

PERSONAL EXPENSE TRACKER APPLICATION

PROJECT REPORT

SUBMITTED BY

KEERTHANA M [111519205021]

HARSINI S R [111519205011]

BARGHANA NISHA S [111519205003]

KAAVYA S [111519205017]

TEAM ID : PNT2022TMID15517

INDUSTRY MENTOR: KUSHBOO

FACULTY MENTOR :DR. R. JOTHILAKSHMI

Project Report

1. INTRODUCTION

- 1.1 Project Overview
- 1.2 Purpose

2. LITERATURE SURVEY

- 2.1 Existing problem
- 2.2 References
- 2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming
- 3.3 Proposed Solution
- 3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

- 4.1 Functional requirement
- 4.2 Non-Functional requirements

5. PROJECT DESIGN

- 5.1 Data Flow Diagrams
- 5.2 Solution & Technical Architecture
- 5.3 User Stories

6. PROJECT PLANNING & SCHEDULING

- 6.1 Sprint Planning & Estimation
- 6.2 Sprint Delivery Schedule
- 6.3 Reports from JIRA

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

- 7.1 Feature 1
- 7.2 Feature 2
- 7.3 Database Schema (if Applicable)

8. TESTING

8.1 Test Cases

8.2 User Acceptance Testing

9. RESULTS

9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

INTRODUCTION

Personal finance management is an important part of people's lives. However, everyone does not have the knowledge or time to manage their finances in a proper manner. And, even if a person has time and knowledge, they do not bother with tracking their expenses as they find it tedious and time-consuming.

Now, you don't have to worry about managing your expenses, as you can get access to an expense tracker that will help in the active management of your finances. This System divides the Income based on daily expenses. If it exceeds the day's expense, the system will calculate income and will provide a new daily expense allowed amount. Daily expense tracking System will generate a report at the end of month to show the Income-Expense graph.

And employees send reports to the manager for verification. Manager sends final reports to the administrator. Based on the final reports system, predict the next month's expenses. It will help to manage overall expenses and income. Businesses utilize expense management software to process, pay, and audit employee-initiated expenses.

The software includes capabilities for employees to input expenses for approval through a form.

Expense management software simplifies and automates a business' expense entry, eliminates paper trail, and reduces administrative effort. Expense management software allows administrators to have full visibility of and track employee use of business financial resources.

Expense management software analyzes overall expenses, identifies cost-saving opportunities, and controls excessive spending. “Expense Tracker” is developed to manage the daily expenses in a more efficient and manageable way. By using this application. We can reduce the manual calculations of the daily expenses and keep track of the expenditure.

1.1 PROJECT OVERVIEW

Once you’ve built up a lot of information about your expenses, you can use it to make a number of different financial decisions. You can easily broadcast your future spending — and plan out a budget. If you aren’t comfortable with the amount of spending you’re doing, you can also use all those expenses you’ve been tracking to help you set limits and find places where you can reduce your spending. If, for instance, you notice a lot of lunches out, you could cut those expenses by committing to brown-bagging on a more regular basis. As long as you already have information on your expenses in hand, you can use it to make a long list of decisions much easier.

1.2 PURPOSE

Personal finance management is an important part of people’s lives. However, everyone does not have the knowledge or time to manage their finances in a proper manner. And, even if a person has time and knowledge, they do not bother with tracking their expenses as they find it tedious and time-consuming. Now, you don’t have to worry about managing your expenses, as you can get access to an expense tracker that will help in the active management of your finances.

LITERATURE SURVEY

ABSTRACT

This is an application that helps users to keep track of their day to day expenses. It will keep track of a user's income and expenses on a daily basis. It enables users to manage and track their finances and have a better control over their expenditure. It helps the users to keep a digital diary and track as they spend. The user will be able to add their expenditures instantly and can review them anywhere and anytime with regular updates. Users can see the accurate duration for how long and how much they spend on a particular category of things. The user will be able to see the detailed analyses with the help of graphical visualizations.

This project will provide a lot of benefits to the users with the help of which they will be surely able to keep track of each penny. It is time to stop using paper and excel sheets, it's not easy to manage. It is common to delete files accidentally or misplace files this may lead to untracked and left out expenses. This expense tracker provides a complete digital solution to this problem. It will save the time of the people and it will assure error-free calculations. So this application helps the user be more aware and to track their finances and to gain better control so there is room for improvement and better management or investment in future.

2.1 EXISTING SYSTEM

There can be many disadvantages of using a manual accounting system. Today, people don't have to worry as there are numerous applications and techniques using which they can manage their expenses. Also called expense manager, an expense tracker is software that facilitates keeping a record of an individual's money inflow and outflow. 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. 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 be changed into a pro if one makes up his or her mind to keep trying to properly manage all finances.

2.2 REFERENCES

- [1] Underwood, D. (2011). A Case Study of Tracking Expenses by Commodity at Widget Farmers' Cooperative.
- [2] Chandini, S., Poojitha, T., Ranjith, D., Akram, V. M., Vani, M. S., & Rajyalakshmi, V. (2019). Online Income and Expense Tracker.
- [3] Rajaprabha, M. N. (2017). Family Expense Manager Application in Android. MS&E, 263(4), 042050 [7] Kan, C., Lynch, J., & Fernbach, P. (2015). How budgeting helps consumers achieve financial goals. ACR North American Advances.

[4] www.researchgate.net/publication/360620084_EXPENSE_TRACKER_MANAGEMENT_SYSTEM

[5] Thanapal, P., Patel, M., Lokesh Raj, T., & Satheesh Kumar, J. (2015). Income and Expense Tracker. Indian Journal Of Science And Technology, 8(S2), 118-122.

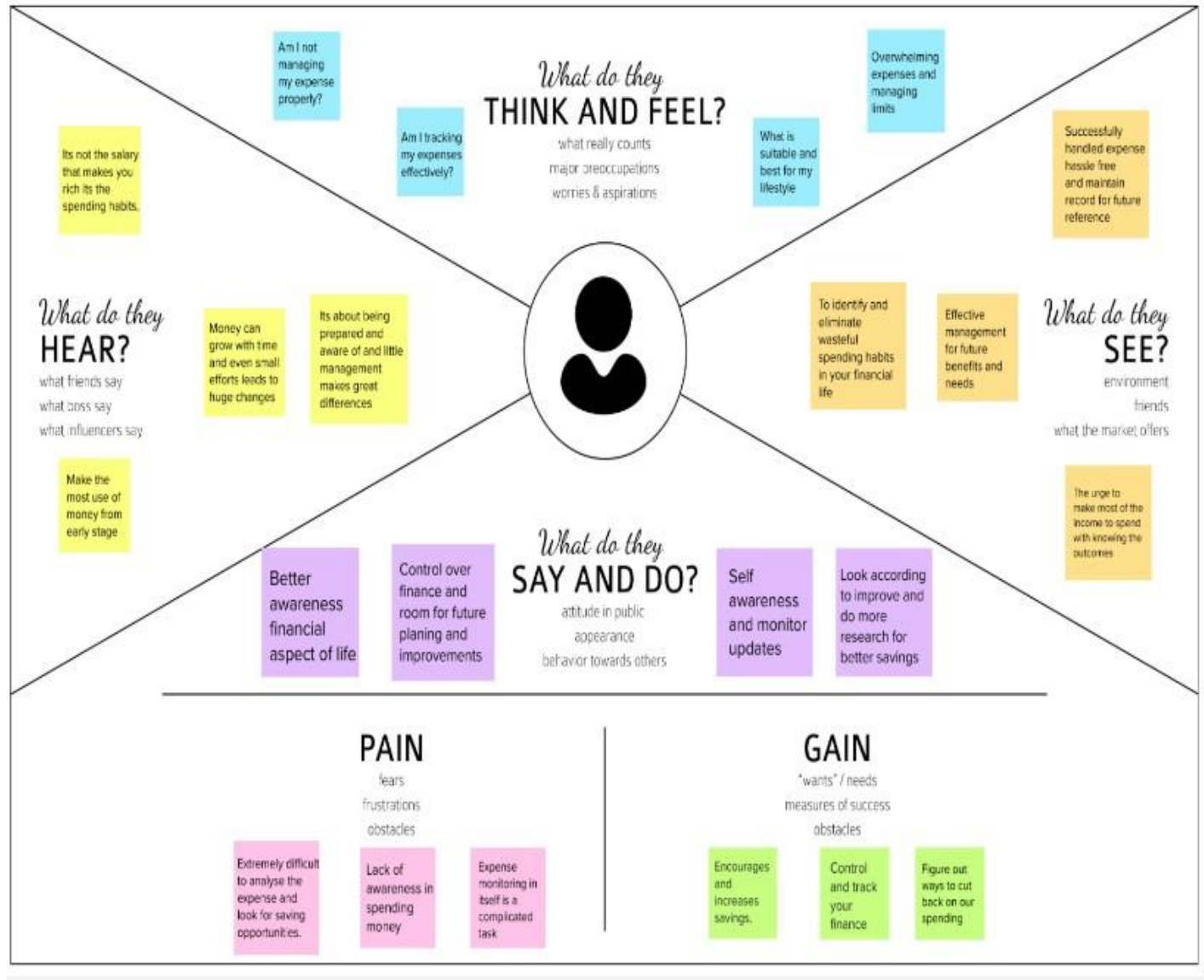
2.3 PROBLEM STATEMENT DEFINITION

It's about being aware of little expenses and management that makes great differences. Often people lose track of where and how much was spent in the long run, ultimately having to live while sustaining the little money they have left for their essential needs. Using an expense tracker can help you keep track of how much money you spend every day and on what. At the end of the month, you will have a clear picture of where your money is going. They have a option to set a limit for the amount to be used for a particular month if the limit is exceeded the user be notified with an email alert.




IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming


Step-1: Team Gathering, Collaboration and Select the Problem Statement



Brainstorm & idea prioritization


Use this template in your next brainstorming session so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.




- 1. 45 minutes (recommended)
- 2. 1 hour (recommended)
- 3. 2 hours (recommended)




Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

 10 minutes


-  **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.
-  **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.
-  **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →



Problem statement

To understand the importance of financial management and make better decisions in the future and decreases the likelihood of falling into debt or financial stress. People in the earning sector need a way to cope up with their financial management and track their expenditure so that they can improve and monitor their spending habits. Help track manage their resources and make smart decisions for leading a better and successful life.


 10minutes

PROBLEM

How the budget help in maintaining the family?
Keep track and of allowances and limits ?







Make the user aware on how much they spend unnecessarily ?

Keep track of spending habits and limits and notify users.



Key rules of brainstorming

To run an smooth and productive session

-  Stay in topic.
-  Encourage wild ideas.
-  Defer judgment.
-  Listen to others.
-  Go for volume.
-  If possible, be visual.

Step-2: Brainstorm, Idea Listing and Grouping

2

Brainstorm
Write down any ideas that come to mind that address your problem statement.

10 minutes

Harsini .S.R

Initiate a way to spend less

Monitor Expenses

Fix Budget based on user expenses

Keerthana.M

Multiple Essence in the app

Suggest to reduce the expenditure

Secure

Security purpose

Single account for multiple members

Effective Management for future benefits

Collaborate any online transaction

Promote Savings

Managing expenses

Kaavya.S

Increases Savings

Financial Awareness

Notify to the user

can access easily

Round trip time Saving

Keep on tracking several transactions

Nisha.S

Seggregated section for each expense

Time saving

Control over finance

Supports financial goals

Better user experience

Alerts user

1

Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes

Accessibility and Security

Can easily access

Manage notification securely and timely updates

Alerts

Notify to the user

Track

Keep a track on several expenses

Managing Expenses

1

Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes

Accessibility and Security

Can easily access

Manage notification securely and timely updates

Alerts

Notify to the user

Track

Keep a track on several expenses

Managing Expenses

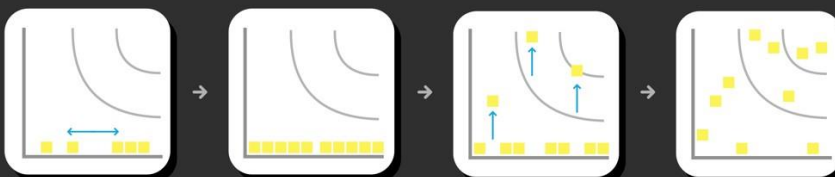
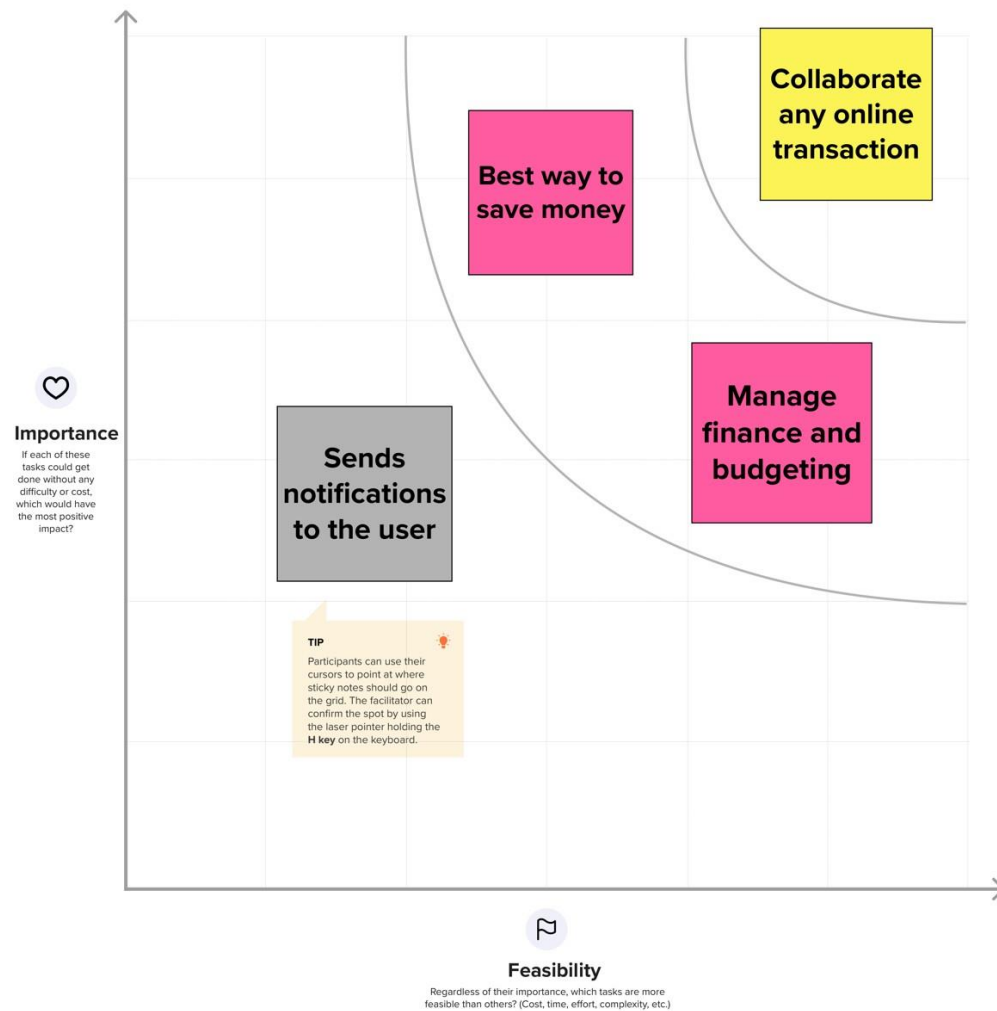
Step-3: Idea Prioritization

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	All people in the earning sector needs a way to manage their financial resources and track their expenditure, so that they can improve and monitor their spending habits. This makes them understand the importance of financial management and makes them better decisions in the future.
2.	Idea / Solution description	An application developed for tracking the user expense based on his/her expenditures. The user creates separate section for each expense and track all his expenses incurred. It is user friendly application for people who can't keep on calculating every expense manually.
3.	Novelty / Uniqueness	The uniqueness in expense tracker app is all based on its security, financial motivations and suggestions, record every expense and controls the inflow and outflow of money.
4.	Social Impact / Customer Satisfaction	It gains customer satisfactory by identifying and eliminating wasteful spending habits in their financial life. More-over it creates an impact on how much of average money is being wasted on unwanted expenses and leads finally to debt.
5.	Business Model (Revenue Model)	We are using the Subscription based model. This model looks at offering a consistent service to a consumer at a monthly fee. The benefits of this model are that you know on an ongoing basis how money should make each month.
6.	Scalability of the Solution	The scalability of the application depends on security, the working of the application even during when the network gets down etc..

3.4 Problem Solution fit

Project Title: Project – Personal Expense Tracker Application

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMID15517

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) CS <p>Working and earning people who spend on a daily basis.</p> <p>Extravagant spender or frequent traveler.</p> <p>Calculative and young adults who are just beginning to earn and handle expenses.</p> <p>Old people to keep track of when, where and how much they spend.</p>	6. CUSTOMER CONSTRAINTS CC <p>Daily habitual tracking.</p> <p>Available and awareness of technology to make this possible.</p> <p>Tardiness even on notification or fear of losing their personal spending information.</p>	5. AVAILABLE SOLUTIONS AS <p>Tracking expenses through pen and paper is an alternative but its not that safe or effective.</p> <p>Random notes lack consistency and there is no proper order or exact details when needed.</p> <p>Calculations done manually and there is no proper record of consolidated effect of spending habits to refelect and improve upon.</p>	Explore AS, differentiate
	2. JOBS-TO-BE-DONE / PROBLEMS J&P <p>Way to notify on time while the budgeting limit they set exceed so that they can be aware of their expenses.</p> <p>Users can get an analysis of their spending habits to improve upon.</p> <p>As there is a way for well recorded and documented records they can better manage and invest into newer fields.</p> <p>An application can easily manage the calculations and tracking than doing it manually</p>	9. PROBLEMS ROOT CAUSE RC <p>Users especially in the earning sector need to manage their finance in a better way for long term needs and benefits so that they do not accidentally fall in debt.</p> <p>People need to know when and how much they spend as there are so many ways and sources for spending in today's world.</p> <p>Keeping track of cash flow is essential for basic money management, and future planning.</p>	7. BEHAVIOUR BE <p>Find a proper system to keep track of their expenses and make it a daily habit to keep track and analyze their spending's manually.</p> <p>People have to make time and overcome tardiness to put in the effort and reflect upon their financial choices.</p>	
Identify strong TR & EM	3. TRIGGERS TR <p>Seeing other people effectively manage money and leading comfortable lives.</p> <p>People who are financially stable and successful, better life choices made possible with minimal effort and careful spending.</p>	10. YOUR SOLUTION SL <p>Users need an effective way to monitor and keep track of their expenses and cash flow and this app helps them to achieve that.</p> <p>Constant notifications and updates and being able to view their records and analyze gives user the control over their finance.</p> <p>Budgeting and tracking become an easy process when its grouped together in a particular place where they can better plan accordingly.</p>	8. CHANNELS OF BEHAVIOUR CH <p>8.1 ONLINE They monitor and analyze for better options and improve their finance</p> <p>8.2 OFFLINE Be more aware and stable to make better financial decisions</p>	Identify strong TR & EM
	4. EMOTIONS: BEFORE / AFTER EM <p>Before:</p> <p>Chaotic and freaked or overwhelmed by their situation and expenses, unable to track and control.</p> <p>After:</p> <p>In control and confident and more aware of their situations.</p> <p>Stress free and planned and settled.</p>			

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	Dash Board Panel	This shows the overview of the app and the features Included by navigating .
FR-2	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-3	User Confirmation	Confirmation via Email Confirmation via OTP
FR-4	Add Bank Account	Add your bank. Give the necessary details for tracking the expense flow.
FR-5	Tracking Flow	App will keep on track the user expenses inflow and outflow. If It exceeds alerts the user.
FR-6	Expense Planner	Plan according to the amount spent in previous month and it gives a clear graph about the expenditure. It gives an idea how to manage the expenses.
FR-7	Notification	User receives notification either via mail or phone number when the limit set is exceeded.

4.2 NON-FUNCTIONAL REQUIREMENTS

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

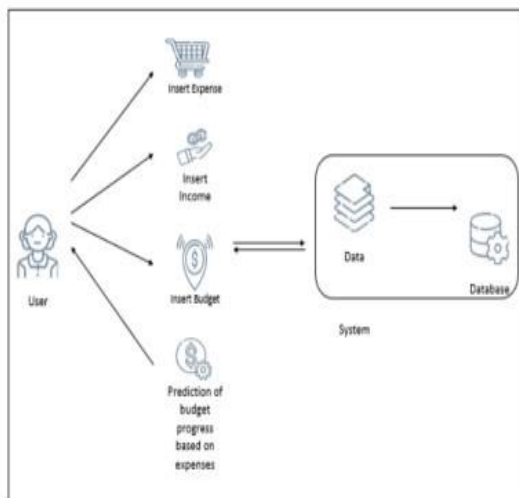
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	How easy user interfaces are to use and improving ease-of-use during the design process
NFR-2	Security	As the users are giving their personnel info and banking info it should be secure from the attacker.
NFR-3	Reliability	Tells about the Service that is performed by the application intended to how much period of time or which will operate without failure
NFR-4	Performance	Indicates how the app is functioning and how responsive the app is to end user.
NFR-5	Availability	Is the extent to which an application is operational ,functional and usable to the user
NFR-6	Scalability	Scalability of the application depends on security ,the working of the application even during when the networks gets down etc....

5.Project Design

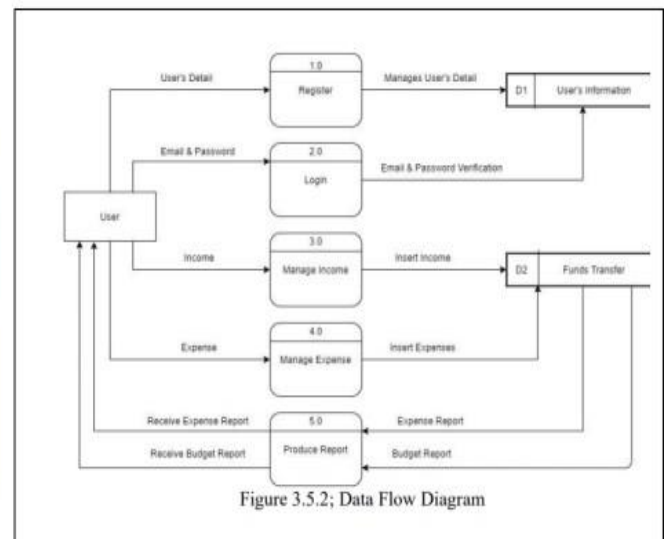
5.1 Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the 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.

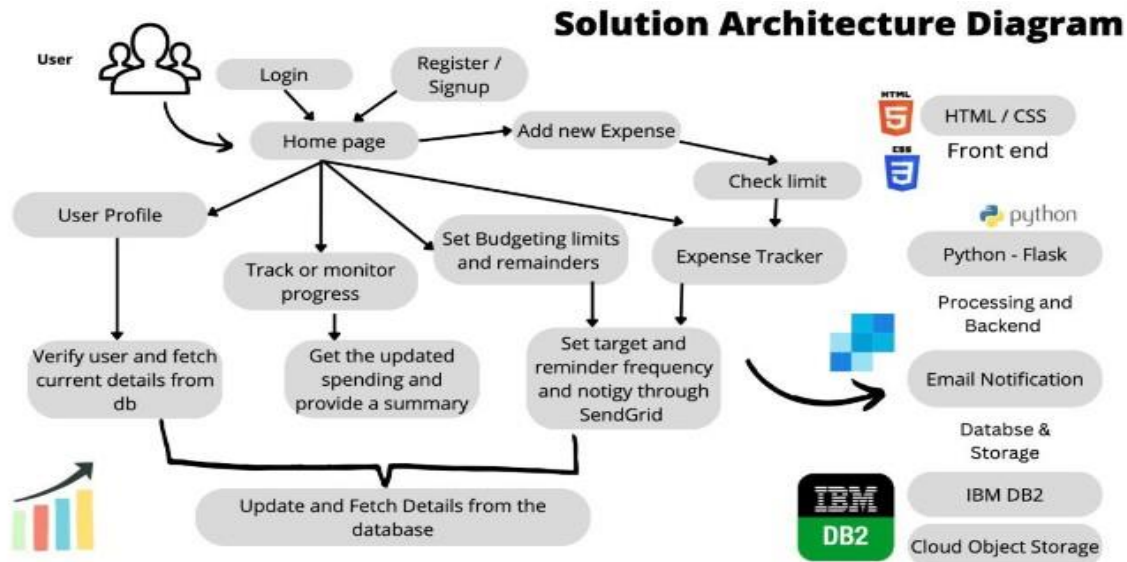
Example: (Simplified)



Example: DFD Level 0 (Industry Standard)

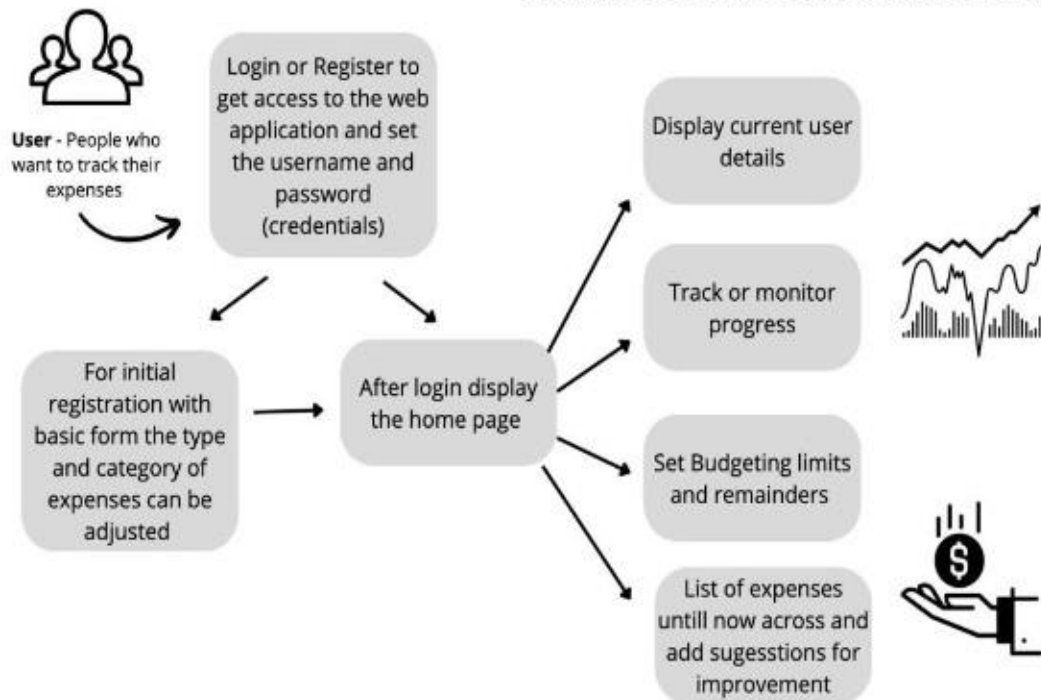


5.2 Solution and Technical Architecture



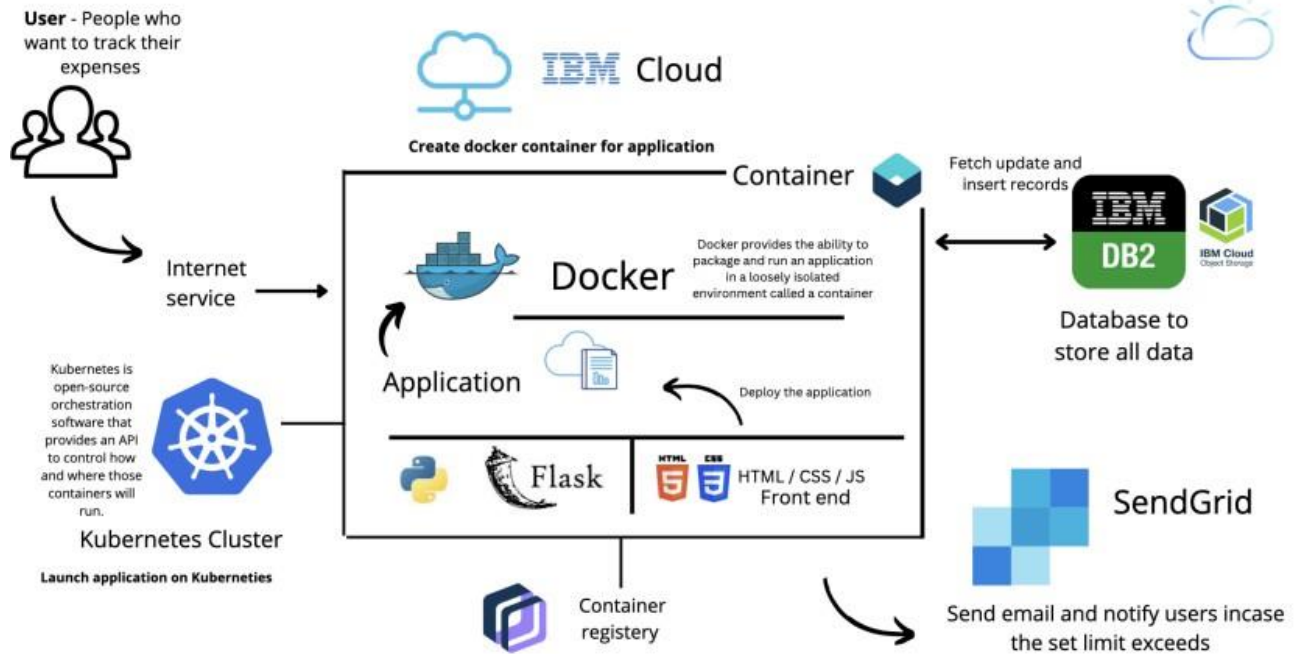
User can login if he/she already have an account else need to create an account by signing in . Homepage contains information about expenses and profile of the user . so based upon the user expense application will set a limit of the amount to be used in a month.

Solution Architecture Diagram



Folks who need to keep track on their expense can use this app by registering and keep monitor on their progress . Which will be helpful to shallow their spending.

Technology Architecture Diagram



5.3 User Stories

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

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	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	Sprint-1
		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	Sprint-1
		USN-3	As a user, I can register for the application through Mobile number	I can register & access the dashboard with Mobile number	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can access my account through gmail account	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can receive login confirmation and login credentials	High	Sprint-1
	Dashboard	USN-6	As a user, can access dashboard my to manage my expenses.	Overall credit outlook	Low	Sprint-1
Customer (Web user)	Web User	USN-7	As a Customer, can access the application using the web based platform also	Can have separate web page form	Medium	Sprint-2
Customer Care Executive	Expense Management		As a Customer care Executive, Periodically update and maintains expense application	Can have the login access when Admin permits	High	Sprint-1
Administrator	Creates and Makes the application into use		As a administrator, is responsible for every expense count management.	I can have the direct access to the application	High	Sprint-2

6.PROJECT PLANNING & SCHEDULING

6.1 SPRINT 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	Harsini , Keerthana
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Keerthana, Kaavya
Sprint-1		USN-3	As a user, I can register for the application through Gmail	2	Medium	Nisha, Harsini
Sprint-1	Login	USN-4	As a user, I can log into the application by entering email & password	1	High	Kaavya, Nisha
Sprint-1	Navigation from login to dashboard	USN-5	Once the login is validated, verified users should be navigated to their dashboard	1	Medium	Keerthana
Sprint-1	Dashboard	USN-6	Dashboard should contain the user expense and profile details. All the functionalities should be available.	2	High	Harsini
Sprint-2		USN-6	Create and display various functionalities of the dashboard.	1	Low	Kaavya
Sprint-2	Profile and Tracking	USN-7	User can view their expenses and track the spent resources. Fetch and display details from database.	2	Medium	Nisha

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Frontend and Backend integration	USN-7	Create a user friendly UI and design dashboard page with html css and js for displaying basic details and features.	2	High	Nisha, Kaavya
Sprint-3	IBM cloud	USN-8	Hosting and integrating with cloud environment.	2	High	Harsini, Keerthana
Sprint-3	Track and monitor expenses.	USN-9	User specific details are fetched and according to the expenditure the details are tracked and listed for preview.	1	Medium	Kaavya, Nisha
	Set limits for expense	USN-9	User according to their spending can set limits to track and have control over their expenditure	2	Medium	Harsini, Nisha, Keerthana
Sprint-3	Feature page navigation – reminders.	USN-10	All modules such as add new expense, track according to categories and setting limits made available in dashboard.	2	High	Keerthana, Kaavya, Harsini
Sprint-4	Expense tracking according to various categories.	USN-10	Using docker, IBM cloud registry and hosting the Flask application and facilitating category wise	2	High	Harsini, Nisha, Keerthana
Sprint-4	New suggestions based on current expenses and planning for better financial management.	USN-11	Add and receive new reminders, while planning and controlling new expenses, updating and displaying up to date information fetched from db to the dashboard.	2	High	Kaavya, Keerthana, Nisha
Sprint-4		USN-11	Use features like Watson to improve and facilitate easy access of information	2	High	Nisha, Harsini, Kaavya

6.2 SPRINT DELIVERY SCHEDULE

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

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

Velocity:

10 -day sprint duration, and the velocity of the team is 20 (points per sprint).
Team's average velocity (AV) per iteration unit (story points per day)

$$AV = \text{Sprint duration} / \text{Velocity} = 20/6 = 3.33$$

Burndown Chart:

Does your team need more from Jira? Get a free trial of our Standard plan.

Projects / PersonalExpenseTracker

Backlog

ExpenseTracker Sprint 1 24 Oct - 29 Oct (4 issues)

Complete basic layout and login process

- EXPTRACKER-3 User login page UI logic design
- EXPTRACKER-4 Verify user login and register user with email password details
- EXPTRACKER-5 New user validation and login/signup logic implementation and design UI
- EXPTRACKER-6 Once the validation is successful navigation to dashboard

ExpenseTracker Sprint 2 31 Oct - 5 Nov (4 issues)

Dashboard functionalities and integration with database

- EXPTRACKER-7 Dashboard UI Design and layout
- EXPTRACKER-8 Display user profile and expense details if exist
- EXPTRACKER-9 Handle new user to add their expenses
- EXPTRACKER-10 Fetch relevant data from database and display in the dashboard

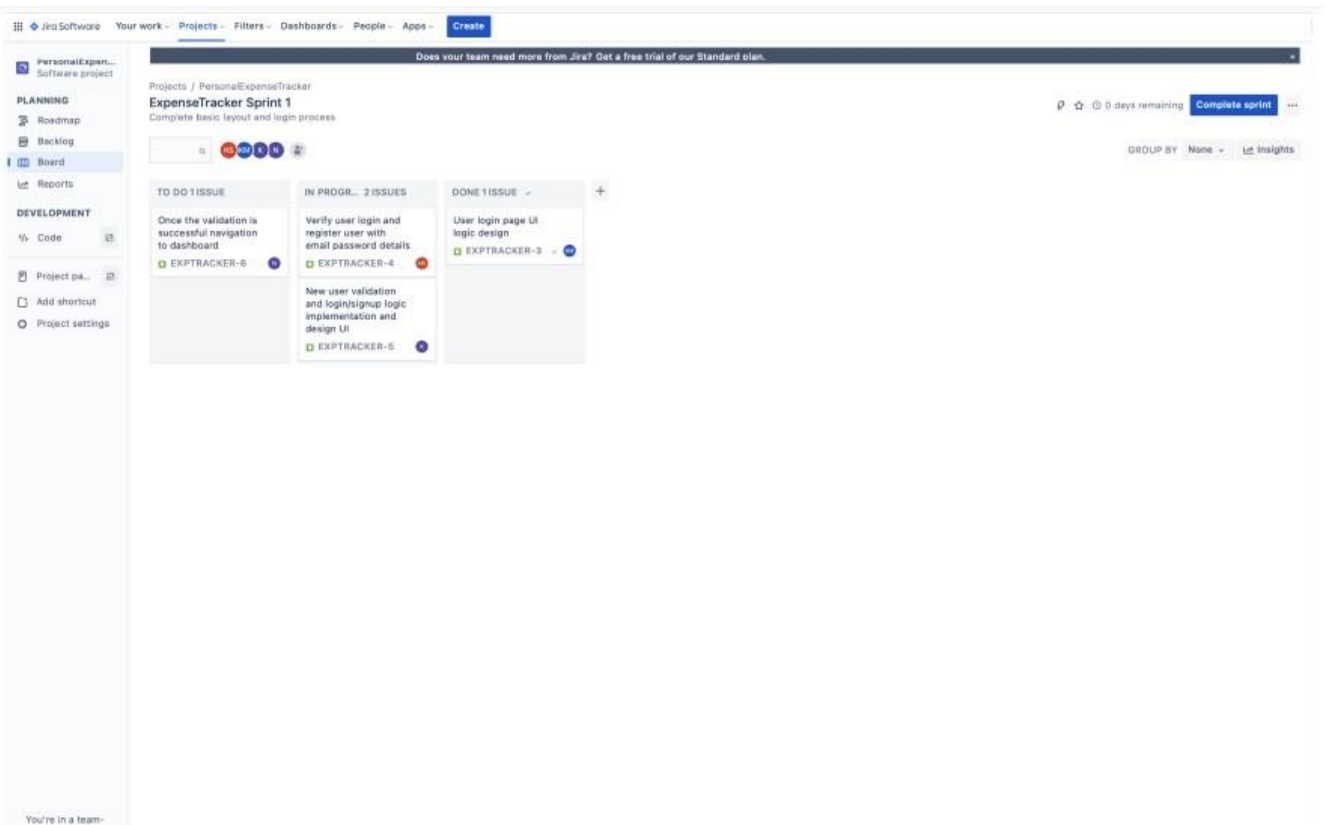
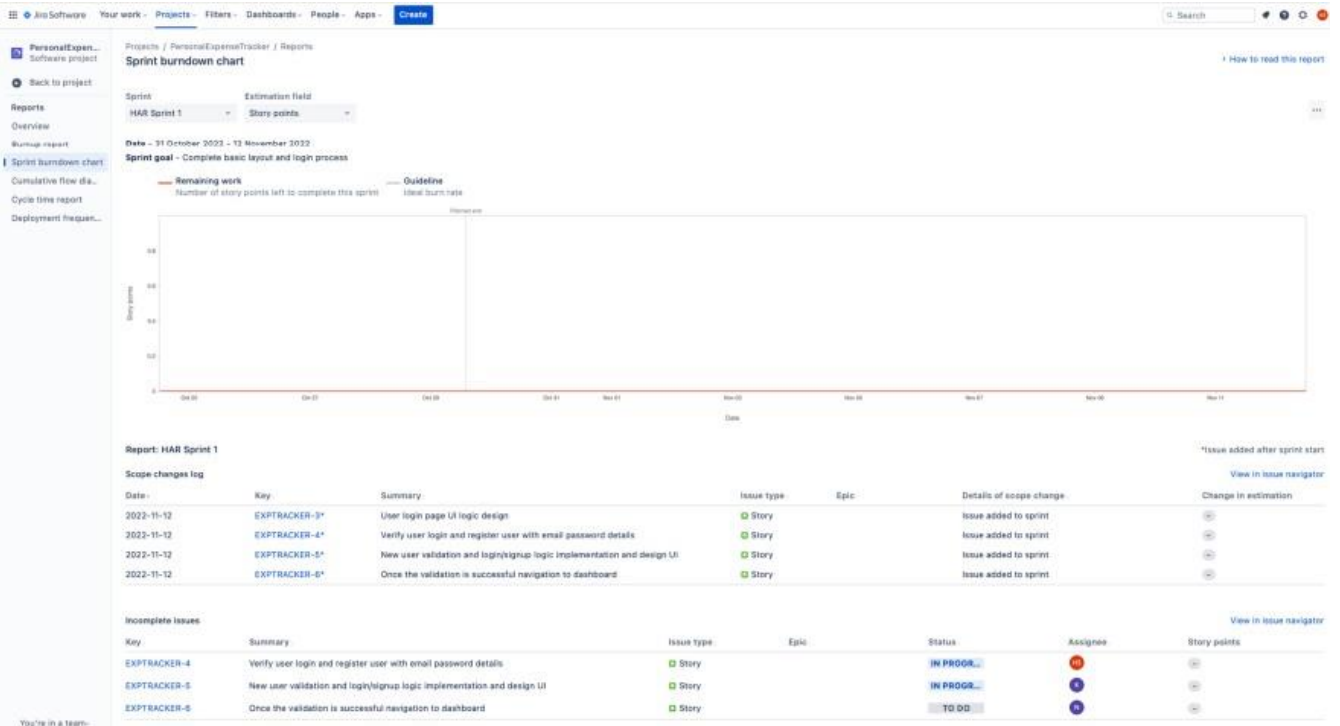
ExpenseTracker Sprint 3 7 Nov - 12 Nov (4 issues)

- EXPTRACKER-11 Make database and UI functional add new details and update in db
- EXPTRACKER-12 Adding new modules to handle expenses under various categories
- EXPTRACKER-16 Set limits for expense
- EXPTRACKER-17 User specific details are fetched and according to the expenditure the details are tracked and listed for preview.

ExpenseTracker Sprint 4 13 Nov - 19 Nov (4 issues)

- EXPTRACKER-13 Track and update new expenses feature
- EXPTRACKER-14 Feature page navigation - reminders.
- EXPTRACKER-15 Expense tracking according to various categories.
- EXPTRACKER-18 Use features like Watson to improve and facilitate easy access of information

You're in a team...



6.3 REPORTS FROM JIRA

The screenshot displays the Jira Software interface for a project named 'PersonalExpenseTracker'. The main view is 'ExpenseTracker Sprint 1', which is a Kanban board. The board is divided into three columns: 'TO DO', 'IN PROGRESS', and 'DONE 4 ISSUES'. The 'DONE' column contains four issues: 'User login page UI logic design' (EXPTRACKER-3), 'Verify user login and register user with email password details' (EXPTRACKER-4), 'New user validation and login/signup logic implementation and design UI' (EXPTRACKER-5), and 'Once the validation is successful navigation to dashboard' (EXPTRACKER-6). The left sidebar shows the project navigation menu with options like 'Roadmap', 'Backlog', 'Board', 'Reports', 'Code', 'Project pages', 'Add shortcut', and 'Project settings'. The top navigation bar includes 'Your work', 'Projects', 'Filters', 'Dashboards', 'People', 'Apps', and a 'Create' button. A search bar is located in the top right corner. A 'Quickstart' button is visible in the bottom right corner.

The screenshot displays the Jira Software interface for a project named 'PersonalExpenseTracker'. The main view is 'ExpenseTracker Sprint 2', which is a Kanban board. The board is divided into three columns: 'TO DO', 'IN PROGRESS', and 'DONE 7 ISSUES'. The 'DONE' column contains seven issues: 'Dashboard UI Design and layout' (EXPTRACKER-7), 'Make sure to add the expense' (EXPTRACKER-20), 'Add expense not redirecting properly' (EXPTRACKER-19), 'Display user profile and expense details if exist' (EXPTRACKER-8), 'Display the use details.' (EXPTRACKER-21), and two other issues. The left sidebar shows the project navigation menu with options like 'Roadmap', 'Backlog', 'Board', 'Reports', 'Code', 'Project pages', 'Add shortcut', and 'Project settings'. The top navigation bar includes 'Your work', 'Projects', 'Filters', 'Dashboards', 'People', 'Apps', and a 'Create' button. A search bar is located in the top right corner. A 'Quickstart' button is visible in the bottom right corner.

Jira Software

Your work

Projects

Filters

Dashboards

People

Apps

Create

Q Search

PersonalExpenseTracker

Software project

PLANNING

Roadmap

Backlog

Board

Reports

DEVELOPMENT

Code

Project pages

Add shortcut

Project settings

You're in a team-managed project

Learn more

Projects / PersonalExpenseTracker

ExpenseTracker Sprint 3

0 days remaining

Complete sprint

GROUP BY

None

Insights

TO DO

IN PROGRESS

DONE 5 ISSUES

Make database and UI functional
add new details and update in db
EXPTRACKER-11

Adding new modules to handle
expenses under various
categories
EXPTRACKER-12

Set limits for expense
EXPTRACKER-16

User specific details are fetched
and according to the
expenditure the details are
tracked and listed for preview.
EXPTRACKER-17

Quickstart

Jira Software

Your work

Projects

Filters

Dashboards

People

Apps

Create

Q Search

PersonalExpenseTracker

Software project

PLANNING

Roadmap

Backlog

Board

Reports

DEVELOPMENT

Code

Project pages

Add shortcut

Project settings

You're in a team-managed project

Learn more

Projects / PersonalExpenseTracker

ExpenseTracker Sprint 4

0 days remaining

Complete sprint

GROUP BY

None

Insights

TO DO

IN PROGRESS

DONE 4 ISSUES

Track and update new expenses
feature
EXPTRACKER-13

Expense tracking according to
various categories.
EXPTRACKER-15

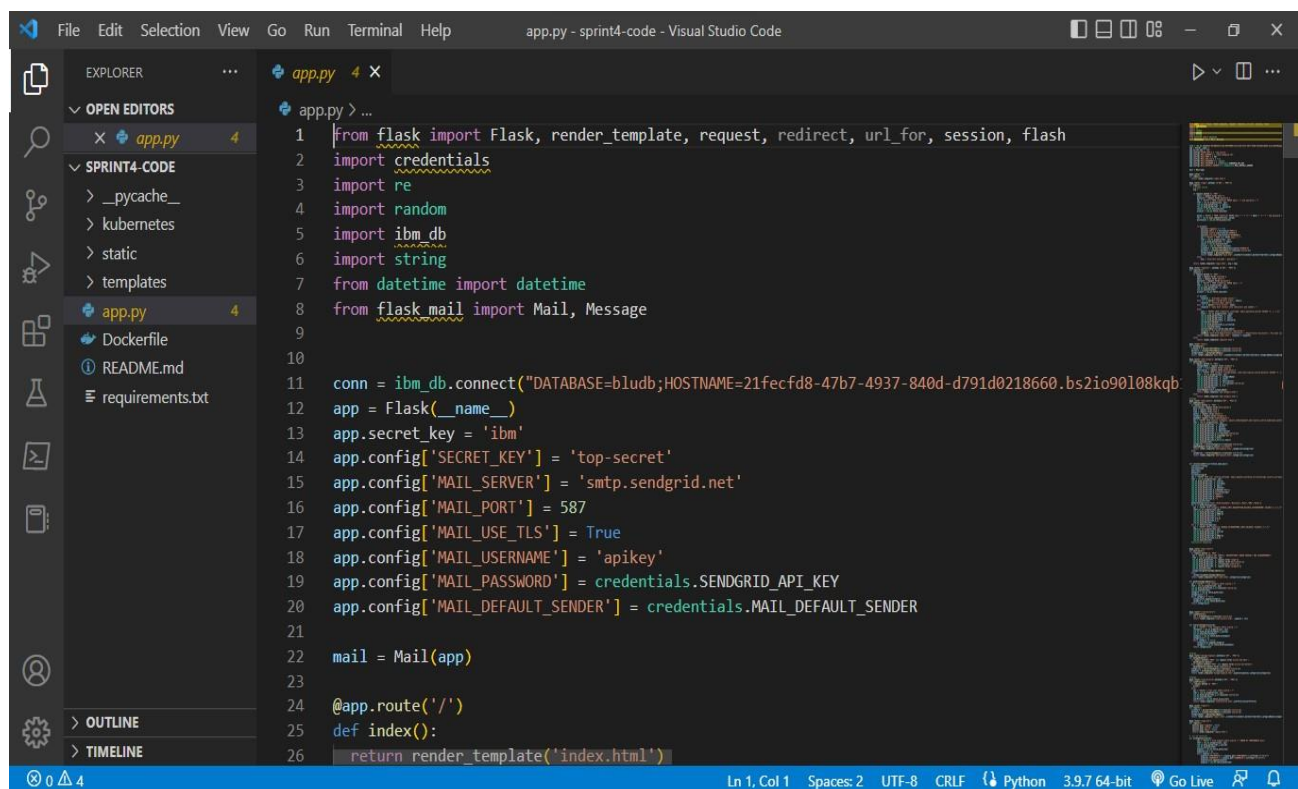
Feature page navigation –
reminders.
EXPTRACKER-14

Use features like Watson to
improve and facilitate easy
access of information
EXPTRACKER-18

Quickstart

7.CODING & SOLUTIONING

We have used flask code to integrate frontend and backend, which allows users to access the features and store the data.



```
1 from flask import Flask, render_template, request, redirect, url_for, session, flash
2 import credentials
3 import re
4 import random
5 import ibm_db
6 import string
7 from datetime import datetime
8 from flask_mail import Mail, Message
9
10
11 conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb
12 app = Flask(__name__)
13 app.secret_key = 'ibm'
14 app.config['SECRET_KEY'] = 'top-secret'
15 app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
16 app.config['MAIL_PORT'] = 587
17 app.config['MAIL_USE_TLS'] = True
18 app.config['MAIL_USERNAME'] = 'apikey'
19 app.config['MAIL_PASSWORD'] = credentials.SENDGRID_API_KEY
20 app.config['MAIL_DEFAULT_SENDER'] = credentials.MAIL_DEFAULT_SENDER
21
22 mail = Mail(app)
23
24 @app.route('/')
25 def index():
26     return render_template('index.html')
```

```
File Edit Selection View Go Run Terminal Help app.py - sprint4-code - Visual Studio Code

EXPLORER
  OPEN EDITORS
    app.py 4
  SPRINT4-CODE
    > __pycache__
    > kubernetes
    > static
    > templates
    app.py 4
    Dockerfile
    README.md
    requirements.txt

  OUTLINE
  TIMELINE

app.py > ...
24 @app.route('/')
25 def index():
26     return render_template('index.html')
27
28 @app.route('/login', methods = ['GET', 'POST'])
29 def login():
30     # global userid
31     msg = ''
32
33     if request.method == 'POST' :
34         email = request.form['email']
35         password = request.form['password']
36         sql = "SELECT * FROM credential WHERE email = ? and password = ?"
37         stmt = ibm_db.prepare(conn, sql)
38         ibm_db.bind_param(stmt, 1, email)
39         ibm_db.bind_param(stmt, 2, password)
40         result = ibm_db.execute(stmt)
41         account = ibm_db.fetch_row(stmt)
42
43         param = "SELECT * FROM credential WHERE email = " + "'" + email + "'" + " and password = ?"
44         res = ibm_db.exec_immediate(conn, param)
45         dictionary = ibm_db.fetch_assoc(res)
46
47         if account:
48             session['loggedin'] = True
```

```
File Edit Selection View Go Run Terminal Help app.py - sprint4-code - Visual Studio Code

EXPLORER
  OPEN EDITORS
    app.py 4
  SPRINT4-CODE
    > __pycache__
    > kubernetes
    > static
    > templates
    app.py 4
    Dockerfile
    README.md
    requirements.txt

  OUTLINE
  TIMELINE

app.py > ...
67 @app.route('/register', methods = ['GET', 'POST'])
68 def register():
69     response = ''
70     if request.method == 'POST' :
71         name = request.form['username']
72         email = request.form['email']
73         password = request.form['password']
74         sql = "SELECT * FROM credential WHERE email = ?"
75         stmt = ibm_db.prepare(conn, sql)
76         ibm_db.bind_param(stmt, 1, email)
77         ibm_db.execute(stmt)
78         account = ibm_db.fetch_row(stmt)
79
80         if account:
81             response = 'Username already exists !'
82         elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
83             response = 'Invalid email address !'
84         elif not re.match(r'[A-Za-z0-9]+', name):
85             response = 'name must contain only characters and numbers !'
86         else:
87             sql2 = "INSERT INTO credential (username, email,password,userid) VALUES (?, ?, ?,?)"
88             stmt2 = ibm_db.prepare(conn, sql2)
89             ibm_db.bind_param(stmt2, 1, name)
90             ibm_db.bind_param(stmt2, 2, email)
91             ibm_db.bind_param(stmt2, 3, password)
92             currentid=createId('')
```

This screenshot shows the Visual Studio Code editor with the file `app.py` open. The Explorer sidebar on the left shows the project structure: `SPRINT4-CODE` containing `__pycache__`, `kubernetes`, `static`, `templates`, `app.py`, `Dockerfile`, `README.md`, and `requirements.txt`. The main editor window displays the `addExpense` function, which is a Flask route for `/add-expense` that handles POST requests. The function logic includes: extracting form data (description, date, time, amount, category, modeofpayment); constructing an SQL `INSERT` statement; binding parameters to the statement; executing the statement on `ibm_db`; updating the budget; and rendering the `add-expense.html` template with the current categories. The status bar at the bottom indicates the cursor is at line 1, column 1, with 2 spaces, UTF-8 encoding, and CRLF line endings.

```
@app.route('/add-expense',methods=['GET', 'POST'])
def addExpense():
    if request.method == 'POST' :
        description= request.form['description']
        date = request.form['date']
        time = request.form['time']
        amount = request.form['amount']
        category = request.form['category']
        paymode= request.form['modeofpayment']
        sql = "INSERT INTO expense (category, amount,modeofpayment,description,userid,expenseid,spent"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, category)
        ibm_db.bind_param(stmt, 2, amount)
        ibm_db.bind_param(stmt, 3, paymode)
        ibm_db.bind_param(stmt, 4, description)
        ibm_db.bind_param(stmt,5,str(session['userid']))
        ibm_db.bind_param(stmt,6,createId('EXP'))
        ibm_db.bind_param(stmt,7,date)
        ibm_db.bind_param(stmt,8,datetime.now())
        ibm_db.execute(stmt)
        categories = overallCategory(str(session['userid']))
        updateBudget(category,amount,"increment")
        return render_template('add-expense.html',categories=categories)
    else:
        categories = overallCategory(str(session['userid']))
        return render_template('add-expense.html',categories=categories)
```

This screenshot shows the Visual Studio Code editor with the file `app.py` open, displaying the `manageExpense` and `viewProfile` functions. The `manageExpense` function is a Flask route for `/manage-expense` that handles POST requests. It checks for 'edit' or 'delete' actions: if 'edit', it calls `editExpense`; if 'delete', it calls `deleteExpense`. It then fetches all expenses for the user and renders the `manage-expense.html` template. The `viewProfile` function is a Flask route for `/view-profile` that handles POST requests by printing the request, querying the database for the user's profile, and rendering the `view-profile.html` template. The status bar at the bottom indicates the cursor is at line 145, column 35, with 2 spaces, UTF-8 encoding, and CRLF line endings.

```
#change
244
245 @app.route("/manage-expense",methods=['GET', 'POST'])
246 def manageExpense():
247     if request.method=='POST' and request.form['action']=='edit':
248         editExpense(request)
249     elif request.method=='POST' and request.form['action']=='delete':
250         deleteExpense(request.form['expenseid'])
251     categories = overallCategory(str(session['userid']))
252     expenses = allExpenses(str(session['userid']))
253     return render_template('manage-expense.html',expense=expenses,categories=categories)
254
255 #change
256 @app.route("/view-profile",methods=['GET', 'POST'])
257 def viewProfile():
258     if request.method == 'POST':
259         print()
260     else:
261         sql = "SELECT * from user where userid = ?"
262         stmt = ibm_db.prepare(conn, sql)
263         ibm_db.bind_param(stmt,1,str(session['userid']))
264         ibm_db.execute(stmt)
265         userProfile = ibm_db.fetch_assoc(stmt)
266         return render_template('view-profile.html',userProfile=userProfile)
267
268 @app.route('/report')
269 def report():
```

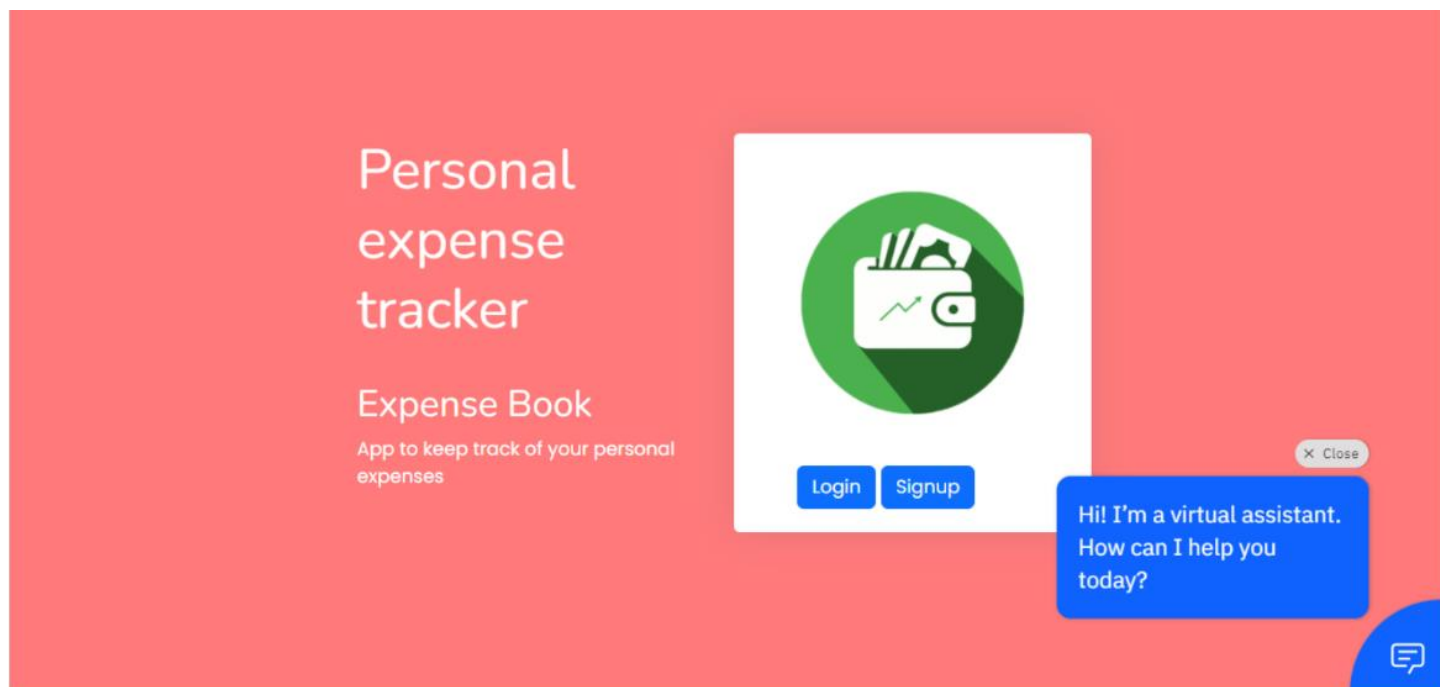


```
File Edit Selection View Go Run Terminal Help app.py - sprint4-code - Visual Studio Code

EXPLORER
  OPEN EDITORS
    app.py 4
  SPRINT4-CODE
    > __pycache__
    > kubernetes
    > static
    > templates
  app.py 4
  Dockerfile
  README.md
  requirements.txt

OUTLINE
TIMELINE

380
381 sql = "UPDATE budgeting SET balance = ? WHERE userid=? and categoryname = ?"
382 stmt = ibm_db.prepare(conn, sql)
383 ibm_db.bind_param(stmt, 1, currBalance)
384 ibm_db.bind_param(stmt, 2, str(session['userid']))
385 ibm_db.bind_param(stmt, 3, categoryName)
386 ibm_db.execute(stmt)
387
388 def getCurrentBalance(categoryName):
389     sql = "SELECT balance, limit from budgeting WHERE userid = ? and categoryname = ?"
390     stmt = ibm_db.prepare(conn, sql)
391     ibm_db.bind_param(stmt, 1, str(session['userid']))
392     ibm_db.bind_param(stmt, 2, categoryName)
393     ibm_db.execute(stmt)
394     balance = ibm_db.fetch_assoc(stmt)
395     return balance
396
397
398 def editExpense(request):
399     currentBalance = getExpenseBalance(request.form['expenseid'])
400     updateBudget(request.form['category'], currentBalance, "decrement")
401     updateBudget(request.form['category'], float(request.form['amount']), "increment")
402     datetimemeans = datetime.strptime(request.form['date'], '%d-%m-%Y')
403     sql = "UPDATE expense SET CATEGORY=?, AMOUNT=?, MODEOFFPAYMENT=?, DESCRIPTION=?, SPENTONDATE=? WHERE EXPENSEID=?"
404     stmt = ibm_db.prepare(conn, sql)
405     ibm_db.bind_param(stmt, 1, request.form['category'])
406     ibm_db.bind_param(stmt, 2, float(request.form['amount']))
407     ibm_db.bind_param(stmt, 3, request.form['modeoffpayment'])
408     ibm_db.bind_param(stmt, 4, request.form['description'])
409     ibm_db.bind_param(stmt, 5, datetimemeans)
410     ibm_db.execute(stmt)
411     return redirect(url_for('home'))
412
413 if __name__ == '__main__':
414     app.run(debug=True)
```



```
Command Prompt
Microsoft Windows [Version 10.0.19044.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rites>ibmcloud login
API endpoint: https://cloud.ibm.com

Email> uit19111@rmd.ac.in

Password>
Authenticating...
OK

Targeted account Harsini SR's Account (ef4be31ad3b24eb8bed57ab1f9b98873)

Select a region (or press enter to skip):
1. au-syd
2. in-che
3. jp-osa
4. jp-tok
5. kr-seo
6. eu-de
7. eu-gb
8. ca-tor
9. us-south
10. us-east
11. br-sao
Enter a number>

API endpoint: https://cloud.ibm.com
Region:
User: uit19111@rmd.ac.in
Account: Harsini SR's Account (ef4be31ad3b24eb8bed57ab1f9b98873)
Resource group: No resource group targeted, use 'ibmcloud target -g RESOURCE_GROUP'
CF API endpoint:
Org:
Space:

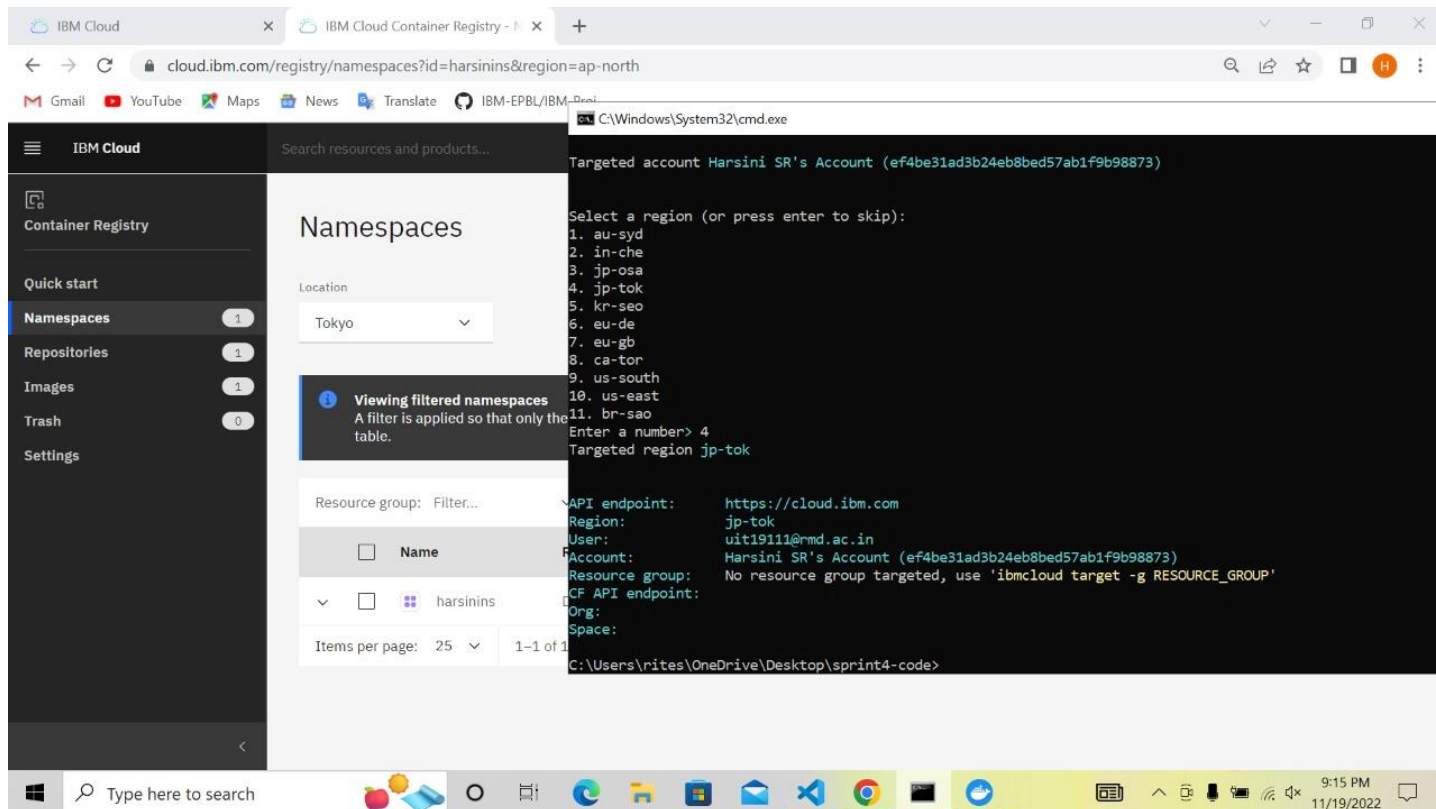
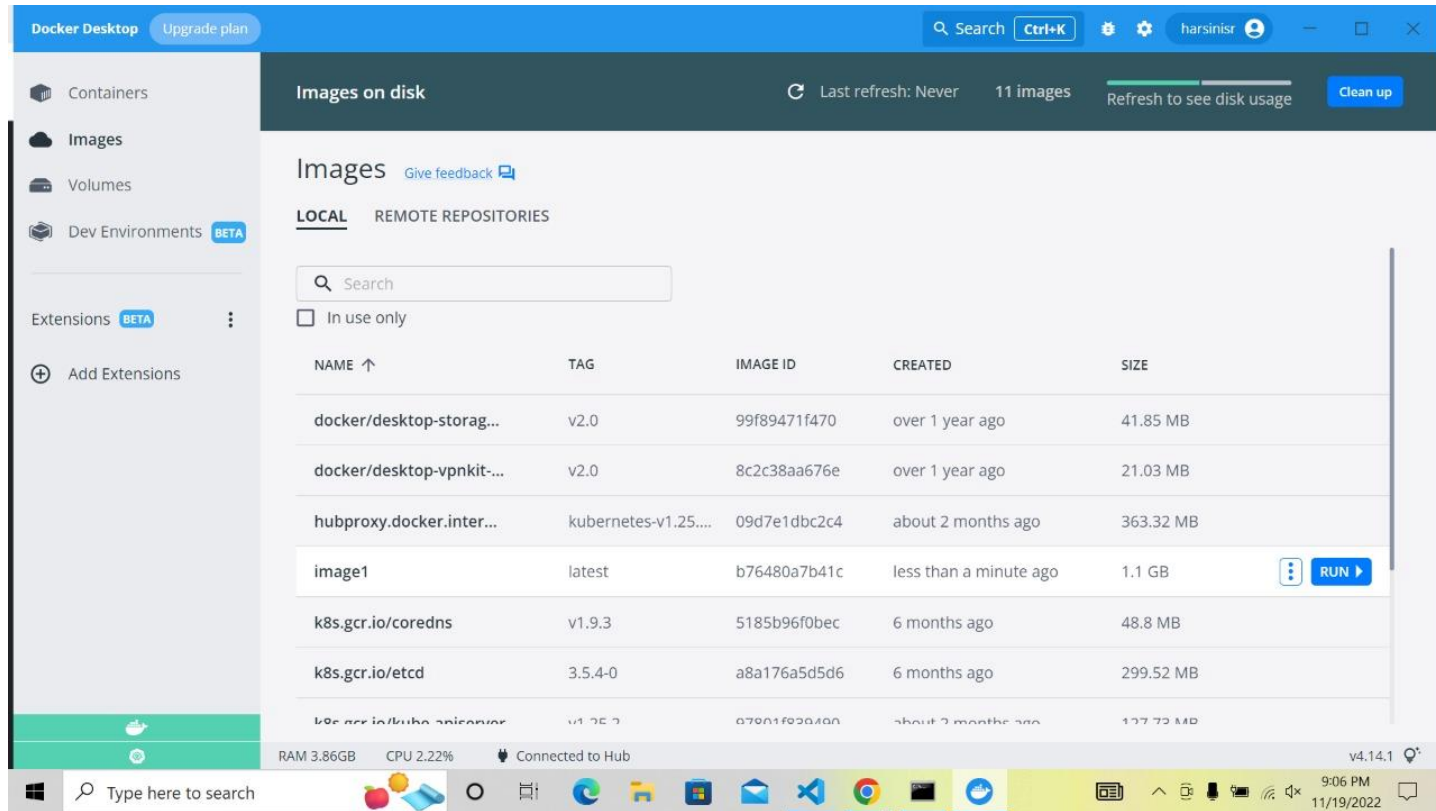
C:\Users\rites>
```

```
Docker Desktop Upgrade plan Search Ctrl+K harsinir

Containers
C:\Windows\System32\cmd.exe
=> sha256:fa9c7528c685216129e8e67bf362a7702e7b1daa585ab85546a41508830657d6 10.88MB / 10.88MB 6.6s
=> sha256:53ad072f9cd16fc8eb93b182b20e758e11acc0ef60babef0bf1043c08de1901a 54.58MB / 54.58MB 28.6s
=> sha256:d6b983117533b718374f1701ef593dd2afa6613c7908c6553be8a2a150e6448a 196.79MB / 196.79MB 59.9s
=> sha256:d8092d56ded5476fe7c302256eb4dc6ff495ae8fb4dd28aa18dbcb7581e24a6c 6.29MB / 6.29MB 28.0s
=> extracting sha256:1671565cc8df8c365c9b661d3fbc164e73d01f1b0430c6179588428f99a9da2e 12.2s
=> sha256:c71afc637d59adc44c5fd3c348504df82b35bbb204f0057ea22c6ac8a1d285a5 20.02MB / 20.02MB 35.7s
=> sha256:864a10b3c704553e08cb5fcd12fbaee1c07048f6365f0fa35e84a285413da40b 234B / 234B 29.0s
=> sha256:4334b2fa8293d19ddc1c3559093aae88f21601a7c85a31c6da6c0dc48fb6ed3c 3.04MB / 3.04MB 31.2s
=> extracting sha256:3e94d13e55e7a4ef17ff21376f57fb95c7e1706931f8704aa99260968d81f6e4 1.8s
=> extracting sha256:fa9c7528c685216129e8e67bf362a7702e7b1daa585ab85546a41508830657d6 1.4s
=> extracting sha256:53ad072f9cd16fc8eb93b182b20e758e11acc0ef60babef0bf1043c08de1901a 12.8s
=> extracting sha256:d6b983117533b718374f1701ef593dd2afa6613c7908c6553be8a2a150e6448a 10.5s
=> extracting sha256:d8092d56ded5476fe7c302256eb4dc6ff495ae8fb4dd28aa18dbcb7581e24a6c 0.4s
=> extracting sha256:c71afc637d59adc44c5fd3c348504df82b35bbb204f0057ea22c6ac8a1d285a5 1.2s
=> extracting sha256:864a10b3c704553e08cb5fcd12fbaee1c07048f6365f0fa35e84a285413da40b 0.0s
=> extracting sha256:4334b2fa8293d19ddc1c3559093aae88f21601a7c85a31c6da6c0dc48fb6ed3c 0.4s
=> [internal] load build context 0.4s
=> transferring context: 850.63kB 0.4s
=> [2/5] WORKDIR /app 5.5s
=> [3/5] COPY requirements.txt ./ 0.7s
=> [4/5] RUN pip install -r requirements.txt 73.8s
=> [5/5] COPY . . 0.8s
=> exporting to image 2.3s
=> exporting layers 1.9s
=> writing image sha256:b76480a7b41c388fe0362f6fe981826d076b365e9e50f58225d23b22e1598436 0.1s
=> naming to docker.io/library/image1 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\rites>OneDrive\Desktop\sprint4-code>
```



IBM Cloud Container Registry - namespaces

cloud.ibm.com/registry/namespaces

Installing binary...
OK
Plug-in 'container-registry 1.0.2' was successfully installed into C:\Users\rites\bluemix\plugins\container-registry. Use 'ibmcloud plugin show container-registry' to show its details.

C:\Users\rites\OneDrive\Desktop\sprint4-code>ibmcloud cr login
Logging 'docker' in to 'jp.icr.io'...
Logged in to 'jp.icr.io'.

OK

C:\Users\rites\OneDrive\Desktop\sprint4-code>docker push jp.icr.io/harsinins/image1
Using default tag: latest
The push refers to repository [jp.icr.io/harsinins/image1]
e0fc7fc38f90: Pushed
10bc04f6ff4b: Pushed
6e4af0e1d21e: Pushed
34cf4bef78f4: Pushed
bfc1deb8136e: Pushed
1f123186824c: Pushed
3d6eb1152931: Pushed
100796cdf3b1: Pushed
54acb5a6fa0b: Pushed
8d51c618126f: Pushed
9ff6e4d46744: Pushed
a89d1d47b5a1: Pushed
655ed1b7a428: Pushed
latest: digest: sha256:3cdb284b3df7c14c39ec84a529392fe641d8e5b54dfa20daa5d6b0081ab9a255 size: 3054

C:\Users\rites\OneDrive\Desktop\sprint4-code>

Visual Studio Code - app.py - sprint4-code

EXPLORER

- SPRINT4-CODE
 - __pycache__
 - kubernetes
 - static
 - templates
 - app.py
 - credentials.py
 - DigiCertGlobalRootCA.crt
 - Dockerfile
 - README.md
 - requirements.txt

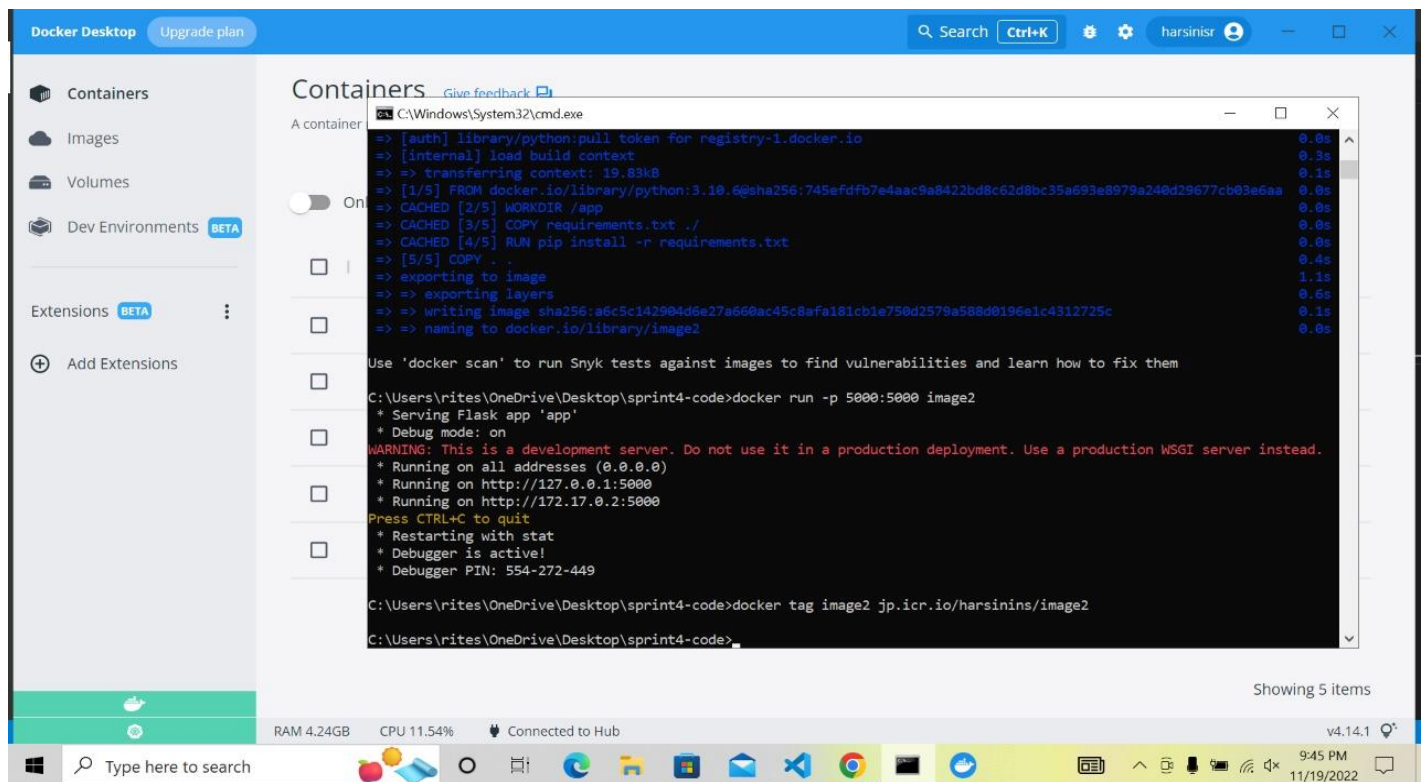
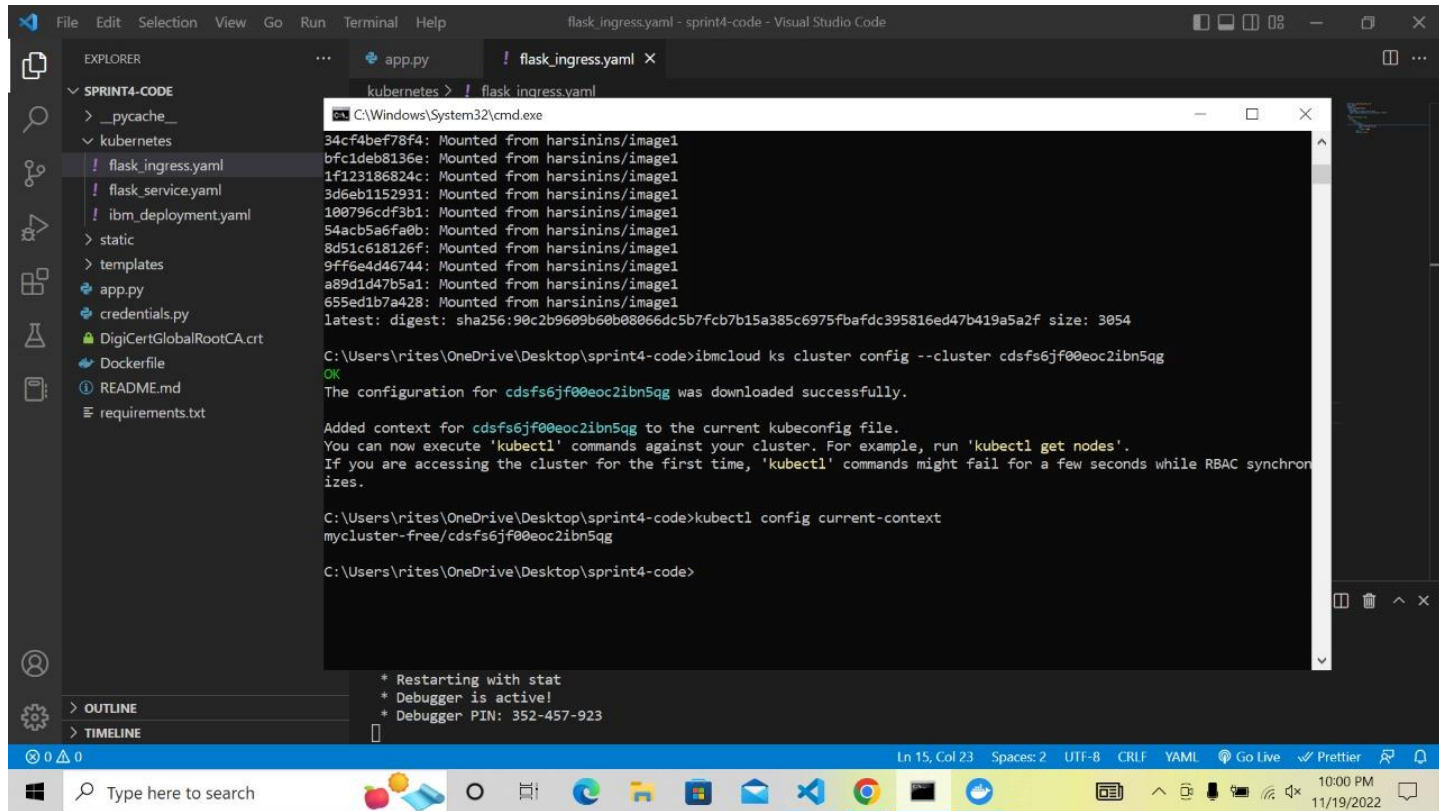
app.py

```
C:\Windows\System32\cmd.exe - docker run -p 5000:5000 image2
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.10.6
=> [auth] library/python:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 19.83kB
=> [1/5] FROM docker.io/library/python:3.10.6@sha256:745efd7b7e4aac9a8422bd8c62d8bc35a693e8979a240d29677cb03e6aa
=> CACHED [2/5] WORKDIR /app
=> CACHED [3/5] COPY requirements.txt ./
=> CACHED [4/5] RUN pip install -r requirements.txt
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:a6c5c142904d6e27a660ac45c8afa181cble750d2579a588d0196e1c4312725c
=> => naming to docker.io/library/image2

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\rites\OneDrive\Desktop\sprint4-code>docker run -p 5000:5000 image2
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
* Restarting with stat

* Debugger is active!
* Debugger PIN: 352-457-923
```

Expense Book

- Dashboard
- Expenses
 - Manage Expense
 - View History
 - Add Expense
- Limiting Expenses
- Report Analysis
- User Profile

```
C:\Windows\System32\cmd.exe
NAME                 TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
flask-app-service    ClusterIP      172.21.59.223 <none>         5000/TCP        3m27s
kubernetes            ClusterIP      172.21.0.1    <none>         443/TCP         25m
personalexpenstracker NodePort        172.21.48.53  <none>         5000:30647/TCP  72s

C:\Users\rites\OneDrive\Desktop\sprint4-code>kubectl get ing
NAME                 CLASS    HOSTS    ADDRESS    PORTS    AGE
flask-app-ingress    <none>   *        80         3m59s

C:\Users\rites\OneDrive\Desktop\sprint4-code>kubectl get nodes -o wide
NAME                 STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION        CONTAINER-RUNTIME
10.144.213.121      Ready    <none>    17m   v1.24.7+IKS  10.144.213.121  169.51.204.118  Ubuntu 18.04.6 LTS   4.15.0-194-generic    containerd/1.6.8

C:\Users\rites\OneDrive\Desktop\sprint4-code>kubectl describe svc personalexpenstracker
Name:                 personalexpenstracker
Namespace:            default
Labels:               <none>
Annotations:          <none>
Selector:             app=personalexpenstracker
Type:                 NodePort
IP Family Policy:     SingleStack
IP Families:          IPv4
IP:                   172.21.48.53
IPs:                  172.21.48.53
Port:                 <unset> 5000/TCP
TargetPort:           5000/TCP
NodePort:             <unset> 30647/TCP
Endpoints:            172.30.163.203:5000
Session Affinity:     None
External Traffic Policy: Cluster
Events:              <none>

C:\Users\rites\OneDrive\Desktop\sprint4-code>
```

IBM - Introduction to Kubernetes

File Edit View Tools Help

View only

91

92

93

94

95

```
C:\Windows\System32\cmd.exe
100796cdf3b1: Mounted from harsinins/image1
54acb5a6fa0b: Mounted from harsinins/image1
8d51c618126f: Mounted from harsinins/image1
9ff6e4d46744: Mounted from harsinins/image1
a89d1d47b5a1: Mounted from harsinins/image1
655ed1b7a428: Mounted from harsinins/image1
latest: digest: sha256:90c2b9609b60b08066dc5b7fcb7b15a385c6975fbafdc395816ed47b419a5a2f size: 3054

C:\Users\rites\OneDrive\Desktop\sprint4-code>ibmcloud ks cluster config --cluster cdsfs6jf0eoc2ibn5qg
OK
The configuration for cdsfs6jf0eoc2ibn5qg was downloaded successfully.

Added context for cdsfs6jf0eoc2ibn5qg to the current kubeconfig file.
You can now execute 'kubectl' commands against your cluster. For example, run 'kubectl get nodes'.
If you are accessing the cluster for the first time, 'kubectl' commands might fail for a few seconds while RBAC synchronizes.

C:\Users\rites\OneDrive\Desktop\sprint4-code>kubectl config current-context
mycluster-free/cdsfs6jf0eoc2ibn5qg

C:\Users\rites\OneDrive\Desktop\sprint4-code>kubectl apply -f kubernetes/ibm_deployment.yaml
deployment.apps/personalexpenstracker created

C:\Users\rites\OneDrive\Desktop\sprint4-code>kubectl apply -f kubernetes/flask_service.yaml
service/flask-app-service created

C:\Users\rites\OneDrive\Desktop\sprint4-code>kubectl apply -f kubernetes/flask_ingress.yaml
ingress.networking.k8s.io/flask-app-ingress created

C:\Users\rites\OneDrive\Desktop\sprint4-code>
```


8.TESTING

8.1 TEST CASES

			Date		12-Nov-22										
			Team ID		PN72022/MD15517										
			Project Name		Project - Personal Expense Tracker										
			Maximum Marks		4 marks										
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By		
Limiting expense Test case 001	Functional	Dashboard	User should be able to add and analyze the expenses.	Existing user	1.Click on limiting expense. 2.Select add category 3.New category clickadd		User should be able to view the analysis.	working as expected	Pass				Hansen S.R		
Manage Expenses test case 002	Functional	Manage Expense	The application should show the details for all the handled expenses.	Existing user	1 Click on the dashboard. 2 Click on the my expense dropdown. 3 Select the handle expense.		The application will show the details of all the handled expenses.	working as expected	pass				Kaarthana M		
Virtual Assistance feature 003	Functional	Virtual Assistant	To check if the Virtual Assistant is responsive.	Virtual Assistant	1 click the VirtualAssistant in bottom right corner. 2 Ask questions to the Virtual Assistant.		The Virtual Assistant should be able to answer all the questions.	Working as expected	Pass				Kaanya S		
My profile Test Case 004	Functional	Profile	The user should be able to edit and update their profile details.	Existing user	1 Click on the profile. 2 Edit the profile.		The user should be able to view the profile.	Working as expected	pass				Borghana Neha S		
My Expense test case 005	Functional	My expense	User should be able to edit and delete the expense details.	Existing user	1 Click on the Expense dropdown box. 2 Click on handle Expense. 3 Click on edit or delete the details.		The user should be able to edit or delete the details.	Working as expected	Pass				Kaarthana M		
My profile picture test case 006	Functional	My Profile	The user should be able to update their picture.	Existing user	1 Click on the profile. 2 Select on edit profile 3 Click on upload. 4 Click limiting Expense. 5 Click the monthly view chart.		The user will be able to update the profile picture.	Working as expected	Pass				Kaanya S		
Stat Analysis 007	Functional	Stat Analysis	The chart should be viewed by the user	Existing user			The application should display the monthly chart.	Working as expected	Pass				Hansen S.R		
View previous Expense Test case 008	Functional	View history	Previous Expense should be able to access by the user.	Existing user	1 Click the View previous expense. 2 Edit the details.		Application should show the previous Expense.	Working as expected	Pass				Kaarthana M		
Dashboard test case 009	Functional	Dashboard	Abble to add expense but cannot able to edit it.	Existing user	Click the expense button under the expense.		Can able to add and change expense in case of any modification	Working as expected	Pass				Borghana Neha S		
Need Help Test case 010	Functional	Need Help	If the user needs any assistance in using the application feature, should be able to use the need help option.	Existing user	1 click on my profile 2 click on need help page		The user should be redirected to the need help page.	Working as expected	Pass				Kaanya S		
Dashboard test case 011	Functional	Dashboard	To manage the expenses and saving history of the user	Existing User	1. Click logged in your account you will see the three horizontal line. 2. Below the dashboard you will see the expense dropdown. 3. On clicking the dropdown you will see the various menu.		View the history of expenses and savings.	Working as expected	Pass				kaarthana M		
Dashboard test case 012	Functional	Dashboard	Abble to see the complete history of an user	Existing User	1. Click the View history under expenses. 2. In the right pane you will see the complete history of an user.		Can able to add and change it incase of any modifications	Working as expected	Pass				Kaanya S		
Dashboard test case 013	Functional	Dashboard	Abble to add the expense but cant able to edit it	Existing User	1. Add expense button under the expense tab.			Working as expected	pass				Hansen S.R		

Profile feature 014	Functional	Profile	Add Multiple accounts for multiple user.	Existing User and New User	1. In the top right click the profile icon. 2. In that you will see the multiple options with respect to the user.		User able to logout from the current profile and switch onto another profile.	Working as expected	Pass				Kaarthana M		
Signup page test case 015	Functional	Signup Page	If the user is new he/she can be able to register themselves in the sign up page.	New user	1 Enter URL and click enter. 2 Click the sign up button. 3 Enter the details for registration.		Signup popup should display	Working as expected	Pass				Kaanya S		
LoginPage test case 016	Functional	Sign in Page	Existing user can be able to login using the credentials.	Existing user	1 Click sign in. 2 Enter the credentials. 3 Verify sign in popup with below elements: a email text box b password text box c Login button		The user can be able to login with the credentials.	Working as expected	Pass				Hansen S.R		
Sign up Page test case 017	Functional	Sign up page	Verify the user that he/she is entering the valid password format.	New user	1 Click the sign up button. 2 Enter the username. 3 Enter valid email. 4 Enter valid password.		The application must show that the password should not contain these special characters.	Working as expected	Pass				Borghana Neha S		
Sign up page test case 018	Functional	Sign up page	If the given credentials are not registered in the database it will show a popup.	New user	1 Click the sign up button. 2 Enter the username. 3 Enter valid email. 4 Enter valid password. 5 Click on Sign up button.		Application should redirect to the sign up page.	Working as expected	Pass				Kaarthana M		
Edit category test case 019	Functional	Expenses	The user can edit the category list after it is created.	Existing user	1. Click on the expense. 2 Click on the edit category list. 3 Click the sign up button. 4 Enter the username. 5 Enter valid email. 6 Enter valid password. 7 Click on Sign up button.		The user will be able to edit the category list.	Working as expected	pass				Kaanya S		
Registration test case 020	Functional	Sign in page	When the user registers he/she will be sent a confirmation email.	Existing user			The user will be sent an email.	Working as expected	pass				Borghana Neha S		
Limit Exceed test case 021	Functional	Limiting Expense	When the user exceeds the limit he/she will be sent a Limit Exceeded mail message.	Existing user	1. The user adds the expense. 2. An email will be sent if the limit is exceeded.		The user will be alerted with an email.	Working as expected	pass				Kaarthana M		

8.1 USER ACCEPTANCE TESTING

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	6	6	6	2	20
Duplicate	2	1	2	2	7
External	0	3	4	7	14
Fixed	4	0	1	5	10
Not Reproduced	0	1	2	1	4
Skipped	0	0	5	3	8
Won't Fix	5	6	3	2	16
Totals	17	16	23	22	79

3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	6	0	0	6
Client Application	33	0	0	33
Security	4	0	0	4
Outsource Shipping	6	0	0	6

Exception Reporting	3	0	0	3
Final Report Output	1	0	0	1
Version Control	2	0	0	2

9. RESULTS

Personal finance management is an important part of people's lives. However, everyone does not have the knowledge or time to manage their finances in a proper manner. And, even if a person has time and knowledge, they do not bother with tracking their expenses as they find it tedious and time-consuming. Now, you don't have to worry about managing your expenses, as you can get access to an expense tracker that will help in the active management of your finances.

The project that we have developed is work more efficient than other income and expense tracker. The project successfully avoids the manual calculation for avoiding calculating the income and expense month. The newly developed system consumes less processing time and all the details are updated and processed immediately and made efficient.

10. ADVANTAGES & DISADVANTAGES

10.1 ADVANTAGES

- Prioritize Your Spending
- Become Aware of Poor Spending Habits
- Identify Fraud
- Take Control of Your Finances

10.2 DISADVANTAGES

- Less Secured
- Limited Accessibility

CONCLUSION

Personal Expense Tracker Application is an web based application. We created this application so that a user can accurately calculate his daily cost. Using this application, the user will see the amount of his income and how much a user is spending, and a notification will be sent to the user if he exceeds the limit and also a report is generated.

12.FUTURE SCOPE

Now in our application we covered almost all features but in the future we will add some more futures. The features are below

- Multiple account support.
- Include currency converter.

13 .APPENDIX

13.1 GITHUB LINK

<https://github.com/IBM-EPBL/IBM-Project-34240-1660233307>

13.2 PROJECT DEMO LINK:

https://drive.google.com/file/d/1ndOG_4wLb3FPJtE7agicJ17Ox8H8zgQR/view?usp=sharing

13.3 SAMPLE CODE

App.py

```
from flask import Flask, render_template, request, redirect, url_for,
session, flash
import credentials
import re
import random
import ibm_db
import string
from datetime import datetime
from flask_mail import Mail, Message

conn =
ibm_db.connect("DATABASE=bludb;HOSTNAME=21fecfd8-47b7-4937-840d-d791d02186
60.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31864;SECURITY=SSL;
SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=lvh24264;PWD=gZS5lI5g0AJ
3CrRN",'','')
app = Flask(__name__)
app.secret_key = 'ibm'
app.config['SECRET_KEY'] = 'top-secret'
app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'apikey'
app.config['MAIL_PASSWORD'] = credentials.SENDGRID_API_KEY
app.config['MAIL_DEFAULT_SENDER'] = credentials.MAIL_DEFAULT_SENDER

mail = Mail(app)

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

@app.route('/login',methods =['GET', 'POST'])
def login():
    # global userid
```

```

msg = ''

if request.method == 'POST' :
    email = request.form['email']
    password = request.form['password']
    sql = "SELECT * FROM credential WHERE email = ? and password = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, email)
    ibm_db.bind_param(stmt, 2, password)
    result = ibm_db.execute(stmt)
    account = ibm_db.fetch_row(stmt)

    param = "SELECT * FROM credential WHERE email = " + "\"" + email +
"\\" + " and password = " + "\"" + password + "\""
    res = ibm_db.exec_immediate(conn, param)
    dictionary = ibm_db.fetch_assoc(res)

    if account:
        session['loggedin'] = True
        session['email'] = dictionary["EMAIL"]
        session['userid']=dictionary["USERID"]
        session['username']=dictionary["USERNAME"]
        sql = "SELECT * FROM user WHERE email = ? "
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        result = ibm_db.execute(stmt)
        account = ibm_db.fetch_row(stmt)
        lineChart = barChartFetchData(dictionary["USERID"])
        barChart = lineChartFetchData(str(session['userid']))
        categoryNames = getCategoryNames()

        return
    render_template('base.html',lineChart=lineChart,barChart=barChart,category
Names=categoryNames,size = len(barChart))
    else:
        msg = 'Incorrect username / password !'

    return render_template('login.html', msg = msg)

@app.route('/register', methods =['GET', 'POST'])

```

```

def register():
    response = ''
    if request.method == 'POST' :
        name = request.form['username']
        email = request.form['email']
        password = request.form['password']
        sql = "SELECT * FROM credential WHERE email = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_row(stmt)

        if account:
            response = 'Username already exists !'
        elif not re.match(r'^@]+@[^@]+\.[^@]+', email):
            response = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', name):
            response = 'name must contain only characters and numbers !'
        else:
            sql2 = "INSERT INTO credential (username,
email,password,userid) VALUES (?, ?, ?,?)"
            stmt2 = ibm_db.prepare(conn, sql2)
            ibm_db.bind_param(stmt2, 1, name)
            ibm_db.bind_param(stmt2, 2, email)
            ibm_db.bind_param(stmt2, 3, password)
            currentid=createId('')
            ibm_db.bind_param(stmt2,4,currentid)
            ibm_db.execute(stmt2)
            initializeUser(currentid,name,email)
            response = 'You have successfully registered !'
            sendMail('Send Grid Registration Successful!!','Registration
Successful','You have successfully created an account in Personal Expense
Tracker!',email)
            return render_template('login.html', response = response)
        else:
            return render_template('register.html')

@app.route('/base')
def dashboard():
    lineChart = barChartFetchData(str(session['userid']))

```

```

    barChart = lineChartFetchData(str(session['userid']))
    categoryNames = getCategoryNames()

    return render_template('base.html',lineChart=lineChart,barChart=barChart,categoryNames=categoryNames,size = len(barChart))

@app.route('/add-category',methods=['GET', 'POST'])
def addCategory():
    if request.method == 'POST':
        categoryName = request.form['category']
        limit = float(request.form['range'])
        description = request.form['description']
        sql = "INSERT INTO category (categoryname,limit,description,userid,balance) VALUES (?, ?, ?,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, categoryName)
        ibm_db.bind_param(stmt, 2, limit)
        ibm_db.bind_param(stmt, 3, description)
        ibm_db.bind_param(stmt, 4, str(session['userid']))
        ibm_db.bind_param(stmt, 5, 0.0 )
        ibm_db.execute(stmt)
        insertBudget(limit,categoryName)
        return render_template('add-category.html')
    else:
        return render_template('add-category.html')
#change
@app.route('/add-expense',methods=['GET', 'POST'])
def addExpense():
    if request.method == 'POST' :
        description= request.form['description']
        date = request.form['date']
        time = request.form['time']
        amount = request.form['amount']
        category = request.form['category']
        paymode= request.form['modeofpayment']
        sql = "INSERT INTO expense (category,amount,modeofpayment,description,userid,expenseid,spentondate,addondate)VALUES (?, ?, ?,?,?,,?,?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, category)

```

```

        ibm_db.bind_param(stmt, 2, amount)
        ibm_db.bind_param(stmt, 3, paymode)
        ibm_db.bind_param(stmt, 4, description)
        ibm_db.bind_param(stmt, 5, str(session['userid']))
        ibm_db.bind_param(stmt, 6, createId('EXP'))
        ibm_db.bind_param(stmt, 7, date)
        ibm_db.bind_param(stmt, 8, datetime.now())
        ibm_db.execute(stmt)
        categories = overallCategory(str(session['userid']))
        updateBudget(category, amount, "increment")
        return render_template('add-expense.html', categories=categories)
    else:
        categories = overallCategory(str(session['userid']))
        return render_template('add-expense.html', categories=categories)

def initializeUser(currentid, name, mail):
    userid=currentid
    username=name
    email=mail
    phoneno=""
    currentsavings=0
    sql = "INSERT INTO user (userid, username, email, phoneno, walletid, currentsavings, country, currency, targetdesc) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, userid)
    ibm_db.bind_param(stmt, 2, username)
    ibm_db.bind_param(stmt, 3, email)
    ibm_db.bind_param(stmt, 4, phoneno)
    ibm_db.bind_param(stmt, 5, createId('WID'))
    ibm_db.bind_param(stmt, 6, currentsavings)
    ibm_db.bind_param(stmt, 7, "INDIA")
    ibm_db.bind_param(stmt, 8, "RUPEES")
    ibm_db.bind_param(stmt, 9, "")
    ibm_db.execute(stmt)

defaultCategories=["Food", "Entertainment", "Business", "Rent", "EMI", "Other"]
for i in defaultCategories:

```

```

        sql = "INSERT INTO category
(USERID,LIMIT,DESCRIPTION,BALANCE,CATEGORYNAME) VALUES (?, ?, ?, ?, ?) "
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,userid)
        ibm_db.bind_param(stmt,2,5000.0)
        ibm_db.bind_param(stmt,3,"")
        ibm_db.bind_param(stmt,4,0.0)
        ibm_db.bind_param(stmt,5,i)
        ibm_db.execute(stmt)
    for i in defaultCategories:
        sql = "INSERT INTO budgeting (USERID,CATEGORYNAME,LIMIT,BALANCE)
VALUES (?, ?, ?, ?) "
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,userid)
        ibm_db.bind_param(stmt,2,i)
        ibm_db.bind_param(stmt,3,5000.0)
        ibm_db.bind_param(stmt,4,0.0)
        ibm_db.execute(stmt)
    #initialize budgeting

@app.route("/edit-limit")
def editLimit():
    if request.method == 'POST' :
        sql = "UPDATE category SET LIMIT=?, DESCRIPTION=? WHERE USERID=? AND
CATEGORYNAME=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, request.form['range'])
        ibm_db.bind_param(stmt, 2, request.form['description'])
        ibm_db.bind_param(stmt, 3, str(session['userid']))
        ibm_db.bind_param(stmt, 4, request.form['category'])
        ibm_db.execute(stmt)
        categories=getAllCategoryDetails()
    else:
        categories=getAllCategoryDetails()
    return render_template('edit-limit.html',categories=categories)

def getAllCategoryDetails():
    sql = "SELECT * from category where userid = ?"
    stmt = ibm_db.prepare(conn, sql)

```



```

    ibm_db.bind_param(stmt,1,str(session['userid']))
    ibm_db.execute(stmt)
    category = ibm_db.fetch_both(stmt)
    categoryList = []
    while category != False:
        categoryList.append(category)
        category = ibm_db.fetch_both(stmt)
    return categoryList

@app.route("/view-history")
def viewHistory():
    res = allExpenses(str(session['userid']))
    return render_template('view-history.html' ,expense = res)

def overallCategory(userid):
    sql = "SELECT * from category where userid = ?"
    statement = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(statement,1,userid)
    ibm_db.execute(statement)
    category = ibm_db.fetch_both(statement)
    categoryList = []
    while category != False:
        categoryList.append(category)
        category = ibm_db.fetch_both(statement)
    return categoryList

#change
@app.route("/manage-expense",methods=['GET', 'POST'])
def manageExpense():
    if request.method=='POST' and request.form['action']=='edit':
        editExpense(request)
    elif request.method=='POST' and request.form['action']=='delete':
        deleteExpense(request.form['expenseid'])
    categories = overallCategory(str(session['userid']))
    expenses = allExpenses(str(session['userid']))

```

```

return
render_template('manage-expense.html',expense=expenses,categories=categories)

#change
@app.route("/view-profile",methods=['GET', 'POST'])
def viewProfile():
    if request.method == 'POST':
        print()
    else:
        sql = "SELECT * from user where userid = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,str(session['userid']))
        ibm_db.execute(stmt)
        userProfile = ibm_db.fetch_assoc(stmt)
        return render_template('view-profile.html',userProfile=userProfile)

@app.route('/report')
def report():
    lineChart = barChartFetchData(str(session['userid']))
    barChart = lineChartFetchData(str(session['userid']))
    categoryNames = getCategoryNames()

    return
render_template('report.html',lineChart=lineChart,barChart=barChart,categoryNames=categoryNames,size = len(barChart))

@app.route('/sign-out')
def logout():
    session.pop('loggedin', None)
    session.pop('userid', None)
    session.pop('email', None)
    return render_template('logout.html')

# all db functions
def allExpenses(userid):
    sql = "SELECT * from expense where userid = ? ORDER BY SPENTONDATE
desc"

    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,userid)
    ibm_db.execute(stmt)

```

```

        expense = ibm_db.fetch_both(stmt)
        expensesList = []
        while expense != False:
            expense['SPENTONDATE'] = expense.get('SPENTONDATE').strftime("%d-%m-%Y")
            expense['ADDONDATE'] = expense.get('ADDONDATE').strftime("%d-%m-%Y")
            expensesList.append(expense)
            expense = ibm_db.fetch_both(stmt)
        return expensesList

def barChartFetchData(userid):
    epenseMonthwise = {
        '1':0.0,'2':0.0,'3':0.0,'4':0.0,'5':0.0,'6':0.0,'7':0.0,'8':0.0,'9':0.0,'10':0.0,'11':0.0,'12':0.0}
    expenses = allExpenses(userid)
    for expense in expenses:
        date = expense.get('SPENTONDATE')
        datemonth = datetime.strptime(date, "%d-%m-%Y")
        epenseMonthwise[str(datemonth.month)] = epenseMonthwise[str(datemonth.month)] + expense.get('AMOUNT')
    return list(epenseMonthwise.values())

def categoryChart(userid):
    epenseMonthwise = {
        '1':0.0,'2':0.0,'3':0.0,'4':0.0,'5':0.0,'6':0.0,'7':0.0,'8':0.0,'9':0.0,'10':0.0,'11':0.0,'12':0.0}
    expenses = allExpenses(userid)
    for expense in expenses:
        date = expense.get('SPENTONDATE')
        datemonth = datetime.strptime(date, "%d-%m-%Y")
        epenseMonthwise[str(datemonth.month)] = epenseMonthwise[str(datemonth.month)] + expense.get('AMOUNT')
    return list(epenseMonthwise.values())

def getExpenseForCategory():
    sql="SELECT DISTINCT CATEGORY FROM expense WHERE USERID=? "
    stmt = ibm_db.prepare(conn, sql)

```

```

    ibm_db.bind_param(stmt,1,str(session['userid']))
    ibm_db.execute(stmt)
    result = ibm_db.fetch_assoc(stmt)

def createId(pre):
    return pre+''.join([random.choice(string.ascii_letters+ string.digits)
for n in range(32)])

def getCategoryNames():
    sql="SELECT DISTINCT CATEGORY FROM expense WHERE USERID=? "
    statement = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(statement,1,str(session['userid']))
    ibm_db.execute(statement)
    category = ibm_db.fetch_assoc(statement)
    categoryNames = []
    while category != False:
        categoryNames.append(category.get('CATEGORY'))
        category = ibm_db.fetch_both(statement)
    return categoryNames

def lineChartFetchData(userid):
    categoryData = []
    categories = getCategoryNames()
    expenses = allExpenses(userid)
    for category in categories:
        epenseMonthwise = {}
        {'1':0.0,'2':0.0,'3':0.0,'4':0.0,'5':0.0,'6':0.0,'7':0.0,'8':0.0,'9':0.0,'10':0.0,'11':0.0,'12':0.0}
        for expense in expenses:
            if(expense.get('CATEGORY')==category):
                date = expense.get('SPENTONDATE')
                datemonth = datetime.strptime(date, "%d-%m-%Y")
                epenseMonthwise[str(datemonth.month)] = 0
                epenseMonthwise[str(datemonth.month)] += expense.get('AMOUNT')
            categoryData.append(list(epenseMonthwise.values()))
    return categoryData

def sendMail(body,htmlHead,message,mailId):
    recipient = []
    recipient.append(mailId)

```

```

msg = Message('Twilio SendGrid', recipients=recipient)
msg.body = (body)
msg.html = ('<h1>'+htmlHead+'</h1><p>'+message+'</p>')
mail.send(msg)
flash(f'A test message was sent to {recipient}.')

def insertBudget(limit, categoryName):
    sql = "INSERT INTO budgeting (userid, limit, balance, categoryName) VALUES
    (?, ?, ?, ?)"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, str(session['userid']))
    ibm_db.bind_param(stmt, 2, limit)
    ibm_db.bind_param(stmt, 3, 0.0)
    ibm_db.bind_param(stmt, 4, categoryName)
    ibm_db.execute(stmt)

def updateBudget(categoryName, amount, function):
    balance = getCurrentBalance(categoryName)
    if(function == 'increment'):
        currBalance = balance.get('BALANCE')+float(amount)
    elif(function == 'decrement'):
        currBalance = abs(balance.get('BALANCE')- float(amount))

    if(balance.get('LIMIT')<currBalance):
        sendMail('Hey there! limit alert .... The Category: '+categoryName+'
has been crossed!',' Category: '+categoryName+' limit has exceeded','The
limit set on '+categoryName+' has been reached. Kindly check and keep track
of your expenses, do exercise caution and avoid spending
more.',str(session['email']))
    sql = "UPDATE budgeting SET balance = ? WHERE userid=? and categoryname
= ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, currBalance)
    ibm_db.bind_param(stmt, 2, str(session['userid']))
    ibm_db.bind_param(stmt, 3, categoryName)
    ibm_db.execute(stmt)

def getCurrentBalance(categoryName):

```

```

    sql = "SELECT balance,limit from budgeting WHERE userid = ? and
categoryname = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, str(session['userid']))
    ibm_db.bind_param(stmt, 2, categoryName)
    ibm_db.execute(stmt)
    balance = ibm_db.fetch_assoc(stmt)
    return balance

def editExpense(request):
    currentBalance = getExpenseBalance(request.form['expenseid'])
    updateBudget(request.form['category'],currentBalance,"decrement")

updateBudget(request.form['category'],float(request.form['amount']),"incre
ment")

    datetimeans = datetime.strptime(request.form['date'], '%d-%m-%Y')
    sql = "UPDATE expense SET CATEGORY=?, AMOUNT=?, MODEOFPAYMENT=?,
DESCRIPTION=?,SPENTONDATE=? WHERE EXPENSEID=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, request.form['category'])
    ibm_db.bind_param(stmt, 2, request.form['amount'])
    ibm_db.bind_param(stmt, 3, request.form['modeofpayment'])
    ibm_db.bind_param(stmt, 5, datetimeans)
    ibm_db.bind_param(stmt, 4, request.form['description'])
    ibm_db.bind_param(stmt, 6, request.form['expenseid'])
    ibm_db.execute(stmt)

def deleteExpense(expenseid):

updateBudget(request.form['category'],request.form['amount'], "decrement")
    sql="DELETE FROM expense WHERE EXPENSEID = ?;"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,expenseid)
    ibm_db.execute(stmt)

def getExpenseBalance(expenseid):
    sql = "Select amount from expense where expenseid= ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, expenseid)

```



```
    ibm_db.execute(stmt)
    amount = ibm_db.fetch_assoc(stmt)
    return amount.get('AMOUNT')

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

CREDENTIALS.PY:

```
SENDGRID_API_KEY="SG.ErCql4TuTdW0BooJ3OrDXw.UZpYp5EQzgVQzo2Lkpai9avBIeWBtb  
u7TiRY1eCUQp0"  
MAIL_DEFAULT_SENDER="harsini.ramalingam@gmail.com"
```