1. INTRODUCTION

1.1. Project Overview

The most demanded skills for data scientists Python, R, SQL, and the list goes on and on. There are many surveys and reports that show some good statistics on popular data skills. In this post, I am going to gather first-hand information by scraping data science jobs from indeed.ca, analyze top skills required by employers, and make job recommendations by matching skills from resume to posted jobs. It will be fun!Python code can be found on my GitHub.This post is part of a series of people analytics experiments I am putting together:Job skill match (Recruitment) Employee attrition prediction (Employee Management) Pay gap by gender, ethnicity, profession (Employee Compensation) FUTURE WORK Organizational network analysis (ONA) FUTURE WORK Web Scraping I like scraping data from Internet because it is free! But you should not abuse it just because it is free. There are basic rules to follow so we do not get our cyber friends and reject ourselves from visiting their websites. All can be summed in two words: BE NICE. Read their terms and conditions on the website first, and space out your requests so that their website does not get hit too hard. Here is a very good web scraping 101.I scraped data science jobs from indeed.ca, the largest global recruiting website. I gathered data scientist/engineer/analyst jobs posted in the last 30 days (09/19/2018 – 10/19/2018), in 6 major Canadian cities, i.e. Toronto, Montreal, Vancouver, Ottawa, Calgary, and Edmonton. I used Selenium Webdriver to automate web scraping and saved results in a local JSON file.In total 367 job postings were retrieved, and it took about 20 minutes.Job Description Keyword Extraction For each job, I tokenized its job description, cleaned up the list by removing words defined in the NLTK list of stopwords, and finally filtered on a list of popular data science related skill words.Bar chart below shows top 30 data skills required by most employers . More than 60% jobs requires SQL, 48% requires Python, and 32% requires R. No surprise. It is worth noting that Excel is actually quite a common requirement, which I would not normally considered a serious data skill. 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

Money View App reads all of the transactional SMS messages and provides you with real-time visibility into your finances. This app unearths the hidden financial data that sits idly in SMS logs and makes excellent use of it.This personal finance manager app acts as a proactive budget planner, assisting you in staying on top of your budget, bills, and finances. The personal finance app was designed for simple, real-time budget and financial tracking, making it one of the best expense tracker apps in India.You can use the budget planner and spending tracker to keep track of your personal and business financial transactions, review financial data on a daily/weekly/monthly basis, and manage your assets.Monefy tracks the user's expenses and compares them to the monthly income and the budget planner. Monefy's money manager app keeps your monthly budget in top shape. As a result, it could also serve as the best expense tracker app.Wallet can automatically track your daily expenses by syncing your bank account, view weekly expense reports, plan your shopping expenses, and share specific features with your loved ones. You can manage your money with a wallet from anywhere and at any time.Walnut automates and secures the tracking of your monthly expenses. You can stay within your budget, pay your bills on time, and save more money each month by using the Walnut app. They also provide personal loans.

2.2. References

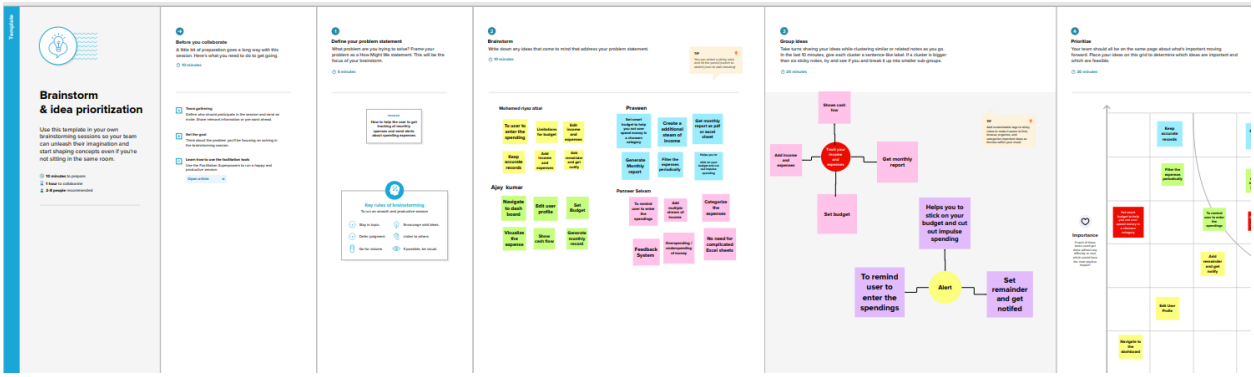
1. https://moneyview.in/insights/best-personal-fnance-management-a pps-in-india
2. https://www.factmr.com/report/personal-fnance-mobile-app-marke t
3. https://www.moneytap.com/blog/best-money-management-apps/
4. https://relevant.software/blog/personal-fnance-app-like-mint/
5. https://www.onmanorama.com/lifestyle/news/2022/01/11/fnancial-l 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

Modern education does not focus on finance management. This is primarily due to lack of resources and the Indian value system on giving money to children. Failing to teach this valuable knowledge had left many Indians to recklessly spend their income and fall into vicious cycles of EMI and debt. Many of them are just a month's salary away from bankruptcy.This issue is tackled by providing a web application for where people can plan their monthly expenses into categories, set alerts and get visual insights from their spending patterns. Who does the problem affect Young adults and earning middle class citizens? What is the issue? Lack of financial literacy among people When does the issue occur Primarily when the person moves from college to job and starts earning their own money.Where is the issue occurring Especially among young engineers who are newly exposed to consumer centric market and services

3. IDEATION & PROPOSED SOLUTION 3.1. Empathy Map Canvas

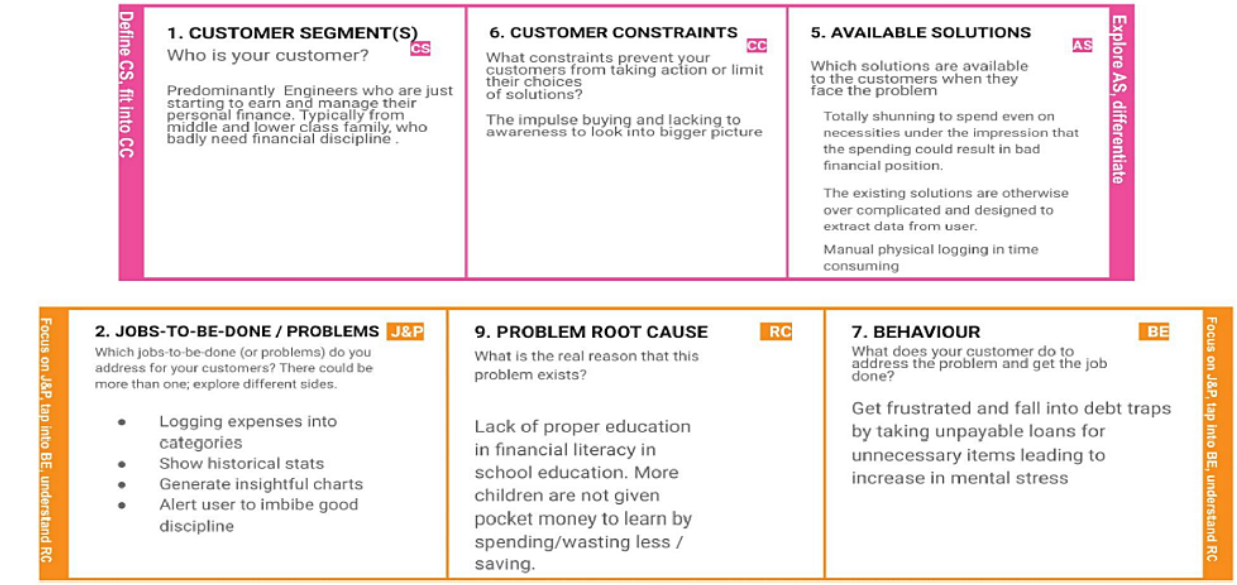
3.2. Ideation & Brainstorming



3.3. Proposed Solution

Problem Statement Building a personal finance tracking application that will imbibe good spending habits into students.(Problem to be solved)Idea / Solution description To build a web application that is deployed in IBM cloud and leverage mailing service like sendgrid to implement the same Novelty / Uniqueness The stats generated with visual graphs are more effective than log books. It also helps in using technology to gain better insights from Social Impact / Customer Satisfaction Better financial knowledge is gained. Gamified approach can be used to give self satisfaction.Reduced chances of bad debt in future Business Model (Revenue Model) Subscription can be incorporated to access premium tools within the app. Scalability of the Solution as deployment.As the application is containerized fr It can be easily scaled in a cloud service provider like IBM.

3.4. Problem Solution fit



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?

Predominantly Engineers who are just starting to earn and manage their personal finance. Typically from middle and lower class family, who badly need financial discipline .

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions?

The impulse buying and lacking to awareness to look into bigger picture

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem

Totally shunning to spend even on necessities under the impression that the spending could result in bad financial position.

The existing solutions are otherwise over complicated and designed to extract data from user.

Manual physical logging in time consuming

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? There could be more than one; explore different sides.

- Logging expenses into categories
- Show historical stats
- Generate insightful charts
- Alert user to imbibe good discipline

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?

Lack of proper education in financial literacy in school education. More children are not given pocket money to learn by spending/wasting less / saving.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?

Get frustrated and fall into debt traps by taking unpayable loans for unnecessary items leading to increase in mental stress

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

3. TRIGGERS TR

What triggers customers to act?

Frequent sales in e-commerce platforms and seamless shopping experience online.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?

Dejected and paranoid about the future as they would need relatively more money to provide for a family and to handle unexpected financial needs.

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.

If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behavior.

Graphical Application with simple UI and to the point clutter free objective. Avoids provision to pay through the app, to minimize the spending and ensure that only necessary spendings are made. The aim is to make the spending process harder throughout the

8.CHANNELS of BEHAVIOUR CH

8.1 ONLINE

What kind of actions do customers take online? Extract online channels from #7

1. Shop from e-commerce
2. Subscribe to OTT platforms
3. Order food frequently

8.2 OFFLINE

What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

1. Shop in malls during sales

Identify strong TR & EM

4. 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.

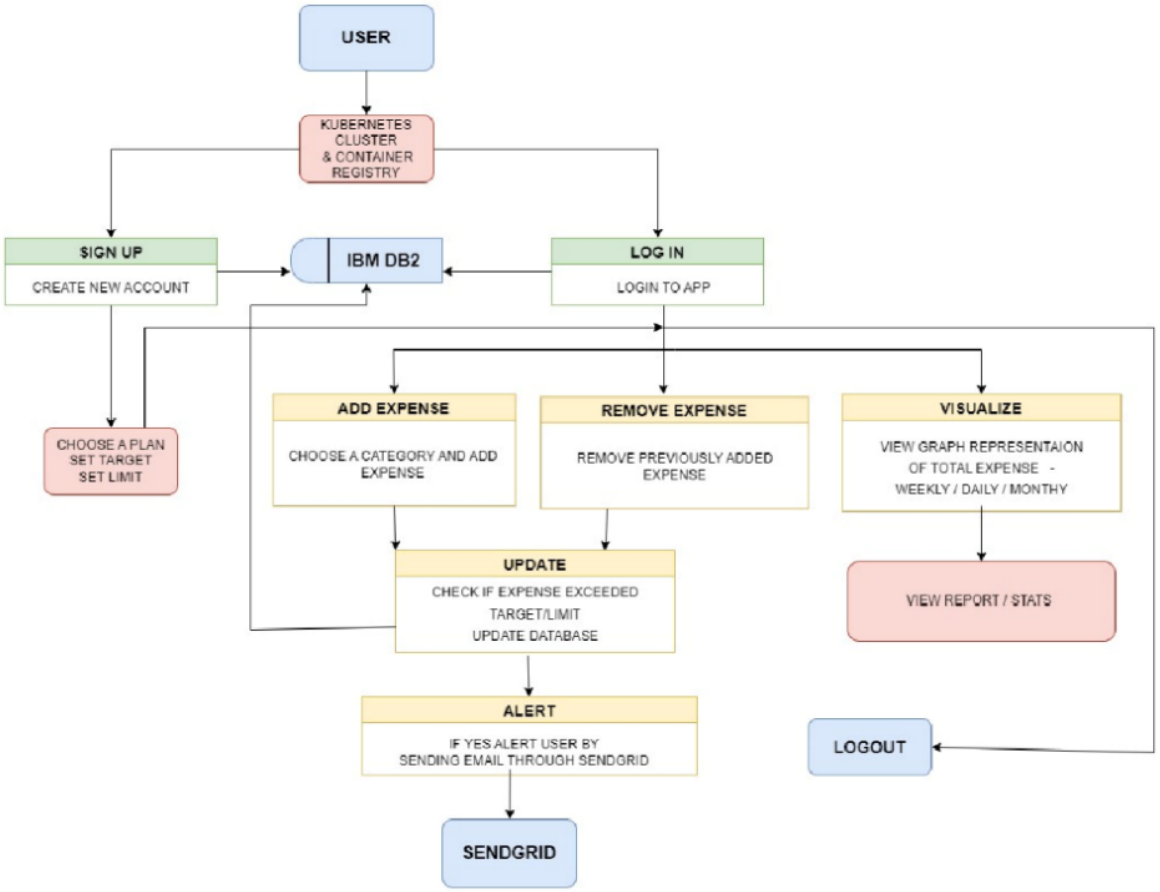
4.2. Non-Functional requirements

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

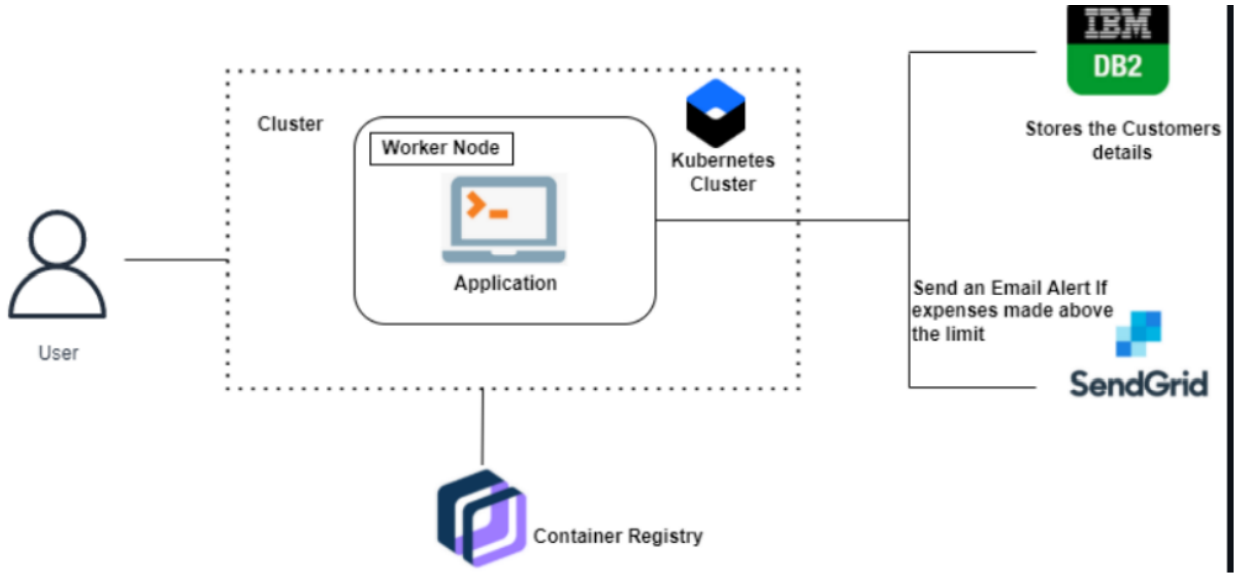
FR No	Non-Functional Requirement	Description
NFR-1	Usability	A simple web application which is accessible across devices
NFR-2	Security	The OAuth Google sign in and email login are secure with hashed and salted secure storage of credentials.
NFR-3	Reliability	Containerized service ensures that new instance 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

5. PROJECT DESIGN

5.1. Data Flo w Diagram



5.2. Solution & Technical Architecture

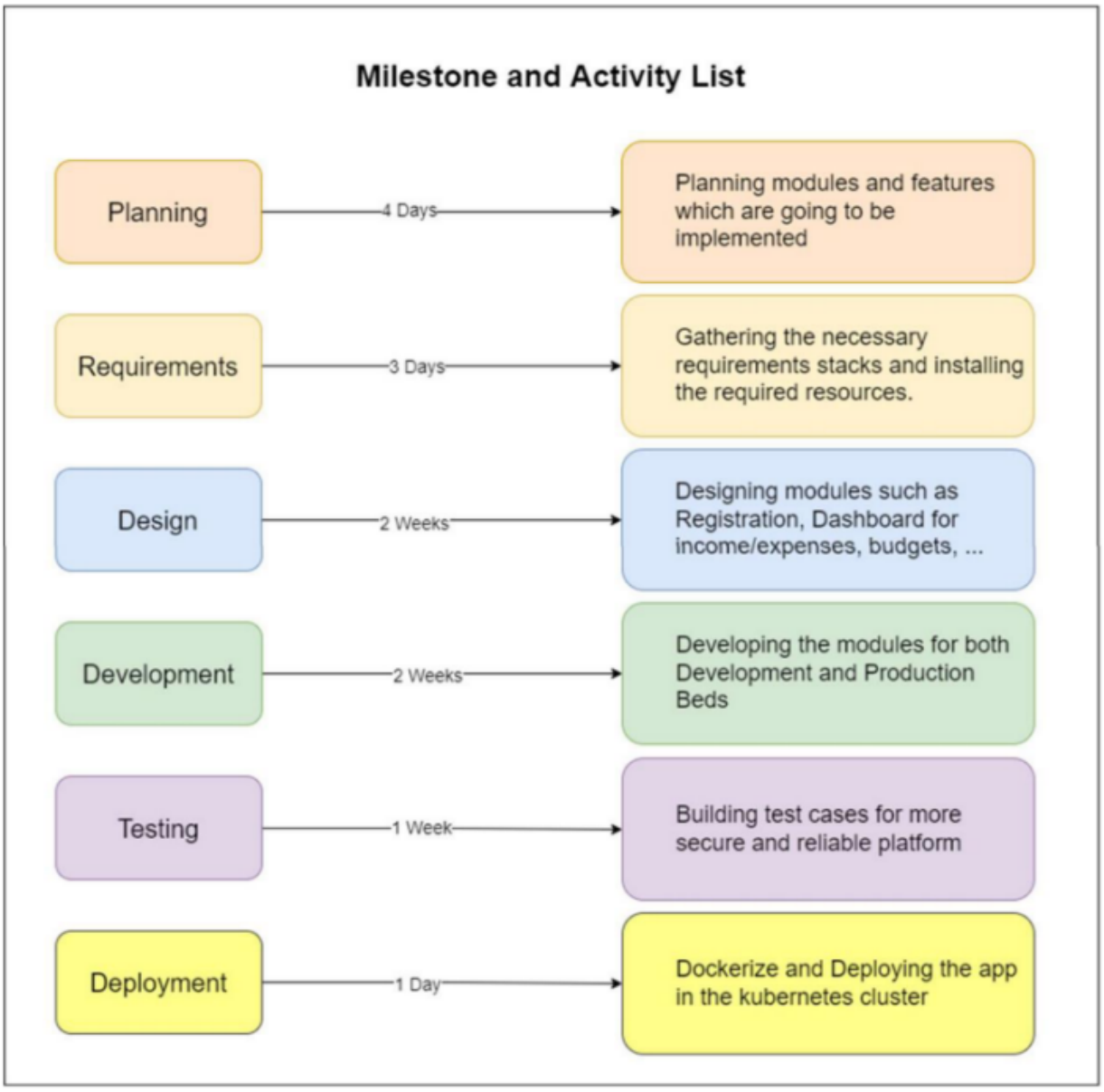


5.3. User Stories

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

6. PROJECT PLANNING & SCHEDULING

6.1. Sprint Planning & Estimation

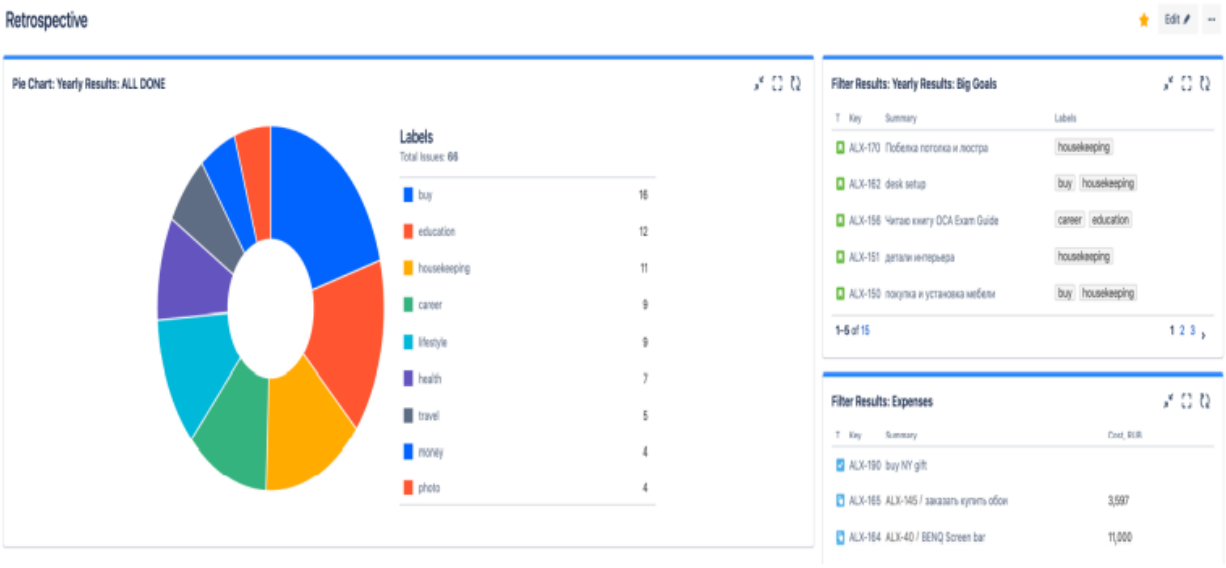


6.2. Sprint Delivery Schedule

SPRINTS	FUNCTIONAL REQUIREMENT	USER STORY NO.	USER STORY	STORY POINTS	PRIORITY	TEAM MEMBERS
ST–1	User Registration	USN-1	Create an account for the users to get access to all the features	10	High	Ajay A Abdul Rahman V
ST–1	User Wallets	USN-2	Wallets hold the users money	5	High	Ajay A
ST–1	User Category Management	USN-3	Users can customize their income and expense categories	5	Medium	Anthony Jeffery William C
ST–2	User Income	USN-4	Users can attach their income to the wallets	7	High	Gowtham T
ST–2	User Expenses	USN-5	Users can deduct their amount from the wallets	7	High	Abdul Rahman V
ST–2	User Budget Alerts	USN-6	Users can set alerts and limit their expenses	6	Medium	Ajay A , Abdul Rahman V
ST–3	Analytics	USN-7	Users can get a visualization on their income and expenses	6	High	Gowtham T, Anthony Jeffery William C
ST–3	Multilingual Support	USN-8	Users should be able to use the application in their languages	4	Low	Abdul Rahman V, Gowtham T
ST–3	File Management	USN-9	Users should be able to attach files to their income or expenses	6	Medium	Ajay A, Anthony Jeffery William C
ST–3	Economic News	USN-10	Users should be alerted with economic news	4	Low	Gowtham T
ST–4	Debt and Investment Calc	USN-11	Users can calculate their returns and risks	7	Medium	Ajay A
ST–4	Dockerization and Deploy the	USN-13	Container the application and deploy to the kubernetes cluster	13	High	Ajay A, Abdul Rahman V, Gowtham T, Anthony Jeffery

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



7. CODING & SOLUTIONING

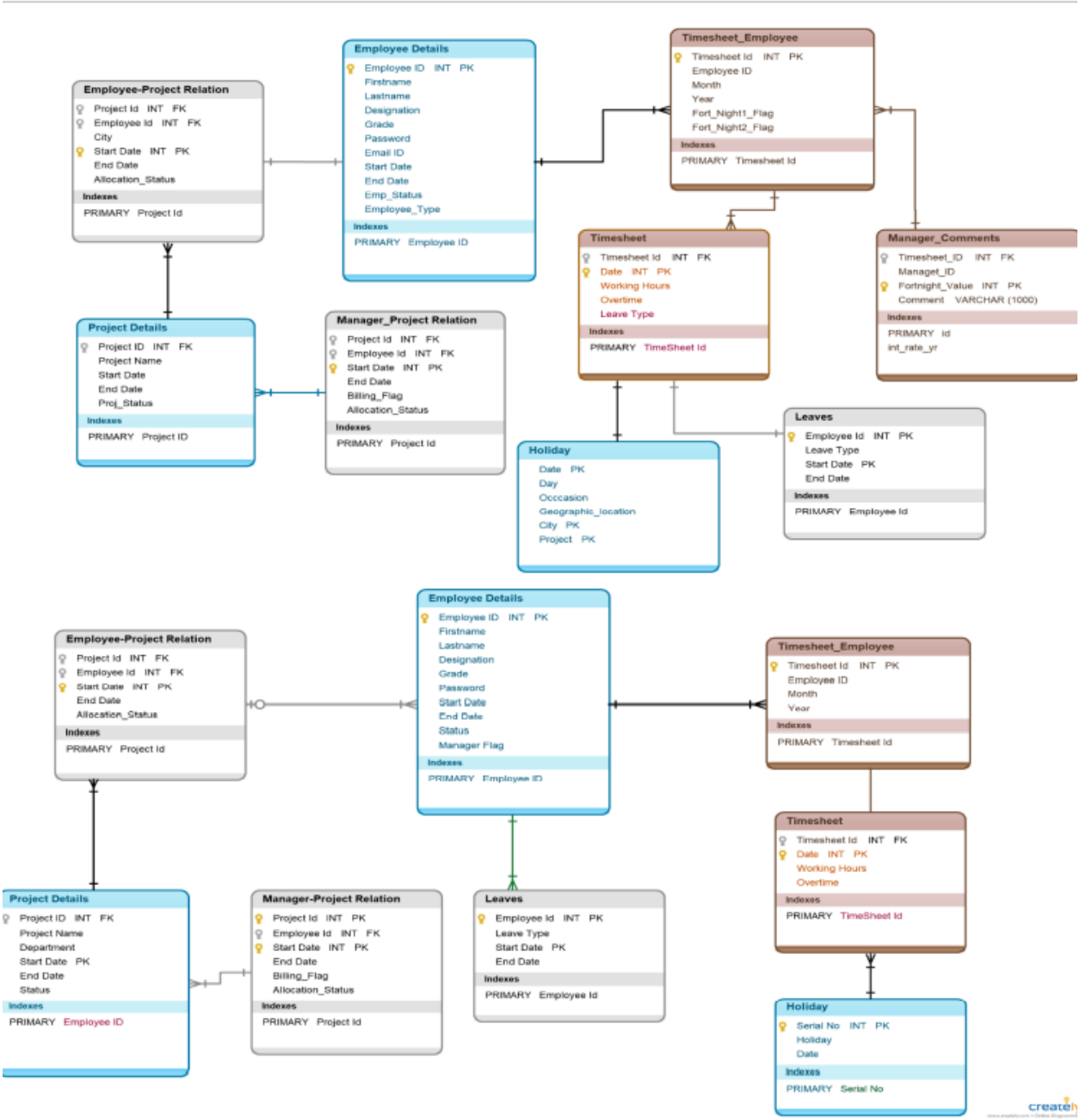
7.1. Feature 1

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



8. 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

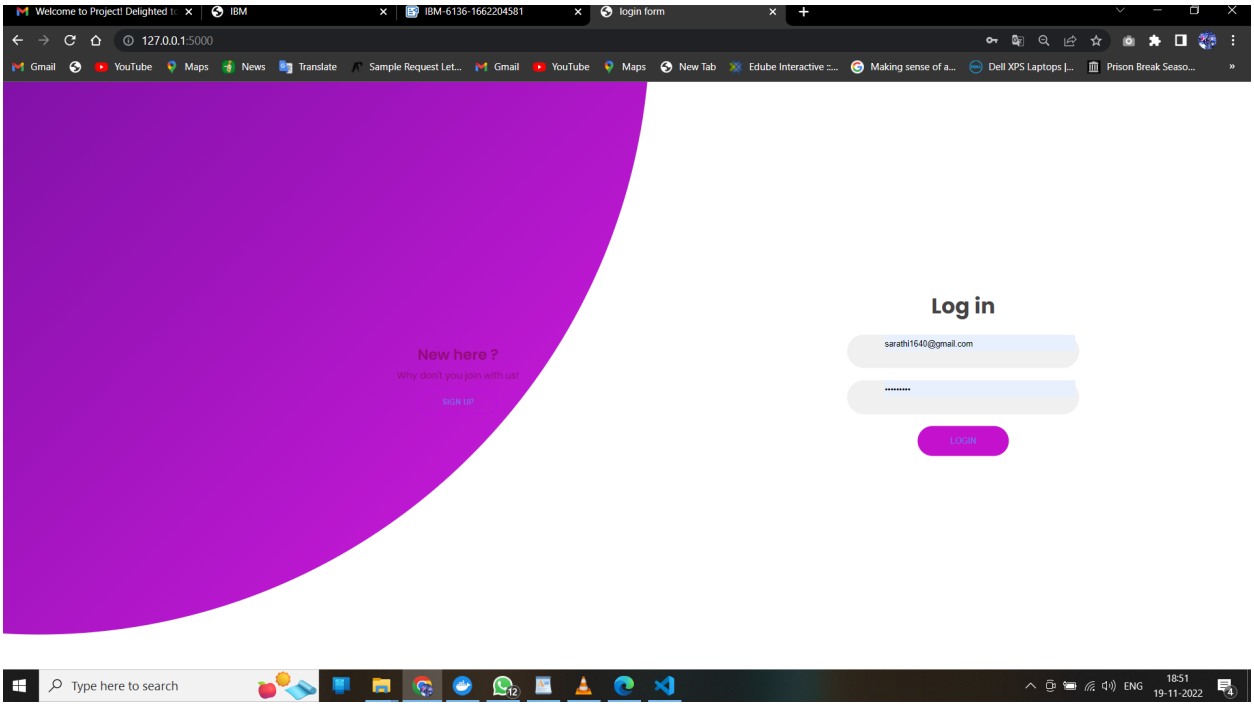
8.2. User Acceptance Testing

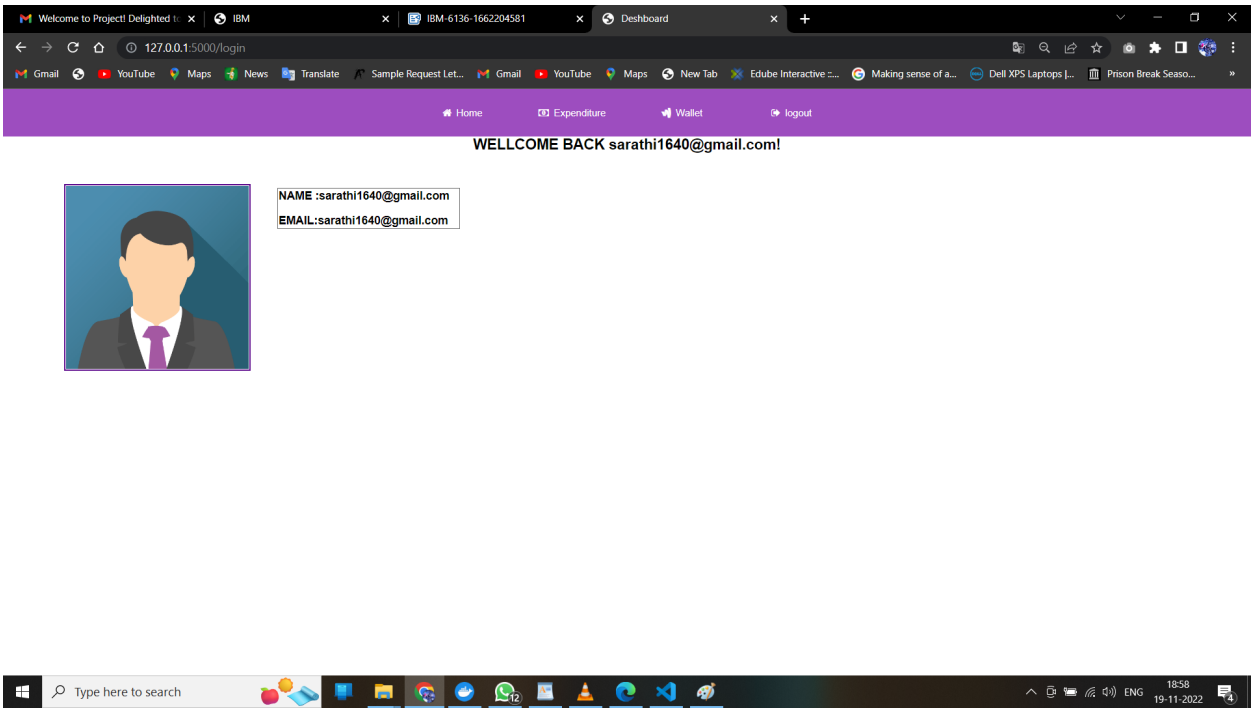
Must-have features of aJob/Skill recommender app I wanted to address the user pain points by including (and improving) the core features of the application.Personal profiles After downloading the app, a user needs to register and create an account. At this stage, users should fill in personal information 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

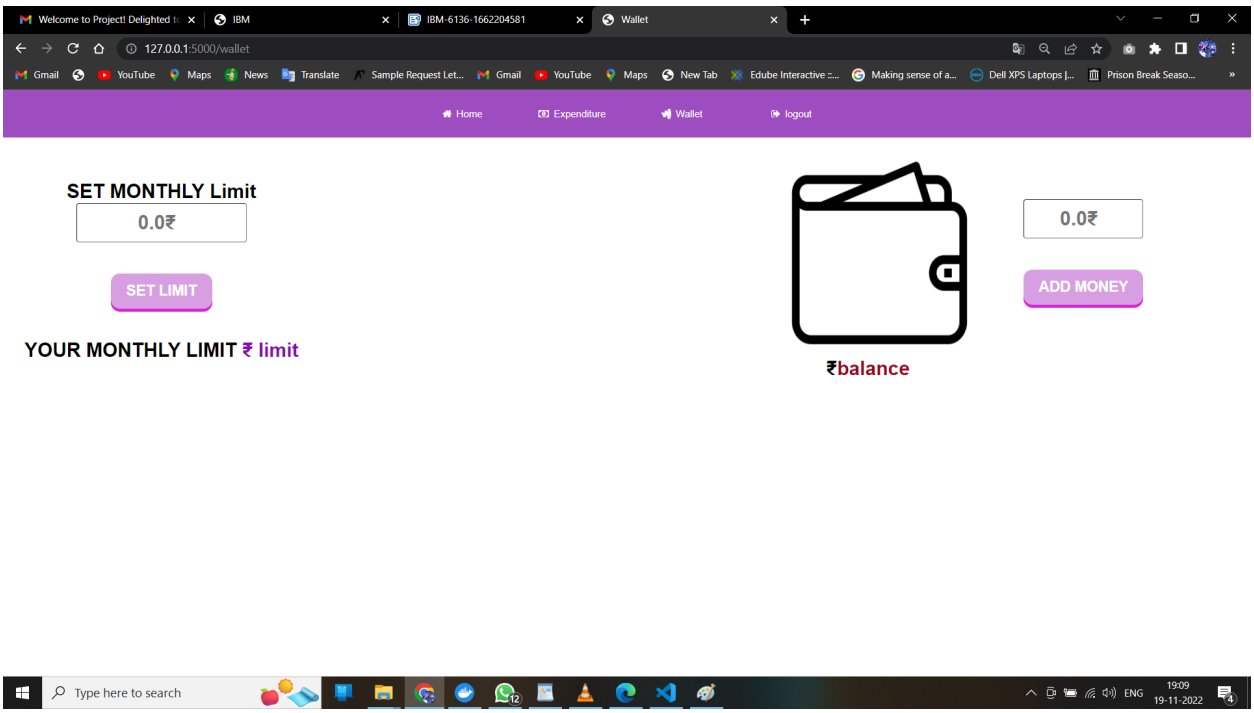
them to log what they spend on.money counter Enabling the application to calculate spent amount of users have gone and done based on the data they've logged.Barcode scanner Let users count money and see accurate spend information via a built-in barcode scanner.

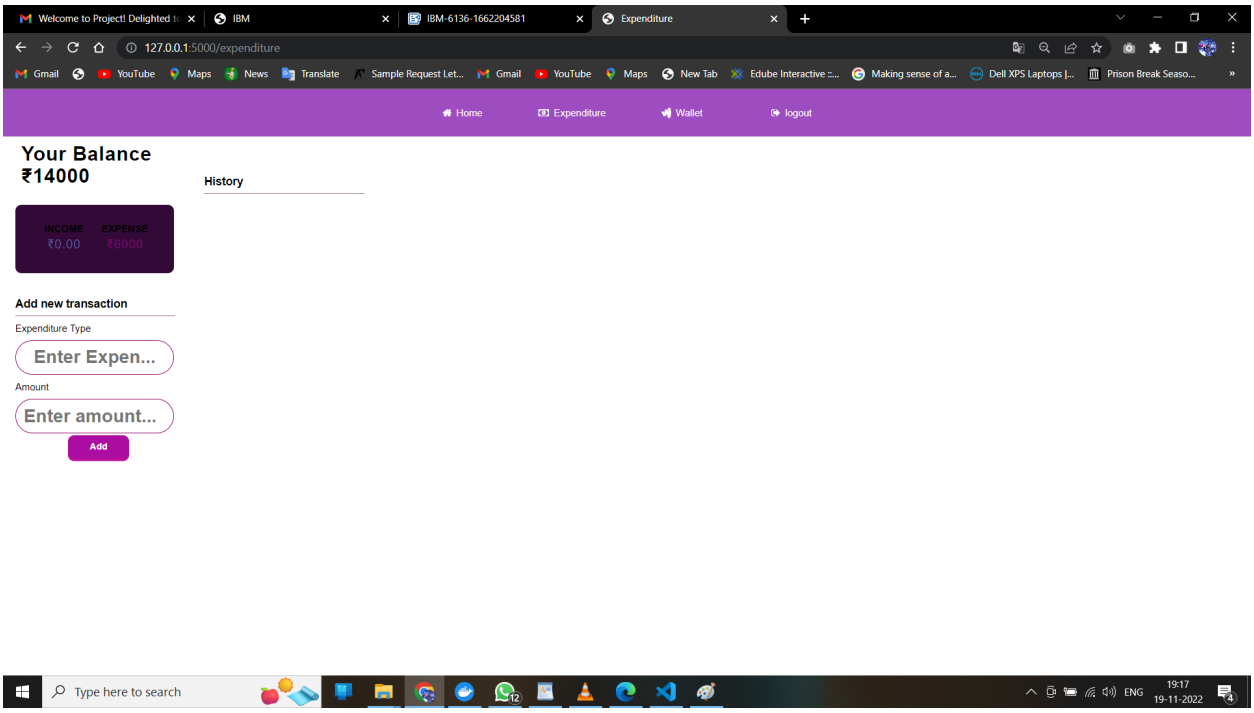
9. RESULTS

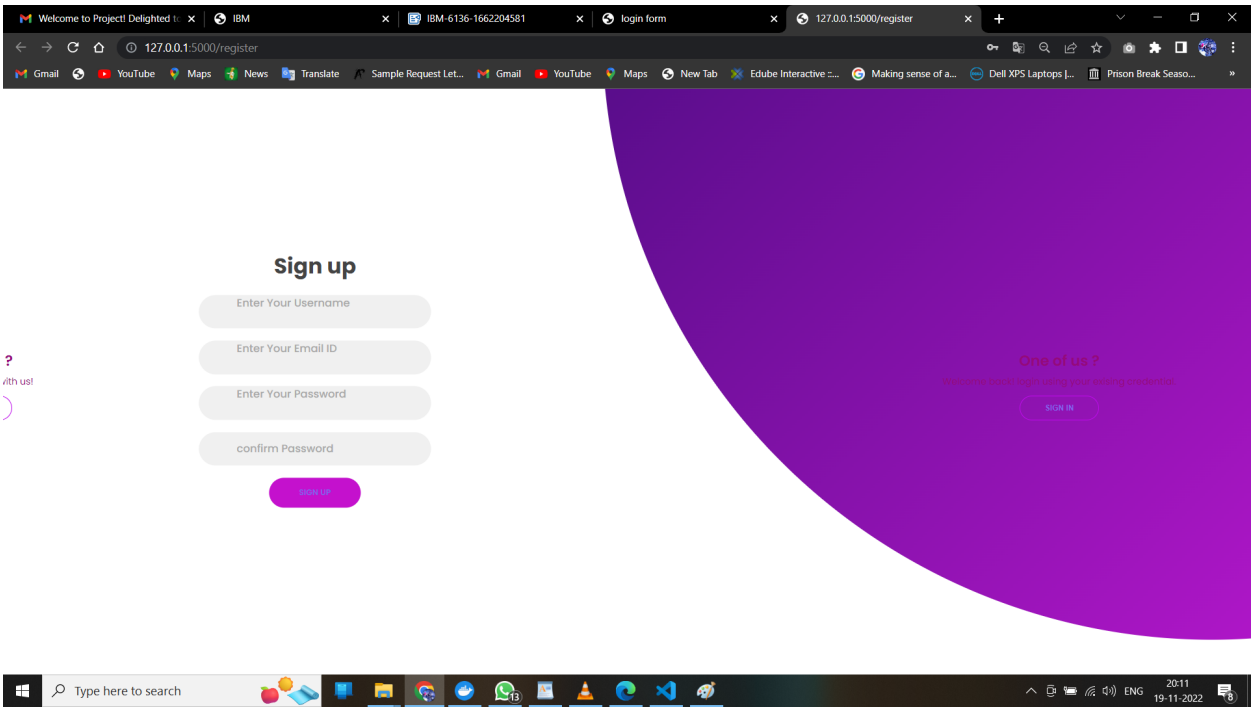
9.1. Performance Metrics




Welcome to Project! Delighted t
IBM
IBM-6136-1662204581
Deshboard
127.0.0.1:5000/login
Home
Expenditure
Wallet
logout
WELLCOME BACK sarathi1640@gmail.com!
NAME :sarathi1640@gmail.com
EMAIL:sarathi1640@gmail.com
Type here to search
ENG
18:58
19-11-2022


Welcome to Project! Delighted t
IBM
IBM-6136-1662204581
Wallet
127.0.0.1:5000/wallet
Home
Expenditure
Wallet
logout
SET MONTHLY Limit
0.0₹
SET LIMIT
YOUR MONTHLY LIMIT ₹ limit
0.0₹
ADD MONEY
₹balance
Type here to search
ENG
19:09
19-11-2022


Welcome to Project! Delighted t
IBM
IBM-6136-1662204581
Expenditure
127.0.0.1:5000/expenditure
Gmail
YouTube
Maps
News
Translate
Sample Request Let...
Gmail
YouTube
Maps
New Tab
Edube Interactive :...
Making sense of a...
Dell XPS Laptops |...
Prison Break Seaso...
Home
Expenditure
Wallet
logout
Your Balance
₹14000
INCOME
₹0.00
EXPENSE
₹6000
Add new transaction
Expenditure Type
Enter Expen...
Amount
Enter amount...
Add
History
Type here to search
ENG
19:17
19-11-2022



10. 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 identity theft, 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 about what you're spending 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.

11. 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.

12. FUTURE SCOPE

Changing face of expense management software Year-on-year, modern expense management software underwent a continuous evolution from traditional back-offic

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.

13. APPENDIX

13.1. Source Code

```
   # Store this code in 'app.py' file

from flask  import Flask, render_template, request, redirect, url_for, session
import re
import ibm_db
import sendgrid
import os
app = Flask(__name__)


app.secret_key = 'a'
hostname="fbd88901-ebdb-4a4f-a32e-
9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
uid="htm29672"
pwd="Q5MZGFK2R7T6dBrx"
driver="{IBM DB2 ODBC DRIVER}"
db="bludb"
port="32731"
protocol="TCPIP"
cert="Certificate.crt"

dsn=(
    "DATABASE={0};"
    "HOSTNAME={1};"
    "PORT={2};"
    "UID={3};"
    "SECURITY=SSL;"
    "SSLServerCerttificate={4};"
    "pwd={5};"
```

```
).format(db,hostname,port,uid,cert,pwd)
print(dsn)
try:
   conn=ibm_db.connect(dsn,"","")
   print("CONNECTED")
except:
   print("Unable to connect",ibm_db.conn_errormsg())



class data:
        def __init__(self, email = ''):
                self.email = email


        # setter method
        def set_email(self, x):
                self.email = x
        def set_name(self, x):
                self.name = x

data = data()




@app.route('/', methods =["GET", "POST"])
@app.route('/login', methods =['GET', 'POST'])
def login():
        print(request.form)
        msg = ''
        if request.method == 'POST':
                email = request.form['email']
                password = request.form['password']
                sql="SELECT * FROM USER_DETAILS WHERE EMAIL =? AND PASSWORD =?"
                stmt = ibm_db.prepare(conn,sql)
                ibm_db.bind_param(stmt,1,email)
                ibm_db.bind_param(stmt,2,password)
                ibm_db.execute(stmt)
                account =ibm_db.fetch_assoc(stmt)
                print(account)
                print("account----------")
```

```
                data.set_email(email)
                print(data.email)



                if account:
                        sql="SELECT * FROM USER_DETAILS WHERE EMAIL =?"
                        stmt = ibm_db.prepare(conn,sql)
                        ibm_db.bind_param(stmt,1,email)
                        ibm_db.execute(stmt)
                        username =ibm_db.fetch_assoc(stmt)
                        name=username.get("USER_NAME")
                        data.set_name(username.get("USER_NAME"))







                        return render_template('home.html',name=data.name,email=data.email)
                else:
                        msg = 'Incorrect username / password !'
        return render_template('login.html', msg = msg)


@app.route('/register', methods =['GET', 'POST'])
def register():
        msg = ''
        if request.method == "POST":
                username = request.form['username']
                password = request.form['password']
                email = request.form['email']
                sql ="SELECT * FROM USER_DETAILS WHERE USER_NAME = ?"
                stmt = ibm_db.prepare(conn,sql)
                ibm_db.bind_param(stmt,1,username)
                ibm_db.execute(stmt)
                account = ibm_db.fetch_assoc(stmt)
                print(account)
                print("account----------")

                if account:
                        msg = 'Account already exists ! Please Login'
                        return render_template('login.html', msg = msg)
```

```
                elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
                        msg = 'Invalid email address !'
                elif not re.match(r'[A-Za-z0-9]+', username):
                        msg = 'Username must contain only characters and numbers !'
                else:
                        inser_sql="INSERT INTO USER_DETAILS VALUES(?,?,?)"
                        prep_stmt = ibm_db.prepare(conn, inser_sql)
                        ibm_db.bind_param(prep_stmt,1,username)
                        ibm_db.bind_param(prep_stmt,2,email)
                        ibm_db.bind_param(prep_stmt,3,password)
                        ibm_db.execute(prep_stmt)
                        inser_sql="INSERT INTO BALANCE VALUES(?,?,?,?)"
                        prep_stmt = ibm_db.prepare(conn, inser_sql)
                        ibm_db.bind_param(prep_stmt,1,0)
                        ibm_db.bind_param(prep_stmt,2,0)
                        ibm_db.bind_param(prep_stmt,3,0)
                        ibm_db.bind_param(prep_stmt,4,email)
                        ibm_db.execute(prep_stmt)


                        msg = 'You have successfully registered !'
                        return render_template('login.html', msg = msg)

        elif request.method == 'POST':
                msg = 'Please fill out the form !'
        return render_template('register.html', msg = msg)

@app.route('/home')
def dash():
        sql="SELECT * FROM USER_DETAILS WHERE EMAIL =?"
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,data.email)
        ibm_db.execute(stmt)
        username =ibm_db.fetch_assoc(stmt)
        name=username.get("USER_NAME")
        email=username.get("EMAIL")

        #name=data.email
        return render_template('home.html',name=name,email=email)

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

```
def expenditure():
	account=[]
	if request.method == "GET":
		sql ="SELECT EMAIL FROM EXPENSE"
		stmt = ibm_db.prepare(conn,sql)
		#ibm_db.bind_param(stmt,1,data.email)
		ibm_db.execute(stmt)
		#account.append(ibm_db.fetch_assoc(stmt))
		account=ibm_db.free_result(stmt)
		print(account)
		print("get")
		print(request.form)
		sql="SELECT * FROM BALANCE WHERE EMAIL =?"
		stmt = ibm_db.prepare(conn,sql)
		ibm_db.bind_param(stmt,1,data.email)
		ibm_db.execute(stmt)
		username =ibm_db.fetch_assoc(stmt)
		expense=username.get("EXPENSE")
		balance=username.get("BALANCE")
		return render_template('expenditure.html',expense=expense,balance=balance)


	else:
		print("----------POST----------")
		print(request.form)
		expense=request.form["expense"]
		etype=request.form["type"]
		sql="UPDATE BALANCE SET EXPENSE=EXPENSE+?,BALANCE=BALANCE-?
WHERE EMAIL=?"
		prep_stmt = ibm_db.prepare(conn,sql)
		ibm_db.bind_param(prep_stmt,1,expense)
		ibm_db.bind_param(prep_stmt,2,expense)
		ibm_db.bind_param(prep_stmt,3,data.email)
		ibm_db.execute(prep_stmt)
		inser_sql="INSERT INTO EXPENSE VALUES(?,?,?)"
		prep_stmt = ibm_db.prepare(conn, inser_sql)
		ibm_db.bind_param(prep_stmt,1,data.email)
		ibm_db.bind_param(prep_stmt,2,etype)
		ibm_db.bind_param(prep_stmt,3,expense)
		ibm_db.execute(prep_stmt)

		sql="SELECT * FROM BALANCE WHERE EMAIL =?"
```

```
                stmt = ibm_db.prepare(conn,sql)
                ibm_db.bind_param(stmt,1,data.email)
                ibm_db.execute(stmt)
                username =ibm_db.fetch_assoc(stmt)
                balance=username.get("BALANCE")
                expense=username.get("EXPENSE")
                limit=username.get("LIMIT")
                if expense > limit:
                        sg =
sendgrid.SendGridAPIClient(api_key='SG.e6jaCTWzQCW57Vegrqxp2Q.hch_gxXWcC8jsve_Lnfm_
ZcDBrnYHtS6dT1AGAEZg90')
                        dat = {
          "personalizations": [
            {
              "to": [
                {
                  "email":data.email
                }
              ],
              "subject": "EXPENDITURE LIMIT Alert!"
            }
          ],
          "from": {
            "email": "sathesh9988@gmail.com"
          },
          "content": [
            {
              "type": "text/plain",
              "value": "Your monthly expense limit exit your balance:"+balance+"
expense:"+expense+" limit:"+limit
            }
          ]
        }
                        response = sg.client.mail.send.post(request_body=dat)
                        print(response.status_code)
                        print(response.body)
                        print(response.headers)
        return render_template('expenditure.html',expense=expense,balance=balance)
```

```
@app.route('/wallet',methods =['GET','POST'])
def wallet():
        if request.method == "GET":
                print("get")
                print(request.form)
                sql="SELECT * FROM BALANCE WHERE EMAIL =?"
                stmt = ibm_db.prepare(conn,sql)
                ibm_db.bind_param(stmt,1,data.email)
                ibm_db.execute(stmt)
                username =ibm_db.fetch_assoc(stmt)
                limit=username.get("LIMIT")
                balance=username.get("BALANCE")
                return render_template('wallet.html',limit=limit,balance=balance)


        else:
                print("post")
                print(request.form)
                limit=request.form["limit"]
                balance=request.form["balance"]
                if limit!='':
                        sql="UPDATE BALANCE SET LIMIT=? WHERE EMAIL=?"
                        prep_stmt = ibm_db.prepare(conn,sql)
                        ibm_db.bind_param(prep_stmt,1,limit)
                        ibm_db.bind_param(prep_stmt,2,data.email)
                        ibm_db.execute(prep_stmt)
                        sql="SELECT * FROM BALANCE WHERE EMAIL =?"
                        stmt = ibm_db.prepare(conn,sql)
                        ibm_db.bind_param(stmt,1,data.email)
                        ibm_db.execute(stmt)
                        username =ibm_db.fetch_assoc(stmt)
                        limit=username.get("LIMIT")
                        balance=username.get("BALANCE")
                        return render_template('wallet.html',limit=limit,balance=balance)
                elif balance!='':
                        sql="UPDATE BALANCE SET BALANCE=BALANCE+? WHERE EMAIL=?"
                        prep_stmt = ibm_db.prepare(conn,sql)
                        ibm_db.bind_param(prep_stmt,1,balance)
                        ibm_db.bind_param(prep_stmt,2,data.email)
                        ibm_db.execute(prep_stmt)
                        sql="SELECT * FROM BALANCE WHERE EMAIL =?"
```

```
                stmt = ibm_db.prepare(conn,sql)
                ibm_db.bind_param(stmt,1,data.email)
                ibm_db.execute(stmt)
                username =ibm_db.fetch_assoc(stmt)
                limit=username.get("LIMIT")
                balance=username.get("BALANCE")
                return render_template('wallet.html',limit=limit,balance=balance)
        else:
                sql="SELECT * FROM BALANCE WHERE EMAIL =?"
                stmt = ibm_db.prepare(conn,sql)
                ibm_db.bind_param(stmt,1,data.email)
                ibm_db.execute(stmt)
                username =ibm_db.fetch_assoc(stmt)
                limit=username.get("LIMIT")
                balance=username.get("BALANCE")
                return render_template('wallet.html',limit=limit,balance=balance)



@app.route('/apply',methods =['GET','POST'])
def apply():
        msg=''
        if request.mwthod == 'POST':
                username = request.form['username']
                email=request.form['email']
                qualification=request.form['qualification']
                skills=request.form['skills']
                jobs = request.form['s']
                ################################################
                msg='YOU ARE SUCESFULLY APPLIED '
                session['loggedin']=True
        elif request.method == 'POST':
                msg='please fill the form'
                return render_template('apply.html',msg = msg)
@app.route('/logout')
def logout():
  return render_template('home.html')

if __name__=='__main__':
        app.run(host='0.0.0.0')
```

13.2. GitHub & Project Demo Link

GITHUB: https://github.com/IBM-EPBL/IBM-Project-6136-1658823978

PROJECT DEMO

LINK:https://drive.google.com/drive/folders/1CwmZuAP7MYNZfMbpOA3miTHqwlp89tog