1. INTRODUCTION

1.1. Project Overview

As the name itself suggests, this project is an attempt to manage our daily expenses in a more efficient and manageable way. Sometime we can't remember where our money goes. And we can't handle our cash flow. For this problem, we need a solution that everyone can manage their expenses. So we decided to find an easier way to get rid of this problem. So, our application attempts to free the user with as much as possible the burden of manual calculation and to keep the track of the expenditure. Instead of keeping a diary or a log of the expenses, this application enables the user to not just keep the control on the expenses but also to generate and save reports. With the help of this application, the user can manage their expenses on a daily, weekly and monthly basis. Users can insert and delete transactions as well as can generate and save their reports. The graphical representation of the application is the main part of the system as it appeals to the user more and is easy to understand.

Agile is also among the top required skills. AWS knowledge is more demanded

than Azure or GCP.

1.2. Purpose

An expense tracker is a desktop application that keeps track of all your expenses and stores all the information regarding them, including the person to whom you have paid the money (also called payee) and the reason why you paid the money.The objective of this project is to create a GUI based Expense Tracker. To build this, you will need an intermediate understanding of the Tkinter library, SQL language and its commands, and basic understanding of the message box module, ttk.Treeview widget and tkcalender library.

2. LITERATURE SURVEY

2.1. Existing problem

The author describes [1] is aimed at developing an android based mobile application capable of monitoring and controlling personal expenses, as well as cautioning the user against reckless and unbudgeted spending. The developed system was designed using system flowchart, use case diagram, sequence diagram, class diagram and system architecture diagram. It was implemented using Java programming language on android studio and My SQL. The developed system was evaluated based on basic functionality tests performed on the individual modules, the integrated testing as well as the overall function testing. The results of testing the functionalities of the developed system showed that all the modules worked properly when tested individually. They rejected invalid inputs and responded promptly to user requests. Database operations such as insert, update, delete and add that were performed yielded expected results, and data consistency / integrity are maintained in the reports generated. Thus, the developed system provides an easy to use, portable and secured means of enhancing financial sustainability and promotes individual and societal economic growth via fiscal discipline.

The author describes [2] is to avoid Income and Expense calculations and in the same manner to remind a person, we develop an android application which may helpful in all the situations and it can be installed in our android phones. It help us to remind and add some information that what are the income comes from other persons and what are all the expenses or payments we have to pay in specific date or month. In expense tracker we have categories like add expense, expenses of each month, add new expense, view categories of expenses, export expenses in a date range, remove export files, view categories wise expenses.

2.2. References

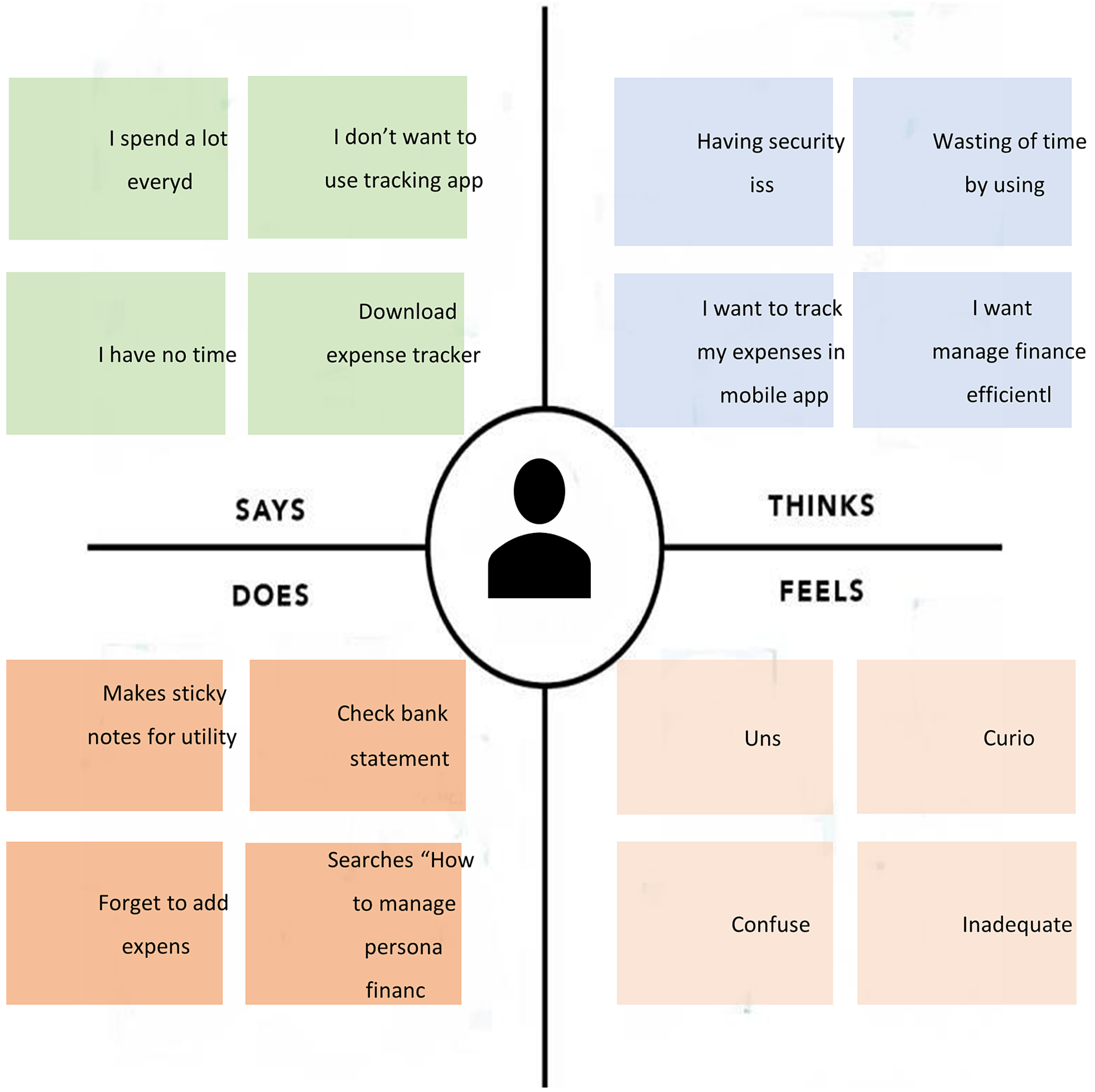
1. https://moneyview.in/insights/best-personal-fnance-management-a pps-in-india
2. https://www.factmr.com/report/personal-fnance-mobile-app-marke t
3. https://www.moneytap.com/blog/best-money-management-apps/
4. https://relevant.software/blog/personal-fnance-app-like-mint/
5. https://www.onmanorama.com/lifestyle/news/2022/01/11/fnancial-l iteracy-trend-among-todays-youth-investment.html
6. https://www.livemint.com/money/personal-fnance/96-indian-parent s-feel-their-children-lack-fnancial-know-how-survey-1166133611085 5.html
7. https://economictimes.indiatimes.com/small-biz/money/importance -of-fnancial-literacy-amongst-youngsters/articleshow/85655134.cms
8. https://www.news18.com/news/education-career/only-27-adults-16-7-of-in dian-teenagers-fnancially-literate-4644893.html

2.3. Problem Statement Definition

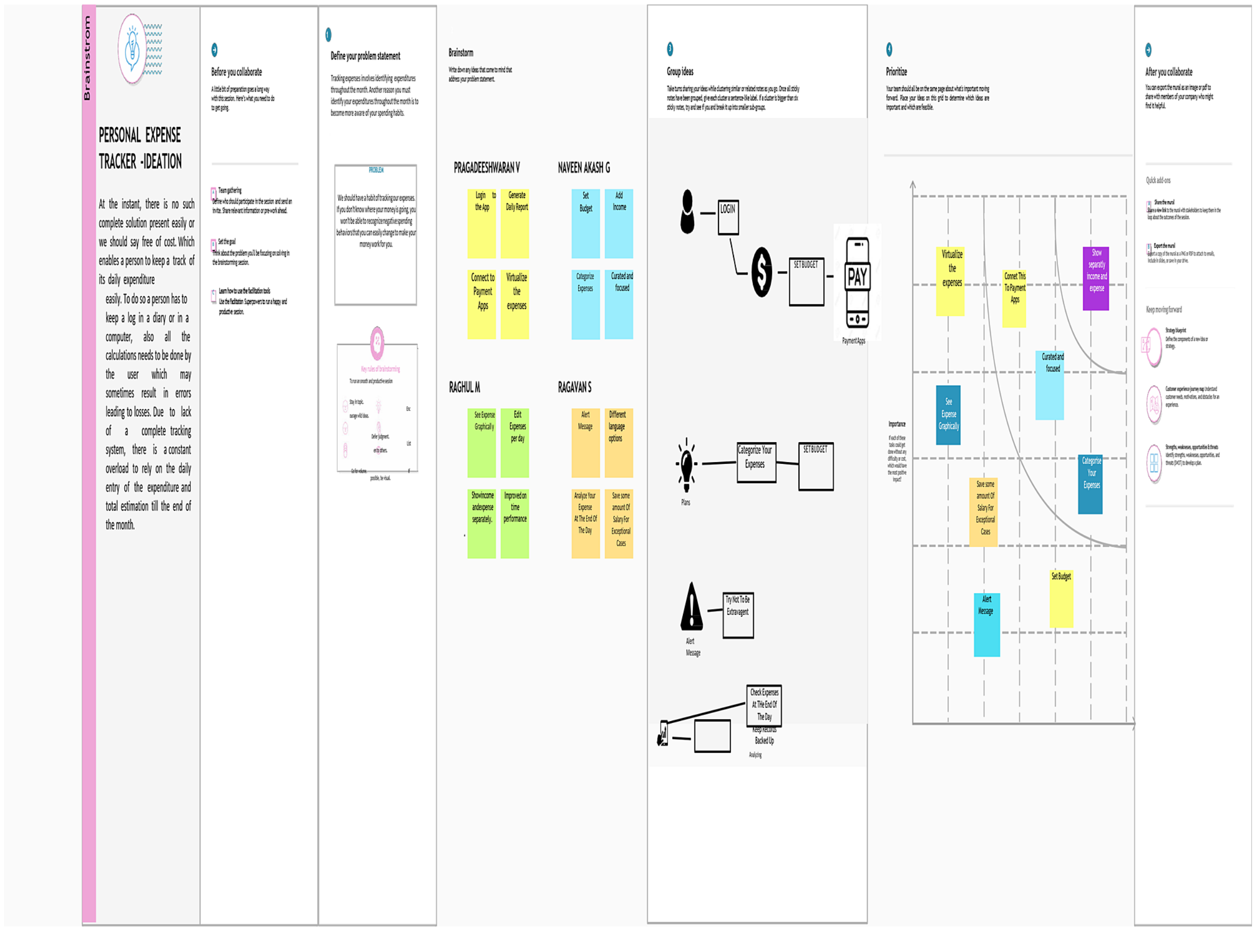
1. Tracking the expenses throughout the month is essential because it provides insight about the way in which the money is spent and also helps to frame a better budgeting plan for the upcoming days. Thus, personal expense tracker application has made tracking and managing expenses a breeze. Who does the problem affect? Investors, savers, big spenders, debtors ,shoppers, budget conscious consumers. What are the boundaries of the problem? Expense tracker for working individuals, students, common people. What is the issue? To be vigilant about the expense spent, increases financial stress. Being indecisive about the finances may result in less financial security and exceed the budget. When does this issue occur? When using wrong budgeting techniques. When not tracking the expenses doesn't help you to know the amount that is actually spent. Where is the issue occurring? Working individuals who find it difficult to track

their expenses Why is it important that we fix the problem? Fixing this issue, brings accountability and helps to be intentional with the income by assign it to spending,saving and giving. This leads to financialstability. Abella ,who is a shopholic ,finds it hard to control her desire to shop .To stop her from overindulging in impulsive purchases, she needs to track her expenses and hold herself accountable. John, who is interested to invest in stocks, finds it difficult to figure out the expense that he can spend on investing stocks. With the help of expense tracking, he can easily plan out the expenses for investing in an efficient way. Akshay, is a high school student, who usually gets a limited allowance from his parents. So tracking his expenses and good budgeting technique allows him to spend on his regular expenses as well as on himself. Udhay ,who is a novice budgeter, finds it tedious to track and manage the expenses amongst his busy schedule . Prioritizing his expenses will help him to curtail his unnecessary expendituresIDEATION & PROPOSED SOLUTION

3.1. Empathy Map Canvas



3.2. Ideation & Brainstorming


Brainstrom

PERSONAL EXPENSE TRACKER -IDEATION

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

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.

A Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

C Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

1 Define your problem statement

Tracking expenses involves identifying expenditures throughout the month. Another reason you must identify your expenditures throughout the month is to become more aware of your spending habits.

PROBLEM

We should have a habit of tracking our expenses. If you don't know where your money is going, you won't be able to recognize negative spending behaviors that you can easily change to make your money work for you.

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.

Brainstorm

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

3 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 and break it up into smaller sub-groups.

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.

Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A Share the mural
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B Export the mural
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.

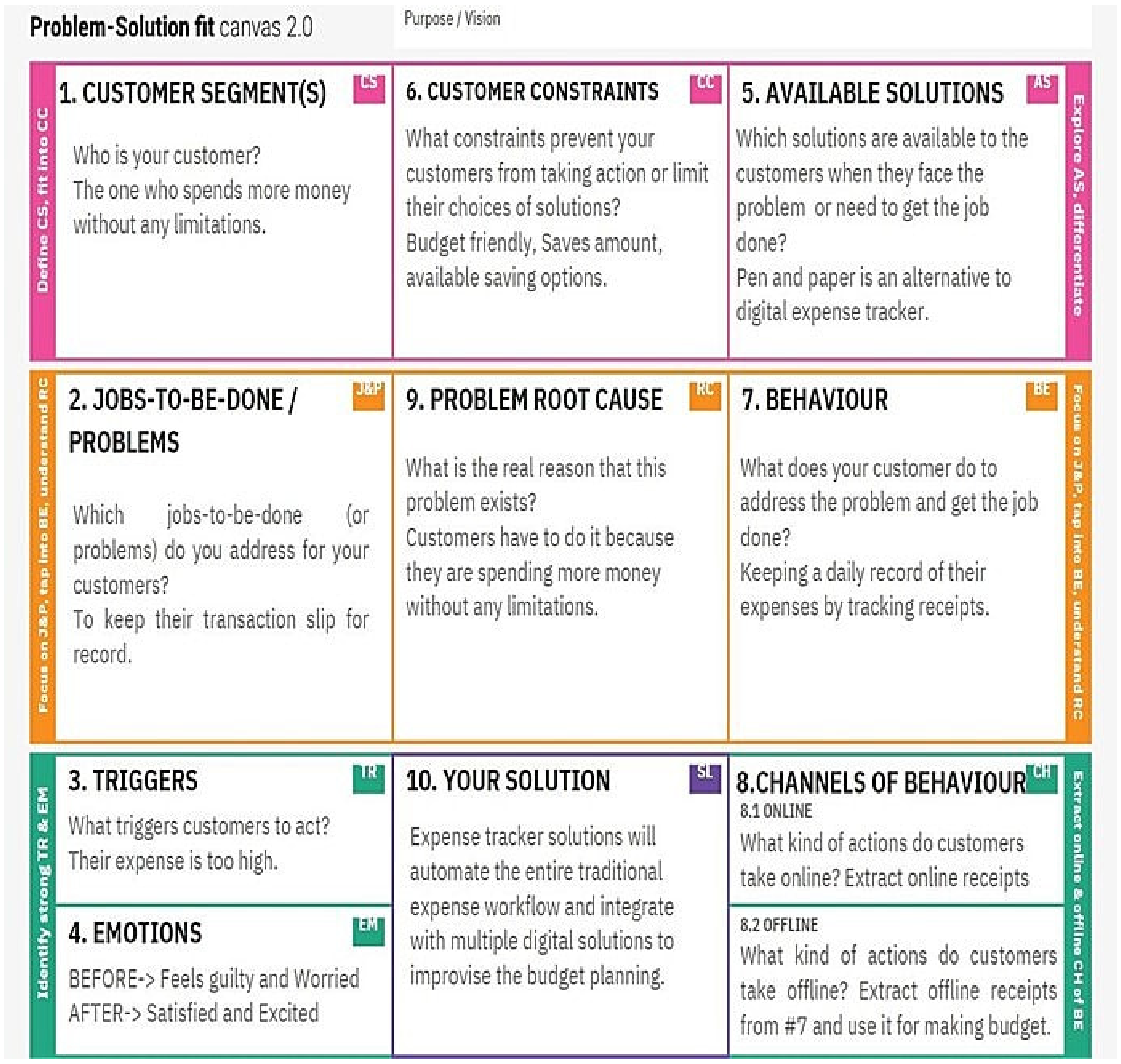
Customer experience journey map Understand customer needs, motivations, and obstacles for an experience.

Strengths, weaknesses, opportunities & threats
Identify strengths, weakness, opportunities, and threats (SWOT) to develop a plan.

3.3. Proposed Solution

IDEA / SOLUTION DESCRIPTION The goal is to move from, at most, a spreadsheet to, at worst, some sort of collection and categorization of money spent. That is something that apps are capable of. The experience, though, differs greatly. These are the most widely used apps for iOS and Android, but you'll probably need to test out a few before you discover one you enjoy. While the majority of the apps on our list are free, you'll probably need to upgrade for a small cost to access all the features. NOVELTY / UNIQUENESS It's time to quit maintaining records of your cash payments and online transactions on paper and in Excel spreadsheets! Papers are more hazardous to the environment and difficult to handle. Excel sheets, on the other hand, might provide an online solution but aren't very helpful for handling money. Therefore, it is preferable to create money management software that gathers information from the data and aids in corporate decision-making. With regards to keeping track of spending, you are never able to locate the cash or digital payments that you have made. Therefore, you should use a business spending tracker app if you want to keep track of your financial assets. By simply clicking on the photos of the receipts in your costs management app, you may store every receipt. Financial affairs handled manually cannot correctly verify each transactional information. Fraud cases occur frequently as a result of this. The procedure of handling money and finances is automated when using a spending monitoring and budgeting app. This not only stops fraud but also improves the process' accuracy and transparency. SOCIAL IMPACT / CUSTOMER SATISFACTION Tracking your spending is often the first step in getting your finances in order.. You can see exactly where your money is going and where you may make savings by analysing method. With the help of mobile spending tracker applications, you can easily include this into your daily routine. These apps do overlap with budgeting tools, but cost tracker apps focus more on your spending while the latter offers a broad overview of your finances. These apps typically classify your expenses and give you a clear picture of your purchasing behaviour. Whether the app to track your expenses that can quickly record every transaction. BUSINESS MODEL (FINANCIAL BENEFITS) Keeping transaction data and receipts organized: One of the main purposes of an expense tracker is to assist you in keeping an orderly and comprehensive record of various expenses. You must take pictures and screenshots of every receipt and save them through the tracker app in order to manage transaction records and arrange them nearly there. The receipts and records that are maintained in the cloud when you create an expenditure tracking app enable you to access and review them whenever necessary. Proper payment of taxes: An expense tracker app also helps you to pay your taxes in time and stay updated on tax deductions. The tax records only need to be uploaded to the app. The planning apps will keep you uninvolved. Processing invoices and payments: An expense tracker app that allows financial transactions through debit cards, bank transfers, credit cards, and net banking will help you make payments quickly against the invoices. Further, a spending tracking software will send payment reminders and link payments to client accounts. Additionally, the app will assist firms in producing bills that appear professional and include the brand logo for distribution through media. Create in-depth reports: The ability to create and distribute thorough data on profit and loss, company revenues, costs, and the balance sheet makesthe development of business expense tracker apps appealing. With the help of the app, a firm may also produce fully personalized reports with a focus on specific financial aspects. SCALABILITY OF SOLUTION ❖ An expense tracker, often known as an expense manager or money manager, is a piece of software or an application that assists in maintaining accurate records of your money coming in and going out. ❖ In India, many people live on a fixed income, and they find that towards the end of the month they don't have sufficient money to meet their needs..

3.4. Problem Solution fit



Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
The one who spends more money without any limitations.

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions?
Budget friendly, Saves amount, available saving options.

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done?
Pen and paper is an alternative to digital expense tracker.

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers?
To keep their transaction slip for record.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
Customers have to do it because they are spending more money without any limitations.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
Keeping a daily record of their expenses by tracking receipts.

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

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act?
Their expense is too high.

4. EMOTIONS EM

BEFORE-> Feels guilty and Worried
AFTER-> Satisfied and Excited

10. YOUR SOLUTION SL

Expense tracker solutions will automate the entire traditional expense workflow and integrate with multiple digital solutions to improvise the budget planning.

8.CHANNELS OF BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online receipts

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline receipts from #7 and use it for making budget.

Extract online & offline CH of BE

3. REQUIREMENT ANALYSIS

4.1. Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Email/SignUp Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Add expenses	Enter the everyday expenses Split it into categories(example : food, petrol,movies)
FR-4	Reminder mail	Sending reminder mail on target (for ex : if user wants a reminder when his/her balance reaches some amount(5000)) Sending reminder mail to the user if he/she has not filled that day's expenses.
FR-5	Creating Graphs	Graphs showing everyday and weekly expenses. Categorical graphs on expenditure.
FR-6	Add salary	Users must enter the salary at the start of the month.
FR-7	Export CSV	User can export the raw data of their expenditure as CSV

4.2. Non-Functional requirements

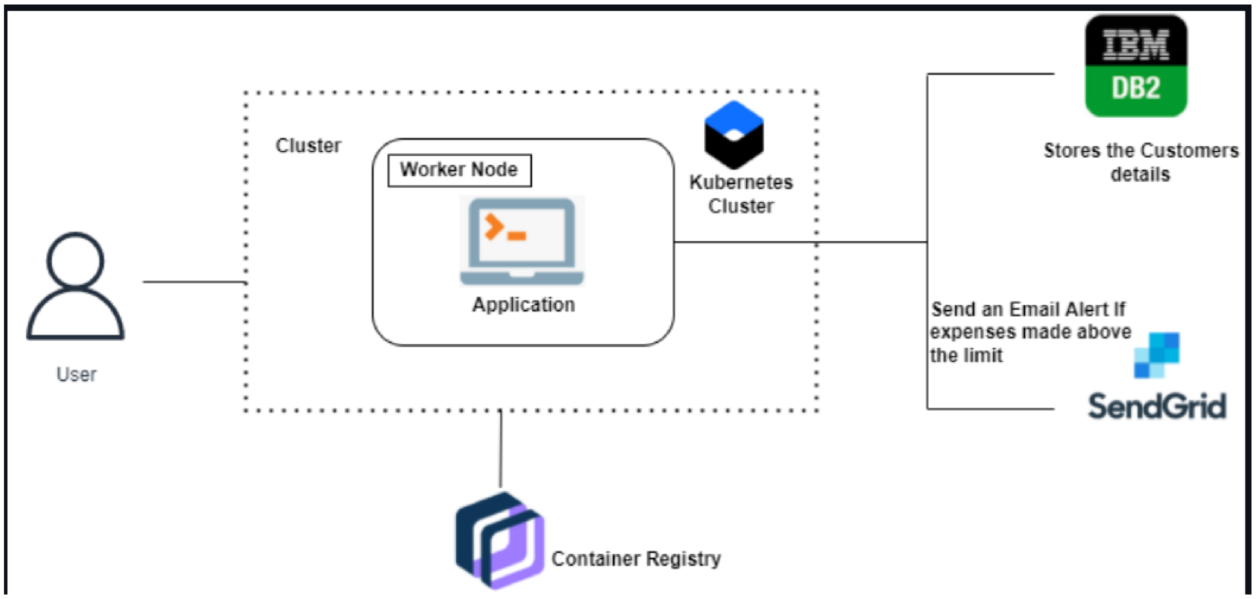
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	A simple web application which is accessible across devices
NFR-2	**Security**	The OAuth Google sign in and email login are secure with hashed and salted secure storage of credentials.
NFR-3	**Reliability**	Containerized service ensures that new instance can kick up when there is a failure
NFR-4	**Performance**	The load is managed through the load balancer used with docker. Thus ensuring good performance

NFR-5	**Availability**	With load balancing and multiple container instances, the service is always available.
NFR-6	**Scalability**	Docker and Kubernetes are designed to accommodate scaling based on need

4. PROJECT DESIGN

5.1. Data Flow Diagrams



5.2. Solution & Technical Architecture+

5.3. User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task
Customer	Registration	USN-1	As a user, I can register for the application

(Mobile user)			by entering my email, password, and confirming my password.
	Login	USN-2	As a user, I can log into the application by entering email & password
	Add	USN -3	As a user , I can add in new expenses.
	Remove	USN–4	As a user , I can remove previously added expenses.
	View	USN-5	As a user , I can view my expenses in the form of graphs and get insights.
	Get alert message	USN-6	As a user , I will get alert messages if I exceed my target amount.
Administrator	Add / remove user	USN–7	As admin , I can add or remove user details on db2 manually.
		USN-8	As admin , I can add or remove user details on sendgrid.

5. PROJECT PLANNING & SCHEDULING

6.1. Sprint Planning & Estimation

Sprint	Functional Requirements (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	PET-1	As a user, I can register for the application by entering my email, password, and confirming my password.	3	High	Pragadeeshwaran V Ragavan S Naveen akash G Raghul M
Sprint-1	Login	PET-2	As a user, I can login to user dashboard and see the information about my incomes and expenses.	3	High	Pragadeeshwarn V Ragavan S Naveen akash G Raghul M

Sprint-2	Dashboard	PET-3	As a user, I can enter my income and expenditure details.	4	High	Pragadeeshwaran V Ragavan S Naveen akash G Raghul M
Sprint	Functional Requirements (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3	Expense Update	PET-4	As a user, I can track my expenses and manage my monthly budget.	2	High	Pragadeeshwaran V Ragavan S Naveen akash G Raghul M
Sprint-3	Email Alert	PET-5	As a user, I can see if there is an excessive expense and if there is such condition, I will be notified via e-mail.	6	Medium	Pragadeeshwaran V Ragavan S Naveen akash G Raghul M
Sprint-4	Customer Care	PET-6	As a customer care executive, I can solve the login issues and other issues of the application.	3	Medium	Pragadeeshwaran V Ragavan S Naveen akash G Raghul M
Sprint-4	Application	PET-7	As an administrator, I can upgrade or update the application.	3	Medium	Pragadeeshwaran V Ragavan S Naveen akash G Raghul M

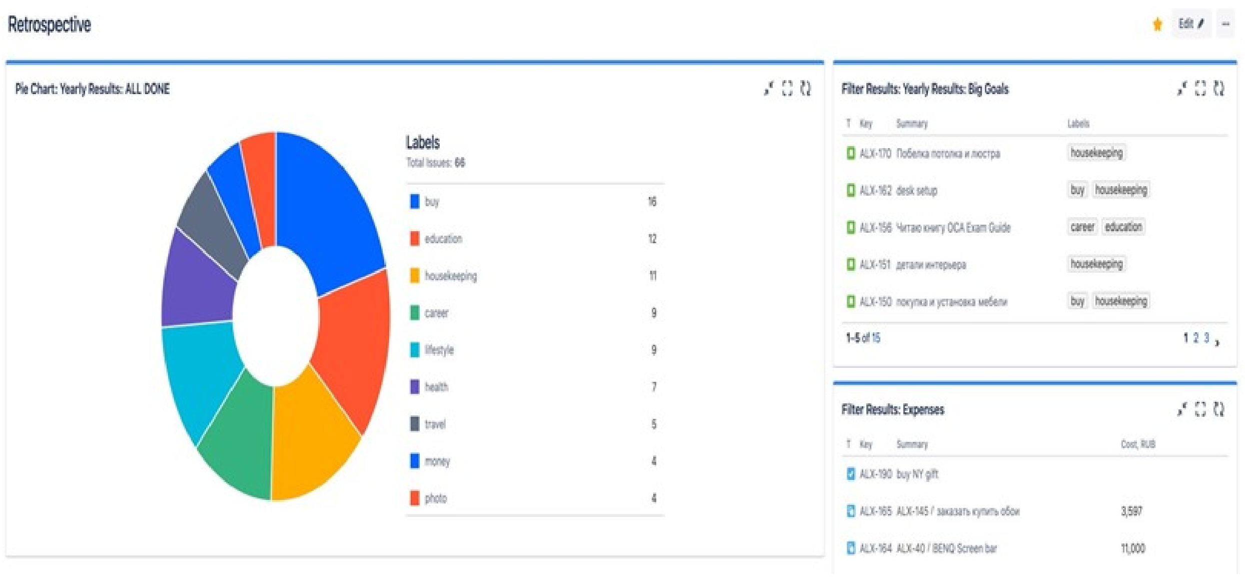
MileStone and Activity List

Sprint	Functional Requirements (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members

Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	3	High	Pragadeeshwaran V Ragavan S Naveen akash G Raghul M
Sprint-1	Login	USN-2	As a user, I can login to user dashboard and see the information about my incomes and expenses.	3	High	Pragadeeshwaran V Ragavan S Naveen akash G Raghul M
Sprint-2	Dashboard	USN-3	As a user, I can enter my income and expenditure details.	4	High	Pragadeeshwaran V Ragavan S Naveen akash G Raghul M
Sprint	Functional Requirements (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3	Expense Update	USN-4	As a user, I can track my expenses and manage my monthly budget.	2	High	Pragadeeshwaran V Ragavan S Naveen akash G Raghul M
Sprint-3	Email Alert	USN-5	As a user, I can see if there is an excessive expense and if there is such condition, I will be notified via email.	6	Medium	Pragadeeshwaran V Ragavan S Naveen akash G Raghul M
Sprint-4	Customer Care	USN-6	As a customer care executive, I can solve the login issues and other issues of the application.	3	Medium	Pragadeeshwaran V Ragavan S Naveen akash G Raghul M
Sprint-4	Application	USN-7	As an administrator, I can upgrade or update the application.	3	Medium	Pragadeeshwaran V Ragavan S Naveen akash G Raghul M

SPRINTS	TOTAL STORY POINTS	DURATION	SPRINT START DATE	SPRINT END DATE	STORY POINTS COMPLETED	SPRINT RELEASE DATE
SPRINT – 1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
SPRINT – 2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
SPRINT – 3	20	6 Days	07 Oct 2022	12 Nov 2022	20	12 Nov 2022
SPRINT – 4	20	6 Days	14 Nov 2022	19 Oct 2022	20	19 Nov 2022

6.3. Reports from JIRA



Labels
Total Issues: 66

🟦	buy	16
🟥	education	12
🟨	housekeeping	11
🟩	career	9
🟦	lifestyle	9
🟪	health	7
⬛	travel	5
🟦	money	4
🟥	photo	4

Filter Results: Yearly Results: Big Goals

T	Key	Summary	Labels
	ALX-170	Побелка потолка и люстра	housekeeping
	ALX-162	desk setup	buy housekeeping
	ALX-156	Читаю книгу OCA Exam Guide	career education
	ALX-151	детали интерьера	housekeeping
	ALX-150	покупка и установка мебели	buy housekeeping

1-5 of 15 1 2 3

Filter Results: Expenses

T	Key	Summary	Cost, RUB
	ALX-190	buy NY gift	
	ALX-165	ALX-145 / заказать купить обои	3,597
	ALX-164	ALX-40 / BENQ Screen bar	11,000

6. CODING & SOLUTIONING

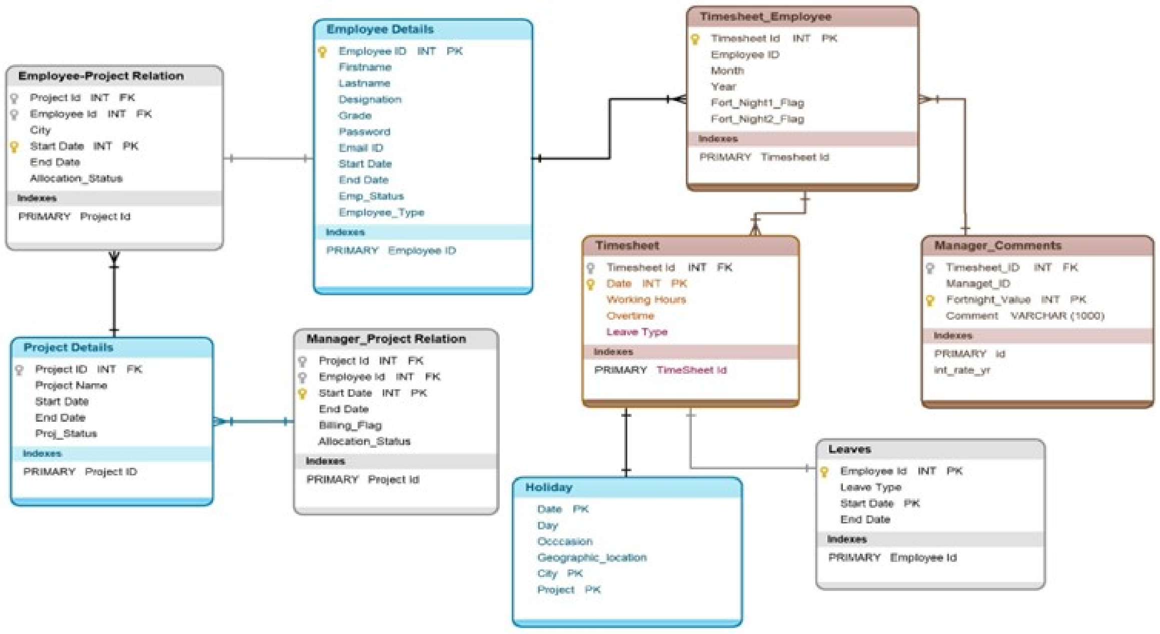
7.1. Feature 1

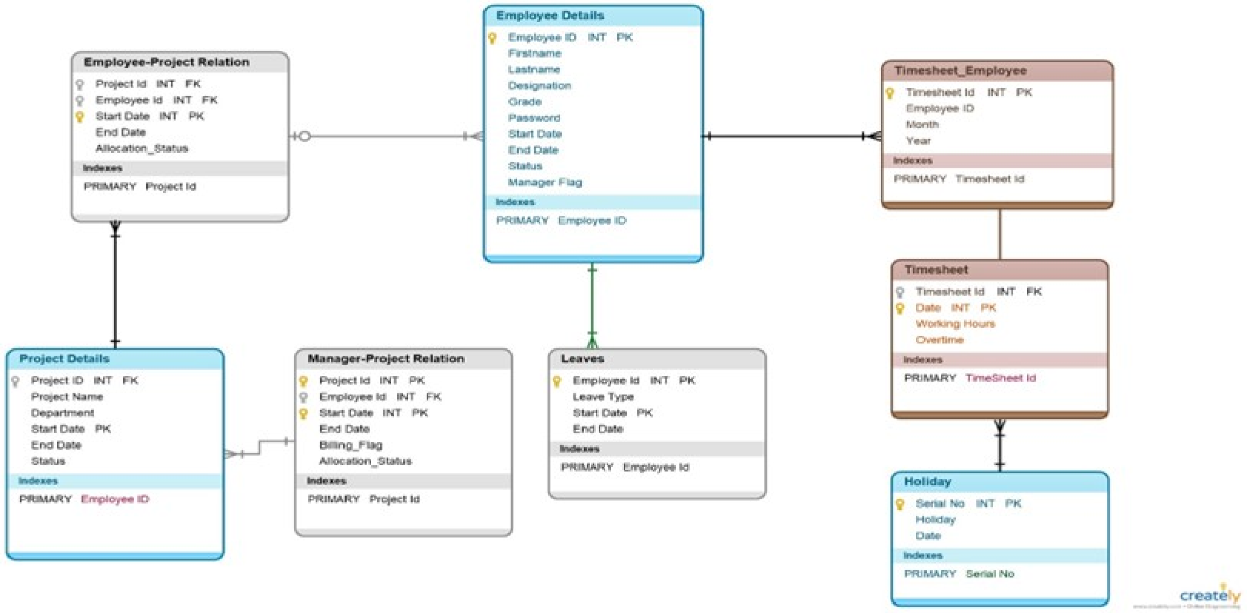
Handle Documents It’s time to stop using paper and excel spreadsheets for keeping records of your cash payments and online transactions! Papers are tough to handle and more dangerous for the environment.On the other hand, excel sheets may offer an online solution but don’t do much help in money handling. So, it’s better to develop money management software that collects insights from the data and helps make business decisions.Tracks Receipts You always can’t find the cash and digital payments made by you and this is a big issue with tracking expenses.So, if you want to keep a track of your monetary investments, you should go using a business expense tracker app. This helps store all receipts by only clicking their images in your expenses handling app.Prevents Data Losses and Frauds Manual handling of personal expenses and finances can’t check every transaction detail accurately.For this reason, fraud cases happen many times. Using an expense tracking and budgeting app, the workflow of money and finance handling becomes automated. This not just prevents fraudulence but also makes the procedure more accurate and transparent.

7.2. Feature 2

Mitigates Human Errors We cannot afford mistakes when it comes to handling budgets and finances. However, humans may make some errors because of misunderstanding, carelessness, or negligence.With the help of an expense handling app, you can lower as well as prevent every mistake caused because of carelessness.Offers Precise Analytics Excel spreadsheets might track and store data and create helpful charts or graphs from it but still have no advanced functionality. On the other hand, human engagement brings the possibility of mistakes.So, it’s best to build a business expense tracker app that carries out prediction analysis and helps you make efficient business decisions.

7.3. Database Schema





TESTING

8.1. Test Cases

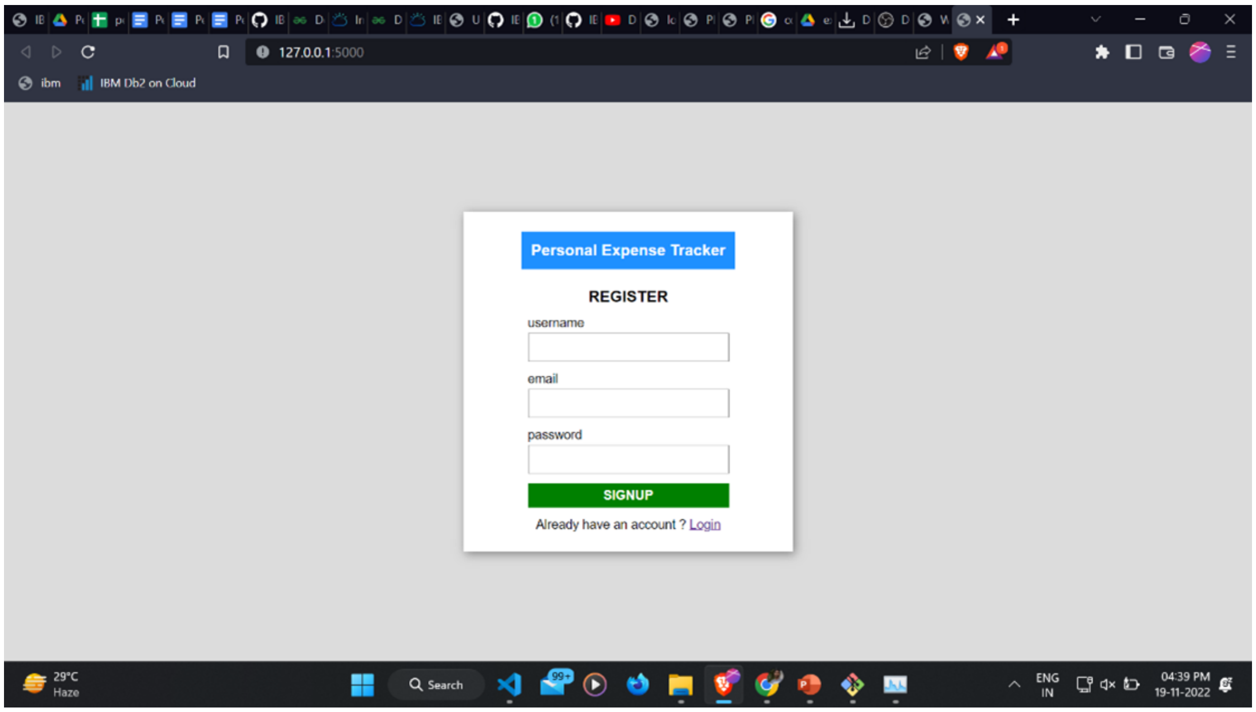
With a concerted effort, I conducted research on general well-being to have a rudimentary grasp on health management, as well as the existing job/skill recommender apps in order to get an understanding of what is already existing in the market, the characteristics, specialties, and usability.There are a considerable number of job/skill recommender apps tracking apps existing in the market. They aim to track daily spends intake by logging info given to achieve users' preset goals. To log spend, users can input the expense in the app, or scan the barcode of a package. Most apps allow users to connect with associated activities apps to track spend progress. With a premium upgrade, users can get access to tailor-made saving according to health goals or specified way to spend.In order to build a realistic initial target group, I wanted to conduct some usability tests with 5 users that regularly engage in buying activity and spending tracking, including both first-time and regular users of money planning . I asked these individuals to perform tasks related to general usage

8.2. User Acceptance Testing

Must-have features of aJob/Skill recommender app I wanted to address the user pain points by including (and improving) the core features of the application.Personal profiles After downloading the app, a user needs to register and create an account. At this stage, users should fill in personal information like name, gender, age, height, weight, spend preferences, spend logging and dashboard Allowing users to analyze their spending habits. They should be able to log expenses and money intake and see their progress on a dashboard that can track overall spends.Push notifications Push notifications are an effective tool for increasing user engagement and retention. To motivate users to keep moving toward their goals, it's pertinent to deliver information on their progress toward the current goal and remind them to log what they spend on.money counter Enabling the application to calculate spent amount of users have gone and done based on the data they've logged.Barcode scanner Let users count money and see accurate spend information via a built-in barcode scanner.

7. RESULTS

9.1. Performance Metrics


127.0.0.1:5000

Personal Expense Tracker

REGISTER

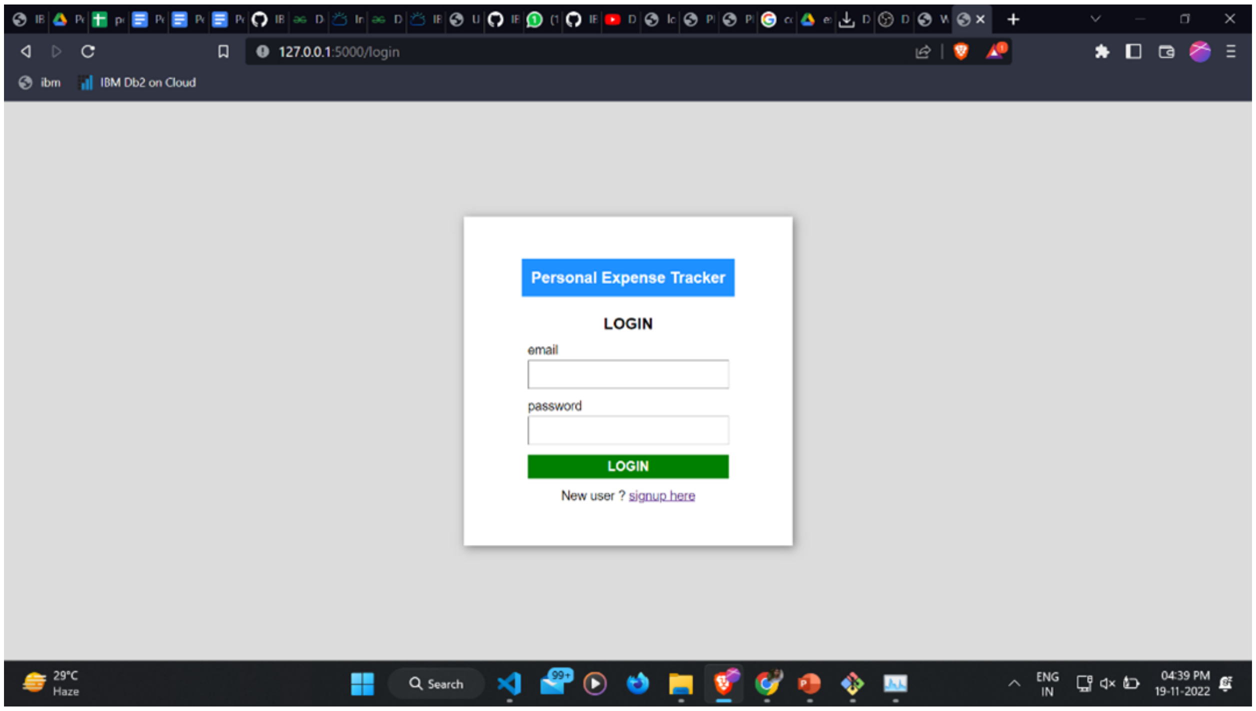
username

email

password

SIGNUP

Already have an account ? Login


127.0.0.1:5000/login

Personal Expense Tracker

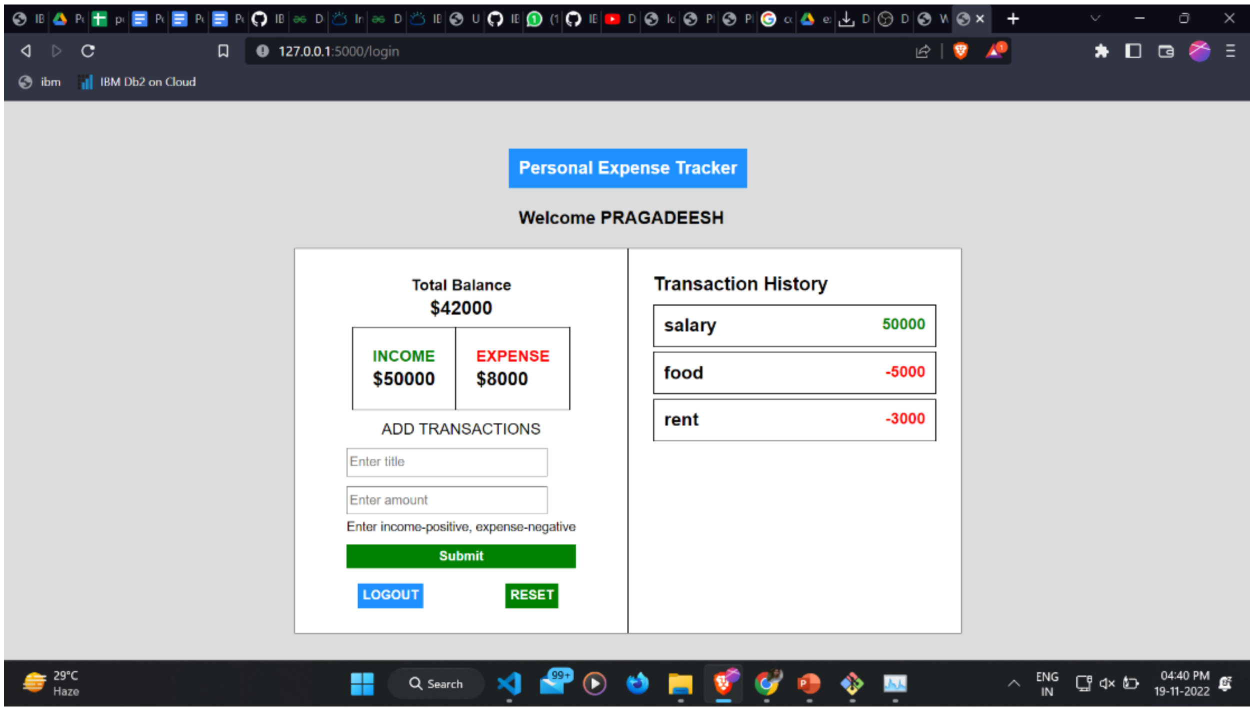
LOGIN

email

password

LOGIN

New user ? signup here


127.0.0.1:5000/login

ibm IBM Db2 on Cloud

Personal Expense Tracker

Welcome PRAGADEESH

Total Balance
$42000

INCOME	EXPENSE
$50000	$8000

ADD TRANSACTIONS

Enter title

Enter amount

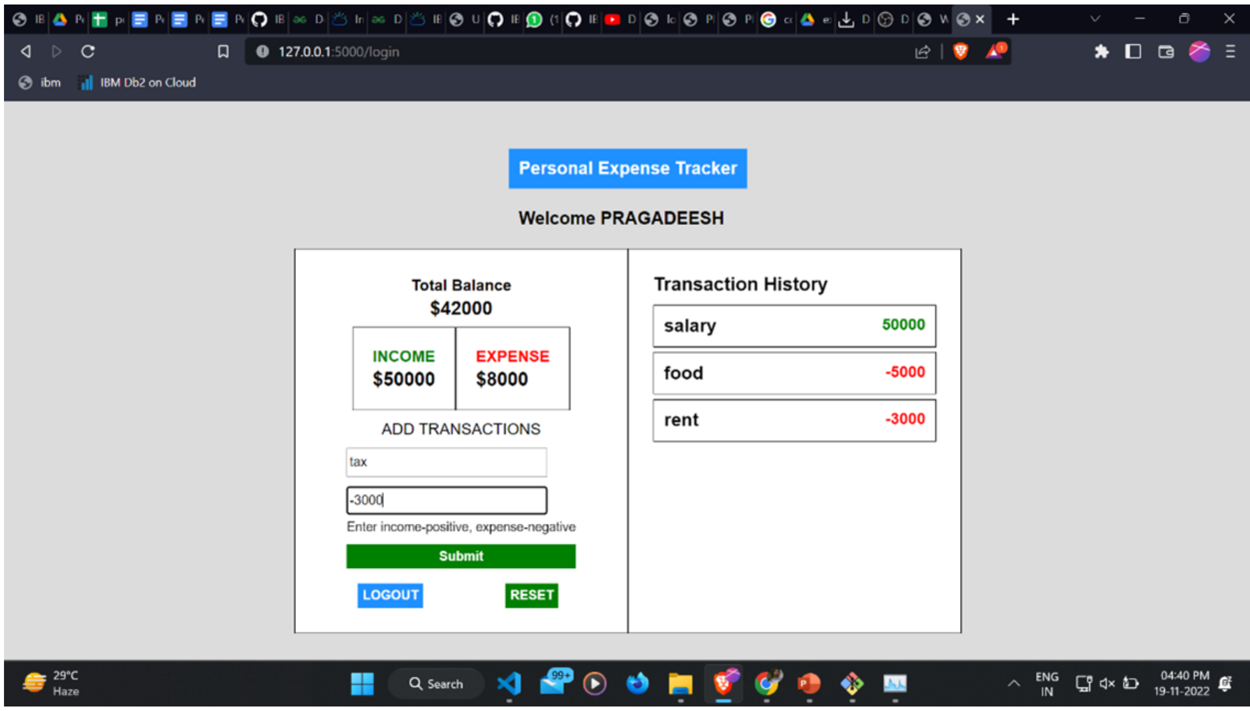
Enter income-positive, expense-negative

Submit

LOGOUT RESET

Transaction History

salary	50000
food	-5000
rent	-3000


127.0.0.1:5000/login

ibm IBM Db2 on Cloud

Personal Expense Tracker

Welcome PRAGADEESH

Total Balance
$42000

INCOME	EXPENSE
$50000	$8000

ADD TRANSACTIONS

tax

-3000

Enter income-positive, expense-negative

Submit

LOGOUT RESET

Transaction History

salary	50000
food	-5000
rent	-3000

8. ADVANTAGES & DISADVANTAGES

10.1 Advantages:-

I have a cheaper cell phone plan, using a smaller provider than the big monopoly providers.I don’t watch tv, so no cable, though my husband has a Netflix account and I’ll watch comedy specials or sci-fi series on occasionI don’t subscribe to any music streaming or video games.Come to think of it, I’ve never been that much into entertainment – I don’t buy tickets to concerts, sports games, or movies (I am in the minority for sure – a lot of people around me love watching movies but I forget movies so fast so I hardly go to the theater). What do I do for fun? I take walks with my family, work in the garden, read, surf online, nap, and I have a lot of occupations to keep me busy.I don’t buy books, I borrow them at the library. I always have a lot of books on hold or on renewal for my family. I listen to a ton of audiobooks when I drive to and from work everyday, they are such a source of delight for me.I don’t have healthcare premiums – I live in Canada.I don’t eat out too much – we do batch cooking, I always bring my own lunch, snacks, and coffee in a thermos to work. I am adamant about perfectly planned meals that incorporate fiber, protein, fats, carbs, and high water content fruit. Plus I am very picky about what I buy at the grocery store. We do not eat processed junk food or soda, I mostly buy high-quality meat, fruit, and dairy. Rice is very economical. I also grind my own coffee beans and bring my coffee to work, I NEVER buy Starbucks and I never eat at the cafeteria. Why should I pay more for inferior food and drink prepared by people who don’t have health as a priority I have a SodaStream, which is one of life’s joys. I love fizzy sparkling water, and now I never have to buy club soda again. I can use tap water and my SodaStream carbonates it for me! I even drink more water now because of it, and I bring it in a water bottle when I’m out and about.I don’t shop for clothes – at age 41, I have enough clothes already and still fit and wear the same size clothes as when I was a teenager. Since I always use a drying rack instead of the dryer, my clothes never wear out either. Over the years I’ve donated the ones that I don’t wear, and kept the ones that I do. Plus, the hospital provides sterile scrubs for work which is free! I am happy with my wardrobe and usually wear cheap Uniqlo leggings with a dress that is 20 years old.

10.2 Disadvantages:-

Your information is less secure, and probably being used and sold. If the service is free, then the product is you. Mint.com, like other financial apps, is a free service. They have to pay their bills somehow, so regardless of what their privacy policy may or may not say, just assume that your spending history and trends are going to be recorded and analyzed, by someone, somewhere. Now, you shouldn’t have to worry about credit card fraud or identitytheft, these companies are large enough and secure enough that you’ll never have to worry about something like that. Just recognize that your information, most likely anonymous, will be used and potentially even sold. Personally, I have no problem with that, but if you do, then make sure you avoid these types of services.

Automating everything to do with your finances can make you financially lazy. If your bills are paid automatically and your finances are track automatically, then what is there left for you to do? Not a lot, to be honest. So you might stop caring aboutwhatyou’respending and where your money is going. Eventually you may look at your Mint data and realize that you’ve blown your budget over the last two months, but by then it is too late. So if you do choose to use this program, ensure that you are also being diligent in checking in on your finances. Set up a weekly or biweekly check for yourself to go through your finances and hit on all the important points.

9. CONCLUSION

In this paper, we proposed a framework for job recommendation task. This framework facilitates the understand-ing of job recommendation process as well as it allows the use of a variety of text processing and recommendation methods according to the preferences of the job recommender system designer. Moreover, we also contribute mak-ing publicly available a new dataset containing job seekers pro les and job vacancies.Future directions of our work will focus on performing a more exhaustive evaluation considering a greater amount of methods and data as well as a comprehensive evaluation of the impact of each professional skill of a job seeker on the received job recommendation.

10. FUTURE SCOPE

Changing face of expense management software Year-on-year, modern expense management software underwent a continuous evolution from traditional back-office function to strategic internal set of processes. But would it be sufficient to meet the needs of next-gen companies? Have you ever thought about what the next-generation software should look like? As the requirements of companies evolve continuously, the software should undergo a series of changes to meet the growing needs of next generation companies.The next-generation travel and expense (T & E) management apps should not only just accelerate the expense management process but also should come with mobile and cloud integration capabilities that add tremendous value to the business bottom line. Future T & E management software should be able to provide greater visibility into spending and standardize critical procedures.Next-gen expense management A recent T & E study unveiled that visibility and intelligence are the two key aspects that companies look forward to understanding spending associated with business travel. Analytics is also on the priority list of the best-in-class organizations. The motto is not just to enhance the existing process but also to leverage analytical capabilities and visibility that can help companies drive efficiency, forecast and plan better for corporate finances.Apparently, as per research, integration, analytics and mobile apps are the three key factors that can help companies succeed at a faster pace. When incorporated, these factors add edge and value to the businesses.Integration Integration between corporate cards and expense management software increases transparency and makes the process effortless throughout the expense report cycle.AnalyticsIncreased intelligence provides you with an unheralded level of visibility into travel spending and enhances overall T & E intelligence. Companies can measure the true performance of any business trip by evaluating the ROI. Efficiency complimented by intelligence proves to be a great way to take the business to new heights.Need for mobile applicationsMobile apps provide employees with the opportunity to manage expenses on the go. It gives both the companies and employees the flexibility that they need in managing the expense related activities. In fact, it increases accuracy as the power of technology is put into the hands of both employees and employers.On a final note, the next generation software should be something that gives users an unprecedented experience in expense management and allows organizations to emphasize on what’s really important to make their businesses better.

11. APPENDIX

13.1. Source Code

Index file

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>PETA - Register</title>



    <link rel="stylesheet" href="static/css/style.css">

</head>

<body>

    <div class="wrapper">
```

```
<div class="register-form">

    <div class="header">

        <header class="topic">Personal Expense Tracker</header>

        <br>

        <header>REGISTER</header>

    </div>

    <form action="{{ url_for('index') }}" method="post">

        <div class="inputField">

            <label for="username">username</label>

            <input type="text" name="username" required>

        </div>

        <div class="inputField">

            <label for="email">email</label>

            <input type="text" name="email" required>

        </div>

        <div class="inputField">

            <label for="password">password</label>

            <input type="text" name="password" required>

        </div>
```

```
                <div class="inputField btn">
                    <input type="submit" name="signup" value="SIGNUP">
                </div>
                <div class="login">
                    <p>Already have an account ? <a href="{{url_for('login')}}">Login</a></p>
                </div>
            </form>
        </div>
    </div>
</body>
</html>
```

Login file

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>PETA - Login</title>



    <link rel="stylesheet" href="/static/css/style.css">

</head>

<body>

    <div class="wrapper">

        <div class="login-form ">

            <div class="header">

                <header class="topic">Personal Expense Tracker</header>

                <br>

                <header>LOGIN</header>

            </div>

            <form action="{{ url_for('loginIn')}}" method="post">

                <div class="inputField">

                    <label for="email">email</label>

                    <input type="text" name="email" required>

                </div>
                <div class="inputField">
                    <label for="password">password</label>
                    <input type="text" name="password" required>
                </div>
                <div class="inputField btn">
                    <input type="submit" name="signup" value="LOGIN">
                </div>
                <div class="signup">
                    <p>New user ? <a href="{{url_for('index')}}">signup here</a></p>
                </div>
            </form>
        </div>
    </div>
</body>
</html>
```

Profile.html

```
<!DOCTYPE html>
```

```
<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>PETA</title>

    <link rel="stylesheet" href="static/css/style.css">

</head>



<body>

    <div class="main-container">

        <div class="name">

            <p class="topic">Personal Expense Tracker</p>

            <br>

            <p class="u-name">Welcome {{name}}</p>

        </div>

    <div class="tracker-container">



        <div class="make-transaction">
```

```
<div class="total-balance">

    <p class="total-balance-title">Total Balance</p>

    <p class="total-balance-amount">${{totalBalance}}</p>

</div>

<div class="income-expense">

    <div class="income">

        <p class="income-title income-expense-title">INCOME</p>

        <p class="income-amount income-expense-amount">${{incomeBalance}}</p>

    </div>

    <div class="expense">

        <p class="expense-title income-expense-title">EXPENSE</p>

        <p class="expense-amount income-expense-amount">${{expenseBalance}}</p>

    </div>

</div>

<div class="add-transactions">

    <form action="{{url_for('add')}}" method="post">

        <div class="add-transactions">

            <header>ADD TRANSACTIONS</header>

        </div>
```

```
            <div class="inputField">

                <input type="text" placeholder="Enter title" name="title" required>

            </div>

            <div class="inputField">

                <input type="text" placeholder="Enter amount" name="amount" required>

            <p class="pon">Enter income-positive, expense-negative</p>

            </div>

            <div class="inputField btn submit">

                <input type="submit">

            </div>

        </form>

    </div>

    <div class="btns">

        <div class="logout">

        <a href="{{url_for('logout')}}"><button>LOGOUT</button></a>

        </div>

        <div class="reset">

            <a href="{{url_for('reset')}}"><button>RESET</button></a>

        </div>
    </div>

</div>

<div class="transaction-history">

    <div class="title">

        <p>Transaction History</p>

    </div>

    <div class="transaction-history-container">

        {%for i in l: %}

         {%if i[2] > 0:%}

         <div class="entery-history single-transaction-history">

            <h3>{{i[1]}}</h3>

            <h4 class="entry-color">{{i[2]}}</h4>

         </div>

         {%else:%}

          <div class="exit-history single-transaction-history">

             <h3>{{i[1]}}</h3>

              <h4 class="exit-color">{{i[2]}}</h4>

            </div>

             {% endif %}
```

```
                        {%endfor%}




                    <!-- <div class="entery-history single-transaction-history">

                        <h3>Food</h3>

                        <h4 class="entry-color">$100</h4>

                    </div>

                    <div class="exit-history single-transaction-history">

                        <h3>Food</h3>

                    <h4 class="exit-color">-$100</h4>

                    </div> -->



                </div>

            </div>

        </div>



        </div>

    </body>

    </html>
```

Style.css

```
*{
    margin: 0;
    font-family: sans-serif;
}
body{
    background-color: gainsboro;
}
.wrapper{
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
}
.header{
    text-align: center;

    font-weight: bolder;

}

.register-form, .login-form{

    background-color: white;

    box-shadow: 2px 2px 10px gray;

    padding: 20px;

    display: flex;

    align-items: center;

    flex-direction:column;

    justify-content: center;

    min-height: 297px;

    min-width: 297px;

}

.inputField{

    display: flex;

    flex-direction: column;

    margin: 10px;

    font-size: small;
```

```
}

.inputField input{

    height: 25px;

    width: 200px;

    border: 1px solid gray;

}

.inputField label{

    margin-bottom: 3px;

}

.btn input{

    font-weight: bold;

    width: 100%;

    background-color: green;

    color: #ffffff;

    border: none;

}

.login, .signup{

    font-size: small;

    text-align: center;
```

```
}


/* PROFILE CSS */

.topic{

    background-color: dodgerblue;

    padding: 10px;

    color: #ffffff;



}

.u-name{

    padding: 0 10px;

}

.name{

    padding: 20px;

    /* margin-left: auto;

    margin-right: auto; */

    margin: auto;

    font-weight: bold;

    font-size: large;
    font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;

    margin-bottom: 0;

    margin-top: 0;

}



.main-container{

    display: flex;

    justify-content: center;

    align-items: center;

    min-height: 100vh;

    flex-direction: column;

}

.tracker-container{

    display: flex;

    background-color: #ffffff;

    max-height: 70vh;

}

.make-transaction, .transaction-history{

    width: 300px;
    border: 1px solid black;

    padding: 20px;

}

.total-balance{

    text-align: center;

    font-weight: bold;

    margin: 8px;

}

.total-balance-title, .income-expense-title{

    font-size: medium;

    margin-bottom: 3px;

}

.total-balance-amount,.income-expense-amount{

    font-size: larger;

}

.income-expense{

    display: flex;

    justify-content: center;

}
```

```
.income, .expense{

    font-weight: bold;

    border: 1px solid black;

    padding: 20px;

}

.income-title{

    color: green;

}

.expense-title{

    color: red;

}

.add-transactions{

    margin: 5px;

    display: flex;

    justify-content: center;

}

.transaction-history-container{

    min-height: 50vh;

    max-height: 50vh;
    overflow-y: auto;
}
.single-transaction-history{
    display: flex;
    justify-content: space-between;
    border: 1px solid black;
    padding: 10px;
    margin: 5px;
}
.submit input{
    background-color: green;
}
.title{
    font-weight: bold;
    font-size: larger;
    margin: 5px;
}
.entry-color{
    color: green;
```

```
}

.exit-color{

    color: red;

}



.btns{

    display: flex;

    justify-content: center;

    text-align: center;

    margin: 5px;

}

.logout, .reset{

    width: 100%;

}

.logout a button{

    background-color: dodgerblue;

    border: none;

    color: #ffffff;

    padding: 5px;
    font-weight: bold;

}

.reset a button{

    background-color: green;

    border: none;

    color: #ffffff;

    padding: 5px;

    font-weight: bold;

}

.pon{

    font-size: small;

    margin-top: 5px;

}


@media screen and ( max-width: 600px){


    *{

        background-color: white;

    }
```

```
.wrapper{

    min-height: 80vh;

}

.register-form, .login-form{

    box-shadow: none;

    width: 100%;

}

.inputField input{

    width: 200px;

    border: 1px solid gray;

}

.btn input{

    width: 100%;

    border: 1px solid gray;

}



.tracker-container{

    display: flex;

    background-color: #ffffff;
```

```
        max-height: 70vh;

        flex-direction: column;

    }



}
```

App.py python file

```
from flask import Flask, render_template, request, redirect, url_for, session

import mysql.connector

# import re

import ibm_db



from sendgrid import SendGridAPIClient

from sendgrid.helpers.mail import Mail




conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=21fecfd8-47b7-4937-840d-
d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31864;SECURITY=S
SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=vpl67398;PWD=c8CTODyXcPc9RJ
Tp'',''.'')

app = Flask(__name__)

app.secret_key = 'your secret key'



# cnx = mysql.connector.connect(user='root', password='', host='127.0.0.1',  database='ibm_db')

# cursor = cnx.cursor(buffered=True)

@app.route('/')



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

def index():

    msg = ''

    if request.method == 'POST' and 'email' in request.form and 'password' in request.form:

        email = request.form['email']

        password = request.form['password']

        username = request.form['username']

        # sql = 'SELECT * FROM accounts WHERE email = %s',(email,)

        # stmt = ibm_db.exec_immediate(conn,sql)

        # dic = ibm_db.fetch_both(stmt)
```

```
sql2 = "SELECT * FROM USERS WHERE EMAIL=?"

stmt2 = ibm_db.prepare(conn, sql2)

ibm_db.bind_param(stmt2, 1, email)

ibm_db.execute(stmt2)

account = ibm_db.fetch_both(stmt2)

# cursor.execute('SELECT * FROM accounts WHERE email = %s',(email,))

# account = cursor.fetchone()

if account:

    msg = 'Account already exists !'

    return "<p>Account already exists !</p>"

# elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

#    return "<p>Invalid Email address !</p>"

else:

    totalBalance = 0

    incomeBalance = 0

    expenseBalance = 0

    # t = int(time.time())

    # ran = random.randrange(t,10000000000)

    # sql = 'INSERT INTO USERS (USERNAME, EMAIL, PASSWORD,
TOTALBALANCE, INCOMEBALANCE, EXPENSEBALANCE) VALUES ( %s, %s, %s, %s,
%s, %s)',(username, email, password, totalBalance, incomeBalance, expenseBalance,)

    # cursor.execute('INSERT INTO accounts (username, email, password, totalBalance,
incomeBalance, expenseBalance) VALUES ( %s, %s, %s, %s, %s, %s)',(username, email,
password, totalBalance, incomeBalance, expenseBalance,))

    # stmt = ibm_db.exec_immediate(conn,sql)

    # cnx.commit()



    #INSERT THE USER



    sql2 = "INSERT INTO USERS (USERNAME, EMAIL, PASSWORD,
TOTALBALANCE, INCOMEBALANCE, EXPENSEBALANCE) VALUES ( ?,?,?,?,?,?)"

    stmt2 = ibm_db.prepare(conn, sql2)

    ibm_db.bind_param(stmt2, 1, username)

    ibm_db.bind_param(stmt2, 2, email)

    ibm_db.bind_param(stmt2, 3, password)

    ibm_db.bind_param(stmt2, 4, totalBalance)

    ibm_db.bind_param(stmt2, 5, incomeBalance)

    ibm_db.bind_param(stmt2, 6, expenseBalance)

    result = ibm_db.execute(stmt2)

    # cursor.execute('SELECT * FROM accounts WHERE email = %s',(email,))
```

```
        # sql = 'SELECT * FROM accounts WHERE email = %s',(email,)

        # account = cursor.fetchone()



        # GET THE USER



        sql2 = "SELECT * FROM USERS WHERE EMAIL=?"

        stmt2 = ibm_db.prepare(conn, sql2)

        ibm_db.bind_param(stmt2, 1, email)

        ibm_db.execute(stmt2)

        account = ibm_db.fetch_both(stmt2)

        name = account['USERNAME'].upper()

        # stmt = ibm_db.exec_immediate(conn,sql)

        # dic = ibm_db.fetch_both(stmt)

        # session['loggedin'] = True

        # session['userID'] = account['userID']

        session['email'] = request.form.get("email")

        return render_template('profile.html',totalBalance=totalBalance,
incomeBalance=incomeBalance,expenseBalance=expenseBalance,len=0, name=name)

    else:
        return render_template('index.html')

@app.route('/login')

def login():

    return render_template('login.html')

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

def loginIn():

    msg = ''

    if request.method == 'POST' and 'email' in request.form and 'password' in request.form:

        email = request.form['email']

        password = request.form['password']

        # sql = 'SELECT * FROM USERS WHERE EMAIL = %s AND PASSWORD =
%s',(email,password,)

        # cursor.execute('SELECT * FROM accounts WHERE email = %s AND password =
%s',(email,password,))

        # account = cursor.fetchone()



        #FETCH THE USER



        sql2 = "SELECT * FROM USERS WHERE EMAIL=? AND PASSWORD=?"

        stmt2 = ibm_db.prepare(conn, sql2)
```

```
ibm_db.bind_param(stmt2, 1, email)

ibm_db.bind_param(stmt2, 2, password)

ibm_db.execute(stmt2)

account = ibm_db.fetch_both(stmt2)



# stmt = ibm_db.exec_immediate(conn,sql)

# dic = ibm_db.fetch_both(stmt)

if account:

    # session['loggedin'] = True

    # session['userID'] = account['userID']

    session['email'] = request.form.get("email")

    email = session['email']

    # cursor.execute('SELECT * FROM accounts WHERE email = %s',(email,))



    #GET THE USER



    sql2 = "SELECT * FROM USERS WHERE EMAIL=?"

    stmt2 = ibm_db.prepare(conn, sql2)

    ibm_db.bind_param(stmt2, 1, email)
    ibm_db.execute(stmt2)

    account = ibm_db.fetch_both(stmt2)



    # sql = 'SELECT * FROM accounts WHERE email = %s',(email,)

    # account = cursor.fetchone()

    # stmt = ibm_db.exec_immediate(conn,sql)

    # dic = ibm_db.fetch_both(stmt)

    name = account['USERNAME'].upper()

    incomeBalance = account['INCOMEBALANCE']

    totalBalance = account['TOTALBALANCE']

    expenseBalance = account['EXPENSEBALANCE']



    sql = "SELECT * FROM HISTORY WHERE EMAIL = " + "\'" + email + "\'"

    stmt = ibm_db.exec_immediate(conn, sql)

    dictionary = ibm_db.fetch_assoc(stmt)

    l=[]

    print(dictionary)

    while dictionary != False:

        print(dictionary)
```

```
                l.append(list(dictionary.values()))

                dictionary = ibm_db.fetch_assoc(stmt)

            print(l[0][0])

            return render_template('profile.html',
totalBalance=totalBalance,incomeBalance=incomeBalance,
expenseBalance=abs(expenseBalance), len=len(l), l=l, name = name)

        else:

            return "<h1>Invalid</h1>"



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

def add():

    if request.method == 'POST' and 'title' in request.form and 'amount' in request.form:

        title = request.form['title']

        amount = request.form['amount']

        email =  session["email"]

        if int(amount) > 0:



            #FETCH THE USER



            sql2 = "SELECT * FROM USERS WHERE EMAIL=?"
            stmt2 = ibm_db.prepare(conn, sql2)

            ibm_db.bind_param(stmt2, 1, email)

            ibm_db.execute(stmt2)

            account = ibm_db.fetch_both(stmt2)

            # cursor.execute('SELECT * FROM accounts WHERE email = %s',(email,))

            # sql = 'SELECT * FROM accounts WHERE email = %s',(email,)

            # account = cursor.fetchone()

            # stmt = ibm_db.exec_immediate(conn,sql)

            # dic = ibm_db.fetch_both(stmt)

            incomeBalance = account['INCOMEBALANCE']

            totalBalance = account['TOTALBALANCE']

            expenseBalance = account['EXPENSEBALANCE']

            incomeBalance += int(amount)

            totalBalance += int(amount)

            # sql = 'UPDATE USERS SET TOTALBALANCE = %s, INCOMEBALANCE = %s
WHERE EMAIL = %s',(totalBalance,incomeBalance,email,)

            # cursor.execute('UPDATE accounts SET totalBalance = %s, incomeBalance = %s
WHERE email = %s',(totalBalance,incomeBalance,email,))



            #UPDATE THE TRANSACTION
            sql2 = "UPDATE USERS SET TOTALBALANCE = ?, INCOMEBALANCE = ?
WHERE EMAIL = ?"

            stmt2 = ibm_db.prepare(conn, sql2)

            ibm_db.bind_param(stmt2, 1, totalBalance)

            ibm_db.bind_param(stmt2, 2, incomeBalance)

            ibm_db.bind_param(stmt2, 3, email)

            ibm_db.execute(stmt2)




            # stmt = ibm_db.exec_immediate(conn,sql)

            # dic = ibm_db.fetch_both(stmt)



            #FETCH THE USER DETAILS



            sql2 = "SELECT * FROM USERS WHERE EMAIL=?"

            stmt2 = ibm_db.prepare(conn, sql2)

            ibm_db.bind_param(stmt2, 1, email)

            ibm_db.execute(stmt2)
```

```
account = ibm_db.fetch_both(stmt2)


# cursor.execute('SELECT * FROM accounts WHERE email = %s',(email,))

# account = cursor.fetchone()

name = account['USERNAME'].upper()

incomeBalance = account['INCOMEBALANCE']

totalBalance = account['TOTALBALANCE']


sql2 = "INSERT INTO HISTORY (EMAIL, TITLE, AMOUNT) VALUES ( ?,?,?)"

stmt2 = ibm_db.prepare(conn, sql2)

ibm_db.bind_param(stmt2, 1, email)

ibm_db.bind_param(stmt2, 2, title)

ibm_db.bind_param(stmt2, 3, amount)

result = ibm_db.execute(stmt2)




sql = "SELECT * FROM HISTORY WHERE EMAIL = " + "\'" + email + "\'"

stmt = ibm_db.exec_immediate(conn, sql)
```

```
            dictionary = ibm_db.fetch_assoc(stmt)

            l=[]

            print(dictionary)

            while dictionary != False:

                print(dictionary)

                l.append(list(dictionary.values()))

                dictionary = ibm_db.fetch_assoc(stmt)

            print(l[0][0])






            # cursor.execute('INSERT INTO history (email, title, amount ) VALUES ( %s, %s,
%s)',(email, title, amount,))

            # cursor.execute('SELECT * FROM history WHERE email = %s',(email,))

            # account = cursor.fetchall()


            # for i in account:

            #     print(i)

            return render_template('profile.html',
totalBalance=abs(totalBalance),incomeBalance=incomeBalance,expenseBalance=abs(expenseBa
lance), len = len(l), name= name, l=l)

    elif int(amount) < 0:

            # cursor.execute('SELECT * FROM accounts WHERE email = %s',(email,))

            # sql ='SELECT * FROM USERS WHERE EMAIL = %s',(email,)

            # account = cursor.fetchone()



            #FETCH THE DETAILS



            sql2 = "SELECT * FROM USERS WHERE EMAIL=?"

            stmt2 = ibm_db.prepare(conn, sql2)

            ibm_db.bind_param(stmt2, 1, email)

            ibm_db.execute(stmt2)

            account = ibm_db.fetch_both(stmt2)



            # stmt = ibm_db.exec_immediate(conn,sql)

            # dic = ibm_db.fetch_both(stmt)

            incomeBalance =account['INCOMEBALANCE']

            totalBalance = account['TOTALBALANCE']
            print(totalBalance)

            expenseBalance = account['EXPENSEBALANCE']

            expenseBalance += int(amount)

            totalBalance -= abs(int(amount))

            print(totalBalance)


            #SEND MAIL

            totalPercent = totalBalance//100

            limit = totalPercent * 70

            print(totalBalance)

            print(limit)



            if abs(expenseBalance) >= limit:

                API =
'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'

                from_email = 'pragadeesh4701@gmail.com'

                to_emails = email

                subject = "Reached Your Limit"

                html_content = "You have expensed more than 70% of your total balance. Please
```

```
reduce your expenses...Thank you..."



        message = Mail(from_email,to_emails,subject,html_content)

        try:

            sg = SendGridAPIClient(API)

            response = sg.send(message)

            print(response.status_code)

            print(response.body)

            print(response.headers)

        except Exception as e:

            print(e.message)








    #UPDATE THE DETAILS



    sql2 = "UPDATE USERS SET TOTALBALANCE = ?, EXPENSEBALANCE = ?
WHERE EMAIL = ?"
```

```
        stmt2 = ibm_db.prepare(conn, sql2)

        ibm_db.bind_param(stmt2, 1, totalBalance)

        ibm_db.bind_param(stmt2, 2, expenseBalance)

        ibm_db.bind_param(stmt2, 3, email)

        ibm_db.execute(stmt2)




        # cursor.execute('UPDATE accounts SET totalBalance = %s, expenseBalance = %s
WHERE email = %s',(totalBalance,expenseBalance,email,))

        # sql = 'UPDATE USERS SET TOTALBALANCE = %s, EXPENSEBALANCE = %s
WHERE EMAIL = %s',(totalBalance,expenseBalance,email,)



        #FETCH THE DETAILS



        sql2 = "SELECT * FROM USERS WHERE EMAIL=?"

        stmt2 = ibm_db.prepare(conn, sql2)

        ibm_db.bind_param(stmt2, 1, email)

        ibm_db.execute(stmt2)

        account = ibm_db.fetch_both(stmt2)
        # cursor.execute('SELECT * FROM accounts WHERE email = %s',(email,))

        # stmt = ibm_db.exec_immediate(conn,sql)

        # dic = ibm_db.fetch_both(stmt)

        # account = cursor.fetchone()

        name= account['USERNAME'].upper()

        expenseBalance = account['EXPENSEBALANCE']

        totalBalance = account['TOTALBALANCE']

        print(totalBalance)

        # cursor.execute('INSERT INTO history (email, title, amount ) VALUES ( %s, %s,
%s)',(email, title, amount,))

        # cursor.execute('SELECT * FROM history WHERE email = %s',(email,))

        # account = cursor.fetchall()

        # for i in account:

        #     print(i)




        sql2 = "INSERT INTO HISTORY (EMAIL, TITLE, AMOUNT) VALUES ( ?,?,?)"

        stmt2 = ibm_db.prepare(conn, sql2)

        ibm_db.bind_param(stmt2, 1, email)
        ibm_db.bind_param(stmt2, 2, title)

        ibm_db.bind_param(stmt2, 3, amount)

        result = ibm_db.execute(stmt2)




        sql = "SELECT * FROM HISTORY WHERE EMAIL = " + "\'" + email + "\'"

        stmt = ibm_db.exec_immediate(conn, sql)

        dictionary = ibm_db.fetch_assoc(stmt)

        l=[]

        print(dictionary)

        while dictionary != False:

            print(dictionary)

            l.append(list(dictionary.values()))

            dictionary = ibm_db.fetch_assoc(stmt)

        print(l[0][0])

        return render_template('profile.html',
totalBalance=abs(totalBalance),incomeBalance=incomeBalance,expenseBalance=abs(expenseBa
lance),len = len(l), l=l, name=name)
```

```
@app.route('/reset')

def reset():

    email =  session["email"]

    totalBalance = 0

    incomeBalance = 0

    expenseBalance = 0

    # cursor.execute('SELECT * FROM accounts WHERE email = %s',(email,))

    #         # stmt = ibm_db.exec_immediate(conn,sql)

            # dic = ibm_db.fetch_both(stmt)

    # account = cursor.fetchone()



    sql2 = "SELECT * FROM USERS WHERE EMAIL=?"

    stmt2 = ibm_db.prepare(conn, sql2)

    ibm_db.bind_param(stmt2, 1, email)

    ibm_db.execute(stmt2)

    account = ibm_db.fetch_both(stmt2)



    name=account['USERNAME']

    # cursor.execute('UPDATE accounts SET totalBalance = %s, expenseBalance = %s,
incomeBalance = %s WHERE email = %s',(0,0,0,email,))



    sql2 = "UPDATE USERS SET TOTALBALANCE = ?, INCOMEBALANCE = ? ,
EXPENSEBALANCE = ? WHERE EMAIL = ?"

    stmt2 = ibm_db.prepare(conn, sql2)

    ibm_db.bind_param(stmt2, 1, totalBalance)

    ibm_db.bind_param(stmt2, 2, incomeBalance)

    ibm_db.bind_param(stmt2, 3, expenseBalance)

    ibm_db.bind_param(stmt2, 4, email)

    ibm_db.execute(stmt2)

    # cursor.execute('DELETE FROM history WHERE email = %s',(email,))



    #HISTORY

    sql = "DELETE FROM HISTORY WHERE EMAIL = " + "\'" + email + "\'"

    stmt = ibm_db.exec_immediate(conn, sql)



    return render_template('profile.html', totalBalance=0,incomeBalance=0,expenseBalance=0,len
= 0,name=name)
```

```
@app.route('/logout')

def logout():

    # session.pop('loggedin', None)

    # session.pop('email', None)

    session["email"] = None

    # session.pop('userID', None)

    return redirect(url_for('login'))
```

sendemail.py

```
import configparser

import ssl

ssl._create_default_https_context = ssl._create_unverified_context

from sendgrid import SendGridAPIClient

from sendgrid.helpers.mail import Mail



API =
'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'
```

```
from_email = 'pragadeesh4701@gmail.com'

to_emails = 'harshaparthiban56789@gmail.com'

subject = "Hello"

html_content = "Happy to Learn.."



# def sendMailUsingSendGrid(API,from_email,to_emails,subject,html_content):

#     if API!=None and from_email!=None and len(to_emails)>0:

message = Mail(from_email,to_emails,subject,html_content)

try:

    sg = SendGridAPIClient(API)

    response = sg.send(message)

    print(response.status_code)

    print(response.body)

    print(response.headers)

except Exception as e:

    print(e.message)
```

# sendMailUsingSendGrid(API,from_email,to_emails,subject,html_content)

13.2. GitHub & Project Demo Link

GITHUB: https://github.com/IBM-EPBL/IBM-Project-39641-1660474299.git

PROJECT DEMO LINK: https://drive.google.com/file/d/14KbasfWXhz3m-LHEXtXCuI-fvCfC5Wyk/view?usp=share_link