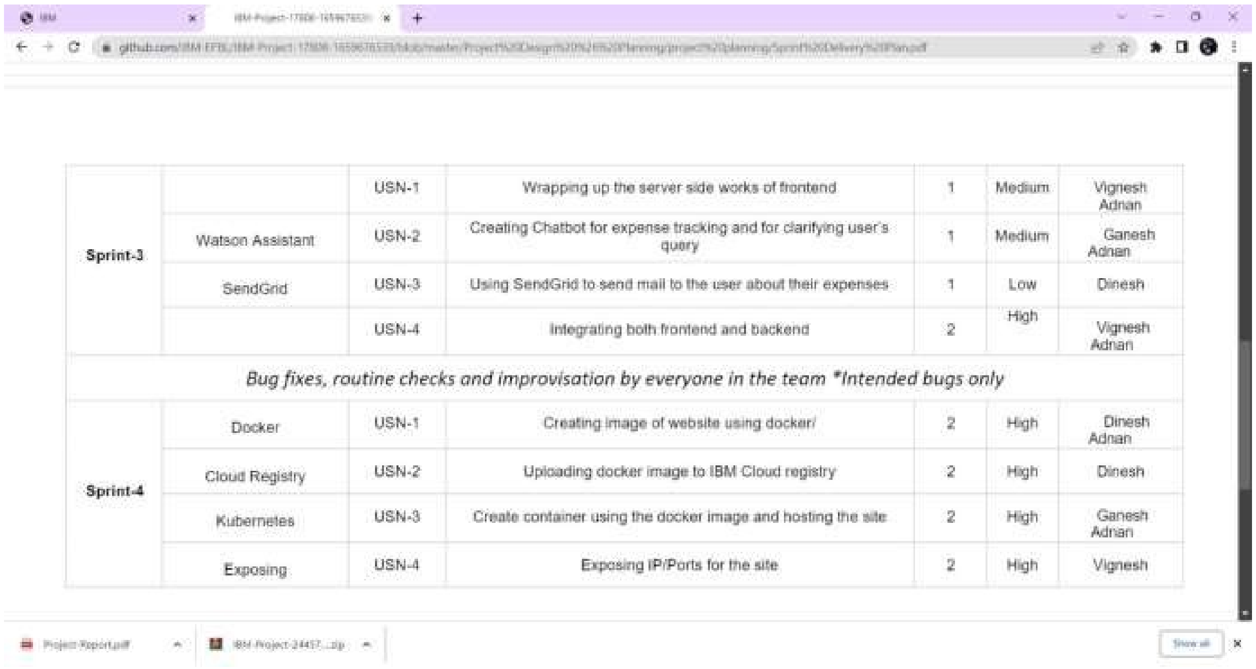
## 6.PROJECT PLANNING & SCHEDULING

### 6.1Sprint Planning & Estimation



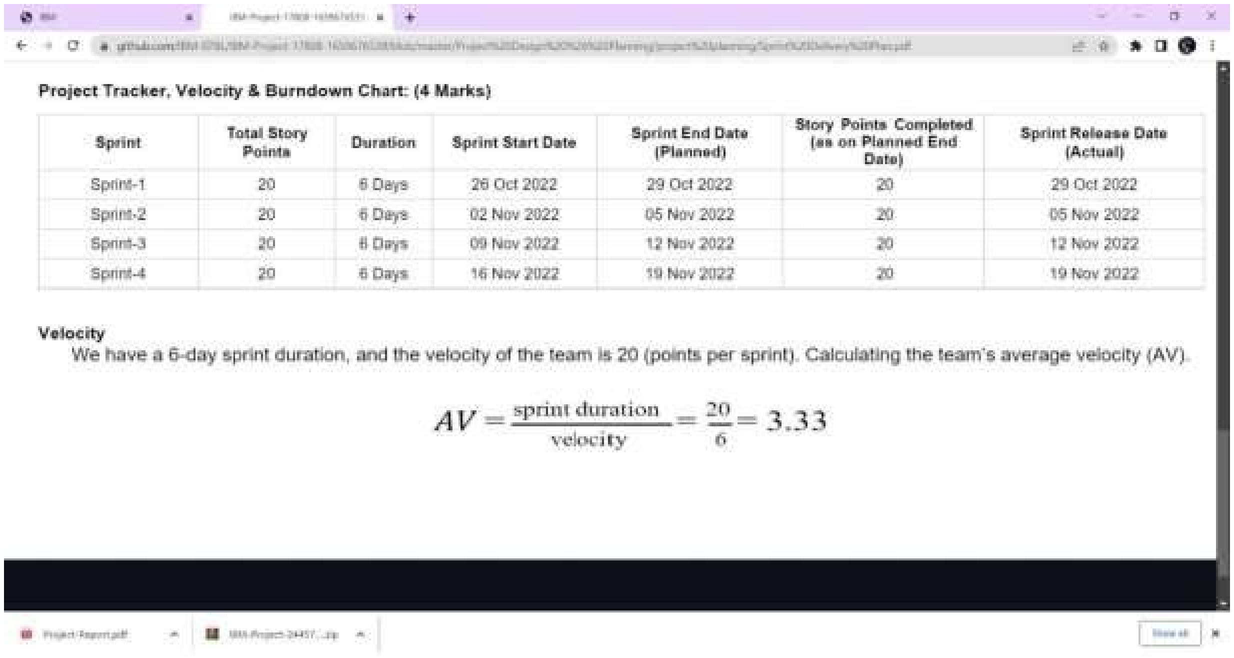
**Product Backlog, Sprint Schedule, and Estimation (4 Marks)**

| 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 | Ganesh |
| | | USN-2 | As a user, I will receive confirmation email once I have registered for the application | 1 | High | Ganesh Dinesh |
| | | USN-3 | As a user, I can register for the application through the gmail | 1 | Medium | Vignesh Adnan |
| | Login | USN-4 | As a user, I can log into the application by entering email & password | 1 | High | Dinesh Ganesh |
| | Dashboard | USN-5 | Logging in takes to the dashboard for the logged user. | 2 | High | Dinesh |
| *Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only* | | | | | | |
| Sprint 2 | Workspace | USN-1 | Workspace for personal expense tracking | 2 | High | Ganesh |
| | Charts | USN-2 | Creating various graphs and statistics of customer's data | 1 | Medium | Vignesh Adnan |
| | Connecting to IBM DB2 | USN-3 | Linking database with dashboard | 2 | High | Dinesh |
| | | USN-4 | Making dashboard interactive with JS | 2 | High | Ganesh Adnan |



| | | | | | | |
|---|---|---|---|---|---|---|
| Sprint-3 | | USN-1 | Wrapping up the server side works of frontend | 1 | Medium | Vignesh Adnan |
| | Watson Assistant | USN-2 | Creating Chatbot for expense tracking and for clarifying user's query | 1 | Medium | Ganesh Adnan |
| | SendGrid | USN-3 | Using SendGrid to send mail to the user about their expenses | 1 | Low | Dinesh |
| | | USN-4 | Integrating both frontend and backend | 2 | High | Vignesh Adnan |
| *Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only* | | | | | | |
| Sprint-4 | Docker | USN-1 | Creating image of website using docker/ | 2 | High | Dinesh Adnan |
| | Cloud Registry | USN-2 | Uploading docker image to IBM Cloud registry | 2 | High | Dinesh |
| | Kubernetes | USN-3 | Create container using the docker image and hosting the site | 2 | High | Ganesh Adnan |
| | Exposing | USN-4 | Exposing IP/Ports for the site | 2 | High | Vignesh |

## 6.2Sprint 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 | 26 Oct 2022 | 29 Oct 2022 | 20 | 29 Oct 2022 |
| Sprint-2 | 20 | 6 Days | 02 Nov 2022 | 05 Nov 2022 | 20 | 05 Nov 2022 |
| Sprint-3 | 20 | 6 Days | 09 Nov 2022 | 12 Nov 2022 | 20 | 12 Nov 2022 |
| Sprint-4 | 20 | 6 Days | 16 Nov 2022 | 19 Nov 2022 | 20 | 19 Nov 2022 |

**Velocity**

We have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). Calculating the team's average velocity (AV).

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{6} = 3.33$$

# 7.Coding and Solutioning:

## 7.1 Features

Feature 1: Add Expense

Feature 2: Update ExpenseFeature 3: Delete Expense Feature 4: Set Limit

Feature 5: Send Alert Emails to users

## 7.2 Other Features

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

# <u>8.TESTING:</u>

## 8.1 TESTING:

•Login Page (Functional)
•Login Page (UI)
•Add Expense Page (Functional)

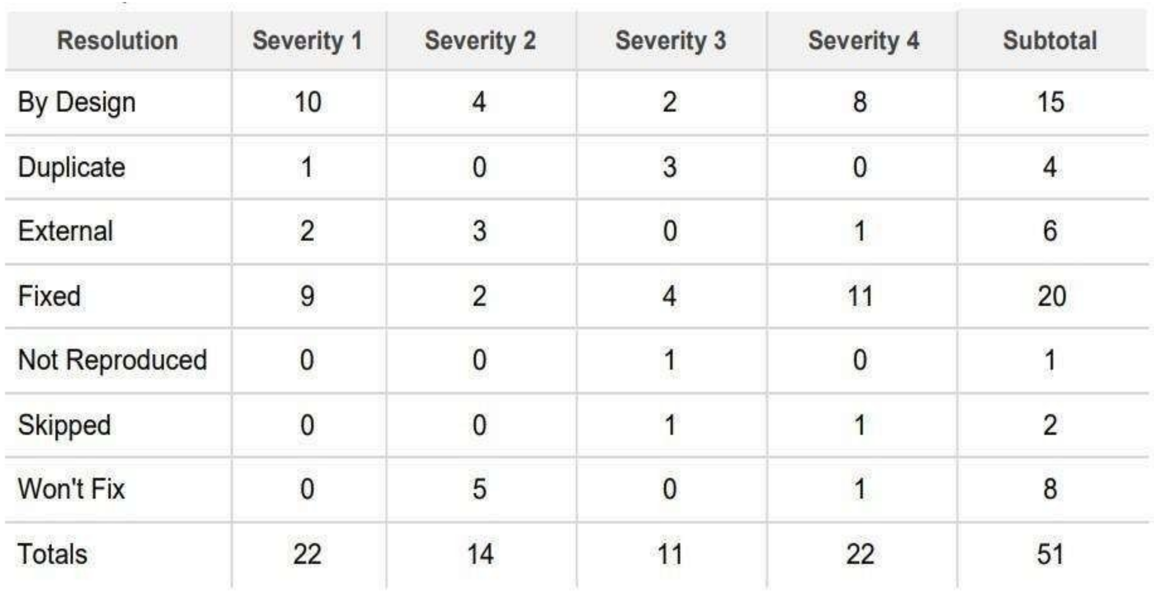
### 8.2 User Acceptance Testing:

## 1.Purpose of Document

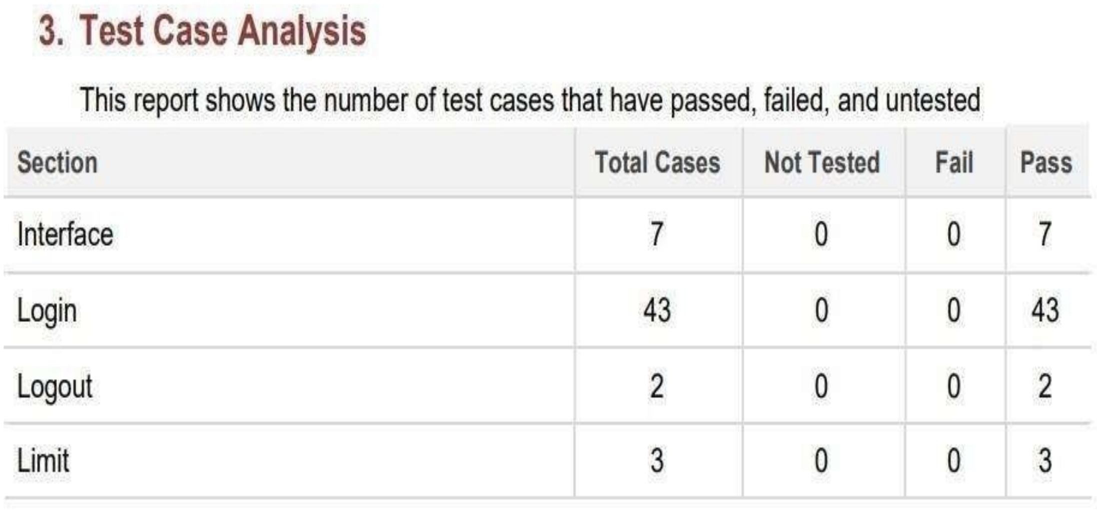
The purpose of this document is to briefly explain the test coverage and open issues of [product name] project time of the release to user acceptance testing (UAT)

## 2.Defect Analysis

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



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



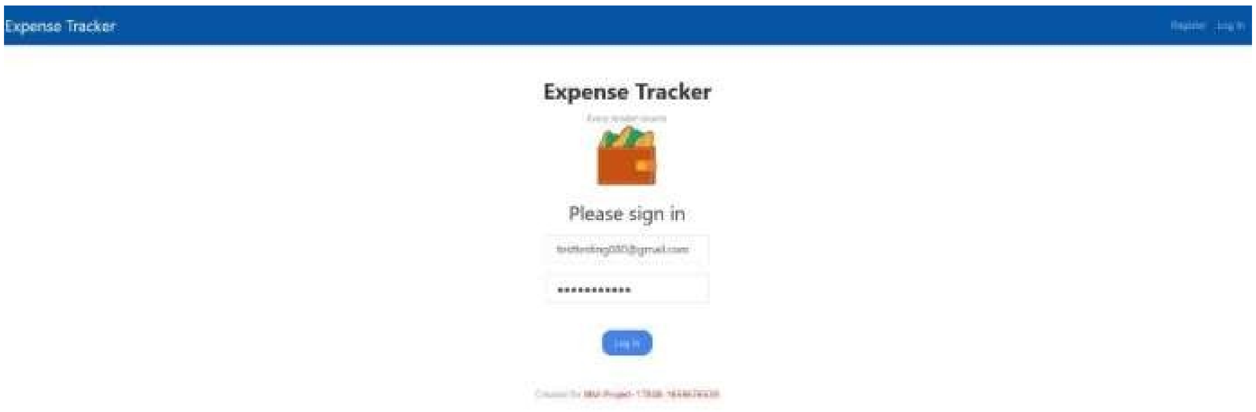
## 3. Test Case Analysis

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

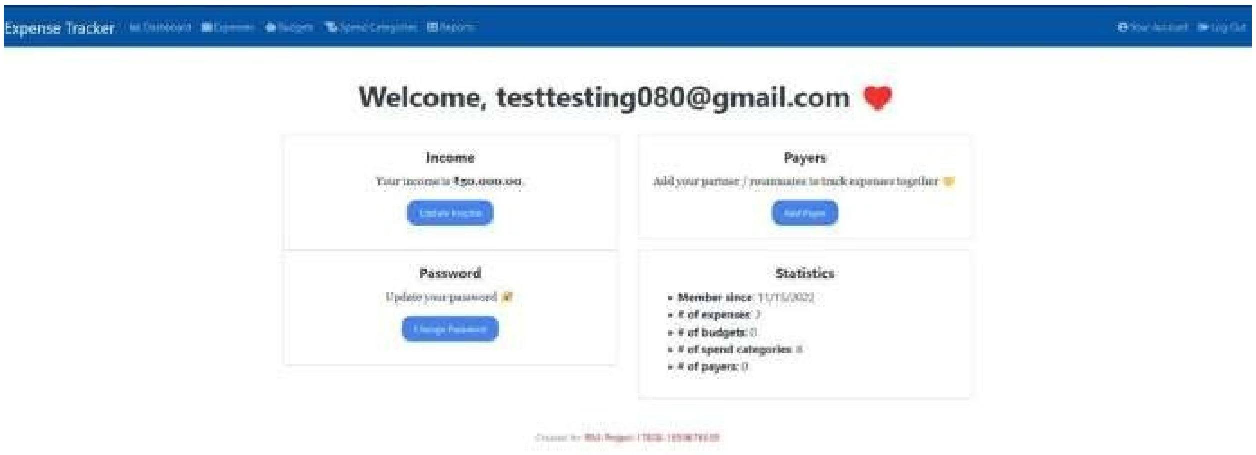
| Section | Total Cases | Not Tested | Fail | Pass |
|---|---|---|---|---|
| Interface | 7 | 0 | 0 | 7 |
| Login | 43 | 0 | 0 | 43 |
| Logout | 2 | 0 | 0 | 2 |
| Limit | 3 | 0 | 0 | 3 |

## 9.2 Sign Up

## 9.3 Login Page:



## Break down of Expense Page:



Expense Tracker Dashboard Expenses Budgets Spend Categories Reports Your Account Log Out

Welcome, testtesting080@gmail.com ❤️

Income

Your income is ₹50,000.00

Update Income

Payers

Add your partner / roommates to track expenses together

Add Payer

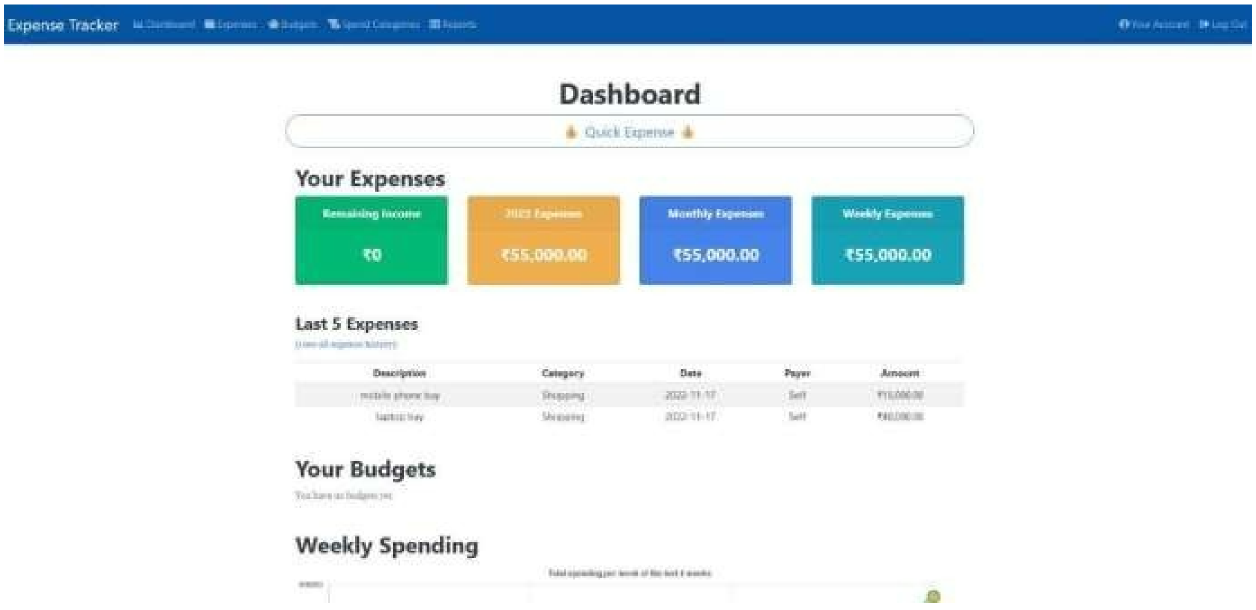
Password

Update your password

Change Password

Statistics

- **Member since**: 11/15/2022
- **# of expenses**: 2
- **# of budgets**: 0
- **# of spend categories**: 8
- **# of payers**: 0



Expense Tracker Dashboard Expenses Budgets Spend Categories Reports Your Account Log Out

Dashboard

Quick Expense

Your Expenses

| Remaining Income | 2022 Expenses | Monthly Expenses | Weekly Expenses |
|---|---|---|---|
| ₹0 | ₹55,000.00 | ₹55,000.00 | ₹55,000.00 |

Last 5 Expenses

(view all expense history)

| Description | Category | Date | Payer | Amount |
|---|---|---|---|---|
| mobile phone buy | Shopping | 2022-11-17 | Self | ₹15,000.00 |
| laptop buy | Shopping | 2022-11-17 | Self | ₹40,000.00 |

Your Budgets

You have no budgets yet

Weekly Spending

Total spending per week of the last 4 weeks

## 10.ADVANTAGES AND DISADVANTAGES

### ADVANTAGES:

One of the major pros of tracking spending is always being aware of the stateof one's personal finances. Tracking what you spend can help you stick to your budget, not just in a general way, but in each category such as housing, food, transportation and gifts. While a con is that manually tracking all cash that is spent can be irritating as well as time consuming, a pro is that doing this automatically can be quick and simple. Another pro is that many automatic spending tracking software programs are available for free. Having the program on a hand-held device can be a main pro since it can be checked before spending occurs in order to be sure of the available budget.

### DISADVANTAGES:

A con with any system used to track spending is that one may start doing it then taper off until it's forgotten about all together. Yet, this is a risk for any new goal such as trying to lose weight or quit smoking. If a person first makes a budget plan, then places money in savings before spending any each new pay period or month, the tracking goal can help. In this way, tracking spending and making sure all receipts are accounted for only needsto be done once or twice a month. Even with constant tracking of one's spending habits, there is no guarantee that financial goals will be met.
Although this can be considered to be a con of tracking spending, it could bechanged into a pro if one makes up his or her mind to keep trying to properly manage all finances.

## 11.CONCLUSION

A comprehensive money management strategy requires clarity and conviction for decision-making. You will need a defined goal and a clear vision for grasping the business and personal finances. That’s when an expense tracking app comes into the picture. An expense tracking app is anexclusive suite of services for people who seek to handle their earnings andplan their expenses and savings efficiently. It helps you track all transactions
like bills, refunds, payrolls, receipts, taxes, etc., on a daily, weekly, andmonthly basis.

## 12.FUTURE SCOPE

•Achieve your business goals with a tailored mobile app that perfectlyfits your business.
•Scale-up at the pace your business is growing.
•Deliver an outstanding customer experience through additionalcontrol over the app.
•Control the security of your business and customer data.

•Open direct marketing channels with no extra costs with methods suchas push notifications.
•Boost the productivity of all the processes within the organization.
•Increase efficiency and customer satisfaction with an app aligned totheir needs.
•Seamlessly integrate with existing infrastructure.
•Ability to provide valuable insights.
•Optimize sales processes to generate more revenue through enhanceddata collection.
•Chats: Equip your expense tracking app with a bot that can understandand answer all user queries and address their needs such as account balance, credit score, etc.
•Prediction: With the help of AI, your mobile app can predict your next purchase, according to your spending behavior. Moreover, it can recommend products and provide unique insights on saving money. Itbrings out the factors causing fluctuations in your expenses.

## Code

```
import os
import re
import expenze_categories
from flask import request, session from flask_session import
Session from sqlalchemy import create_engine
from sqlalchemy.orm import scoped_session, sessionmaker
from datetime import datetime
from helpers import convertSQLToDict
# Create engine object to manage connections to DB, and
scoped session to separate user interactions with DB engine
= create_engine(os.getenv("DATABASE_URL"))
db = scoped_session(sessionmaker(bind=engine))
# Get the users budgetsdef 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:
# Create a dict with budget year as key and empty list as
value which will store all budgets for that year
budgets = {budget['year']: [] for budget inbudgets_query}
# Update the dict by inserting budget info as valuesfor
budget in budgets_query: budgets[budget['year']].append(
{'amount': budget['amount'], 'id': budget['id'],
'name': budget['name']})
return budgetselse:
return None
# Get a users budget by the budget IDdef
getBudgetByID(budgetID, userID):
results = db.execute(
"SELECT name, amount, year, id FROM budgets WHERE
user_id = :usersID AND id = :budgetID", {"usersID":
userID,"budgetID": budgetID}).fetchall()
budget = convertSQLToDict(results)return budget[0]
# Get total amount budgeted by year
def getTotalBudgetedByYear(userID, year=None):
# Default to getting current years budgetsif 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 0else:
return amount
# Generates a budget data structure from the users input
when submitting a new or updated budget
def generateBudgetFromForm(formData):
budget = {"name": None, "year": None, "amount": None,
"categories": []}
counter = 0
# Loop through all of the form data to extract budgetsdetails
and store in the budget dict
for key, value in formData:
counter += 1
# First 3 keys represent the name/year/amount from the
form, all other keys represent dynamically loaded categories
from the form
if counter <= 3:
# Check name for invalid chars and uniquenessif key ==
"name":
```

```
# Invalid chars are all special chars except
underscores, spaces, and hyphens (uses same regex as what's
on the HTML page)
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"}
# Check if year is valid 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 budgetyear: 2020
through {currentYear}"}
# Convert the amount from string to floatelse:
amount = float(value.strip())budget[key] = amount
# All other keys will provide the *category* name /percent
budgeted
else:
# Skip iteration if value is empty (empty means theuser
doesnt want the category in their budget)
if value == '':
continue
# Need to split the key since the HTML elements areloaded
dynamically and named like 'categories.1', 'categories.2', etc.
cleanKey = key.split(".")
# Store the category name and associated % the userwants
budgetd for the category
category = {"name": None, "percent": None}if cleanKey[0]
== "categories":
category["name"] = value.strip()
# Get the percent value and convert to decimal percent =
(int(formData[counter][1].strip()) / 100)category["percent"]
= percent
# Add the category to the list of categories withinthe dict
budget[cleanKey[0]].append(category)
# Pass on this field because we grab the percent
above (Why? It's easier to keep these 2 lines than rewrite
many lines. This is the lowest of low pri TODOs)
elif cleanKey[0] == "categoryPercent":
passelse:
return {"apology": "Only categories and their
percentage of the overall budget are allowed to be stored"}
return budget
```

```
# Create a new budget
# Note: due to DB design, this is a 2 step process: 1) create a
budget (name/year/amount) in budgets table, 2) create 1:M
records in budgetCategories (budgetID + categoryID +
percentAmount)
def createBudget(budget, userID):
# Verify the budget name is not a duplicate of an existing
budget
uniqueBudgetName =
isUniqueBudgetName(budget["name"], None, userID)if not
uniqueBudgetName:
return {"apology": "Please enter a unique budget name,
not a duplicate."}
# Insert new budget into DB
newBudgetID = db.execute("INSERT INTO budgets (name,
year, amount, user_id) VALUES (:budgetName, :budgetYear,
:budgetAmount, :usersID) RETURNING id",
{"budgetName": budget["name"],
"budgetYear": budget["year"], "budgetAmount":
budget["amount"], "usersID": userID}).fetchone()[0]
db.commit()
# Get category IDs from DB for the new budget categoryIDS
= getBudgetCategoryIDS(budget["categories"], userID)
# Insert a record for each category in the new budget
```

```
addCategory(newBudgetID, categoryIDS)
return budget
# When creating or updating a budget, add the spending
categories and % budgeted per category to a budgets record
in the DB
def addCategory(budgetID, categoryIDS):
# Insert a record for each category in the new budgetfor
categoryID in categoryIDS:
db.execute("INSERT INTO budgetCategories (budgets_id,
category_id, amount) VALUES (:budgetID,
:categoryID, :percentAmount)",
{"budgetID": budgetID, "categoryID":
categoryID["id"], "percentAmount":
categoryID["amount"]})db.commit()
# Update an existing budget
# Note: due to DB design, this is a 3 step process: 1) update a
budget (name/year/amount) in budgets table, 2) delete the
existing spending categories for the budget, 3) create 1:M
records in budgetCategories (budgetID + categoryID +
percentAmount)
def updateBudget(oldBudgetName, budget, userID):# Query
the DB for the budget ID
oldBudgetID = getBudgetID(oldBudgetName, userID)
# Verify the budget name is not a duplicate of an existing
```

```
budget
uniqueBudgetName = isUniqueBudgetName(
budget["name"], oldBudgetID, userID)
if not uniqueBudgetName:
return {"apology": "Please enter a unique budget name,not
a duplicate."}
# Update the budget name, year, and amount in DB
db.execute("UPDATE budgets SET name = :budgetName,
year = :budgetYear, amount = :budgetAmount WHERE id =
:oldBudgetID AND user_id = :usersID",
{"budgetName": budget["name"], "budgetYear":
budget["year"], "budgetAmount": budget["amount"],
"oldBudgetID": oldBudgetID, "usersID": userID})
db.commit()
# Delete existing category records for the budget
db.execute("DELETE FROM budgetCategories WHERE
budgets_id = :oldBudgetID",
{"oldBudgetID": oldBudgetID})db.commit()
# Get category IDs from DB for the new budget categoryIDS
= getBudgetCategoryIDS(budget["categories"], userID)
# Insert a record for each category in the new budget
addCategory(oldBudgetID, categoryIDS)
return budget
# Get a budgets associated category ids
```

```
def getBudgetCategoryIDS(categories, userID):
# Get the category IDs from the DB for the updatedbudget
categoryIDS = []
for category in categories:
# Get the category ID
categoryID = db.execute("SELECT categories.id FROM
userCategories INNER JOIN categories ON
userCategories.category_id = categories.id WHERE
userCategories.user_id = :usersID AND categories.name =
:categoryName",
{"usersID": userID, "categoryName":
category["name"]}).fetchone()[0]
# Store the category ID and associated percent amountinto a
dict
id_amount = {"id": None, "amount": None}
id_amount["id"] = categoryID
id_amount["amount"] = category["percent"]
# Add the dictionary to the list of categoryIDs
categoryIDS.append(id_amount)
return categoryIDS
# Delete an existing budget
def deleteBudget(budgetName, userID):# Query the DB for
the budget ID
budgetID = getBudgetID(budgetName, userID)
```

```
if budgetID:
# Delete the records for budgetCategories
db.execute("DELETE FROM budgetCategories WHERE
budgets_id = :budgetID",
{"budgetID": budgetID})db.commit()
# Delete the budget
db.execute("DELETE FROM budgets WHERE id =
:budgetID",
{"budgetID": budgetID})db.commit()
return budgetNameelse:
return None
# Get budget ID from DB
def getBudgetID(budgetName, userID):
# Query the DB for a budget ID based on the user and the
supplied budget name
budgetID = db.execute("SELECT id FROM budgets
WHEREuser_id = :usersID AND name = :budgetName",
{"usersID": userID, "budgetName":
budgetName}).fetchone()[0]
if not budgetID:
return Noneelse:
return budgetID
# Get and return a bool based on whether or not a
new/updated budget name already exists for the user
```

```
def isUniqueBudgetName(budgetName, budgetID, userID):if
budgetID == None:
# Verify the net-new created budget name is not already
existing in the users existing budgets
results = db.execute(
"SELECT name FROM budgets WHERE user_id =
:usersID", {"usersID": userID}).fetchall() existingBudgets =
convertSQLToDict(results)else:
# Verify the updated budget name is not alreadyexisting in
the users existing budgets
results = db.execute(
"SELECT name FROM budgets WHERE user_id =
:usersID AND NOT id = :oldBudgetID", {"usersID":
userID,"oldBudgetID": budgetID}).fetchall()
existingBudgets = convertSQLToDict(results)
# Loop through all budgets and compare names
isUniqueName = True
for budget in existingBudgets:
if budgetName.lower() == budget["name"].lower():
isUniqueName = False
break
if isUniqueName:return True
else:
return False
```

```
# Generate a complete, updatable budget that includes the
budget name, amount, and all categories (selected/unselected
categories and % budgeted for)
def getUpdatableBudget(budget, userID):
# Get the users library of spend categories categories =
expenze_categories.getSpendCategories(userID)
# Get the budget's spend categories and % amount foreach
category
results = db.execute("SELECT DISTINCT categories.name,
budgetCategories.amount FROM budgetCategories INNER
JOIN categories ON budgetCategories.category_id =
categories.id INNER JOIN budgets ON
budgetCategories.budgets_id = budgets.id WHERE
budgets.id = :budgetsID",
{"budgetsID": budget["id"]}).fetchall() budgetCategories =
convertSQLToDict(results)
# Add 'categories' as a new key/value pair to the existing
budget dict
budget["categories"] = []
# Populate the categories by looping through and addingall
their categories
for category in categories:
for budgetCategory in budgetCategories:
# Mark the category as checked/True if it exists in thebudget
```

**that the user wants to update**
**if category["name"] == budgetCategory["name"]:# Convert the percentage (decimal) into a whole integer to be consistent with UX**
**amount = round(budgetCategory["amount"] * 100)**
**budget["categories"].append(**
**{"name": category["name"], "amount": amount,"checked": True})**
**breakelse:**
**budget["categories"].append(**
**{"name": category["name"], "amount": None,"checked": False})**
**return budget**

## Reference Links :

Github Link
https://github.com/IBM-EPBL/IBM-Project-8752-1658928637

Demo link
https://drive.google.com/file/d/1zWr_d2ld6jTsi8M9eL8qW0HEQES-KWW2/view?usp=sharing