PERSONAL EXPENSE TRACKER APPLICATION

IBM-Project-2693-1658481225

NALAIYA THIRAN PROJECT BASED LEARNING ON PROFESSIONAL READLINESS FOR INNOVATION, EMPLOYMENT AND ENTERPRENEURSHIP

A PROJECT REPORT

Submitted By:

SOUMIK RAKSHIT [2019506093]

AARTHI IYER [2019506003]

SHRUTHI MUTHU[2019506088]

BARATHVARAJ [2019506018]

ADITYA V [2019506009]

TEAM ID: PNT2022TMID36032

INDUSTRY MENTOR: Kushboo

FACULTY MENTOR : Eliza Femi Sherley S

EVALUATOR: Kola Sujatha

MADRAS INSTITUITE OF TECHNOLOGY, ANNA UNIVERSITY -600044

Project Report

1. **INTRODUCTION**
    a. Project Overview
    b. Purpose
2. **LITERATURE SURVEY**
    a. Existing problem
    b. References
    c. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
    a. Empathy Map Canvas
    b. Ideation & Brainstorming
    c. Proposed Solution fit
    d. Problem Solution
4. **REQUIREMENT ANALYSIS**
    a. Functional requirement
    b. Non-Functional requirements
5. **PROJECT DESIGN**
    a. Data Flow Diagrams
    b. Solution & Technical Architecture
    c. User Stories
6. **PROJECT PLANNING & SCHEDULING**
    a. Sprint Planning & Estimation
    b. Sprint Delivery Schedule
    c. Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
    a. Feature 1
    b. Feature 2
    c. Database Schema (if Applicable)
8. **TESTING**
    a. Test Cases
    b. User Acceptance Testing

9. **RESULTS**

10. **ADVANTAGES & DISADVANTAGES**

11. **CONCLUSION**

12. **FUTURE SCOPE**

13. **APPENDIX**

    Source Code

    GitHub & Project Demo Link

1. INTRODUCTION

1.a Project overview:

In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. Xpense Tracker, a personal finance app will not only help with budgeting and accounting but also give helpful insights about money management.

The application will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. The application will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert. An application tailored to young adults to start tracking their expenses, splitting their bills, learning to budget and save. This application would be designed to be more portable than traditional systems and help users to efficiently manage and track their expenses.

1.b Purpose:

Personal finance management is an important part of people's lives. However, young adults may lack the knowledge or time to manage their finances in a proper manner. And, even if a person has time and knowledge, they could not be bothered with tracking their expenses as they find it tedious and time-consuming.

An expense tracker is a software or application that helps to keep an accurate record of your money inflow and outflow. Young adults who live on a fixed income are most likely to find themselves with insufficient funds that towards the end of the month to meet their needs. This problem can arise due to an array of reasons, it may also be due to poor money management skills.

People tend to overspend without realizing, and this can prove to be disastrous. Using a daily expense manager can help you keep track of how much you spend every day and on what.. At the end of the month, you will have a clear picture where your money is going. Further, one can also set saving goals and track the progress on that.

An effective tracking application that caters to the needs of people who are new to money and its management with its user friendly design and high portability would prove to be extremely helpful.

2. LITERATURE SURVEY

2.a Existing Problem:

The existing problem can be identified by understanding how people tracked their expenses without any app. Getting notified by your bank each time there is a transaction is a great way to track your finances in real time and to keep a record of these transactions handy for you to conveniently check later on. Categorizing them will help you understand your expenses even better. Segregating your expenses into categories such as monthly bills, loans, shopping, groceries, etc. will ensure that you track your spends effectively.Experts recommend that budget should be reviewed at least every 15 days so that you can stay on top of your finances and any refactoring can be done.People sometimes also create their own Excel sheets. But all of the above measures take voluntary efforts on the user's part or some previous financial management experience.

Newly independent young adults that are just getting into money management need a simple and straightforward system that can make them aware of the potential pitfalls such as letting your expenses exceed your income.The system must also assist the user in creating the best way to tackle this by budgeting and creating a personal spending plan to track the money you have coming in and the money you have going out. . A system that is non intrusive yet can help the users exercise self- control and stick to the spending plan is in need. Before you even get your first paycheck, it's important to understand how income tax works. When a company offers you a starting salary, you need to calculate whether that salary will give you enough money after taxes to meet your financial obligations—and, with smart planning,meet your savings and retirement goals as well.

Some of the existing expense tracking applications in the market include Wallet, Walnut, Mint, Money Manager, AndroMoney, Splitwise, Monefy. Some of the features that these applications are lacking in include data security, incompatibility in some stores, problems with data breach, too many advertisements, absence of multicurrency support, complex UI etc.

2.b References:

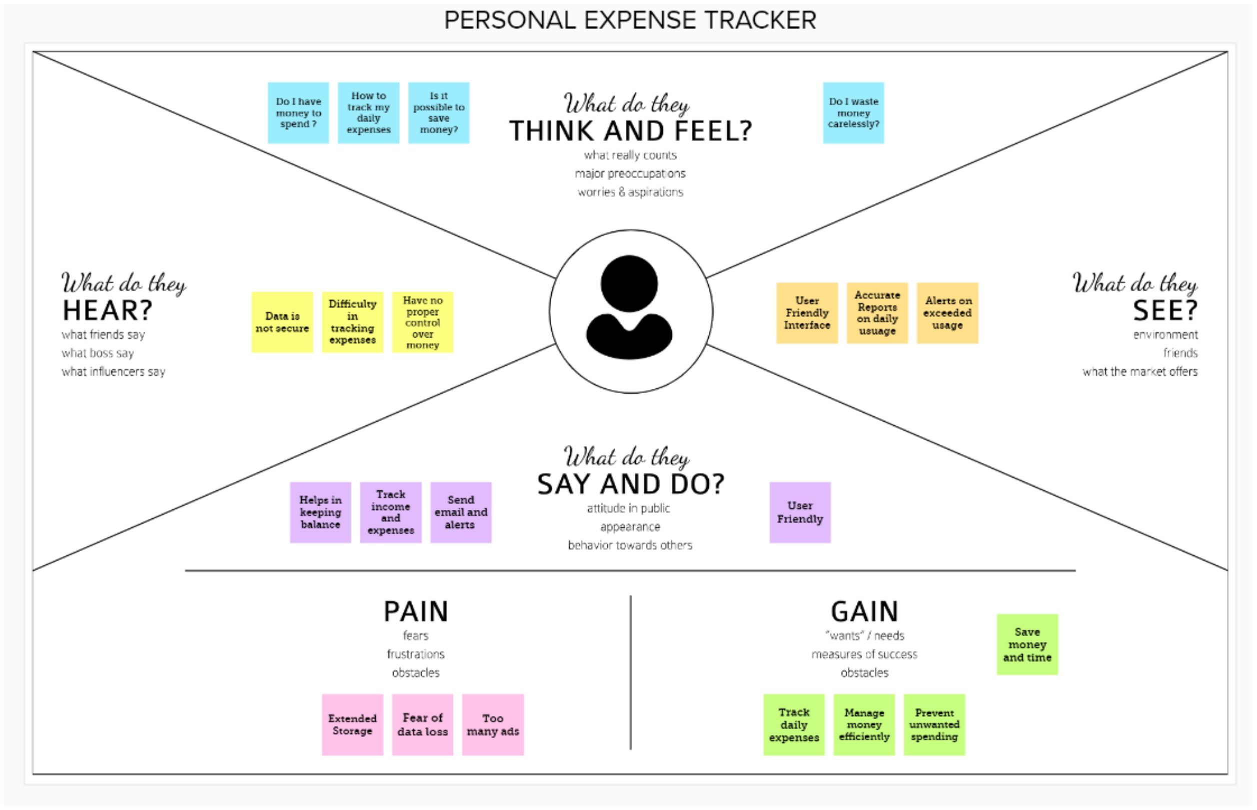
1. https://moneyview.in/blog/how-to-track-your-finances/
2. https://www.investopedia.com/articles/younginvestors/08/eight-tips.asp
3. https://www.cnbc.com/select/best-expense-tracker-apps/
4. https://www.nerdwallet.com/article/finance/expense-tracker-apps
5. https://financialgym.com/financiallynakedpodcast/episode52

2.c Problem Statement Definition:

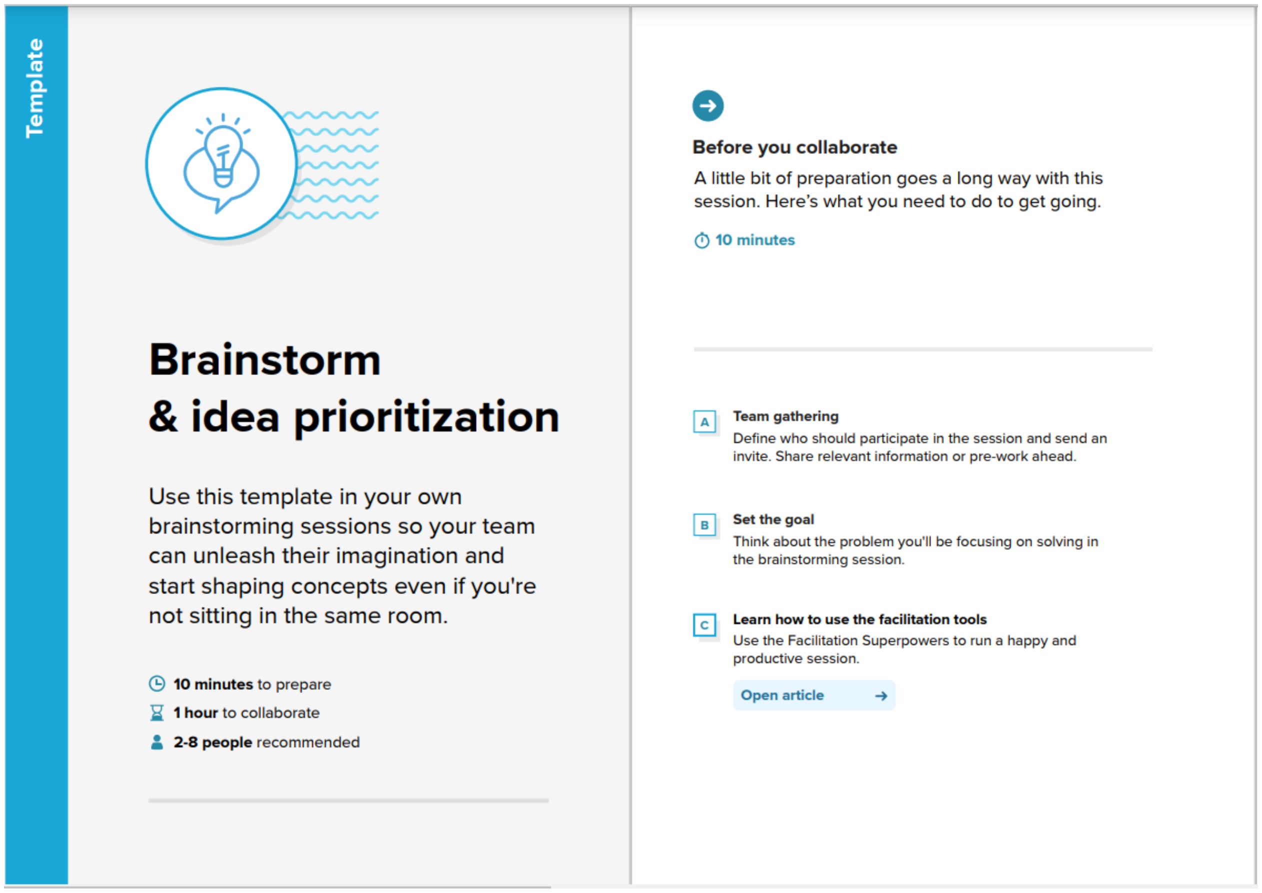
Young Adults often fall into bad financial practices and sometimes even crushing debt since personal finance is not taught in schools and colleges. This is a pressing issue and must be solved through reliable resources and early intervention. An application tailored to young adults to start tracking their expenses, splitting their bills, learning to budget and save is a good starting point.Young adults must also be encouraged to review their spending practices frequently and bring about any necessary changes in order to get a grip on their finances. This application would be designed to be more portable than traditional systems and help users to efficiently manage and track their expenses with required notifications and alerts.

3. IDEATION AND PROPOSED SOLUTION

3.a Empathy Map Canvas:



3.b Ideation & Brainstorming:



Template

Brainstorm
& idea prioritization

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

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

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

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.

Open article →

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

How might we build an application that helps youth get started with tracking and managing their finances easily?

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.

2

Brainstorm

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

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Person 1

- An app for young adults who are just learning how to handle money. Basic budgeting, saving and splitting of bills with in-app payment integrations.
- An app to help people curb online shopping addiction/ impulses
- Realtime Support and Feedback
- Build interactive and responsive UI

Person 2

- An automated expense tracking application that is highly interactive and targets singular users who wish to effectively manage expenses.
- An highly secure expense tracker system that uses fingerprint for data security and privacy
- A help desk to respond to queries and assist people
- Machine Learning based application to predict future spending

Person 3

- An app with simple yet effective UI that anyone can use with ease
- Seamless and smooth transaction of payments by establishing a stable network
- An intelligent AI chatbot for answering any queries put up by user
- A feedback form to be filled by users for imporving their experience with the app

Person 4

- An app with interactive and user friendly interface with different levels of security
- Different levels of registration (basic and professional) to manage different needs
- Involves simple expense reports, integration of payment service with app, assign expense goals categorically.
- Advance reports generated using data analytics, stock performance tracking, multiple currency support

Person 5

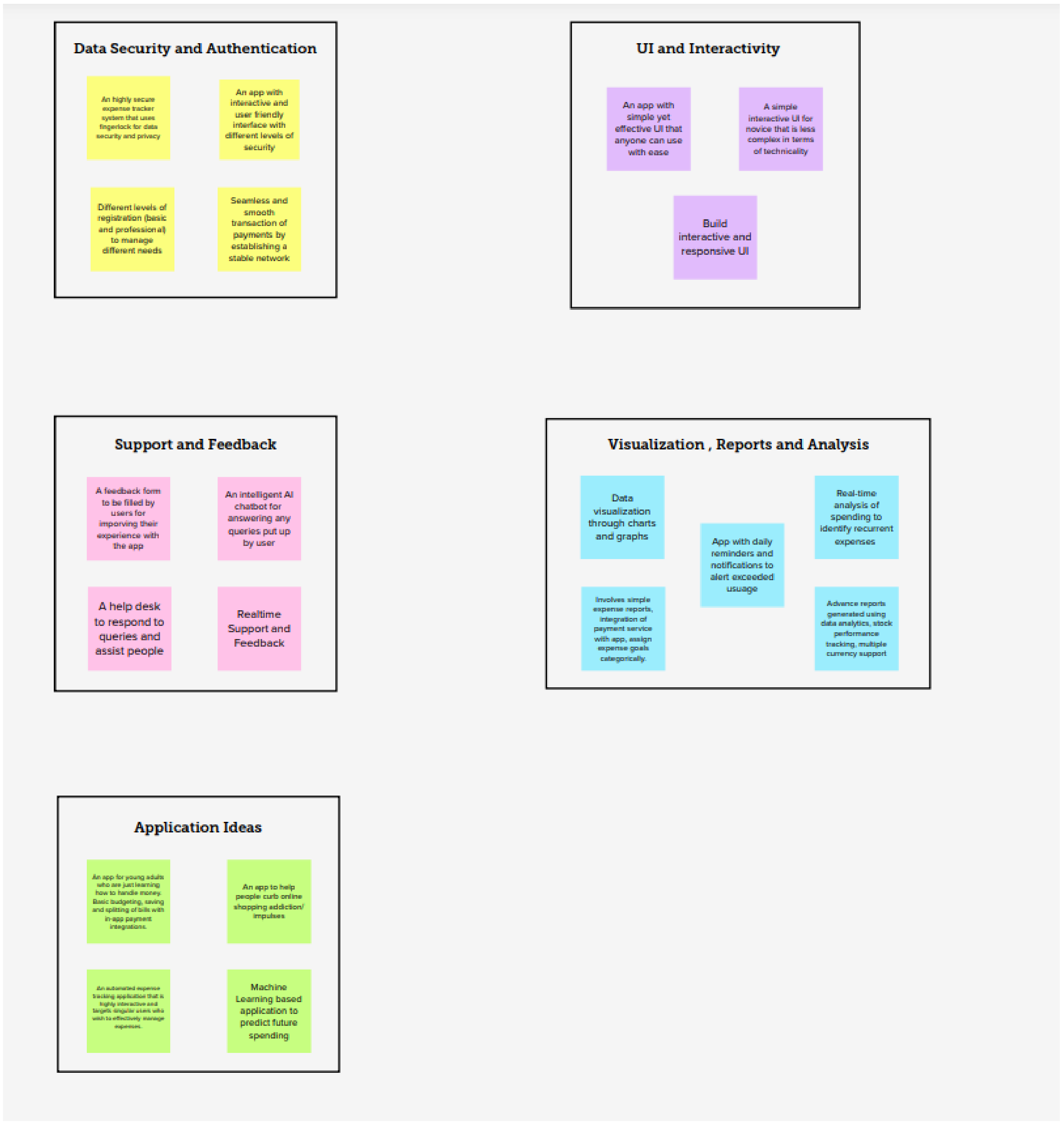
- A simple interactive UI for novice that is less complex in terms of technicality
- Data visualization through charts and graphs
- App with daily reminders and notifications to alert exceeded usuage
- Real-time analysis of spending to identify recurrent expenses

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, 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.

20 minutes



Data Security and Authentication

An highly secure expense tracker system that uses fingerlock for data security and privacy

An app with interactive and user friendly interface with different levels of security

Different levels of registration (basic and professional) to manage different needs

Seamless and smooth transaction of payments by establishing a stable network

UI and Interactivity

An app with simple yet effective UI that anyone can use with ease

A simple interactive UI for novice that is less complex in terms of technicality

Build interactive and responsive UI

Support and Feedback

A feedback form to be filled by users for imporving their experience with the app

An intelligent AI chatbot for answering any queries put up by user

A help desk to respond to queries and assist people

Realtime Support and Feedback

Visualization , Reports and Analysis

Data visualization through charts and graphs

App with daily reminders and notifications to alert exceeded usuage

Real-time analysis of spending to identify recurrent expenses

Involves simple expense reports, integration of payment service with app, assign expense goals categorically.

Advance reports generated using data analytics, stock performance tracking, multiple currency support

Application Ideas

An app for young adults who are just learning how to handle money. Basic budgeting, saving and splitting of bills with in-app payment integrations.

An app to help people curb online shopping addiction/ impulses

An automated expense tracking application that is highly interactive and targets singular users who wish to effectively manage expenses.

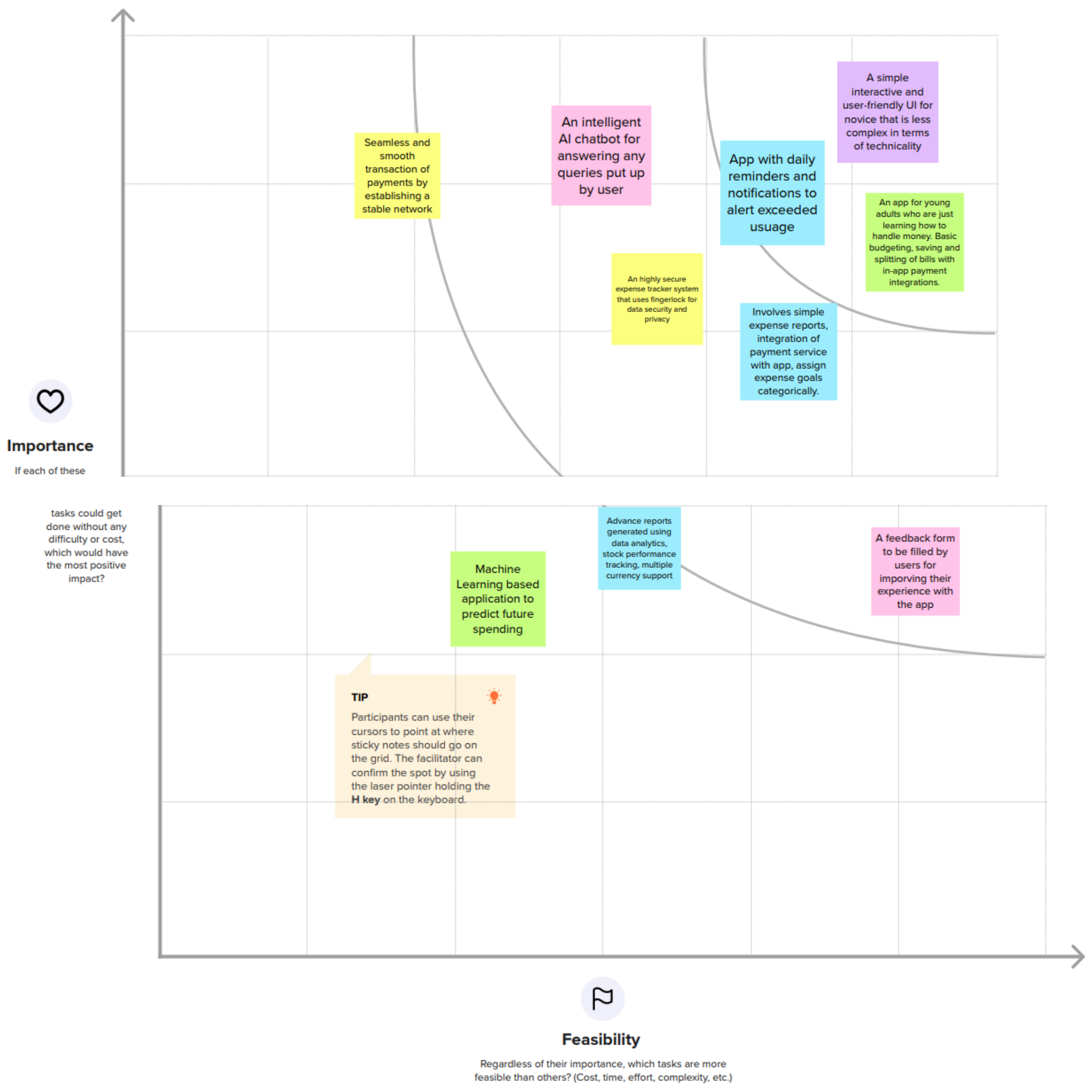
Machine Learning based application to predict future spending

4

Prioritize

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

20 minutes


Seamless and smooth transaction of payments by establishing a stable network
An intelligent AI chatbot for answering any queries put up by user
A simple interactive and user-friendly UI for novice that is less complex in terms of technicality
App with daily reminders and notifications to alert exceeded usage
An app for young adults who are just learning how to handle money. Basic budgeting, saving and splitting of bills with in-app payment integrations.
An highly secure expense tracker system that uses fingerlock for data security and privacy
Involves simple expense reports, integration of payment service with app, assign expense goals categorically.
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Advance reports generated using data analytics, stock performance tracking, multiple currency support
A feedback form to be filled by users for imporving their experience with the app
Machine Learning based application to predict future spending
TIP
Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the H key on the keyboard.
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

3.c Proposed Solution fit:

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

Youngsters around the age of 12-20 who desperately need an app to manage their day to day expenses

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

Customer constraints are spending power and budget

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

They would have tried to keep track of their expenses manually with pen and paper or asking their parents/friends to keep track of it

There are many neo-banking solutions as well but they do not cater to the specific needs of young adults.

This is very difficult as it needs constant human monitoring and is not very feasible

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.

- Making payments
- Tracking expenses
- Splitting their bills
- Learning to budget and save
- Alert users whenever required.
- Security

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists? What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

All neobanking applications as of today are tailored towards adults. Nowadays, youngsters have purchasing needs too and it is necessary that they get financial educated early on. But it is difficult to trust teens with full-fledged payment applications.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

Ask parents to buy for them

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

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbor installing solar panels, reading about a more efficient solution in the news.

Users would feel triggered to act when they have to purchase stationery for school, purchase food or snacks, split bills with friends etc

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

Users feel less in control and they feel reliant on others for making simple decisions
Users will also feel less aware of the financial climate as well as the right personal finance practices that one must carry out if they are not allowed to learn early on.

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.

An application tailored to young adults to start tracking their expenses, splitting their bills, learning to budget and save is a good starting point.. This application would be designed to be more portable than traditional systems and help users to efficiently manage and track their expenses with required notifications and alerts.

8. CHANNELS of BEHAVIOR CH

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

Make payments
Analyze spending
Set financial goals for self

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

Go to shops and restaurants where they make payments

3.d Proposed solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Young Adults often fall into bad financial practices and sometimes even crushing debt owing to lack of awareness about good spending habits and reliable tracking/management resources.
2.	Idea / Solution description	An application tailored to young adults to start tracking their expenses, splitting their bills, learning to budget/save and review their spending practices frequently and bring about any necessary changes in order to get a grip on their finances.
3.	Novelty / Uniqueness	More portable than traditional systems and equipped to help users between the ages of 12-20 to efficiently manage and track their expenses with required notifications and alerts and non-invasive parent involvement and supervision.
4.	Social Impact / Customer Satisfaction	Sense of financial freedom, instill good expense management practices in young adults early on, effective parent/guardian participation in teaching financial responsibilities.
5.	Business Model (Revenue Model)	Data Monetization can be employed.
6.	Scalability of the Solution	Scalability is ensured using micro-service architecture.

4. REQUIREMENT ANALYSIS

4.a. Functional Requirements

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 form for collecting details
FR-2	User Confirmation	Confirmation via Email
FR-3	Login	Entering the valid username and password
FR-4	Wallet	Application must allow user to perform transaction using the wallet
FR-5	Report Generation	Statistical report generation to visualise weekly expenditure
FR-6	Alerts/Notifications	To send alert through emails to notify user if the expenses if crossed a certain limit

4.b Non-functional Requirements

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

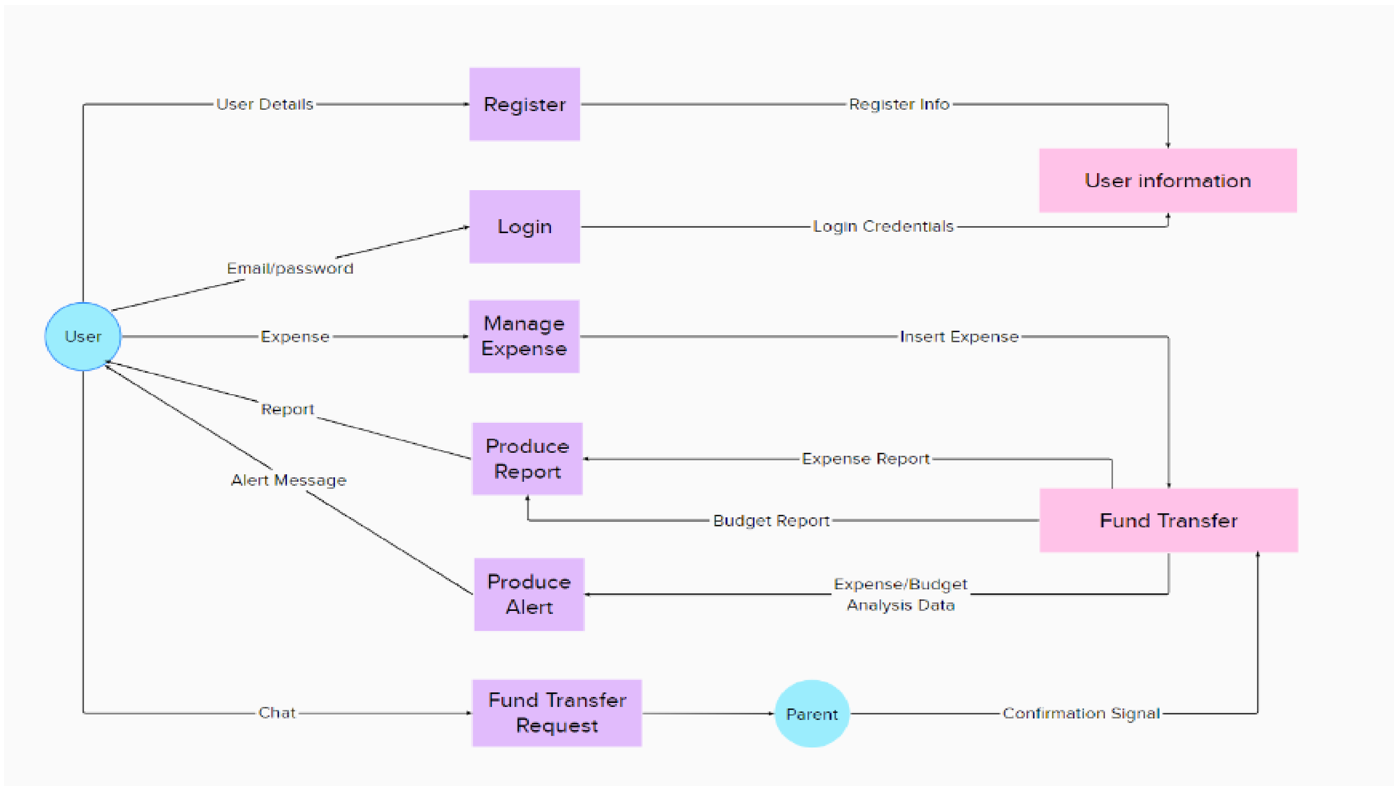
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	User friendly and interactive experience to manage and track day-to-day expenses.
NFR-2	**Security**	User data must be stored in a secure environment. Transactions should be authorised beforehand.
NFR-3	**Reliability**	User data is stored in a well defined database and backups are maintained. Replicas of data are maintained to prevent data loss
NFR-4	**Performance**	Should have high throughput and less latency while performing transactions to maintain data integrity.
NFR-5	**Availability**	The application must be available 24/7 to the users. There should be multiple instances of the server and it should be up and running so that even if one fails the other instances respond to the requests.

NFR-6	**Scalability**	Must be able to handle the increasing needs of users as the number of users increase. Can use kubernetes to create and manage multiple instances of a docker image.

5. PROJECT DESIGN

5.a Data Flow Diagrams

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



5.b Solution and Technical Architecture

Table-1 : Components & Technologies:

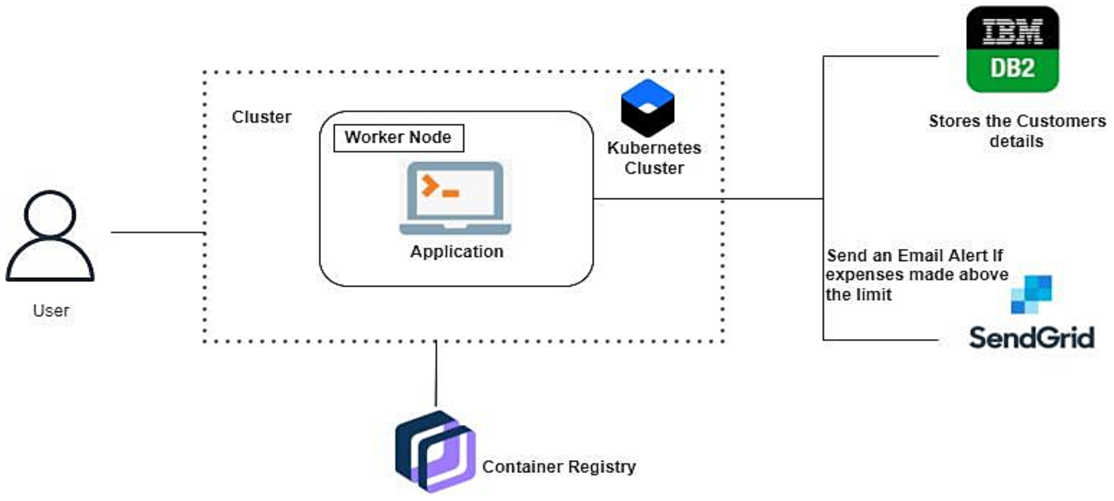
S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g.Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / React Js etc.

2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.
9.	External API-2	Purpose of External API used in the application	Aadhar API, etc.
10.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Technology of Opensource framework
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Technology used

4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Technology used
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Technology used



5.c. User Stories

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

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile/Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1

		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-2
		USN-3	As a user, I can manage expenses through the application by adding all expenditure details(bills etc).	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can receive expense and budget reports on request.	I can view reports.	Medium	Sprint-3
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access the application.	High	Sprint-2
	Dashboard	USN_6	As a user, I can observe live automated analysis metrics on spending habits via visualisations and spending goals list in an understandable format.	I can navigate to other elements of the application from the Dashboard.	High	Sprint-4
Customer Care Executive		USN_7	The customer care executive can view and respond to customer queries and work to fix discrepancies posed by the customers	Provide reliable support to customers at all times.	Medium	Sprint-4
Administrator		USN_8	Admins can update the application, perform necessary fixes and execute timely updates.	Admin support for upgrades and bug fixes.	Medium	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.a Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration and Setup	USN-1	As a new user, I want to see how this product will provide value and see where I can register Also setting up the project environment	20	Low	Aarthi Iyer
Sprint-2	Sign-Up	USN-2	As a user, I want to register using my email ID and get a confirmation about account creation	5	High	Soumik Rakshit
Sprint-2	Social Sign-Up	USN-3	As a user, I want the option to sign up easily using social logins [Google OAuth]	5	Low	Shruthi Muthu
Sprint-2	User information - Onboarding	USN-4	As a user, I want to update my information with respect to spending capacity, budget and savings goal	10	High	Aarthi Iyer, Soumik Rakshit
Sprint-3		USN-5	As a user, I want to be able to update my daily expenses	10	High	Shruthi Muthu, Barathvaraj
	Interactive Dashboard using JS	USN-6	As a user, I want to be able to view visually, my spending behaviour and get insights on how to do it better	10	High	Shruthi Muthu, Aarthi Iyer
Sprint-4	Email Reminders using SendGrid	USN-7	As a user, I want to receive reminders about my activities on the web app	3	Medium	Aditya V

Sprint-4	Watson Assistant Chatbot	USN-8	As a user, I want immediate query resolution	10	Medium	Shruthi Muthu
Sprint-4	Profile	USN-9	As a user, I want to be able to view and update my personal details from my profile page	7	Medium	Aarthi Iyer, Aditya V
Sprint-5	Forgot Password	USN-10	As a user, I want to be able to reset my password in case I forget	4	Medium	Soumik Rakshit, Barathvaraj
Sprint 5	IBM DB2	USN-11	Linking database with dashboard	4	High	Soumik Rakshit
	Integration	USN-12	Integrating frontend and backend	2	High	Soumik Rakshit, Aditya V
	Docker	USN-13	Creating Docker image of web app	3	High	Soumik Rakshit
	Cloud Registry	USN-14	Uploading docker image to IBM cloud registry	3	High	Soumik Rakshit, Shruthi Muthu
	Kubernetes	USN-15	Creating container using docker and hosting the webapp	3	High	Soumik Rakshit, Aarthi Iyer
	Exposing Deployment	USN-16	Exposing IP/Ports for the site	1	Medium	Soumik Rakshit

6.b Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Status
Sprint-1	20	7 Days	20 Oct 2022	27 Oct 2022	Completed
Sprint-2	20	7 Days	27 Nov 2022	4 Nov 2022	Completed
Sprint-3	20	7 Days	4 Nov 2022	11 Nov 2022	Completed
Sprint-4	20	7 Days	11 Nov 2022	18 Nov 2022	Completed

VELOCITY: SPRINT - 1

Sprint duration = 5 days
Velocity of team = 20 points

Average Velocity (AV) = $\frac{\text{Velocity}}{\text{Sprint duration}}$

AV = 20/5 = 4

Average Velocity = 4

VELOCITY: Sprint 1 - 4

Sprint duration = 20 days
Velocity of team = 80 points

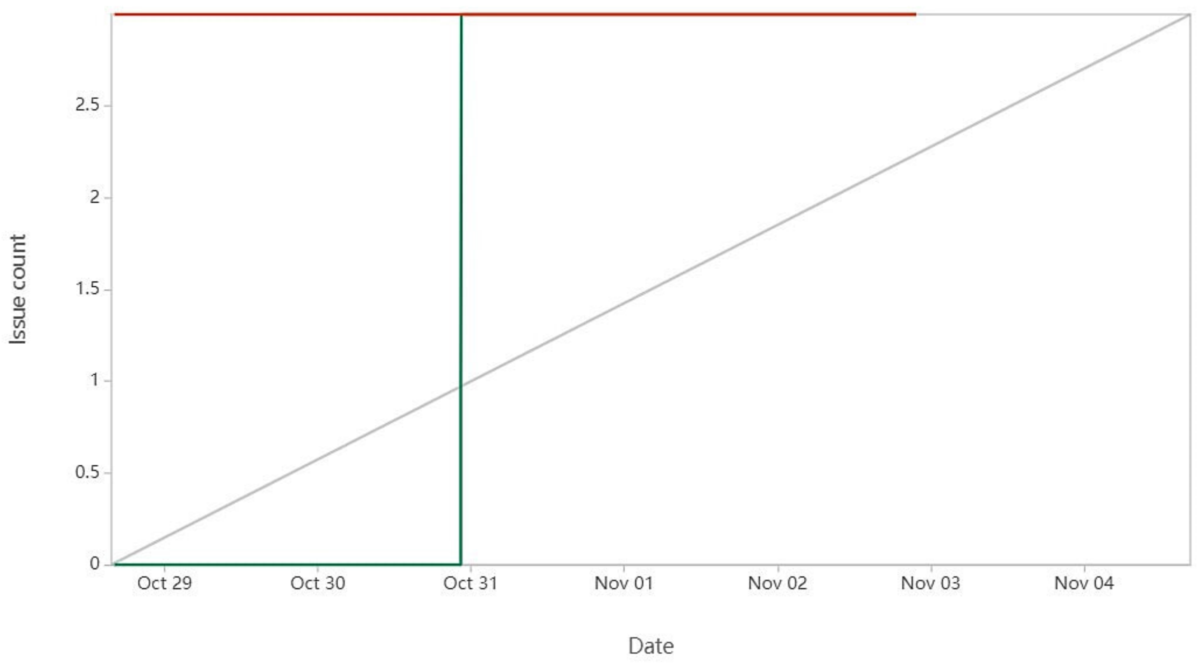
Average Velocity (AV) = $\frac{\text{Velocity}}{\text{Sprint duration}}$

AV = 80/20 = 4

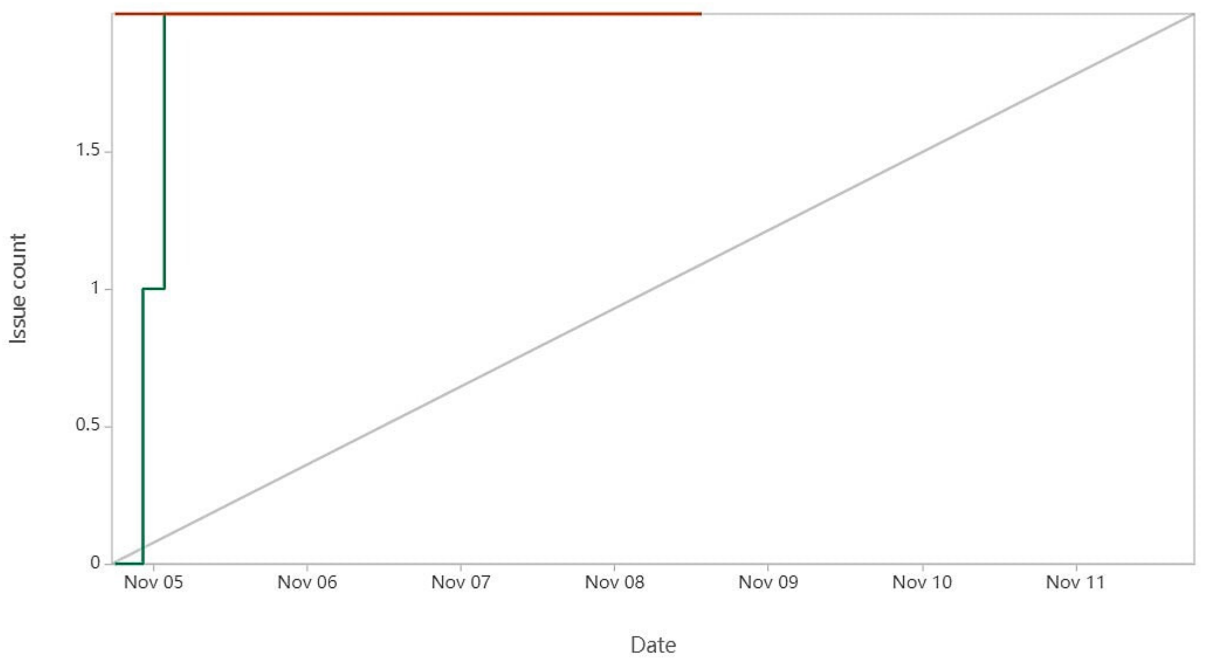
Total Average Velocity = 4

6.c Reports from JIRA

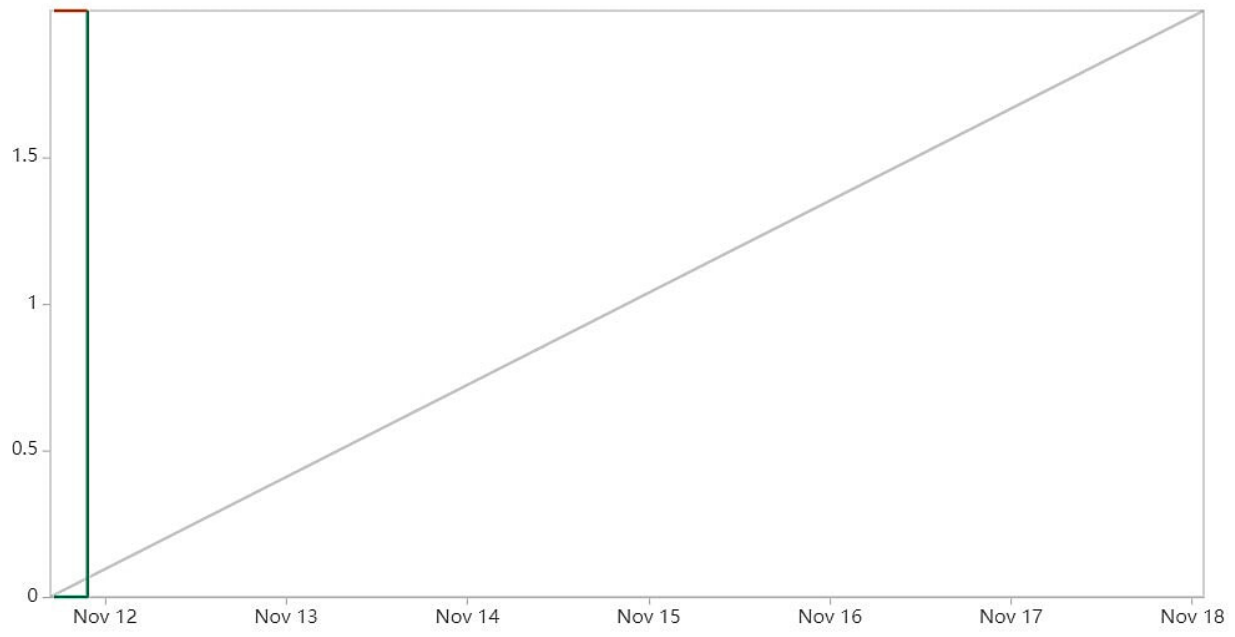
1. BURNUP REPORT - SPRINT 1



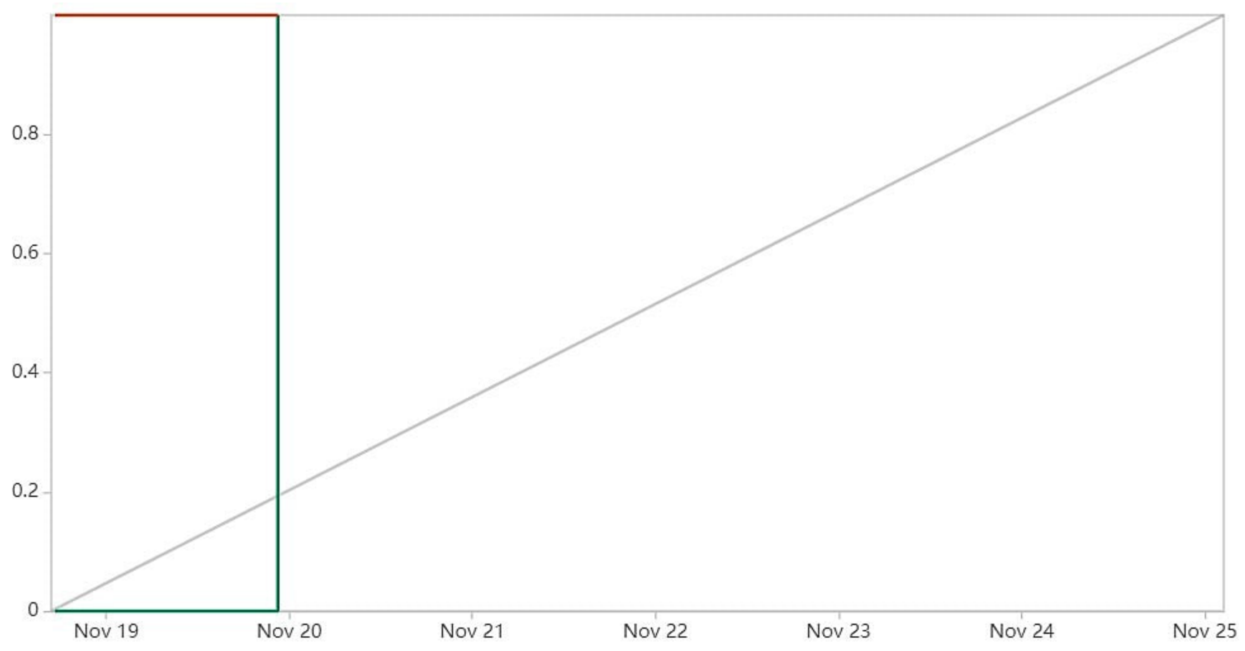
2. BURNUP REPORT - SPRINT 2



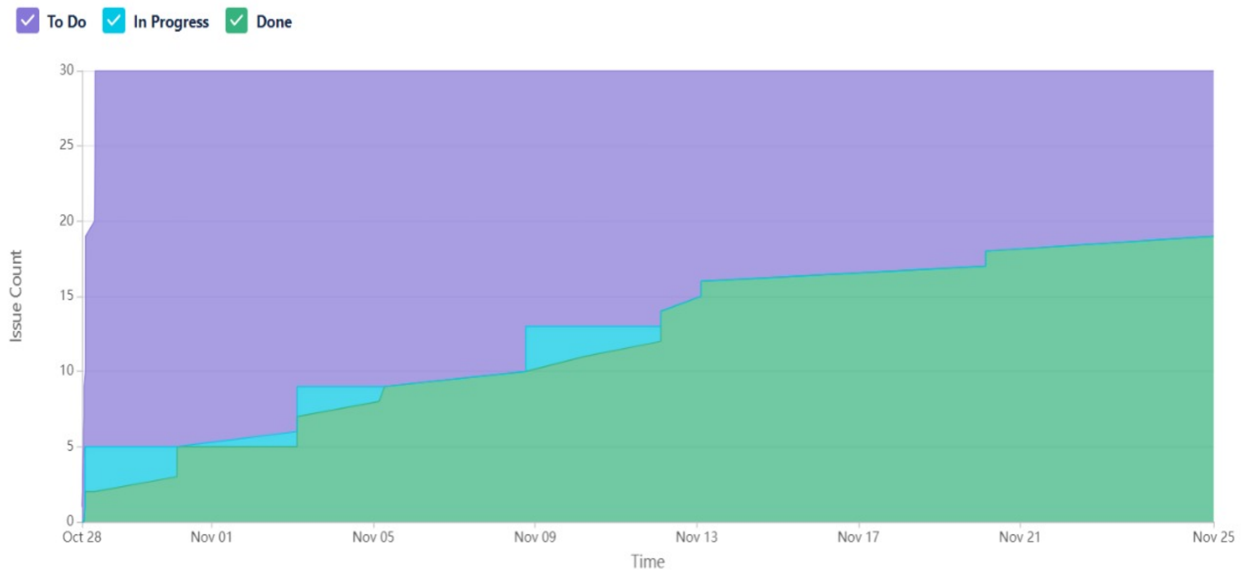
3. BURNUP REPORT - SPRINT 3



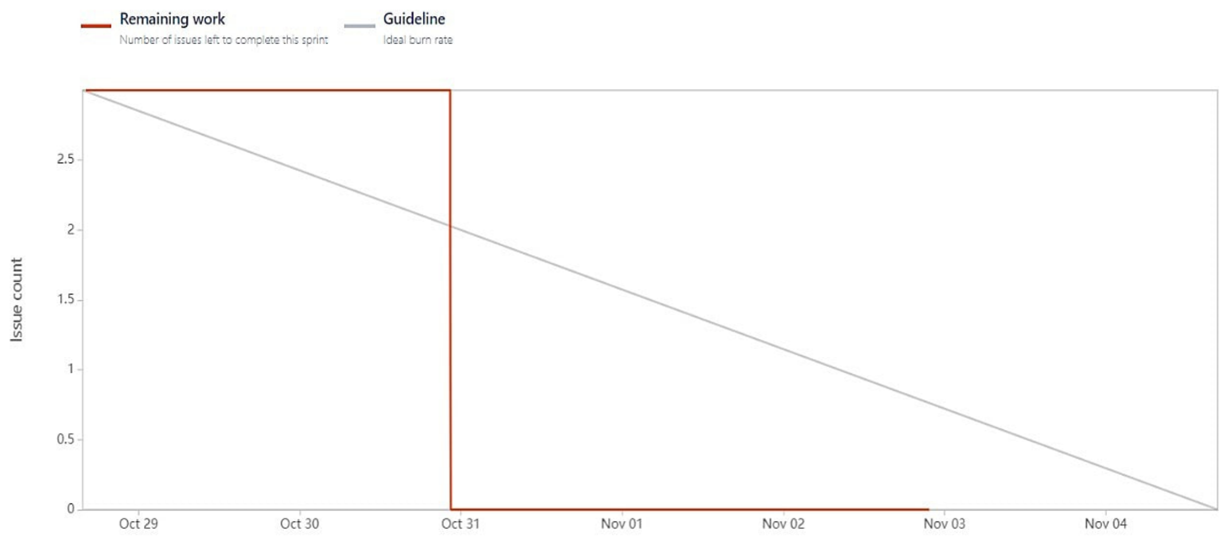
4. BURNUP REPORT - SPRINT 4



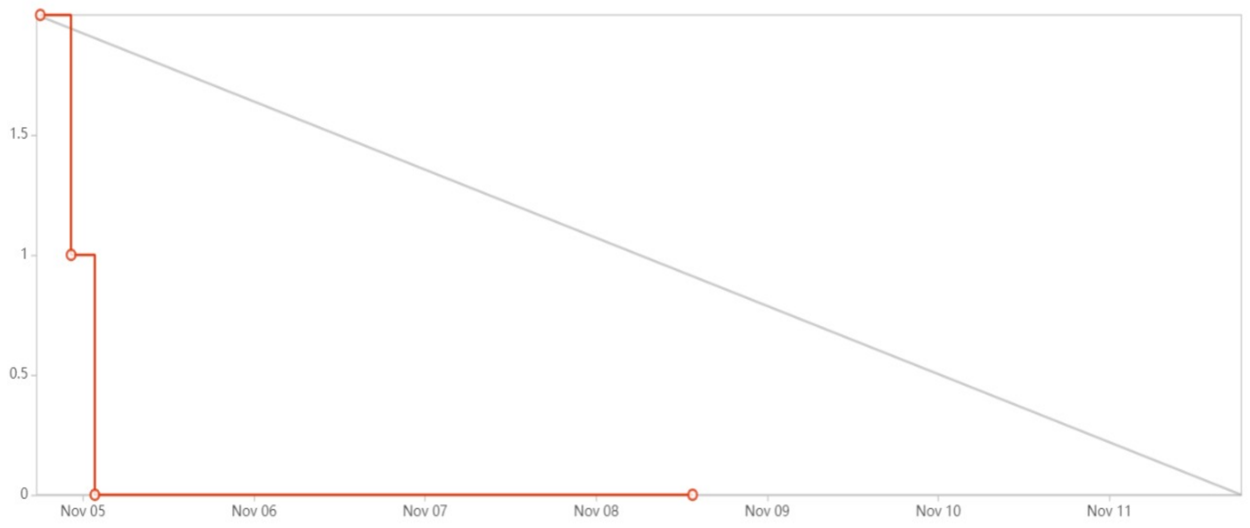
5. CUMMULATIVE FLOW DIAGRAM



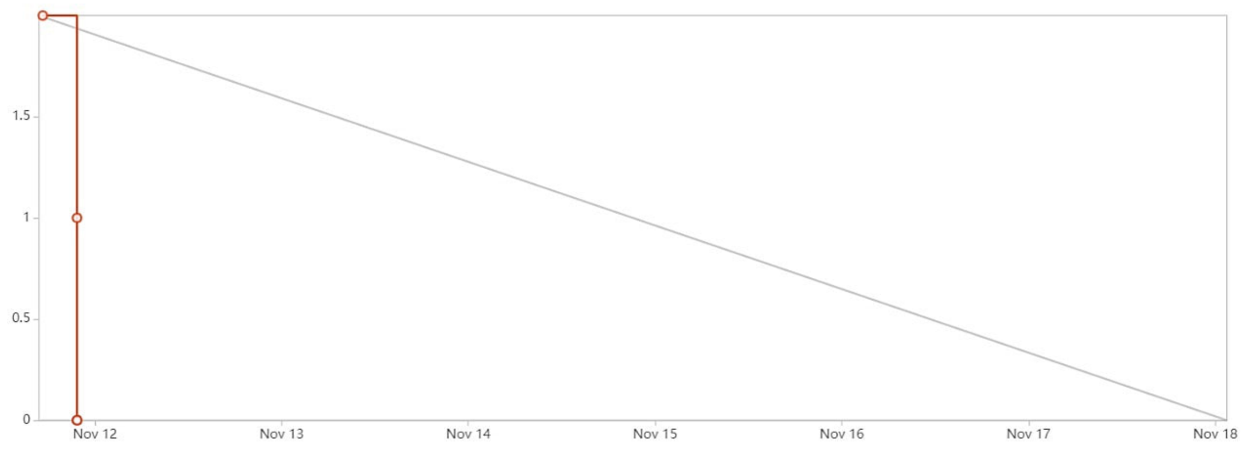
6. BURNDOWN REPORT - SPRINT 1



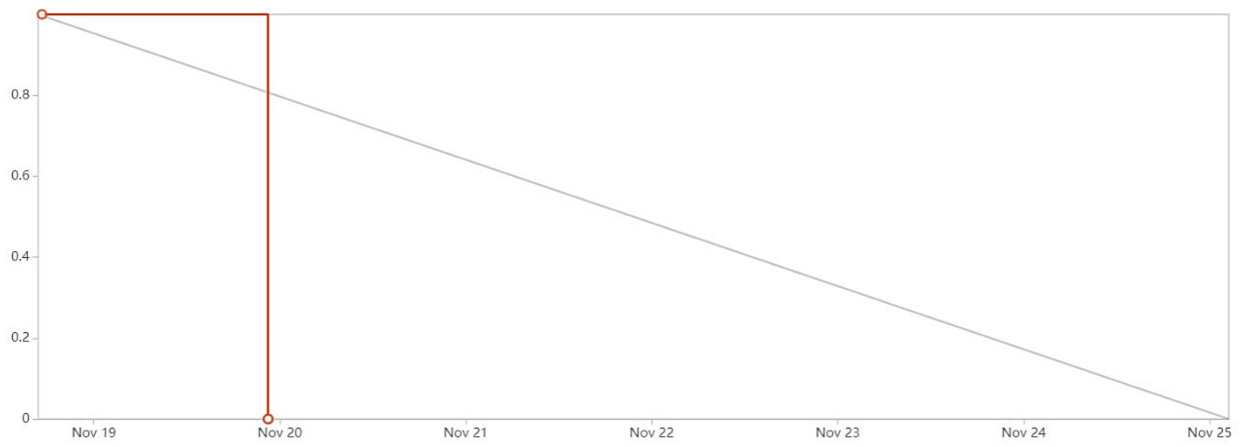
7. BURNDOWN REPORT - SPRINT 2



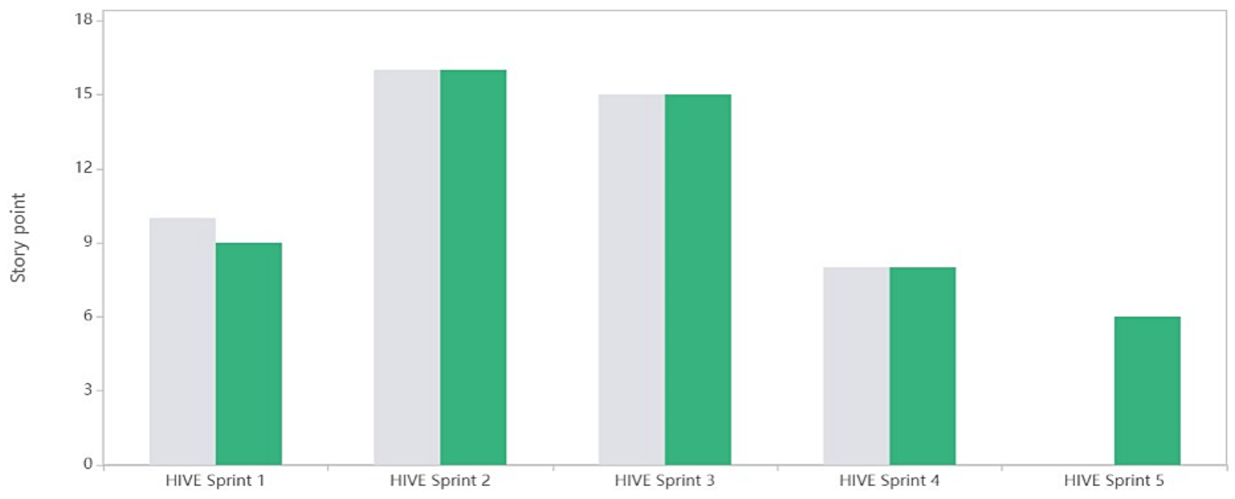
8. BURNDOWN REPORT - SPRINT 3



9. BURNDOWN REPORT - SPRINT 4



10. VELOCITY REPORT



7. CODING & SOLUTIONING

7.a Feature 1

Login and Singup

```
@app.route('/signup', methods=['GET', 'POST'])
def signup():
   if 'logged_in' in session and session['logged_in'] == True:
      flash('You are already logged in', 'info')
      return redirect(url_for('addTransactions'))
   if request.method == 'POST':
      name = request.form.get('name')
      email = request.form.get('email')
      username = request.form.get('username')
      passw = request.form.get('psw')
      rep_pass = request.form.get('psw-repeat')
      if(passw != rep_pass):
         flash('Confirm password doesnot match','error')
         return redirect(url_for('signup'))
      else:
         conn = connection.establish()
      if(connection.useremail_check(conn,email)==False):
         flash('User with email already exists, try again', 'warning')
         return redirect(url_for('signup'))
      else:
         session['usermail'] = email
         session['pwd'] = passw
         connection.insertuser(conn,name,email,username,passw)
         flash('You are now registered', 'success')
         return redirect(url_for('question'))
   else:
      return render_template('register.html')


@app.route('/login', methods=['GET', 'POST'])
def login():
   if 'logged_in' in session and session['logged_in'] == True:
      flash('You are already logged in', 'info')
      return redirect(url_for('addTransactions'))
```

```
        if request.method == 'POST' :
            email = request.form.get('email')
            password_input = request.form.get('psw')
            conn = connection.establish()
            res = connection.user_check(conn,email,password_input)
            print('Hello')x
            if(res!=False):
                print('Login Success')
                session['logged_in'] = True
                session['usermail'] = email
                session['userID'] = res['ID']
                flash('Login Successfull','success')
                return redirect(url_for('addTransactions'))
            else:
                print('Login Failure')
                flash('Incorrect Username/Password','error')
                return redirect(url_for('login'))
        else:
            return render_template('login.html')

    def is_logged_in(f):
        @wraps(f)
        def wrap(*args, **kwargs):
            if 'logged_in' in session:
                return f(*args, **kwargs)
            else:
                flash('Please login', 'info')
                return redirect(url_for('login'))
        return wrap
```

Templates:

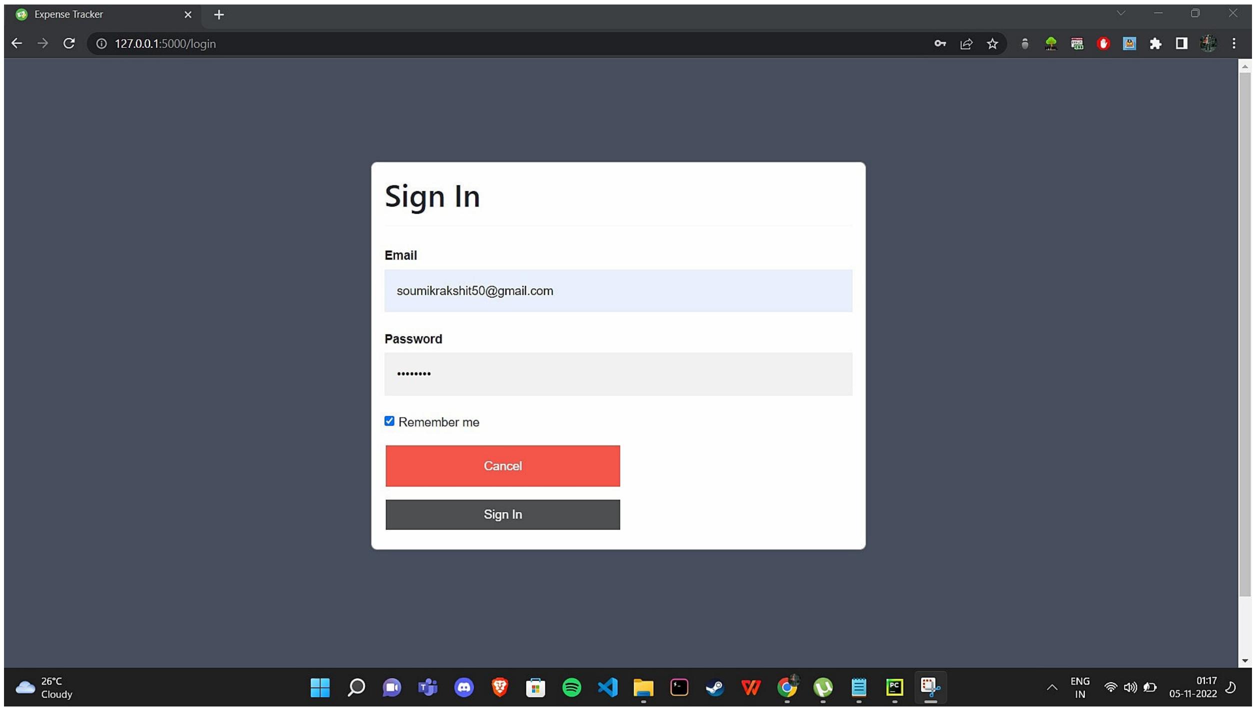
Login :

```
<div class="modal">
  <form class="modal-content" action="/login" method="post">
    <div class="container">
      <h1>Sign In</h1>
      <hr>
      <label ><b>Email</b></label>
      <input type="text" placeholder="Enter Email" name="email" required>
```

```
   <label><b>Password</b></label>
   <input type="password" placeholder="Enter Password" name="psw" required>
   <label>
     <input type="checkbox" checked="checked" name="remember" style="margin-
        bottom:15px"> Remember me
   </label>
   <h5 class="forgot"><a href="{{ url_for('reset_request') }}" class="green-text
        link">Forgot Password?</a></h5>
   <div class="clearfix">
     <a href="/" style=" text-decoration=none;" ><button type="button"
         class="cancelbtn">Cancel</button></a>
     <a href="/login" style=" text-decoration=none;"></a><button type="submit"
        class="signupbtn">Sign In</button></a>
   </div>
  </div>
 </form>
</div>
```

Signup:

```
<div class="modal">
 <form class="modal-content" action="/signup" method="post">
  <div class="container">
   <h1>Sign Up</h1>
   <p>Please fill in this form to create an account.</p>
   <hr>
   <label><b>Name</b></label>
   <input type="text" placeholder="Enter Name" name="name" required>

   <label ><b>Email</b></label>
   <input type="text" placeholder="Enter Email" name="email" required>

   <label><b>Username</b></label>
   <input type="text" placeholder="Enter Username" name="username" required>

   <label><b>Password</b></label>
   <input type="password" placeholder="Enter Password" name="psw" required>

   <label ><b>Confirm Password</b></label>
   <input type="password" placeholder="Confirm Password" name="psw-repeat" required>
   <label>
    <input type="checkbox" checked="checked" name="remember" style="margin-
       bottom:15px"> Remember me
   </label>
   <p>By creating an account you agree to our <a href="#" style="color:dodgerblue">Terms
   & Privacy</a>.</p>
   <div class="clearfix">
    <a href="/" style=" text-decoration=none;" ><button type="button"
        class="cancelbtn">Cancel</button></a>
    <a href="/signup" style=" text-decoration=none;"><button type="submit"
       class="signupbtn">Sign Up</button></a>
   </div>
  </div>
```



Sign In

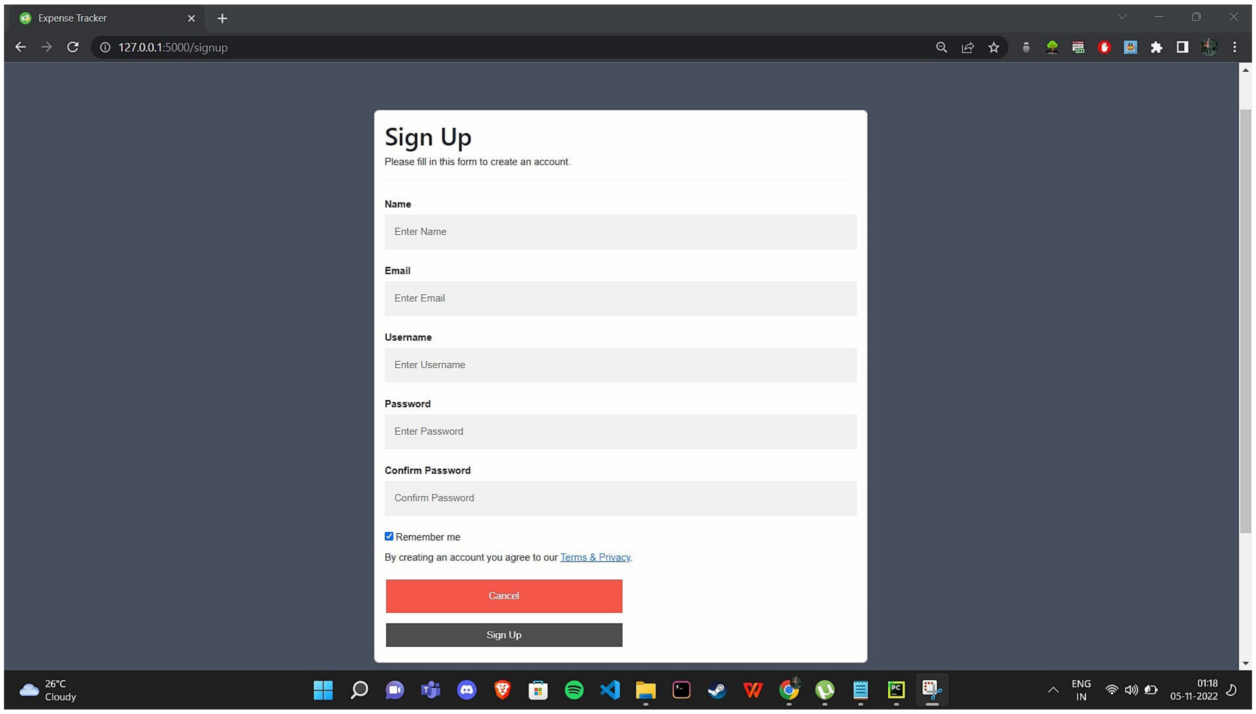
Email

soumikrakshit50@gmail.com

Password

••••••••

☑ Remember me



Sign Up

Please fill in this form to create an account.

Name

Enter Name

Email

Enter Email

Username

Enter Username

Password

Enter Password

Confirm Password

Confirm Password

☑ Remember me

By creating an account you agree to our Terms & Privacy.

7.b Feature 2

Expense Tracker page - To add expenses
Charts - To view statistics

```
@app.route('/addTransactions', methods=['GET', 'POST'])
def addTransactions():
    if request.method == 'POST':
        amount = request.form['amount']
        description = request.form['description']
        category = request.form['category']
        conn = connection.establish()
        connection.inserttransac(conn,session['userID'],amount,description,category)
        flash('Transaction Successfully Recorded', 'success')
        return redirect(url_for('addTransactions'))
    else:
        conn = connection.establish()
        res = connection.gettotalsum(conn,session['userID'])
        total = res['SUM']
        budget = connection.get_budget(conn,session['userID'])
        goal = connection.get_savings(conn,session['userID'])
        if total!=None and total>int(budget['POCKETMONEY']):
            subject = "Exceeded Montly Budget"
            html_content = "You have exceeded your monthly budget, do not spend any more
             money this month!"
            sendEmail(API,from_email,session.get('usermail'),subject,html_content)
        elif total!=None and int(budget['POCKETMONEY'])-total<int(goal['MONTHLYGOAL']):
            subject = "Savings Goal Affected"
            html_content = "You cannot achieve the target savings goal for the month as you
             have exceeded your expenses. Try spending carefully next time!"
            sendEmail(API, from_email, session.get('usermail'), subject, html_content)
        elif total!=None and int(budget['POCKETMONEY'])-total<=200:
            subject = "About to exceed your Monthly Budget"
            html_content = "Use your money carefully as you have only '{}' Rs remaining from your
             monthly budget".format(budget-total)
            sendEmail(API, from_email, session.get('usermail'), subject, html_content)
        dict= connection.getalltransac(conn,session['userID'])
        if len(dict)!=0:
            return render_template('addTransactions.html', totalExpenses=total,
             transactions=dict)
        else:
            return render_template('addTransactions.html', result=dict)
    return render_template('addTransactions.html')

class TransactionForm(Form):
    amount = IntegerField('Amount', validators=[DataRequired()])
    description = StringField('Description', [validators.Length(min=1)])

@app.route('/editCurrentMonthTransaction/<string:id>', methods=['GET', 'POST'])
@is_logged_in
def editCurrentMonthTransaction(id):
    form = TransactionForm(request.form)
    form = TransactionForm(request.form)

    if request.method == 'POST' and form.validate():
        amount = request.form['amount']
        description = request.form['description']
        conn = connection.establish()
        connection.updateTrans(conn,id,amount,description)
        flash('Transaction Updated', 'success')
        return redirect(url_for('addTransactions'))
    return render_template('editTransaction.html', form=form)
```

```
@app.route('/category')
def createBarCharts():
    conn = connection.establish()
    res = connection.gettotalsum(conn, session['userID'])
    total = res['SUM']
    dict = connection.getalltransac(conn, session['userID'])
    if len(dict) > 0:
        values = []
        labels = []
        print(dict)
        for transaction in dict:
            values.append(transaction['amt'])
            labels.append(transaction['cat'])
        print(labels)
        print(values)
        fig = go.Figure(data=[go.Pie(labels=labels, values=values)])
        fig.update_traces(textinfo='label+value', hoverinfo='percent')
        fig.update_layout(title_text='Category Wise Pie Chart For Current Year')
        #fig.show()
    return render_template('chart.html',context={'labels':labels,'value':values})


@app.route('/monthly_bar')
def monthlyBar():
    conn = connection.establish()
    res = connection.gettotalsum(conn, session['userID'])
    total = res['SUM']
    dict = connection.getalltransac(conn, session['userID'])
    if len(dict) > 0:
        year = []
        value = []
        print(dict)
d={'January':0,'February':0,'March':0,'April':0,'May':0,'June':0,'July':0,'August':0,'September':0,'October':0,'
November':0,'December':0}
        for transaction in dict:
            d[transaction['date'].strftime("%B")]+=transaction['amt']
        print(d)
        for t,a in d.items():
            year.append(t)
            value.append(a)
```

```
print(year)
print(value)
fig = go.Figure([go.Bar(x=year, y=value)])
fig.update_layout(title_text='Monthly Bar Chart For Current Year')
#fig.show()
#cur.close()
return render_template('chart1.html',context={'labels':year,'value':value})
```

HTML Templates

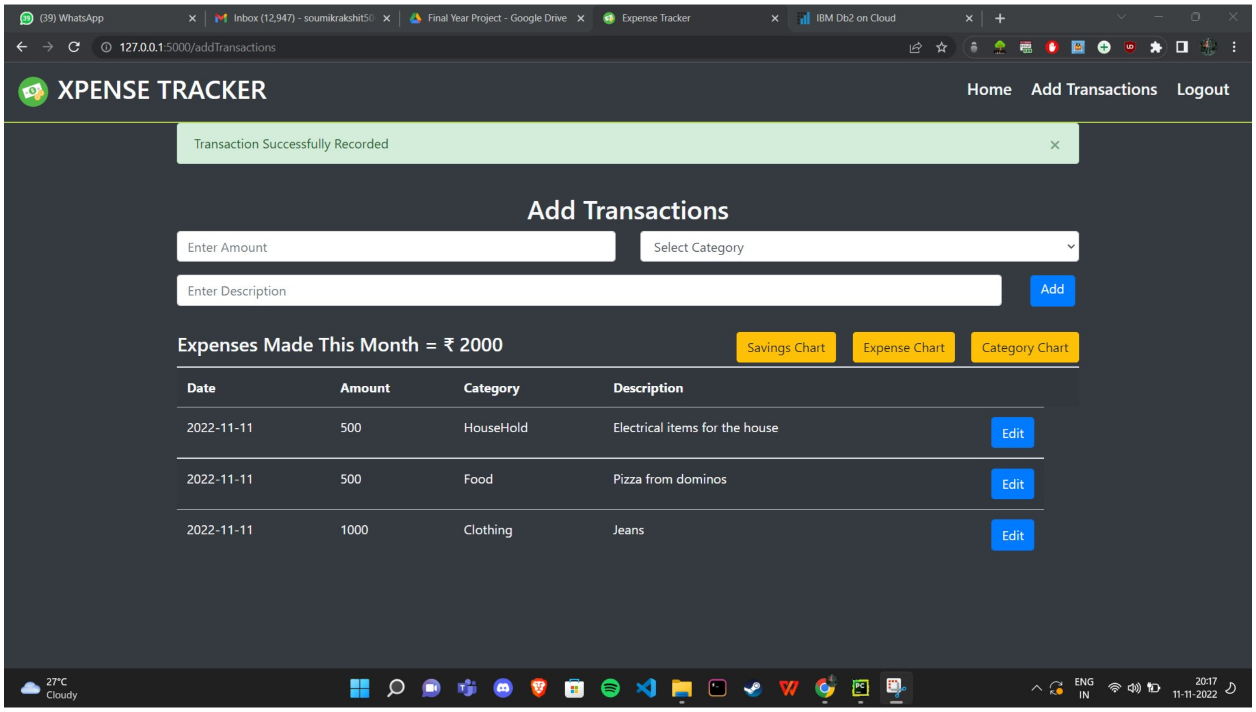
AddTransaction:

```
<div class="add">
<h2 class="text-light">Add Transactions</h2>
{% from "include/formhelpers.html" import render_field %}
<form class="form" method="POST" action="">
        <div class="form-group row">
                <div class="form-group col-md-6">
                        <input
                                type="number"
                                placeholder="Enter Amount"
                                class="form-control"
                                name="amount"
                                value="{{request.form.amount}}"
                        />
                </div>
                <div class="form-group category col-md-6">
                        <select name="category" id="category" class="form-control">
                                <option value="Miscellaneous" selected="selected"
                                disabled>Select Category</option>
                                <option value="Miscellaneous">Miscellaneous</option>
                                <option value="Food">Food</option>
                                <option value="Transportation">Transportation</option>
                                <option value="Groceries">Groceries</option>
                                <option value="Clothing">Clothing</option>
                                <option value="HouseHold">HouseHold</option>
                                <option value="Rent">Rent</option>
                                <option value="Bills and Taxes">Bills and Taxes</option>
                                <option value="Vacations">Vacations</option>
                        </select>
                </div>
                <div class="form-group col-md-10 col-lg-11">
                <input
                        type="text"
                        placeholder="Enter Description"
                        name="description"
                        class="form-control"
                        value="{{request.form.description}}"
                />
            </div>
            <div class="form-group col-md-2 col-lg-1 btn">
                <button type="submit" class="btn btn-primary">Add</button>
            </div>
        </div>
</form>
{% if result != 0%}
<div class="current-month">
        <h4 class="text-light float-left">
            Expenses Made This Month = <span class="green-text expense">₹
                {{totalExpenses}}</span>
        </h4>
        <p class="text-light float-left swipe">Swipe to Edit/Delete</p>
        <a href="/category"  class="btn btn-warning pie_chart float-right">Category
            Chart</a>
        <a href="/monthly_bar" class="btn btn-warning line_chart float-right">Expense
            Chart</a>
        <a href="/monthly_savings" style="margin-right:21px" class="btn btn-warning
    pie_chart float-right"> Savings Chart</a>
</div>
<div class="table-responsive">
        <table class="table table-striped text-light">
            <tr>
                <th>Date</th>
                <th>Amount</th>
                <th>Category</th>
                <th>Description</th>
                <th></th>
                <th></th>
            </tr>
            {% for transaction in transactions %}
            <tr>
                <td>{{transaction.date}}</td>
                <td>{{transaction.amt}}</td>
```

```
                        <td>{{transaction.cat}}</td>
                        <td>{{transaction.des}}</td>
                        <td><a href="editCurrentMonthTransaction/{{transaction.id}}"
                            class="btn btn-primary pull-right">Edit</a></td>
                    </tr>
                    {% endfor %}
                </table>
        </div>
</div>
```

Chart:

```
<h2 style="text-align:center; " class="text-light">Category wise Expense
            Chart</h2>
<br><br><br>
<input type="hidden" id="years" labels="{{context.labels}}">
<input type="hidden" id="values" values="{{context.value}}">
<div style="width:100;height:100%">
  <canvas id="myChart"></canvas>
</div>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>
let years = document.getElementById('years').getAttribute('labels')
years = years.slice(1,years.length-1).split(",")
let values = document.getElementById('values').getAttribute('values')
values = values.slice(1,values.length-1).split(",")
console.log(years)
const ctx = document.getElementById('myChart').getContext('2d');
var barColors = [];
for (let index = 0; index < years.length; index++) {
 const r = Math.floor(Math.random()*255);
 const g = Math.floor(Math.random()*255);
 const b = Math.floor(Math.random()*255);
 barColors.push('rgba('+r+','+g+','+b+', 0.4)')
}
const data = {
 labels: years,
 datasets: [{
  label: 'My Expenses',
  data: values,
  backgroundColor: barColors,
  hoverOffset: 4
 }]};
Chart.defaults.color = "white";
var myChart = new Chart(ctx, {
   type: 'pie',
   data: data,
    options: {
   legend: {
     display: false
   },
   maintainAspectRatio: false,
 }
});
</script>
```



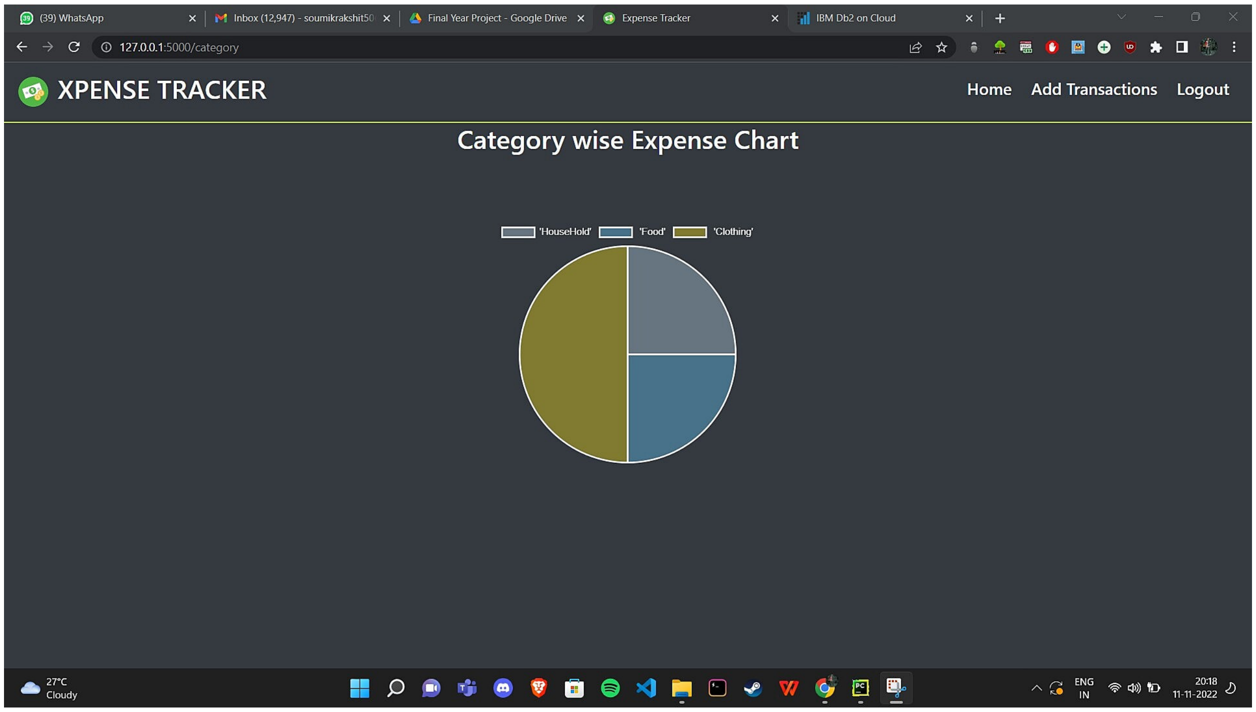
XPENSE TRACKER

Transaction Successfully Recorded

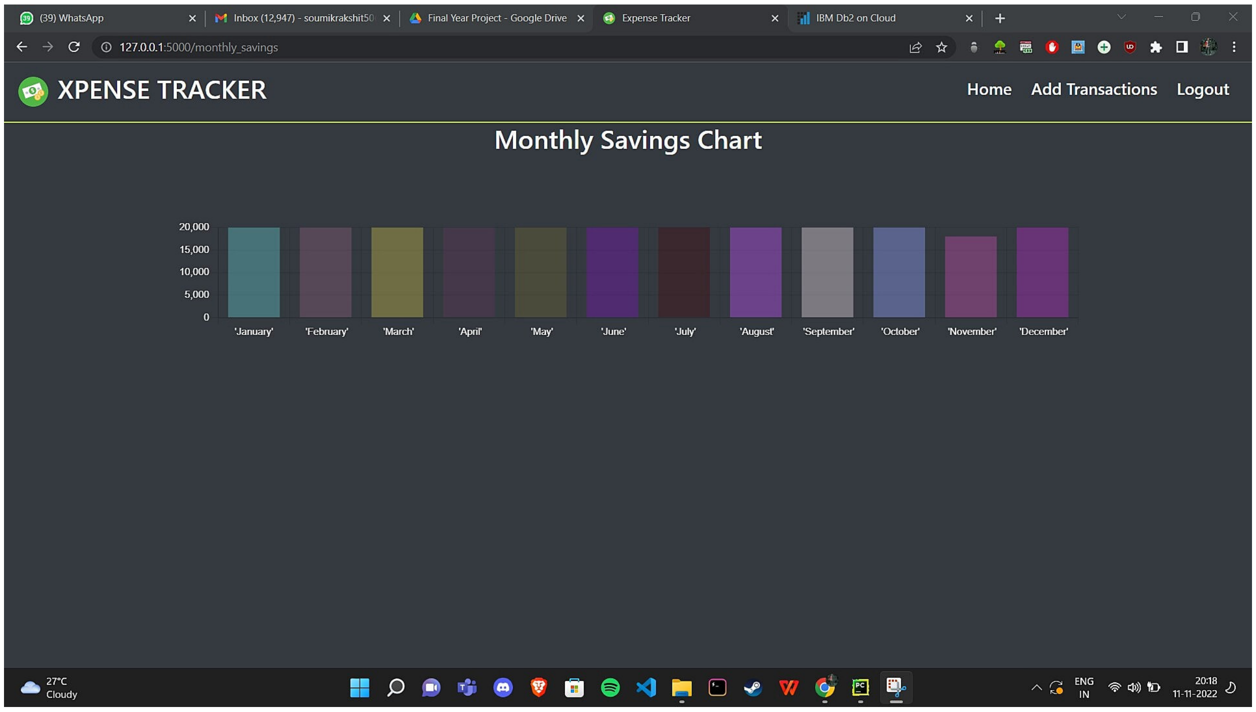
Add Transactions

Expenses Made This Month = ₹ 2000

Date	Amount	Category	Description
2022-11-11	500	HouseHold	Electrical items for the house
2022-11-11	500	Food	Pizza from dominos
2022-11-11	1000	Clothing	Jeans



Category wise Expense Chart



Monthly Savings Chart

7.c Feature 3

ChatBot - Watson Assistant
Forgot Password

Watson Assistant

```
<script>
  window.watsonAssistantChatOptions = {
    integrationID: "715f07f7-4ee6-489c-a26d-619cd0c1ab2e", // The ID of this integration.
    region: "au-syd", // The region your integration is hosted in.
    serviceInstanceID: "e2dd9860-6728-4bdd-82cf-681b659835ef", // The ID of your
service instance.
    onLoad: function(instance) { instance.render(); }
  };
  setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
  });
</script>
```

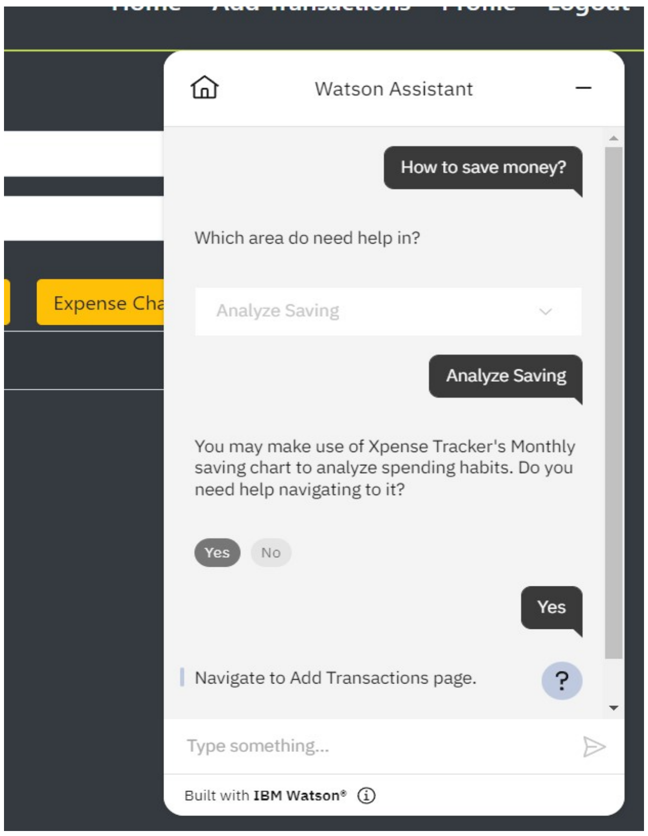
Forgot Passowd:

```
<div class="signUp container text-white">
  {% from "include/formhelpers.html" import render_field %}
  <h1 class="text-center green-text"><u>FORGOT PASSWORD</u></h1>
  <div class="text-center">
    <p>Enter the registered email address of your account and we shall send you an
email.</p>
    <p>The email will consist of a link. Click on the link to reset your password.</p>
    <small>The link will expire in 30 minutes</small>
  </div>
      <form action="" method="post">
            <h4 class="green-text form-group">
                  {{render_field(form.email, class_="form-control")}}
            </h4>
            <p class="text-center"><input class="btn btn-info" type="submit"
            value="Submit" /></p>
      </form></div>
```

app.py

```
class RequestResetForm(Form):
   email = EmailField('Email address', [validators.DataRequired(), validators.Email()])

@app.route("/reset_request", methods=['GET', 'POST'])
def reset_request():
   if 'logged_in' in session and session['logged_in'] == True:
      flash('You are already logged in', 'info')
      return redirect(url_for('index'))
   form = RequestResetForm(request.form)
   if request.method == 'POST' and form.validate():
      email = form.email.data
      conn = connection.establish()
      res = connection.get_useralId(conn,email)
      if res == False:
         flash('There is no account with that email. You must register first.', 'warning')
         return redirect(url_for('signup'))
      else:
         user_id = res['ID']
         user_email = res['EMAIL']
         s = Serializer(app.secret_key, 1800)
         token = s.dumps({'user_id': user_id}).decode('utf-8')
         sub = 'Password Reset Request'
         html_content = f'''To reset your password, visit the following link:
{url_for('reset_token', token=token, _external=True)}
If you did not make password reset request then simply ignore this email and no
changes will be made.Note:This link is valid only for 30 mins from the time you
requested a password change request. '''sendEmail(API,'dragodark223@gmail.com',
user_email, sub, html_content)
         flash('An email has been sent with instructions to reset your password.', 'info')
         return redirect(url_for('login'))
   return render_template('reset_request.html', form=form)

class ResetPasswordForm(Form):
   password = PasswordField('Password', [
      validators.DataRequired(),
      validators.EqualTo('confirm', message='Passwords do not match')
   ])
   confirm = PasswordField('Confirm Password')
```

```
@app.route("/reset_password/<token>", methods=['GET', 'POST'])
def reset_token(token):
    if 'logged_in' in session and session['logged_in'] == True:
        flash('You are already logged in', 'info')
        return redirect(url_for('index'))
    s = Serializer(app.secret_key)
    try:
        user_id = s.loads(token)['user_id']
    except:
        flash('That is an invalid or expired token', 'warning')
        return redirect(url_for('reset_request'))
    form = ResetPasswordForm(request.form)
    if request.method == 'POST' and form.validate():
        password = str(form.password.data)
        conn = connection.establish()
        connection.reset_pass(conn,password,user_id)
        flash('Your password has been updated! You are now able to log in', 'success')
        return redirect(url_for('login'))
    return render_template('reset_token.html', title='Reset Password', form=form)
```



How to save money?

Which area do need help in?

Analyze Saving

Analyze Saving

You may make use of Xpense Tracker's Monthly saving chart to analyze spending habits. Do you need help navigating to it?

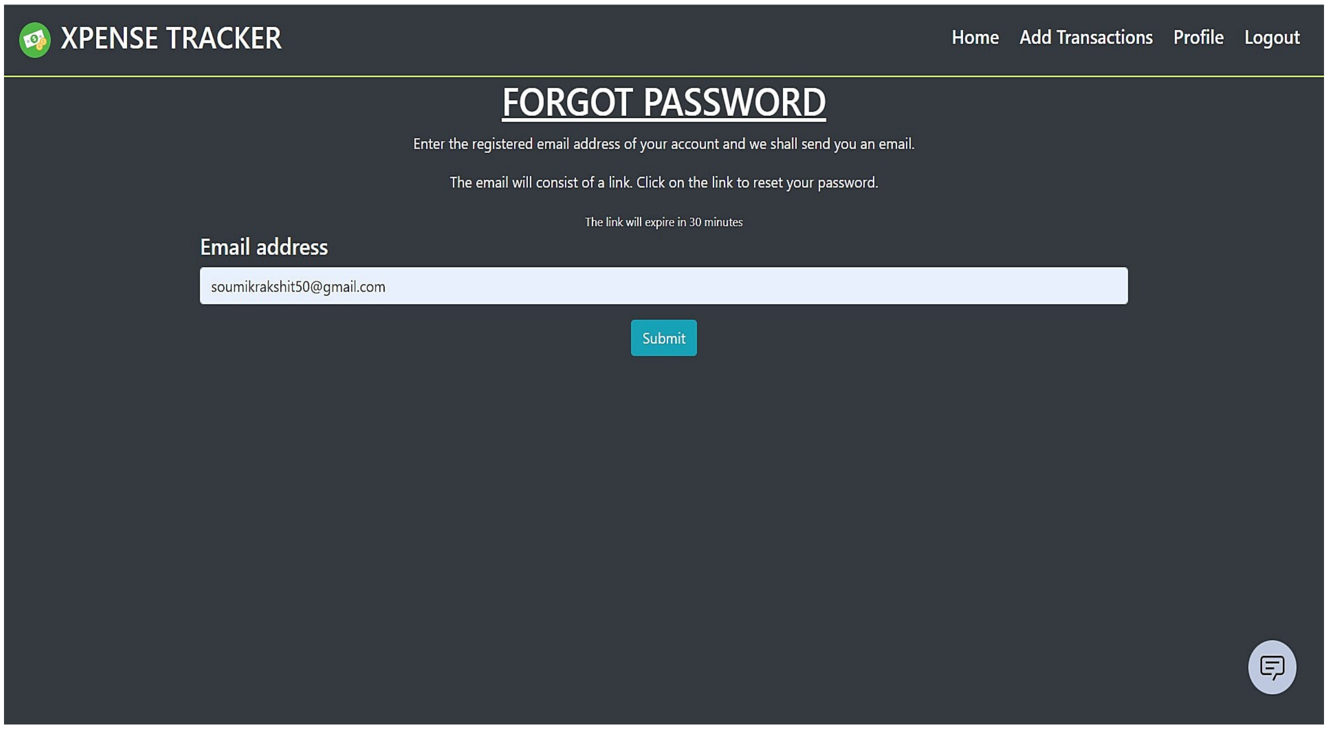
Yes No

Yes

Navigate to Add Transactions page.

Type something...

Built with IBM Watson®


XPENSE TRACKER

FORGOT PASSWORD

Enter the registered email address of your account and we shall send you an email.

The email will consist of a link. Click on the link to reset your password.

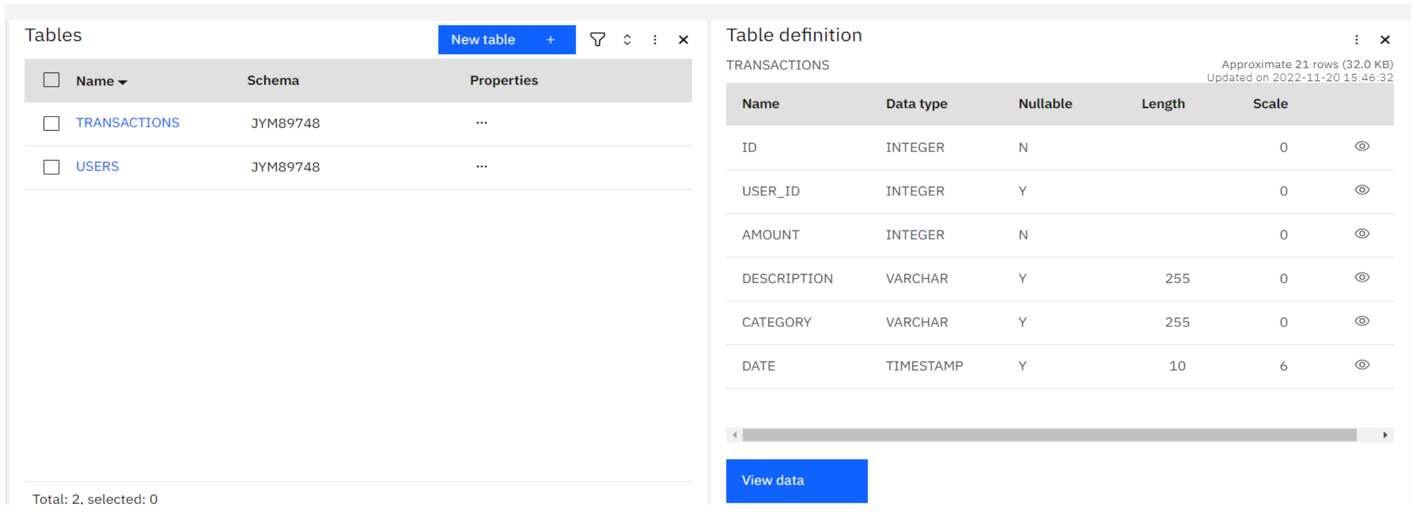
The link will expire in 30 minutes

Email address: soumikrakshit50@gmail.com

7.d Database Schema

```
import ibm_db as db
from datetime import datetime

dbname = "bludb"
hostname = "3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"
port = 31498
protocol = "TCPIP"
username = "jym89748"
password = "8fXXBe0fBoZmJKgG"
cert = "DigiCertGlobalRootCA.crt"
```

```
# establish connection
def establish():
    try:
        conn = db.connect(
f"DATABASE={dbname};HOSTNAME={hostname};PORT={port};PROTOCOL={protocol};UI
D={username};PWD={password}; SECURITY=SSL; SSLServerCertificate={cert};",
            "", "")
        print("Connected to database")
        return conn
    except:
        print("Error connecting to database")

# to insert a new user
def insertuser(conn1, name, email, user, passw):
    sql = "INSERT INTO users(name,email,username,password) VALUES
('{}','{}','{}','{}')".format(name, email, user, passw)
    try:
        stmt = db.exec_immediate(conn1, sql)
        print("Number of affected rows: ", db.num_rows(stmt))
    except:
        print("cannot insert user to database")

# to check if user exists with given email
def useremail_check(conn,email):
    sql = "SELECT * FROM users WHERE email='{}' ".format(email)
    stmt = db.exec_immediate(conn, sql)
    results = db.fetch_assoc(stmt)
    if results == False:
        return True
    else: return False

# to check if user exists with given username and password
def user_check(conn,email,passw):
    sql = "SELECT * FROM users WHERE email='{}' AND
password='{}'".format(email,passw)
    stmt = db.exec_immediate(conn, sql)
    results = db.fetch_both(stmt)
    return results

#set basic details of each user
def setuser(conn,money,budget,goal,email,pwd):
    sql = "UPDATE USERS SET(pocketmoney,budget,monthlygoal) = ('{}','{}','{}') WHERE
email='{}' AND password='{}'".format(money,budget,goal,email,pwd)
    try:
        stmt = db.exec_immediate(conn,sql)
        print("Number of affected rows: ", db.num_rows(stmt))
    except:
        print("Error inserting data to database")

#insert a new transaction of a user
def inserttransac(conn,id,amt,des,cat):
    sql = "INSERT INTO TRANSACTIONS(user_id,amount,description,category)
VALUES('{}','{}','{}','{}')".format(id,amt,des,cat)
    try:
        stmt = db.exec_immediate(conn, sql)
        print("Number of affected rows: ", db.num_rows(stmt))
    except:
        print("Error inserting data to database")

#to get the total mount spent for the month
def gettotalsum(conn,id):
    sql = "SELECT SUM(amount) as SUM FROM transactions WHERE MONTH(date) =
MONTH(CURRENT DATE) AND YEAR(date) = YEAR(CURRENT DATE) AND
user_id='{}'".format(id)
    try:
        stmt = db.exec_immediate(conn,sql)
        res = db.fetch_both(stmt)
        return res
    except:
        print("Error while fetching")

#to get all transactions of a user
def getalltransac(conn,id):
    sql = "SELECT * FROM transactions WHERE MONTH(date) = MONTH(CURRENT DATE)
AND YEAR(date) = YEAR(CURRENT DATE) AND user_id='{}' ORDER BY date
DESC".format(id)
    try:
        stmt = db.exec_immediate(conn,sql)
        res = db.fetch_both(stmt)
        if(res==False):
            return []
        else:
            res['DATE'] = res['DATE'].date()
            dict = [{'id':res['ID'], 'date': res['DATE'], 'amt': res['AMOUNT'], 'cat': res['CATEGORY'],
'des':res['DESCRIPTION']}]
            res= db.fetch_both(stmt)
            while(res!=False):
                res['DATE'] = res['DATE'].date()
                dict.append({'id':res['ID'],'date': res['DATE'], 'amt': res['AMOUNT'], 'cat':
res['CATEGORY'], 'des':res['DESCRIPTION']})
                res = db.fetch_both(stmt)
            return dict
    except:
        print("Error while fetching")

#to delete a transaction
def deletetrans(conn,id):
    sql = "DELETE FROM transactions WHERE id='{}'".format(id)
    try:
        stmt = db.exec_immediate(conn, sql)
        print("Number of affected rows: ", db.num_rows(stmt))
    except:
        print("Error deleting data from database")

#to update a transaction
def updateTrans(conn,id,amt,des):
    sql = "UPDATE transactions SET AMOUNT='{}',DESCRIPTION = '{}' WHERE
ID='{}'".format(amt,des,id)
    try:
        stmt = db.exec_immediate(conn, sql)
        print("Number of affected rows: ", db.num_rows(stmt))
    except:
        print("Successfully updated transaction")

#get monthly budget
def get_budget(conn,id):
    sql = "SELECT POCKETMONEY FROM users WHERE ID='{}' ".format(id)
    stmt = db.exec_immediate(conn, sql)
    results = db.fetch_assoc(stmt)
    return results

#get daily budget
def get_savings(conn,id):
    sql = "SELECT MONTHLYGOAL FROM users WHERE ID='{}' ".format(id)
    stmt = db.exec_immediate(conn, sql)
    results = db.fetch_assoc(stmt)
    return results


#get user details
def get_userdetails(conn,id):
    sql = "SELECT * FROM users WHERE ID='{}' ".format(id)
    stmt = db.exec_immediate(conn, sql)
    results = db.fetch_assoc(stmt)
    return results

#get user details
def get_useralld(conn,email):
    sql = "SELECT * FROM users WHERE email='{}' ".format(email)
    stmt = db.exec_immediate(conn, sql)
    results = db.fetch_assoc(stmt)
    return results

# reset password
def reset_pass(conn,passw,id):
    sql = "UPDATE users SET password = '{}' WHERE id = '{}' ".format(passw, id)
    try:
        stmt = db.exec_immediate(conn, sql)
        print("Number of affected rows: ", db.num_rows(stmt))
    except:
        print("Not able to fetch")
```



	Name	Schema	Properties
☐	TRANSACTIONS	JYM89748	...
☐	USERS	JYM89748	...

Total: 2, selected: 0

TRANSACTIONS

Approximate 21 rows (32.0 KB)
Updated on 2022-11-20 15:46:32

Name	Data type	Nullable	Length	Scale
ID	INTEGER	N		0
USER_ID	INTEGER	Y		0
AMOUNT	INTEGER	N		0
DESCRIPTION	VARCHAR	Y	255	0
CATEGORY	VARCHAR	Y	255	0
DATE	TIMESTAMP	Y	10	6

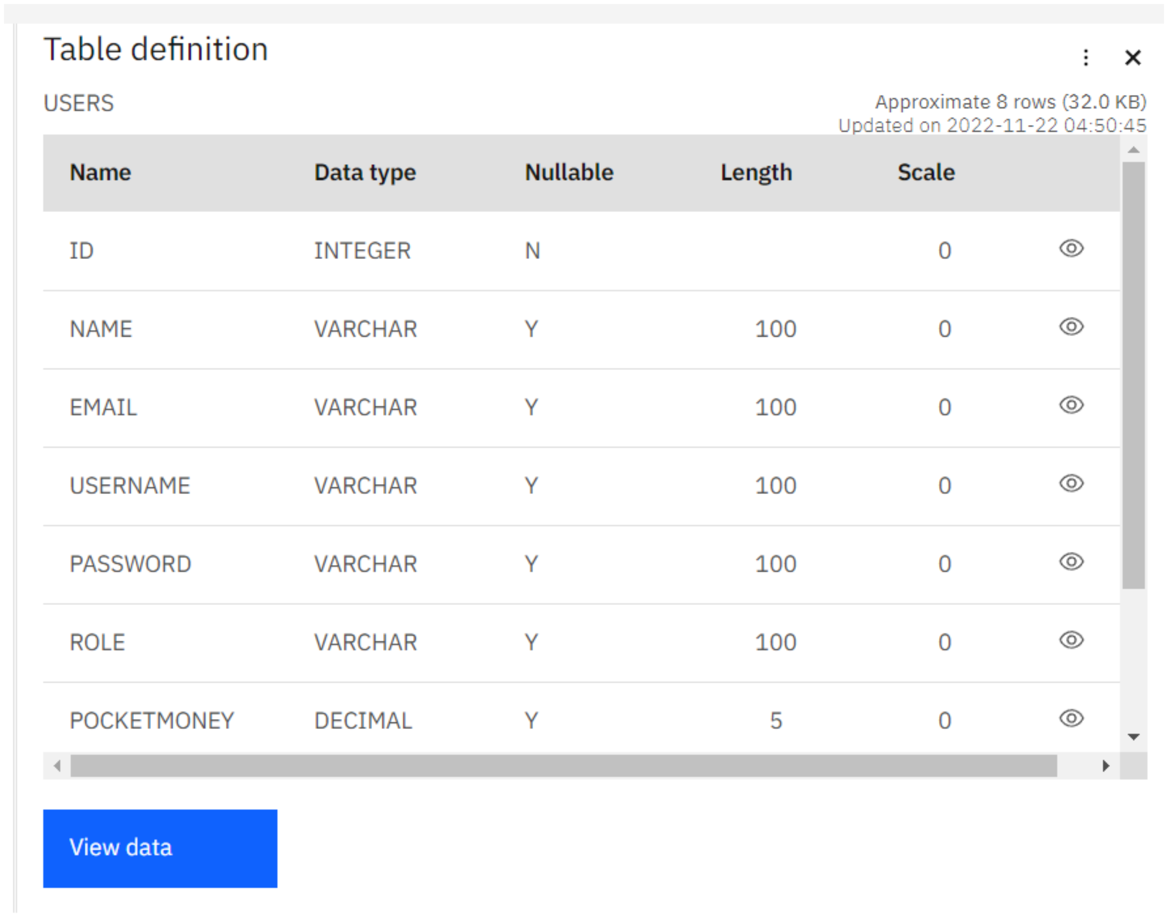


Table definition

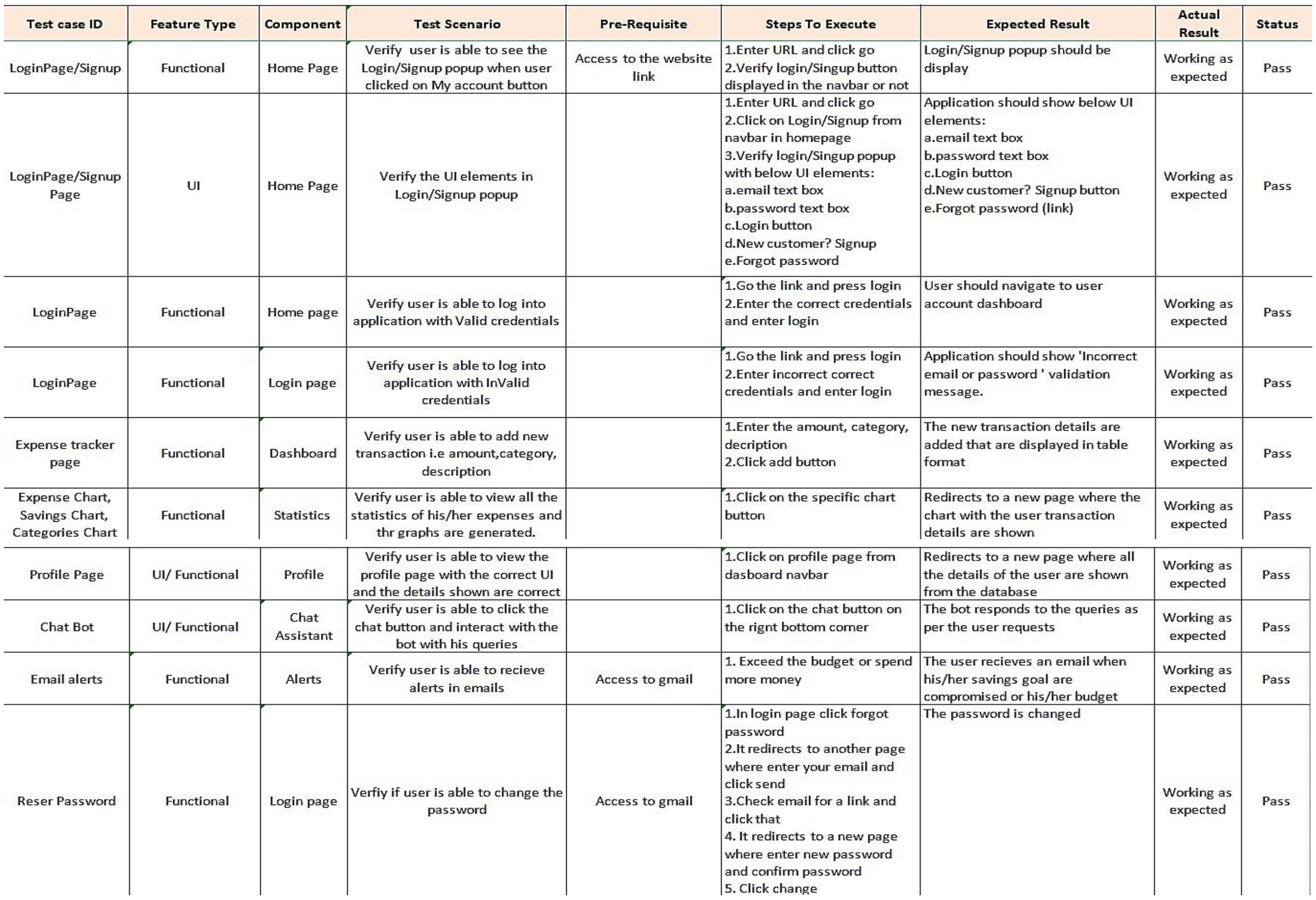
USERS

Approximate 8 rows (32.0 KB)
Updated on 2022-11-22 04:50:45

Name	Data type	Nullable	Length	Scale
ID	INTEGER	N		0
NAME	VARCHAR	Y	100	0
EMAIL	VARCHAR	Y	100	0
USERNAME	VARCHAR	Y	100	0
PASSWORD	VARCHAR	Y	100	0
ROLE	VARCHAR	Y	100	0
POCKETMONEY	DECIMAL	Y	5	0

8. TESTING

a. Testcases



Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Expected Result	Actual Result	Status
LoginPage/Signup	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button	Access to the website link	1.Enter URL and click go 2.Verify login/Singup button displayed in the navbar or not	Login/Signup popup should be display	Working as expected	Pass
LoginPage/Signup Page	UI	Home Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on Login/Signup from navbar in homepage 3.Verify login/Singup popup with below UI elements: a.email text box b.password text box c.Login button d.New customer? Signup e.Forgot password	Application should show below UI elements: a.email text box b.password text box c.Login button d.New customer? Signup button e.Forgot password (link)	Working as expected	Pass
LoginPage	Functional	Home page	Verify user is able to log into application with Valid credentials		1.Go the link and press login 2.Enter the correct credentials and enter login	User should navigate to user account dashboard	Working as expected	Pass
LoginPage	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Go the link and press login 2.Enter incorrect correct credentials and enter login	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass
Expense tracker page	Functional	Dashboard	Verify user is able to add new transaction i.e amount,category, description		1.Enter the amount, category, decription 2.Click add button	The new transaction details are added that are displayed in table format	Working as expected	Pass
Expense Chart, Savings Chart, Categories Chart	Functional	Statistics	Verify user is able to view all the statistics of his/her expenses and thr graphs are generated.		1.Click on the specific chart button	Redirects to a new page where the chart with the user transaction details are shown	Working as expected	Pass
Profile Page	UI/ Functional	Profile	Verify user is able to view the profile page with the correct UI and the details shown are correct		1.Click on profile page from dasboard navbar	Redirects to a new page where all the details of the user are shown from the database	Working as expected	Pass
Chat Bot	UI/ Functional	Chat Assistant	Verify user is able to click the chat button and interact with the bot with his queries		1.Click on the chat button on the rignt bottom corner	The bot responds to the queries as per the user requests	Working as expected	Pass
Email alerts	Functional	Alerts	Verify user is able to recieve alerts in emails	Access to gmail	1. Exceed the budget or spend more money	The user recieves an email when his/her savings goal are compromised or his/her budget	Working as expected	Pass
Reser Password	Functional	Login page	Verfiy if user is able to change the password	Access to gmail	1.In login page click forgot password 2.It redirects to another page where enter your email and click send 3.Check email for a link and click that 4. It redirects to a new page where enter new password and confirm password 5. Click change	The password is changed	Working as expected	Pass

b. User Acceptance Testing

1. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	3	2	2	12
Duplicate	1	0	1	0	2

External	2	1	0	1	4
Fixed	6	2	5	10	23
Not Reproduced	0	1	0	0	1
Skipped	1	0	0	0	1
Won't Fix	0	3	2	1	6
Totals	15	10	10	14	49

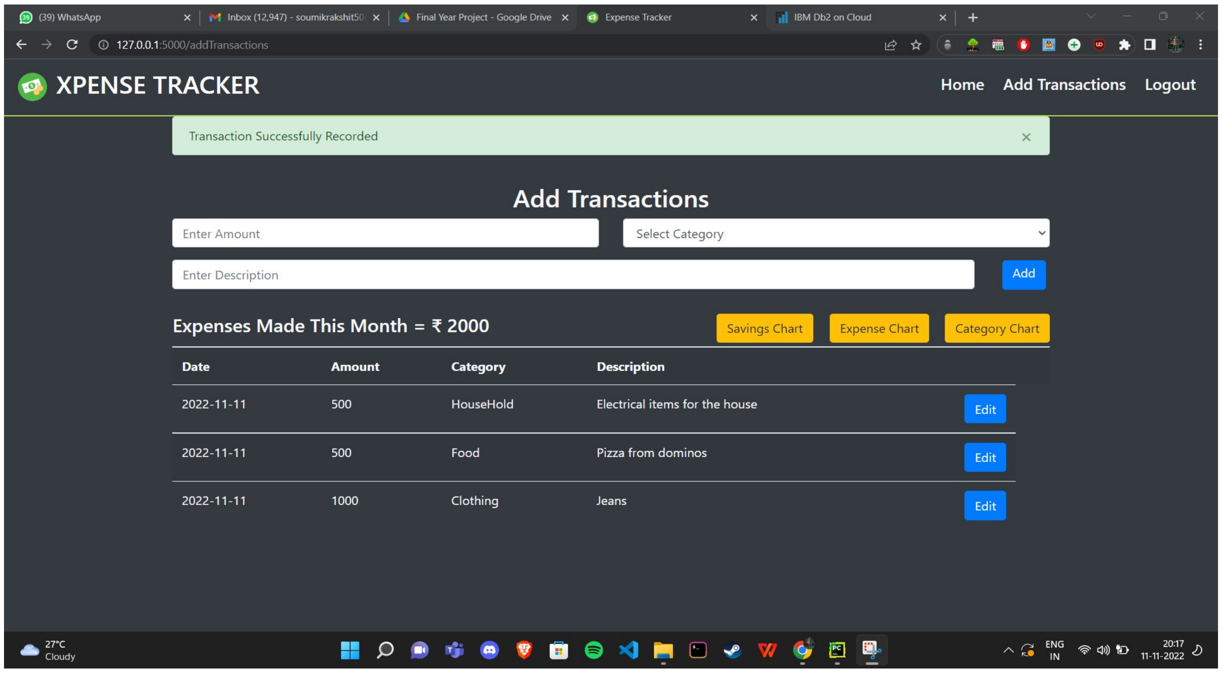
2. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	3	0	0	3
Client Application	20	0	0	20
Security	3	1	0	4
Outsource Shipping	1	0	0	1
Exception Reporting	4	0	0	4
Final Report Output	2	0	0	2
Version Control	3	0	0	3

9. RESULTS

a. Created the track expenses page which users can use to the track their expenses and edit them as they use.



Transaction Successfully Recorded

Add Transactions

Enter Amount | Select Category

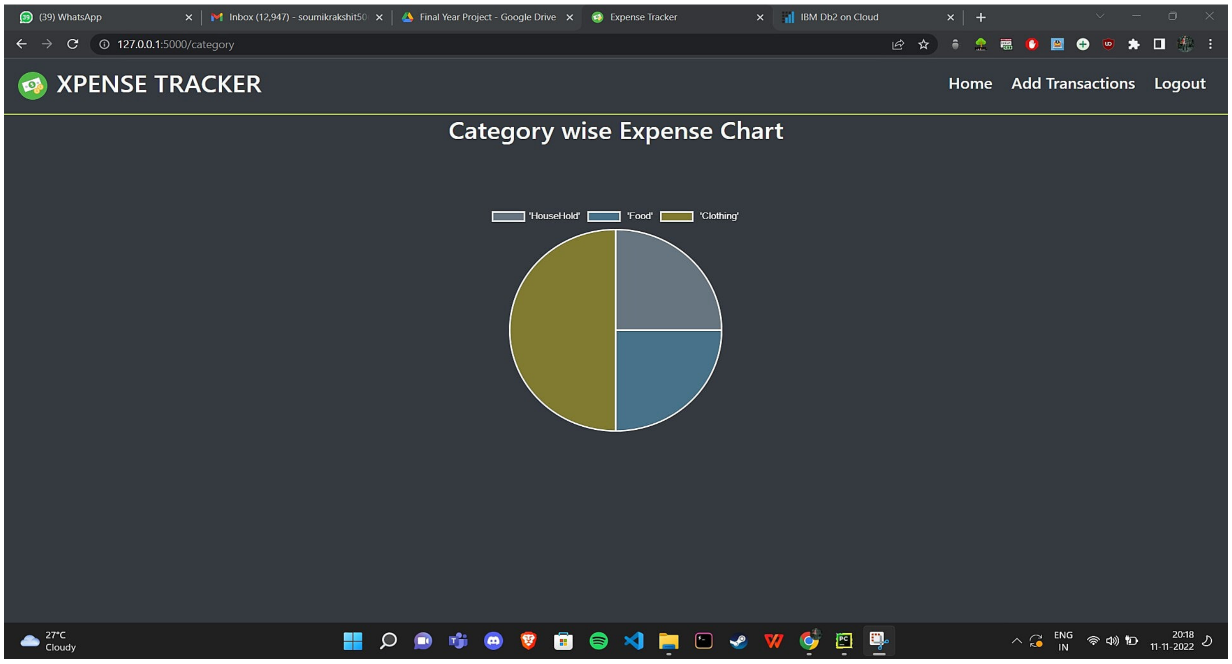
Enter Description | Add

Expenses Made This Month = ₹ 2000

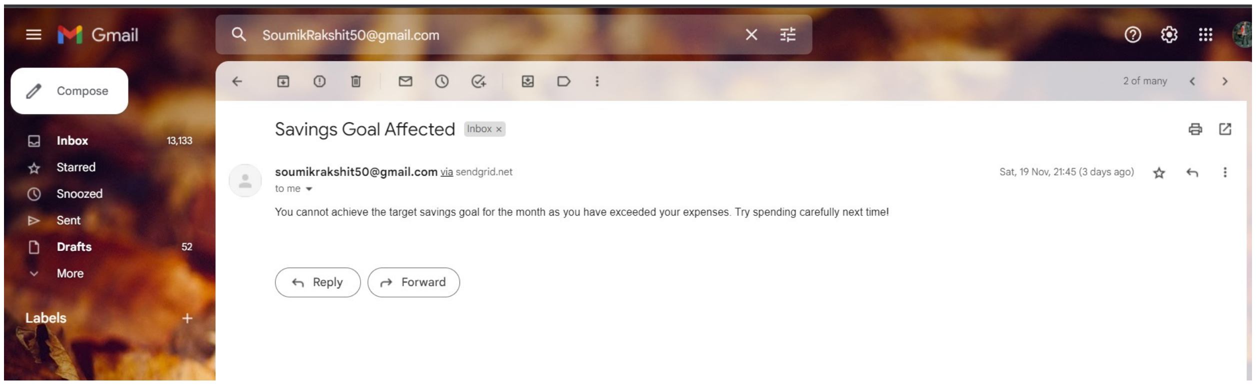
Savings Chart | Expense Chart | Category Chart

Date	Amount	Category	Description	
2022-11-11	500	HouseHold	Electrical items for the house	Edit
2022-11-11	500	Food	Pizza from dominos	Edit
2022-11-11	1000	Clothing	Jeans	Edit

b. Can view statistics and charts to analyse the our spending



c. Created email alerts using sendgrid

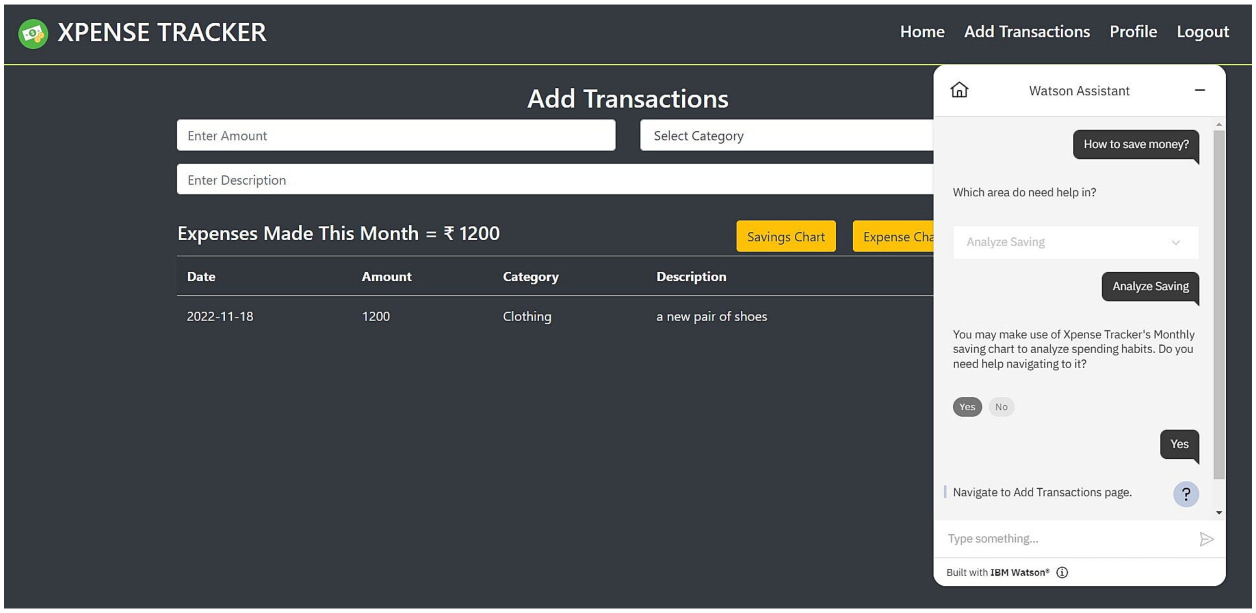


Savings Goal Affected Inbox ×

soumikrakshit50@gmail.com via sendgrid.net
to me
Sat, 19 Nov, 21:45 (3 days ago)

You cannot achieve the target savings goal for the month as you have exceeded your expenses. Try spending carefully next time!

d. Chatbot to interact and guide the users



Expenses Made This Month = ₹ 1200

Date	Amount	Category	Description
2022-11-18	1200	Clothing	a new pair of shoes

How to save money?

Which area do need help in?

Analyze Saving

You may make use of Xpense Tracker's Monthly saving chart to analyze spending habits. Do you need help navigating to it?

Yes

Navigate to Add Transactions page.

The expense tracker website was created and all the pages were tested. The website is designed to track expenses, view stats and analyse spending, chatbot to interact with user and forgot password feature to recover users accound. The different functionalities were integrated with the database and tested. Finally the app was successfully deployed using docker and kubernetes.

10. ADVANTAGES AND DISADVANTAGES

Advantages :

- This application helps you manage your expenses
- It helps you meet your financial expenses
- It provides an interactive user experience
- It helps meet your future goals by helping you save money
- It alerts the user from time to time to ensure that the user sticks to the routine.

Disadvantages:

- User might not get time to update his expenses daily
- Determining the right process might be difficult
- Might be stressful
- Maintainace is high

11. CONCLUSION

Xpense Tracker is a finance tracking and management application tailor made for Young Adults who are just getting into income and savings management and experienced users alike. The application is designed and developed to be much more portable than traditional systems. It has a user friendly design that simplifies the complex and disciplined task of money management. Upon registration the users are prompted to enter their budget and saving goal information. The dashboard is equipped to let the users enter their expenses category wise. It allows provides provision for visualising and tracking expense and savings. The application is designed to notify users regarding expense exceeding situations.

12. FUTURE SCOPE

Xpense Tracker is a fully functional finance tracking web application deployed in IBM cloud equipped with db2 in its backend. Although the application proves efficient in catering to the basic finance tracking needs of users, it can still be upgraded to include features such as extracting expense from bill images, integration with bank account, paying via the app, supporting multiple currencies etc. A mobile app equipped with data analytics that can achieve the same use as Xpense Tracker would prove helpful.

13. APPENDIX

Github link:

https://github.com/IBM-EPBL/IBM-Project-2693-1658481225

Demolink:
https://drive.google.com/file/d/14oLTg3OQJqB8o3FtfFBv2WN3y5aHZeTn/view?usp=drivesdk