SMART FASHION RECOMMENDER APPLICATION

HX8001 - PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP

A NALAIYATHIRAN - PROJECT REPORT

Submitted by

S.No	NAME	REGISTER NO
1	MEERA V	815119106021
2	NANDHINI R	815119106026
3	NISHA S	815119106027
4	RAGASWETHA M	815119106031

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

DHANALAKSHMI SRINIVASAN INSTITUTE OF TECHNOLOGY
SAMAYAPURAM, TIRUCHIRAPALLI - 621112

ANNA UNNIVERSITY

2022 - 2023

CONTENTS

ACKNOWLEDGEMENT

The completion of this project could not have been possible without the participation of individuals contributing to this project. However, we would like to express our deep appreciation to our mentors and evaluators for their endless support, kindness, and understanding during the project duration.

We would like to extend our gratitude to the **IBM** for **Nalaiya Thiran** project for providing us with all the facilities that were required.

We are highly indebted to our **Faculty Mentor** and **Industry Mentor** for their guidance and supervision. We would like to thank them for providing the necessary information and resources for this project.

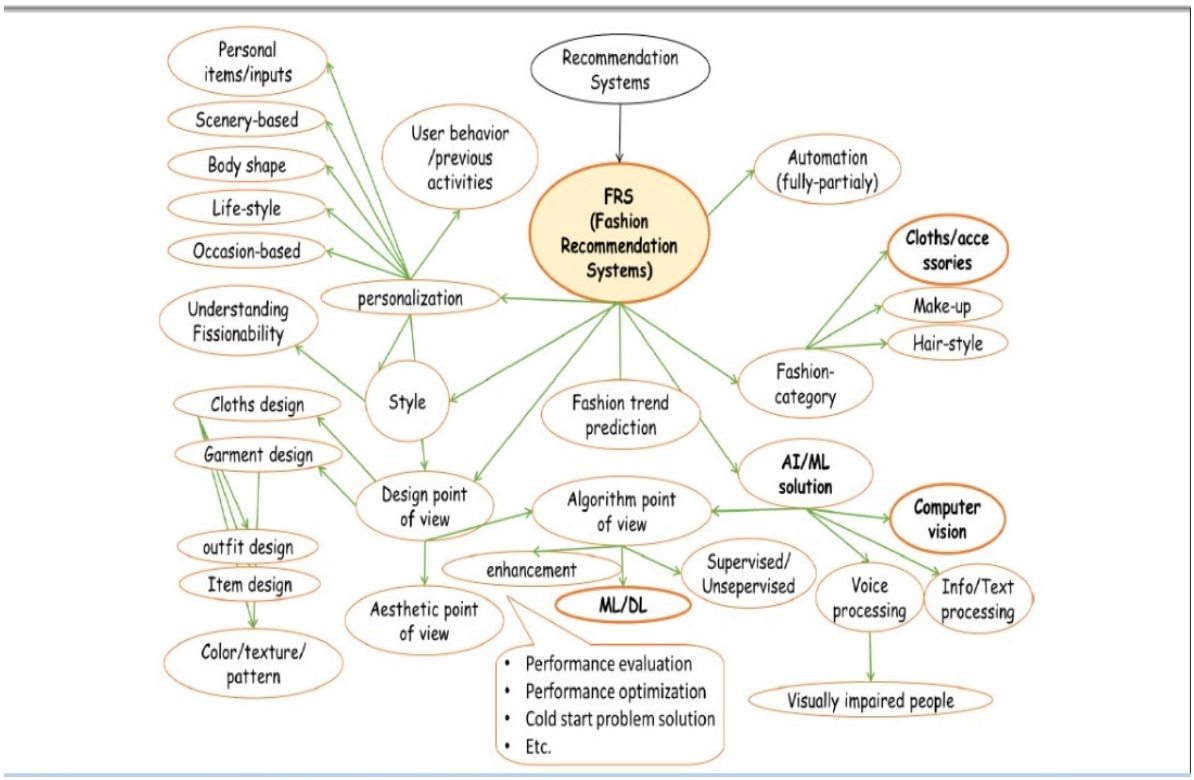
It was a great learning experience. We would like to take this opportunity to express our gratitude.

ABSTRACT

The textile and fashion industries have witnessed an enormous amount of growth in fast fashion. On e-commerce platforms, where numerous choices are available, an efficient recommendation system is required to sort, order, and efficiently convey relevant product content or information to the users. Smart fashion recommendation systems (SFRSs) have attracted a huge amount of attention from fast fashion retailers as they provide a personalized shopping experience to consumers. With the technological advancements, Web application exhibits a tremendous amount of potential in processing, parsing, classification, and segmentation. This is the review of state of art of fashion recommendation systems. In addition, this review also explores various potential models that could be implemented to develop fashion recommendation systems.

1. INTRODUCTION

In fashion, clothing is a kind of symbol that represents people's internal perceptions through their outer appearance. It conveys information about their choices, faith, personality, profession, social status, and attitude towards a life. Therefore, clothing is believed to be a nonverbal way of communicating and a major part of people's outer appearance. Recent technological advancements have enabled consumers to track current fashion trends around the globe, which influence their choices. The fashion choices of consumers depend on many factors, such as demographics, geographic location, individual preferences, interpersonal influences, age, gender, season, and culture



Fashion Recommendation System

Moreover, previous fashion recommendation research shows that fashion preferences vary not only from country to country but also from city to city. The combination of fashion preferences and the above mentioned factors associated with clothing choices could transmit the image features for a better understanding of consumers' preferences. Therefore analyzing consumers' choices and recommendation is valuable to fashion designers and retailers. Additionally, consumers' clothing choices and product preference data have become available on the Internet in the form of text or opinions and images or pictures. Since these images contain information about people from all around the world, both online and offline fashion retailers are using these platforms to reach billions of users who are active on the Internet. Therefore, e-commerce has become the predominant channel for shopping in recent years.

Recommendation system is referred to as a decision-making approach for users under a multidimensional information environment. RS has also been defined as an e-commerce tool, which helps consumers search based on knowledge that is related to a consumer's choices and preferences. RS also assists in augmenting social processes by using the recommendations of other users when there is no abundant personal information or knowledge of the alternatives. RS handles the complication of information overload that consumers usually encounter by offering customized service, exclusive content, and personalized recommendations. There are multiple phases involved in the recommendation system that develop the foundation of any state-of-the-art recommendation system. These are defined as the information collection phase, the learning phase, and the recommendation phase. The interrelationship of these phases involved in the recommendation process. It shows that information collection is the initial stage of RS, which is followed by the learning phase and the recommendation phase. The recommendation provided in the last phase can be generated based on information gathered .

1.1 Project Overview

In this project, we need a below mentioned framework to build a Recommendation System.

- **Flask :** Flask is a popular micro framework for building a web application. Flask is used for developing web application using python.
- **Front End Framework :** By using Bootstrap, a open source front-end framework we can create web application

  HTML - Hyper Text Markup Language

  CSS - Cascading Style Sheets

  JS - Java Script
- **CLI :** Command line interface is a text based user to run program, manage computer files and interact with the computer.
- **Sendgrid :** Sendgrid is a cloud-based SMTP provider that allows you to send emails without having to maintain email servers.
- **Docker :** Docker is an open source containerization plat from. Docker is used to build, test and deploy application.
- **Kubernetes :** Kubernetes automates operational tasks of container. Make it easier to manage applications.
- **IBM Services :** IBM provide some set of service to implement Web Application

IBM Cloud : Cloud service is the delivery of computing service including servers, storage, databases, networking, software, analytics.

IBM Watson Assistant : Developing a ChatBot and integrate the ChatBot with the Application.

IBM DB2 : It is a family of data management products, including DB2 relational database.

1.2 Purpose

A new intelligent fashion recommender system to select the most relevant garment design scheme for a specific consumer in order to deliver new personalized garment products. This system integrates emotional fashion themes and human perception on personalized body shapes and professional designers' knowledge. The corresponding perceptual data are systematically collected from professional using sensory evaluation techniques. The perceptual data of consumers and designers are formalized mathematically using fuzzy sets and fuzzy relations. The complex relation between human body measurements and basic sensory descriptors, provided by designers, is modeled using fuzzy decision trees. The fuzzy decision trees constitute an empirical model based on learning data measured and evaluated on a set of representative samples. The complex relation between basic sensory descriptors and fashion themes, given by consumers, is modeled using fuzzy cognitive maps. The combination of the two models can provide more complete information to the fashion recommender system, making it possible to evaluate if a specific body shape is relevant to a desired emotional fashion theme and which garment design scheme can improve the image of the body shape. The proposed system has been validated in a customized design and mass market selection through the evaluations of target consumers and fashion experts using a method frequently used in marketing study.

2. Literature Survey

A literature review is a comprehensive summary of previous research on a topic. The literature review surveys scholarly articles, books, and other sources relevant to a particular area of research. The review should enumerate, describe, summarize, objectively evaluate and clarify this previous research. It should give a theoretical base for the research and help you determine the nature of your research.

2.1 Existing problem

Electronic commerce or e-commerce includes the service and good exchange through electronic support like the Internet. It plays a crucial role in today's business and users' experience. Also, e-commerce platforms produce a vast amount of information. So, Recommender Systems (RSs) are a solution to overcome the information overload problem. They provide personalized recommendations to improve user satisfaction. The present article illustrates a comprehensive and Systematic Literature Review (SLR) regarding the papers published in the field of e-commerce recommender systems. We reviewed the selected papers to identify the gaps and significant issues of the RSs' traditional methods, which guide the researchers to do future work. So, we provided the traditional techniques, challenges, and open issues concerning traditional methods of the field of review based on the selected papers. This review includes five categories of the RSs' algorithms, including Content-Based Filtering (CBF), Collaborative Filtering (CF), Demographic-Based Filtering (DBF), hybrid filtering, and Knowledge-Based Filtering (KBF).

Predicting customer future purchases and lifetime value is a key metrics for managing marketing campaigns and optimizing marketing spend.

- ◎ This task is specifically challenging when the relationships between the customer and the firm are of a non contractual nature and therefore the future purchases need to be predicted based mostly on historical purchases.

- ◎ This work compares two approaches to predict customer future purchases, first using a "buy-till-you-die" statistical model to predict customer behavior and later using a neural network on the same dataset and comparing the results.

2.2 References

[I] Title : Implementation of e-commerce used on cloud computing using asp.net technology

Author : Samson Oluwaseun Fadiya, Acheme Odeh, Emeka Joshua Chukwuemeka :2016

Project Description : In this paper, the client is given an e-commerce website that is utilized as a part of a cloud domain to discover the store and its locations online. To actualize this as a web application, we utilized ASP.NET as the Technology. ASP.NET has a few preferences, for example, improved execution, scalability, built-in security and simplicity.To build any web application utilizing ASP.NET we require a programming language, for example, C#, VB.NET, J# and so on. VB.NET was the language used to build this application. For the customer browser to associate with the ASP.NET engine, we utilized Microsoft's Internet Information Services (IIS) as the Web Server. ASP.NET utilizes ADO.NET to interact with the database as it gives in-memory caching that takes out the need to contact the database server as often as possible and it can without much of a stretch send and keep up an ASP.NET application. MSSQL was utilized as back-end database since it gives quick data access, easy installation, and simplicity.

[II] Title : A Case Study on Recommendation Systems Based on Big Data

Author : M. Sandeep Kumar and J. Prabhu :2019

Project Description : Recommender systems mainly utilize for finding and recover contents from large datasets; it has been determining and analysis based on the scenario—Big Data. In this paper, we describe the process of recommendation system using big data with a clear explanation in representing the operation of map reduce. We demonstrate the various stage of recommendation namely data collection rating, types of filtering.

Analysis Scenario based drug recommender system, it consists of three components namely drug storage, cloud server, and recommender server. The system is evaluating with specific parameters like F score, Precision, and recall. Finally, we describe the challenge of recommendation systems like data sparsity, cold start, sentimental analysis and No surprise.

[III] Title : Building an e-commerce recommendation system by using Big Query Machine Learning

Author : Farah Tawfiq Abdul Hussien , Abdul Monem S. Rahma :2021

Project Description : The technological development in the devices and services provided via the Internet and the availability of modern devices and their advanced applications, for most people, have led to an increase in the expansion and a trend towards electronic commerce. The large number and variety of goods offered on e-commerce websites sometimes make the customers feel overwhelmed and sometimes make it difficult to find the right product. These factors increase the amount of competition between global commercial sites, which increases the need to work efficiently to increase financial profits. The recommendation systems aim to improve the e-commerce systems performance by facilitating the customers to find the appropriate products according to their preferences. There are lots of recommendation system algorithms that are implemented for this purpose. However, most of these algorithms suffer from several problems, including: cold start, sparsity of user-item matrix, scalability, and changes in user interest. This paper aims to develop a recommendation system to solve the problems mentioned before and to achieve high realistic prediction results this is done by building the system based on the customers' behavior and cooperating with the statistical analysis to support decision making, to be employed on an e-commerce site and increasing its performance. The project contribution can be shown by the experimental results using

precision, recall, F-function, mean absolute error (MAE), and root mean square error (RMSE) metrics, which are used to evaluate system performance.

1) Amazon "Made for You" Custom T-shirt. https://www.amazon.com/Made-for-You-Custom-T-shirt/dp/B08N6J8G5M. Accessed: 2021-04-17

2) G. Mohammed Abdulla, Shreya Singh, and Sumit Borar. 2019. Shop your Right Size: A System for Recommending Sizes for Fashion products. In Companion of the 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019. ACM, 327–334.

3) Adewole Adewumi, Adebola Taiwo, Sanjay Misra, Rytis Maskeliunas, Robertas Damasevicius, Ravin Ahuja, and Foluso a. Ayeni. 2019. A United Framework for Outfit Design and Advice. 31–41.

4) Pankaj Agarwal, Sreekanth Vempati, and Sumit Borar. 2018. Personalizing Similar Product Recommendations in a. Fashion E-commerce. CoRR abs/1806.11371 (2018).

5) Debopriyo Banerjee, Krothapalli Sreenivasa Rao, Shamik Sural, and Niloy Ganguly. 2020. BOXREC: Recommending a Box of Preferred Outfits in Online Shopping. ACM Trans. Intell. Syst. Technol. 11, 6 (2020), 69:1–69:28.

6) The references that follow are cited in the above text. Additional references for the papers filtered from results of the searches over the selected databases are listed at https://figshare.com/s/4ec9f1d08f6a1a362b7c [Google Scholar]

7) Aarthi, N. G. (2020). Chatbot for retail shop evaluation. International Journal of Computer Science and Mobile Computing, 9(3), 69–77. [Google Scholar]

8) Ameen, N., Hosany, S., & Tarhini, A. (2021). Consumer interaction with cutting-edge technologies: Implications for future research. Computers in Human Behavior, 120, 106761. [Crossref], [Web of Science ®], [Google Scholar]

9) Chaves, A. P., & Gerosa, M. A. (2020). How should My Chatbot interact? A survey on social characteristics in human–chatbot interaction design. International Journal of Human–Computer Interaction, 0(0), 1–30. [Taylor & Francis Online], [Google Scholar]

10) Chen, S., Li, C., Ji, F., Zhou, W., & Chen, H. (2019). Review-driven answer generation for product-related questions in E-commerce. Proceedings of WSDM '19. ACM, New York, NY, USA, 411–419. [Crossref], [Google Scholar]

11) Kathleen Bardovi-Harlig. 1996. Pragmatics and Language Teaching: Bringing Pragmatics and Pedagogy Together. (1996).

12) Sam Blum. 2017. Amazon's Newest Feature Gives You Fashion Advice from a Real Stylist. amazon-prime-outfit-compare-judges-your-outfits-for-you.amazon-prime-outfit-compare-judges-

your-outfits-for-you. (2017).https://www.thrillist.com/news/nation/

13) Yuh-Fang Chang. 2009. How to say no: An analysis of cross-cultural difference and pragmatic transfer. Language Sciences (2009).

14) Maurice Chu, Brinda Dalal, Alan Walendowski, and Bo Begole. 2010. Countertop Responsive Mirror: Supporting Physical Retail Shopping for Sellers, Buyers and Companions. In Proc. CHI.

15) Paolo Cremonesi, Antonella Di Rienzo, Franca Garzotto, Luigi Oliveto, and Pietro Piazzolla. 2016. Dynamic and Interactive Lighting for Fash.

2.3 Problem Statement Definition

An Effective recommendation system is necessary to properly sort, order, and communicate relevant product material or information to users. Effective fashion RS can have a noticeable impact on billions of customers' shopping experiences and increase sales and revenues on the provider-side. The aim of the project is to build a model capable of doing fashion recommendation by just looking at its image. The model accepts a image and first determines whether the image contains a fashion product or not and recommend it accordingly.

The main objective of this work is to :

- Develop a fashion recommendation system which answers the queries related to fashion shopping.
- To identify the fashion type of given input image.
- If the given fashion image is valid then similar set of clothing will be recommended.
- Retrieving the similar search query products from different website.

Problem of Purchaser :

Problem of Retailer :

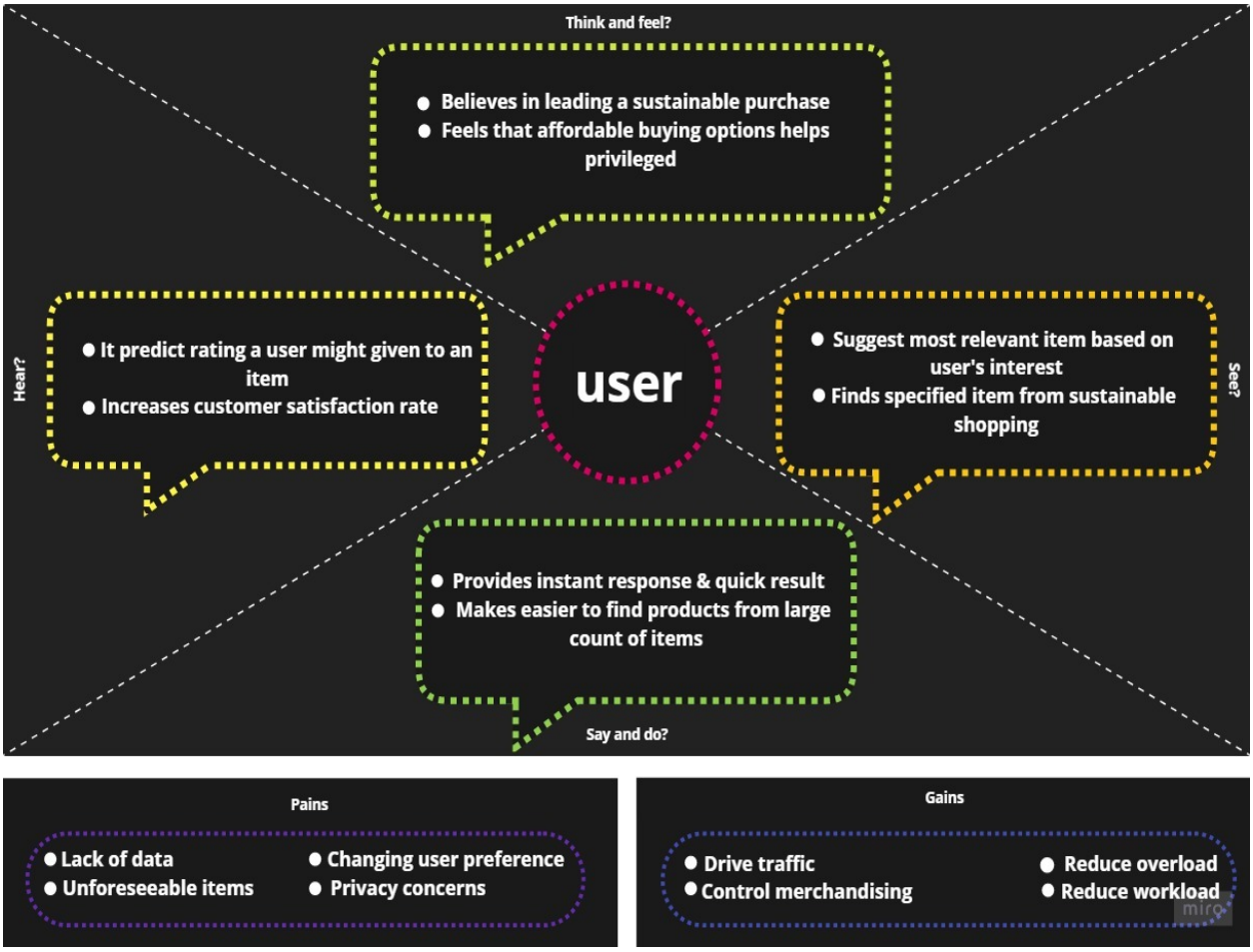
Problem of Service Provider :

3. IDEATION & PROPOSED SOLUTION

Ideation is the process where you generate ideas and solutions throughsessionssuchas Sketching, Prototyping, Brainstorming, Brainwriting, Worst Possible Idea, and a wealth of other ideation techniques provided by the Implementation agency in response to the requirements and the objectives of the Project.

3.1 Empathy Map Canvas

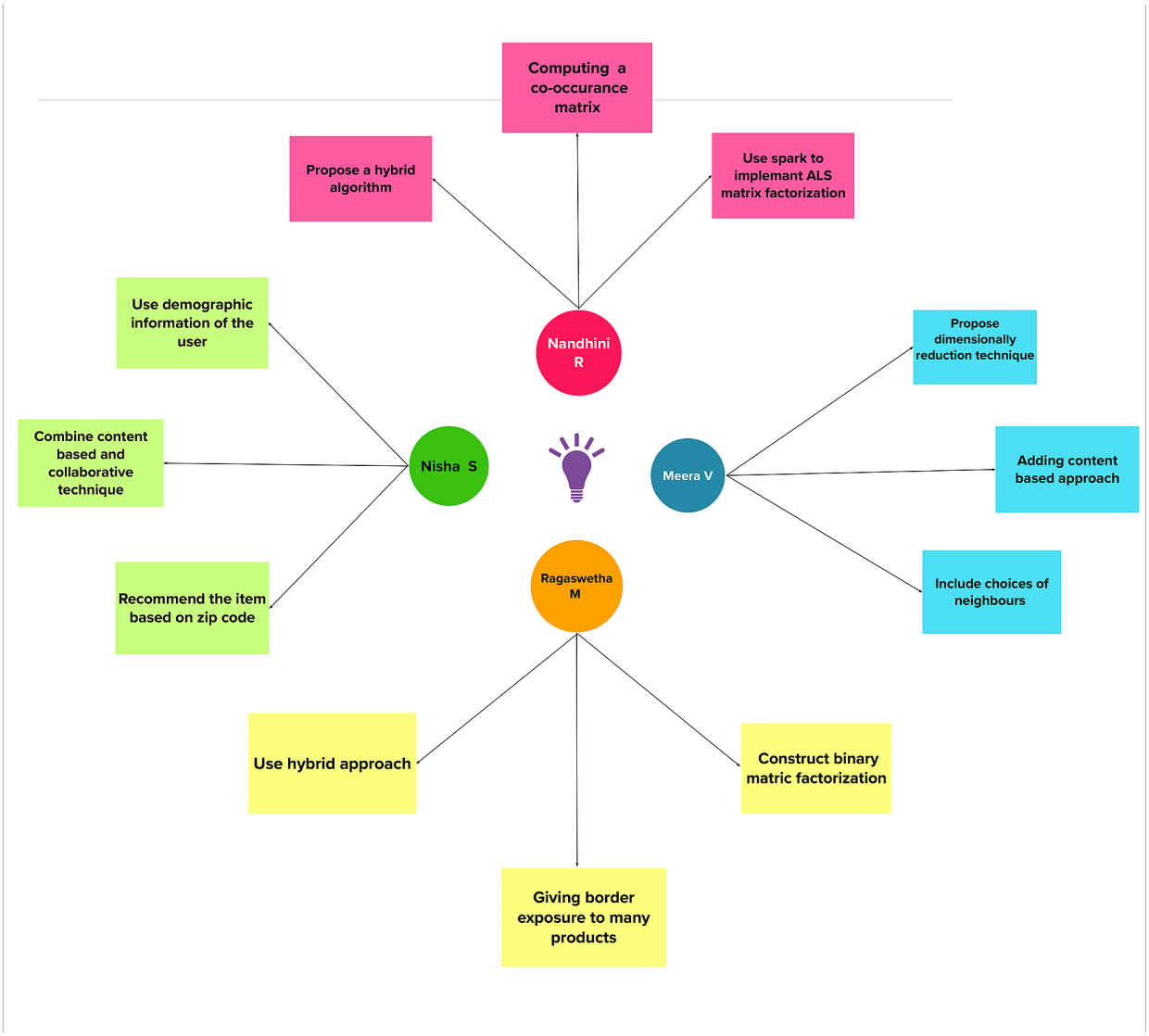
An empathy map helps to map what a design team knows about the potential customer. The empathy map has four sections namely Says, Here, Thinks, Does. It also includes the user pains and gains.



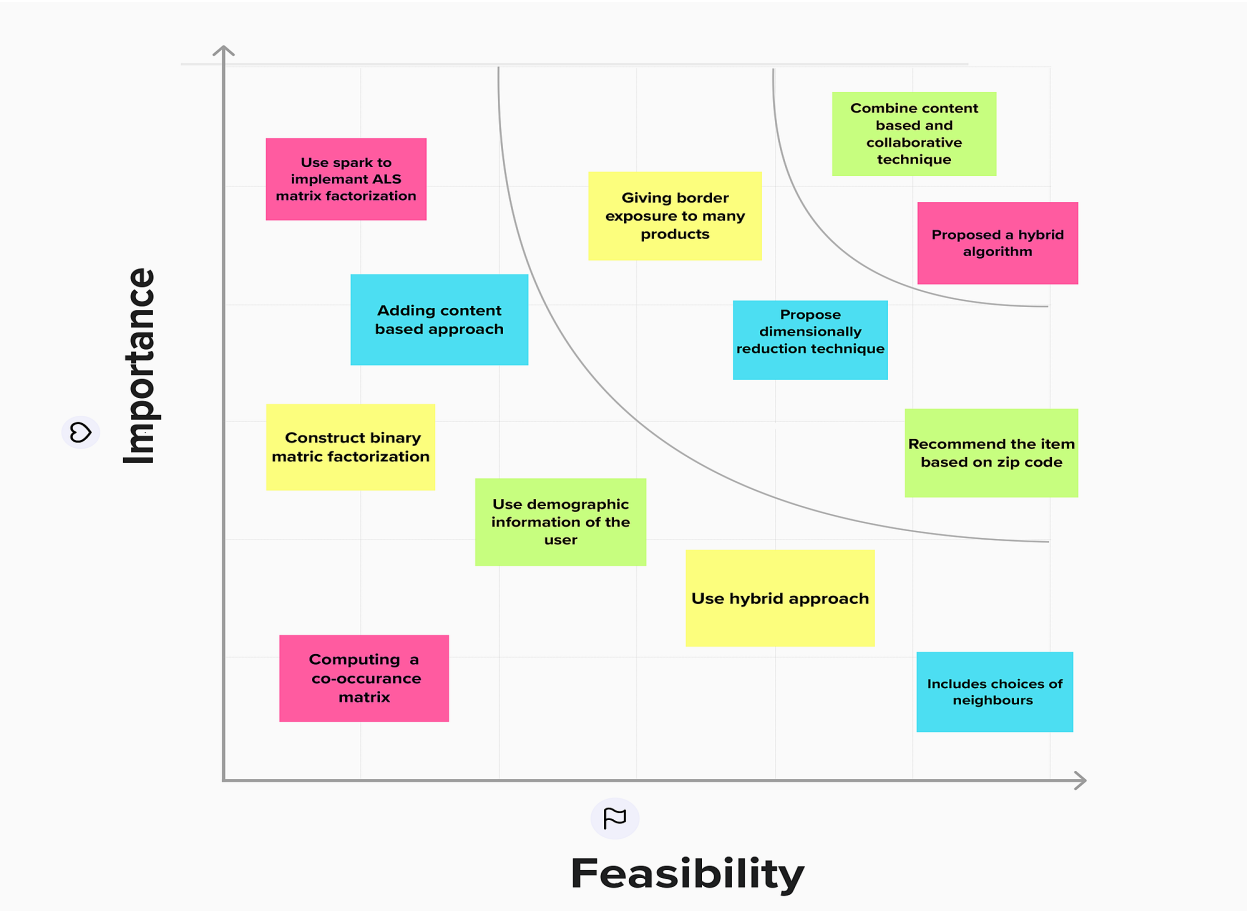
3.2 Ideation & Brainstorming

Brainstorming is a group problem-solving method that involves the spontaneous contribution of creative ideas and solutions. This technique requires intensive, freewheeling discussion in which every member of the group is encouraged to think aloud and suggest as many ideas as possible based on their diverse knowledge

Idea List



Idea Prioritization



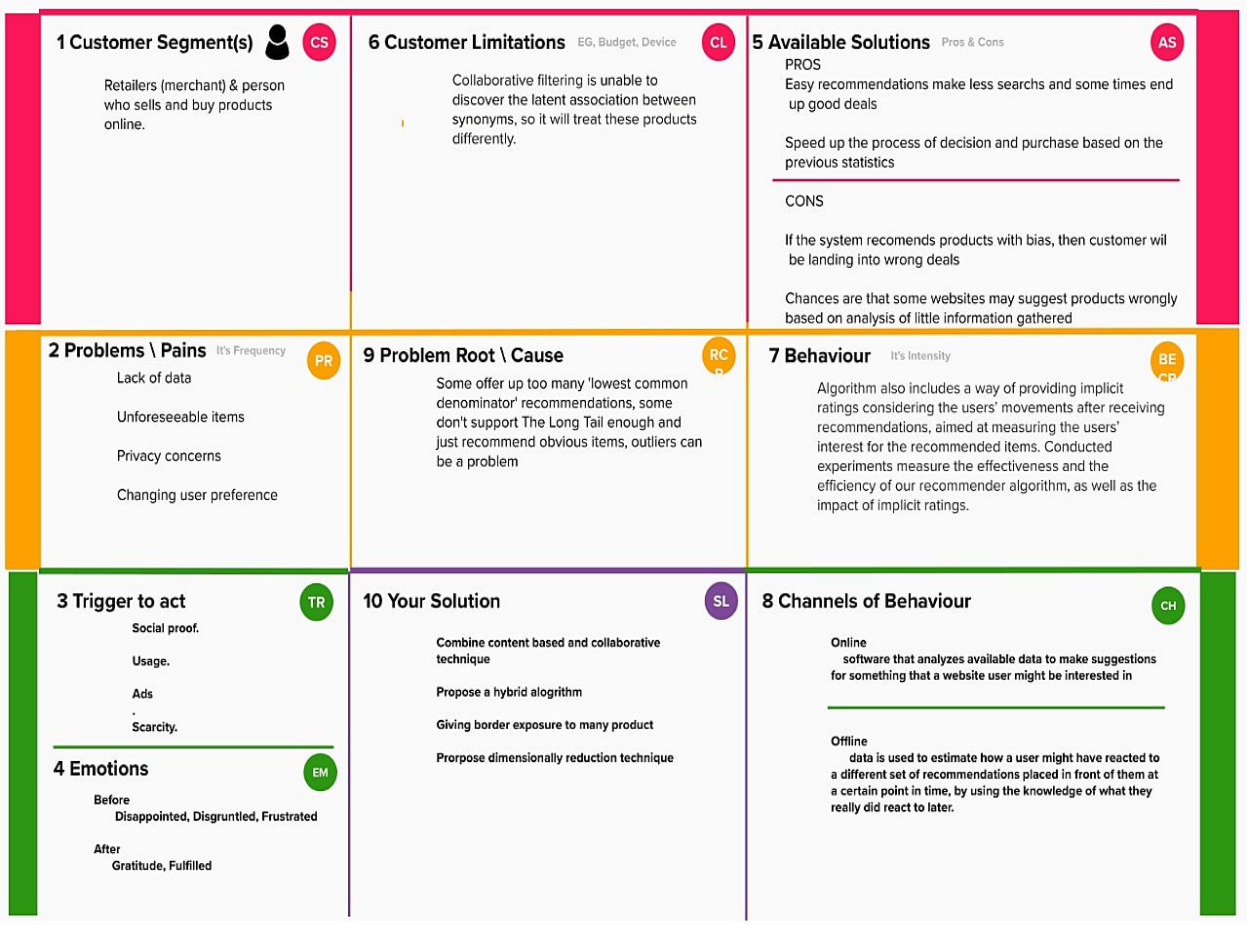
3.3 Proposed Solution

Proposed Solution means the technical solution to be provided by the Implementation agency in response to the requirements and the objectives of the Project.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	● Improving users privacy with minimum imposition of accuracy loss on the recommendation. ● In the recommendation system the problem is trying to forecast the opinion of user will have on dissimilar substance and be able to recommend the finest items to each user.
2.	Idea / Solution description	● System will recommend the items based on the zip code. ● As the user click on link of any item a time session will be started to record how much time he has spent on particular page.
3.	Novelty / Uniqueness	● Recommender system are very powerful to help a user find good products or items. the important thing the goals of recommender systems are to serve information
4.	Social Impact / Customer Satisfaction	● Recommender systems has consistently suggested that customer satisfaction will be highest when the recommendation algorithm is accurate diversity of items.
5.	Business Model (Revenue Model)	● Researchers have studied and generate many algorithms to learn increasing rate for an online customer like Amazon site.
6.	Scalability of the Solution	● A recommendation technique that is efficient when the number of dataset is limited may be unable to generate satisfactory number of recommendations when the volume of dataset is increased.

3.4 Problem Solution fit

The Problem-Solution Fit Canvas is a template to help identify solutions with higher chances of solution adoption, reduce time spent on testing and get a better overview of the current situation.



4. REQUIREMENT ANALYSIS

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed.

4.1 Functional requirement

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through mail Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login using username & Password
FR-4	Personal Details	Personal details through Form Personal details through UI Tab
FR-5	Delivery Confirmation	Confirmation via Email Confirmation via Phone

All these functionalities need to be necessarily incorporated into the system as a part of the contract. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

4.2 Non-Functional requirements

Non-Functional Requirements deal with issues like scalability, maintainability, performance, portability, security, reliability, and many more. Non-Functional Requirements address vital issues of quality for software systems.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Ease of use of the application for the user
NFR-2	Security	User privacy is the highest priority of the application. Security measures are undertaken for the user
NFR-3	Reliability	It can handle more than 2000 users at a time. It can process and initialize most functions.
NFR-4	Performance	The application can handle complex tasks and supports multi-tasking.
NFR-5	Availability	It is a free web and application available on all platforms.
NFR-6	Scalability	With higher workloads the user will experience a 10 to 17% drop in performance.

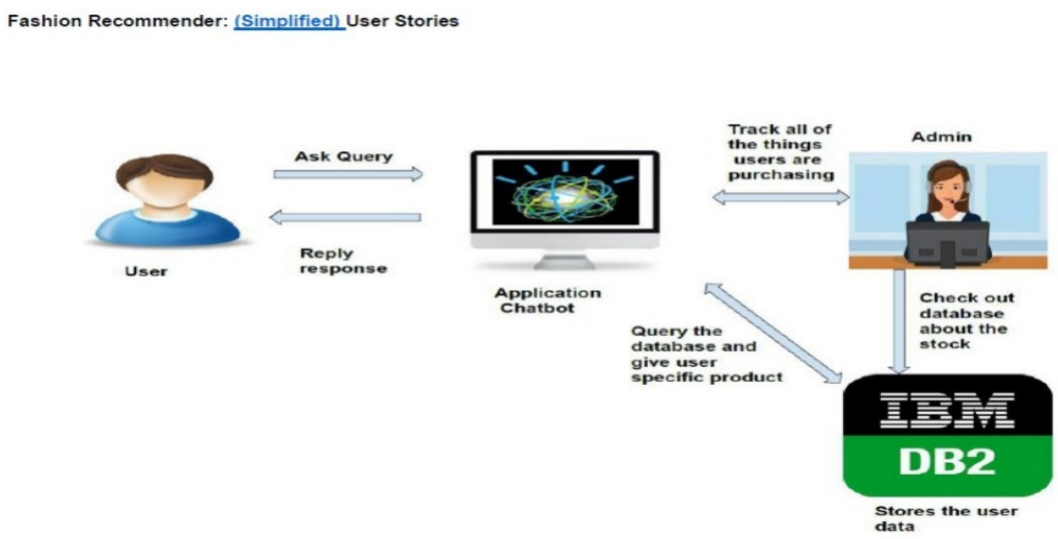
5. PROJECT DESIGN

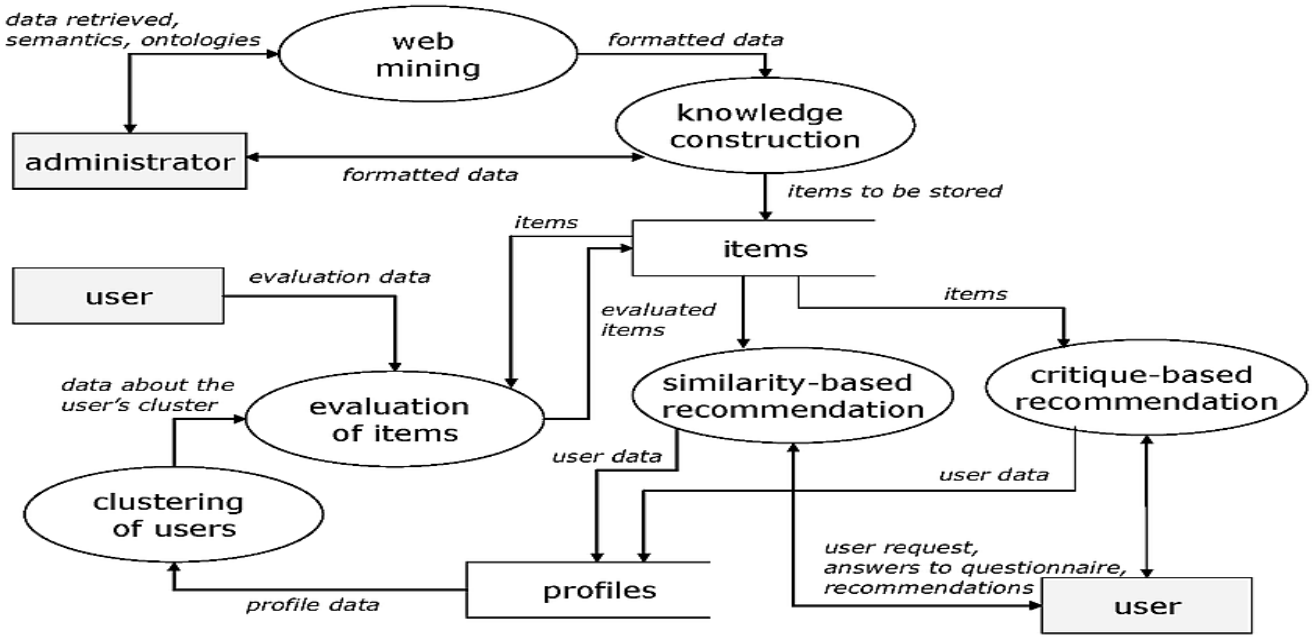
Project design is an early phase of the project lifecycle where ideas, processes, resources, and deliverables are planned out. A project design comes before a project plan as it's a broad overview whereas a project plan includes more detailed information.

A project design is a method of organizing ideas, materials, and processes in order to achieve a specific goal. Project managers rely on smart design to avoid mistakes and offer parameters to keep key components of the project, such as the Project Timeline.

5.1 Data Flow Diagrams

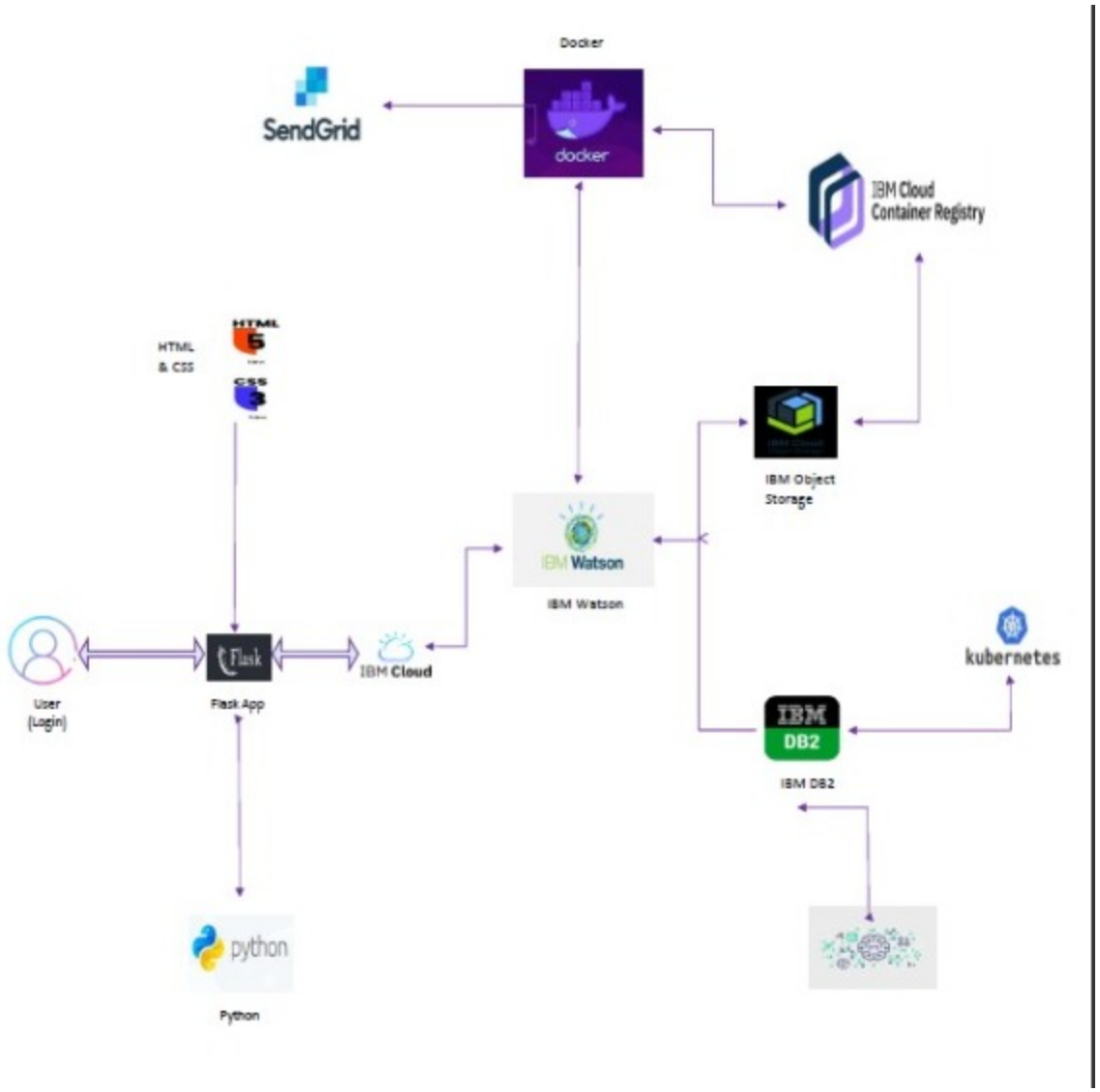
DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart.





5.2 Solution & Technical Architecture

Technical architecture—which is also often referred to as application architecture, IT architecture, business architecture, etc.—refers to creating a structured software solution that will meet the business needs and expectations while providing a strong technical plan for the growth of the software application through its lifetime.



5.3 User Stories

A user story is a well-formed, short and simple description of a software requirement from the perspective of an end-user, written in an informal and natural language. It is the main artifact used in the agile software development process to capture user requirements.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High
		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
		USN-3	As a user, I can register for the application through Email	I can register & access the dashboard with Email Login	Low
		USN-4	As a user, I can register for the application through Gmail	I can register & access the dashboard with Gmail Login	Medium
	Login	USN-5	As a user, I can log into the application by entering my username/email & password	I can login into the application with Gmail Login	High
	Dashboard	USN-5	As a user, I can log access the Dashboard of the application by logging into the application	I can access the Dashboard by logging into the application	High
Customer (Web user)	Registration	USN-1	As a user, I can register for the web-page by entering my email, password, and confirming my password.	I can access my account / dashboard	High
		USN-2	As a user, I will receive confirmation email once I have registered for the web-page	I can receive confirmation email & click confirm	High

		USN-3	As a user, I can register for the web-page through Email	I can register & access the dashboard with Email Login	Low
		USN-4	As a user, I can register for the web-page through Gmail	I can register & access the dashboard with Gmail Login	Medium
	Login	USN-5	As a user, I can log into the web-page by entering my username/email & password	I can login into the application with Gmail Login	High

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority
	Dashboard	USN-5	As a user, I can log access the Dashboard by logging into the web-page	I can access the Dashboard by logging into the web-page.	High
Customer Care Executive	Login	USN-1	As a Customer Care Executive, I can log into the application by entering my Executive email id & password	I can login into the application with Gmail Login	High
	Dashboard	USN-1	As a Customer Care Executive, I can access the Dashboard of the application by logging into the application	I can access the Dashboard by logging into the application	High
	Service	USN-1	As a Customer Care Executive, I can access the Customer service page of the application by logging and accessing the page	I can access the Service page by logging & accessing the page	High
Administrator	Login	USN-1	As a Administrator, I can log into the application by entering my Administer email id & password	I can login into the application with Gmail Login	High
	Dashboard	USN-1	As a Administrator, I can access the Dashboard of the application by logging into the application	I can access the Dashboard by logging into the application	High

6. PROJECT PLANNING & SCHEDULING

The basis of project planning is the entire project. Unlikely, project scheduling focuses only on the project-related tasks, the project start/end dates and project dependencies. Thus, a 'project plan' is a comprehensive document that contains the project aims, scope, costing, risks, and schedule. And a project schedule includes the estimated dates and sequential project tasks to be executed.

6.1 Sprint Planning & Estimation

Sprint planning is an event in scrum that kicks off the sprint. The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved. Sprint planning is done in collaboration with the whole scrum team.

What – The product owner describes the objective(or goal) of the sprint and what backlog items contribute to that goal. The scrum team decides what can be done in the coming sprint and what they will do during the sprint to make that happen.

How – The development team plans the work necessary to deliver the sprint goal. Ultimately, the resulting sprint plan is a negotiation between the development team and product owner based on value and effort.

Who – You cannot do sprint planning without the product owner or the development team. The product owner defines the goal based on the value that they seek. The development team needs to understand how they can or cannot deliver that goal. If either is missing from this event it makes planning the sprint almost impossible.

Inputs – A great starting point for the sprint plan is the product backlog as it provides a list of 'stuff' that could potentially be part of the current sprint. The team should also look at the existing work done in the increment.

Outputs – The most important outcome for the sprint planning meeting is that the team can describe the goal of the sprint and how it will start working toward that goal. This is made visible in the sprint backlog.

6.2 Sprint Delivery Schedule

A sprint schedule is a document that outlines sprint planning from end to end. It's one of the first steps in the agile sprint planning process and something that requires adequate research, planning, and communication.

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story points	Priority	Team Members
Sprint-1	Setting up App environment	USN-1	As a user, I can register in ICTA Academy and create IBM cloud account.	2	High	Nisha S Meera V
Sprint-1		USN-2	As a user, I will create a flask project	1	Low	Nandhini R Ragaswetha
Sprint-1		USN-3	As a user, I will install IBM Cloud CLI	2	Medium	Nisha S Nandhini R
Sprint-2	Setting up App environment	USN-4	As a user, I can install Docker CLI	1	Low	Ragaswetha Meera V
Sprint-2		USN-5	As a user, I will Create an account in sendgrid	2	Medium	Nisha S Ragaswetha
Sprint-3	Implementing web application	USN-6	As a user, ICreate UI to interact with the app	1	High	Nandhini R Meera V
Sprint-3		USN-7	As a user, I Create IBM DB2 & connect with Python	3	High	Nisha S

Sprint-3	Integrating sendgrid service	USN-8	As a user, I will integrating sendgrid with python code	2	High	Nandhini R
Sprint-3	Developing a chatbot	USN-9	As a user,I have to build a chatbot and Integrate to application	1	Medium	Ragaswetha
Sprint-4	Development of App in IBM Cloud	USN-10	As a user, I will Containerize the App	1	Low	Meera V
Sprint-4		USN-11	As a user, I will upload image to IBM Container registry	2	Medium	Nandhini R
Sprint-4		USN-12	As a user, I will deploy App in Kebernetes cluster	3	High	Nisha S
Sprint-4	User panel		As a user ● Register, Login, Email, Verification ● Manual Search ● Order placement, Order Details	3	High	Nisha S Nandhini R Ragaswetha Meera V

Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date (Planned)**	**Story Points Completed (as on Planned End Date)**	**Sprint Release Date (Actual)**
Sprint-1	18	6 Days	24 Oct 2022	29 Oct 2022	24	29 Oct 2022
Sprint-2	18	6 Days	31 Oct 2022	05 Nov 2022	24	05 Nov 2022
Sprint-3	18	6 Days	07 Nov 2022	12 Nov 2022	24	12 Nov 2022
Sprint-4	18	6 Days	14 Nov 2022	19 Nov 2022	24	19 Nov 2022

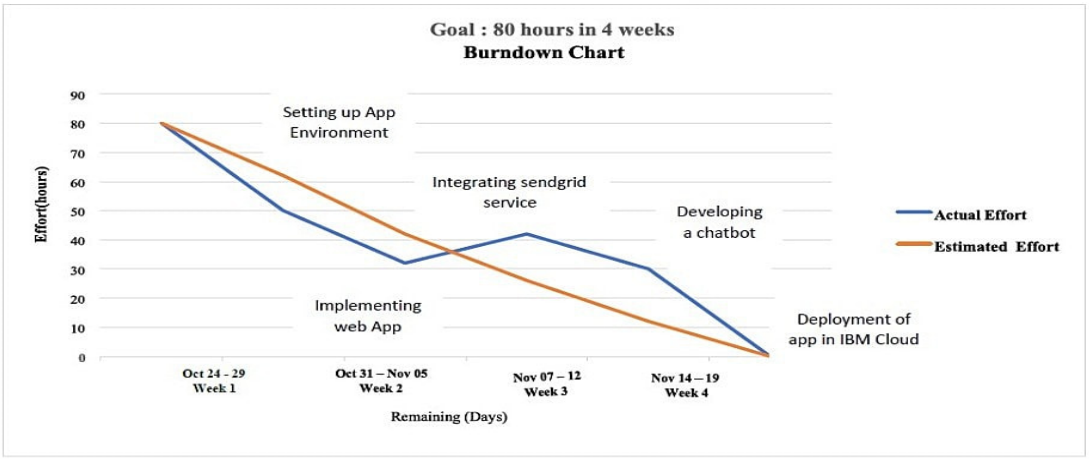
6.3 Reports from JIRA

Jira is a software application used for issue tracking and project management. The tool, developed by the Australian software company Atlassian, has become widely used by agile development teams to track bugs, stories, epics, and other tasks.

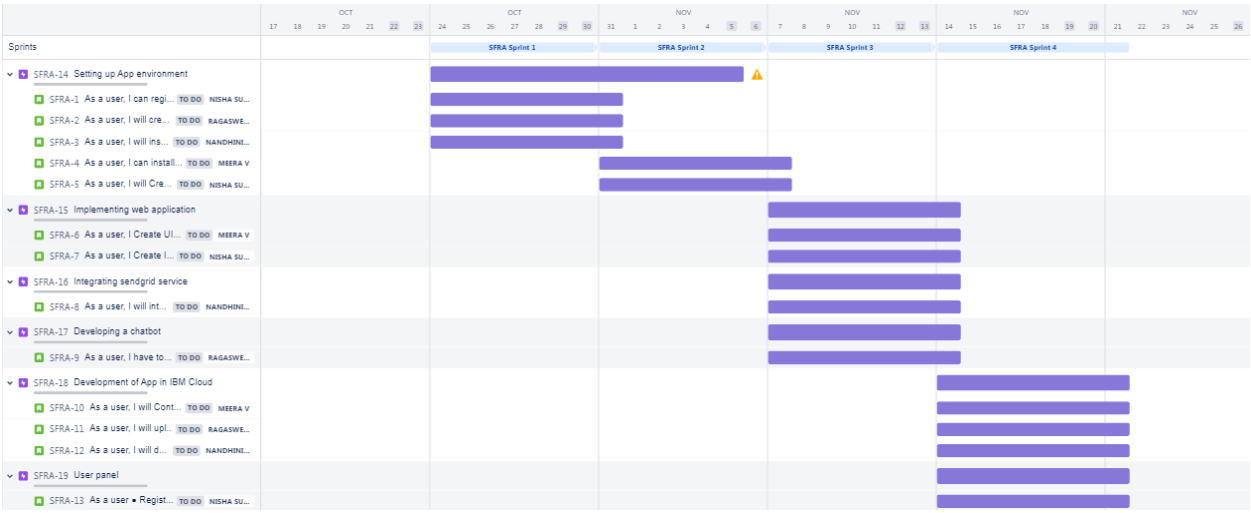
Sprint Burndown Chart

Burndown Chart

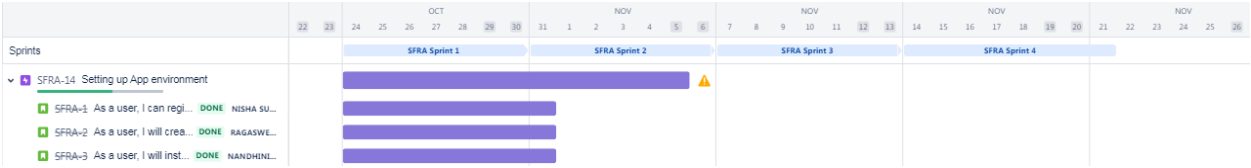
A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



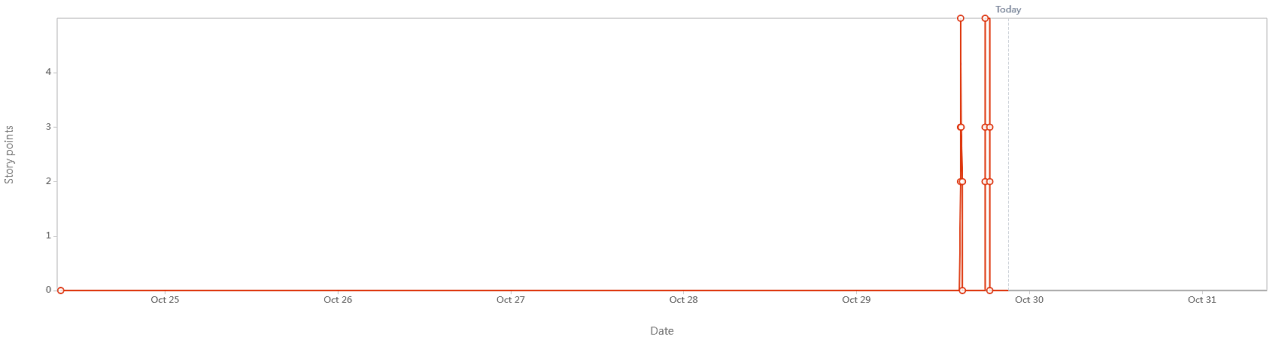
Sprint Roadmap



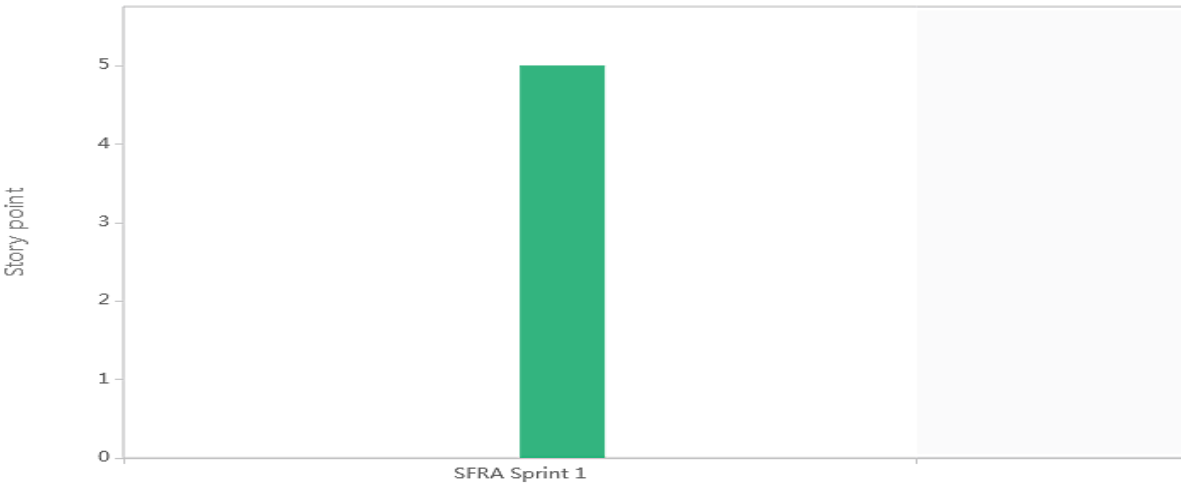
Sprint I Jira Files



Sprint I Road Map

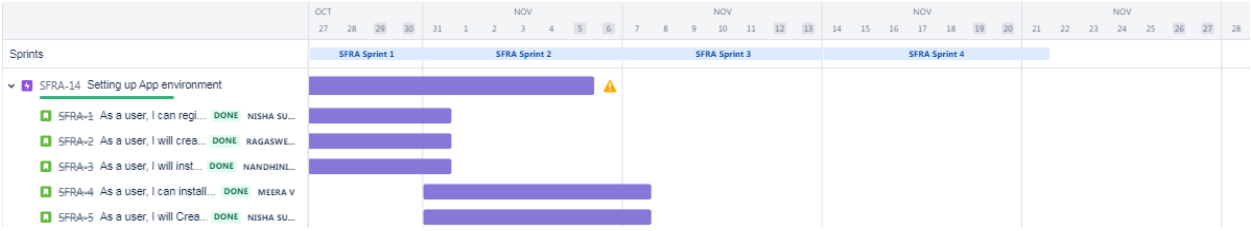


Sprint I Burndown Chart

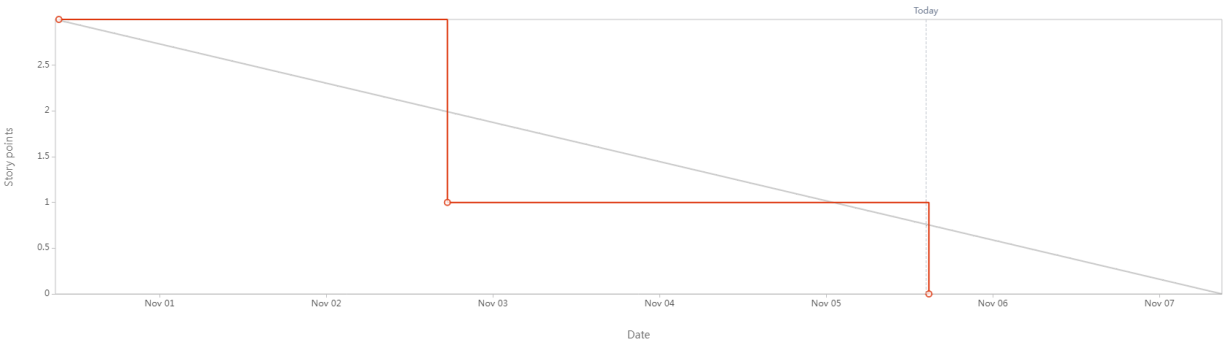


Sprint I Velocity Chart

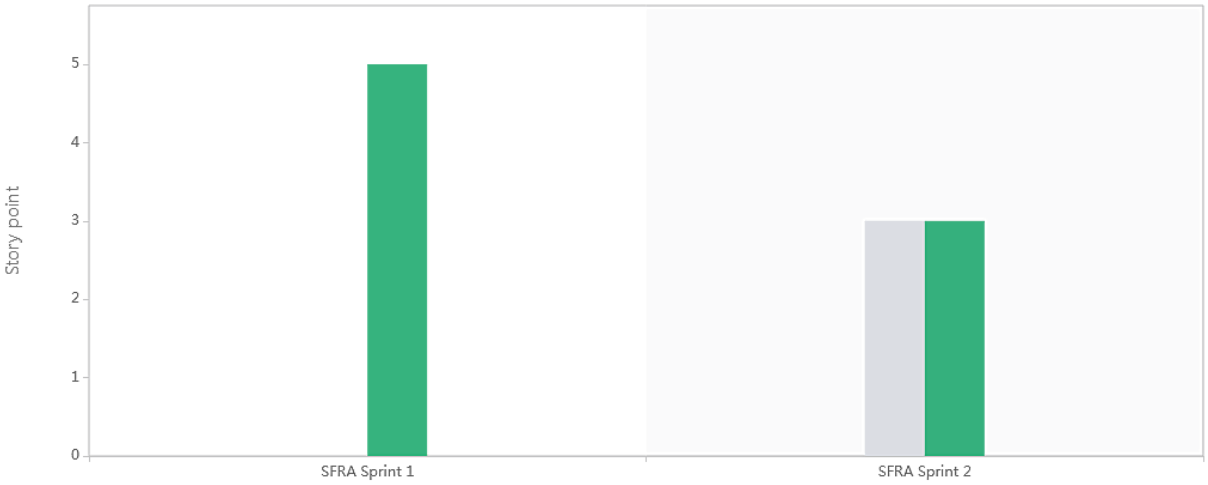
Sprint II Jira Files



Sprint II Road Map

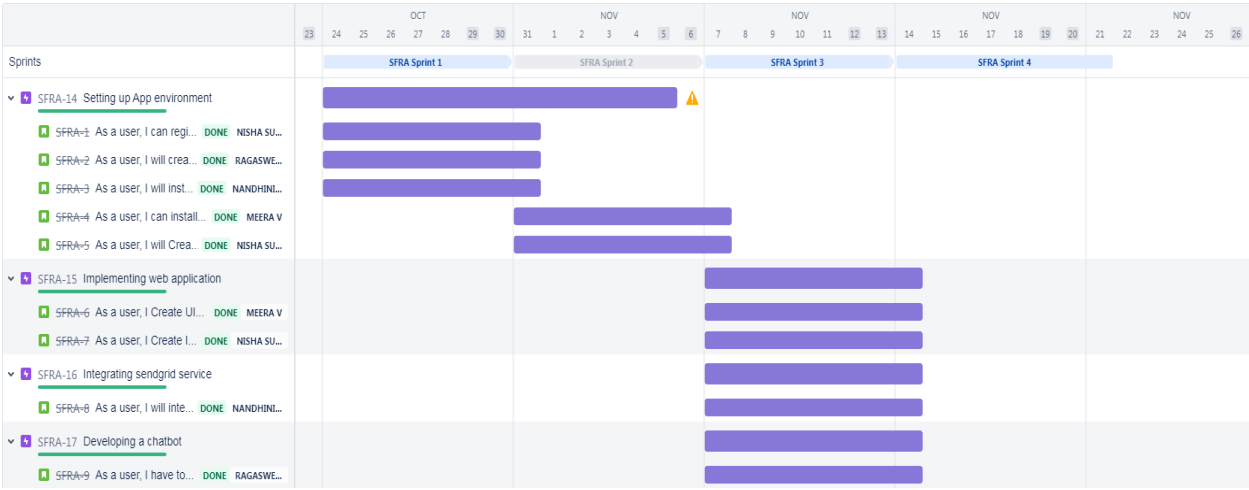


Sprint II Burndown Chart

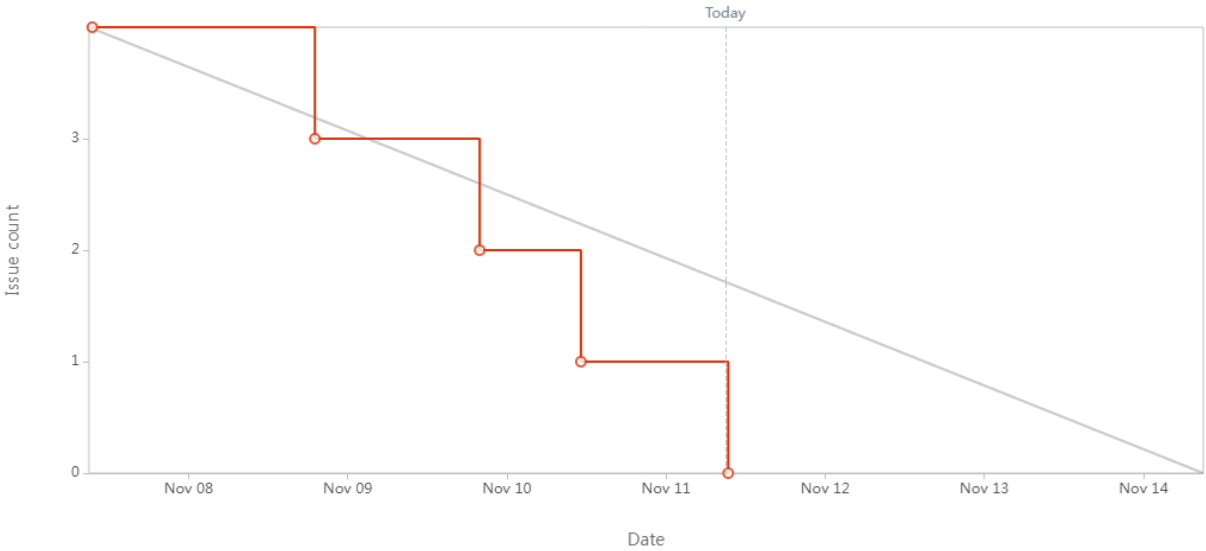


Sprint II Velocity Chart

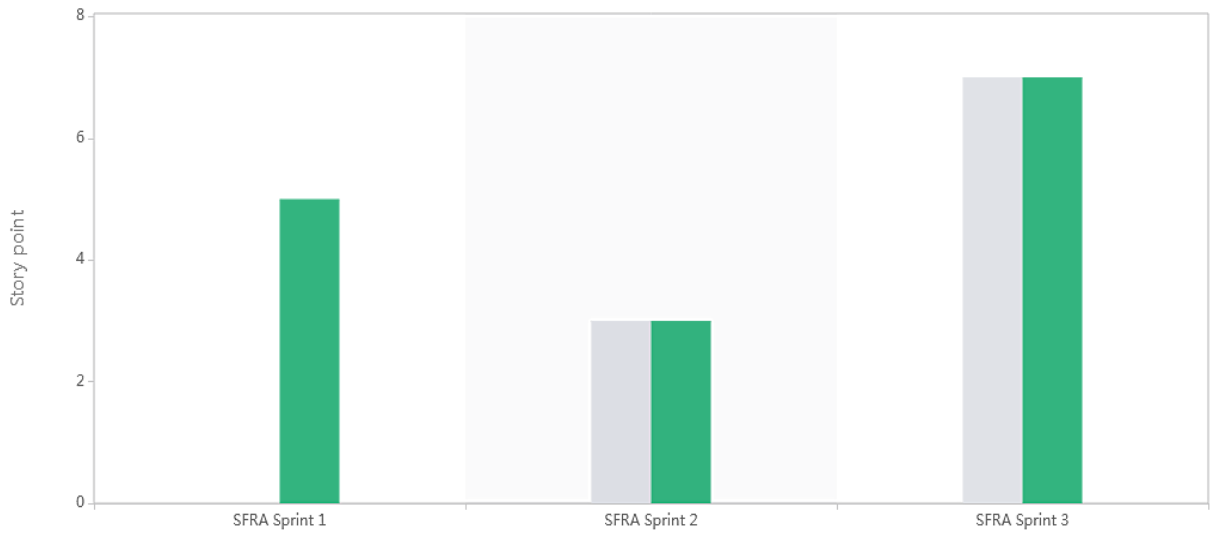
Sprint III Jira Files



Sprint III Road Map

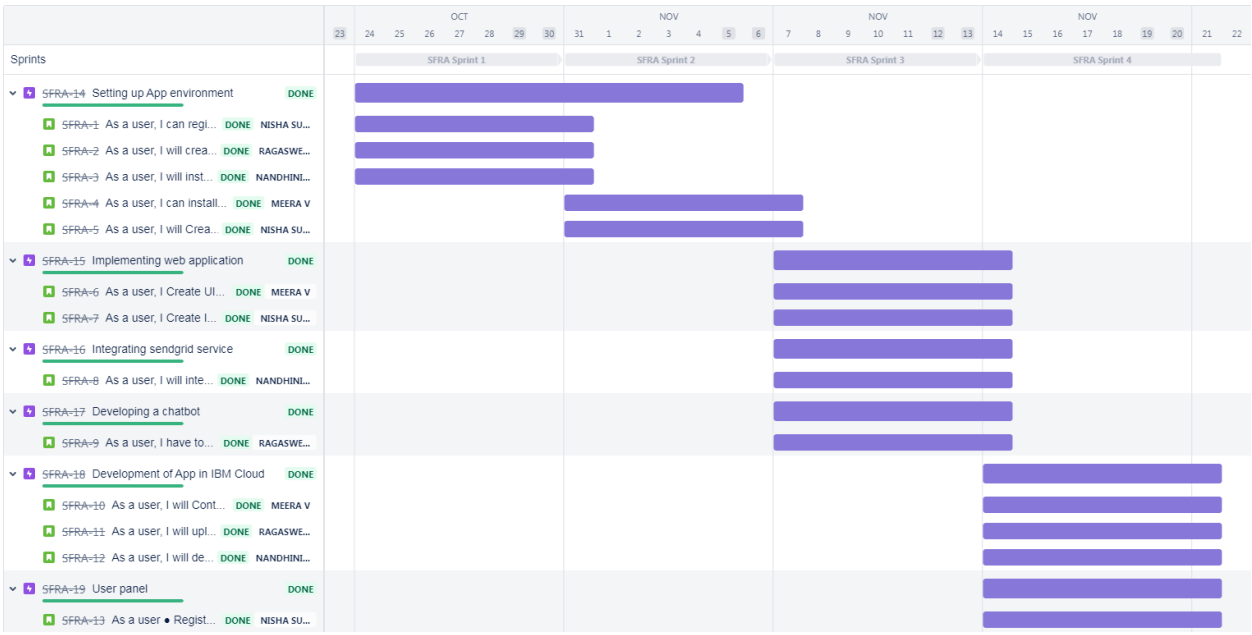


Sprint III Burndown Chart

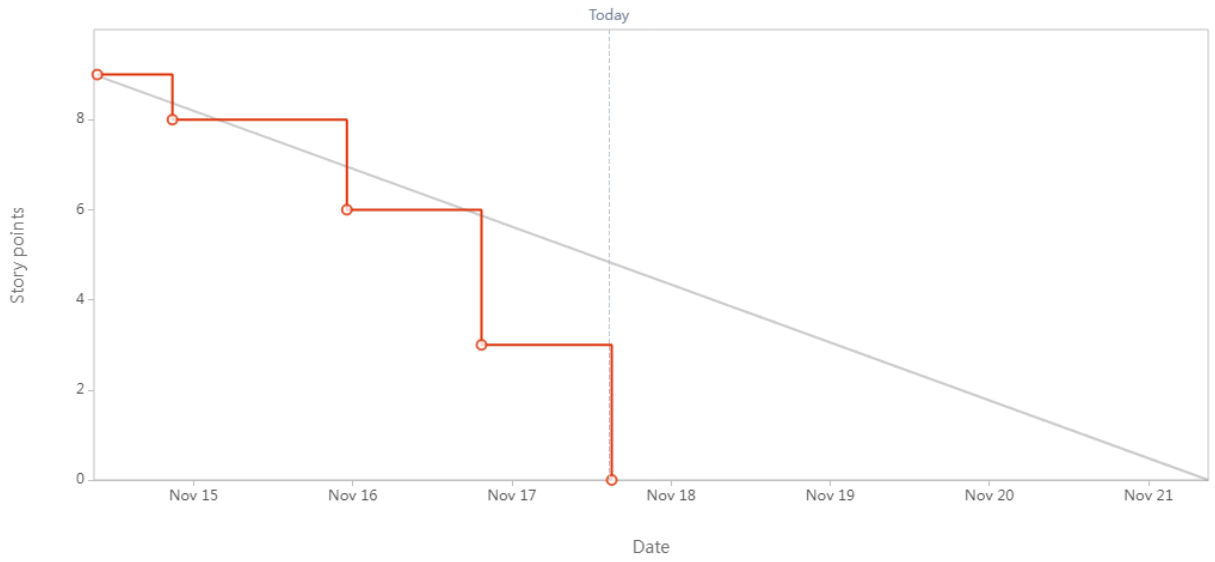


Sprint III Velocity Chart

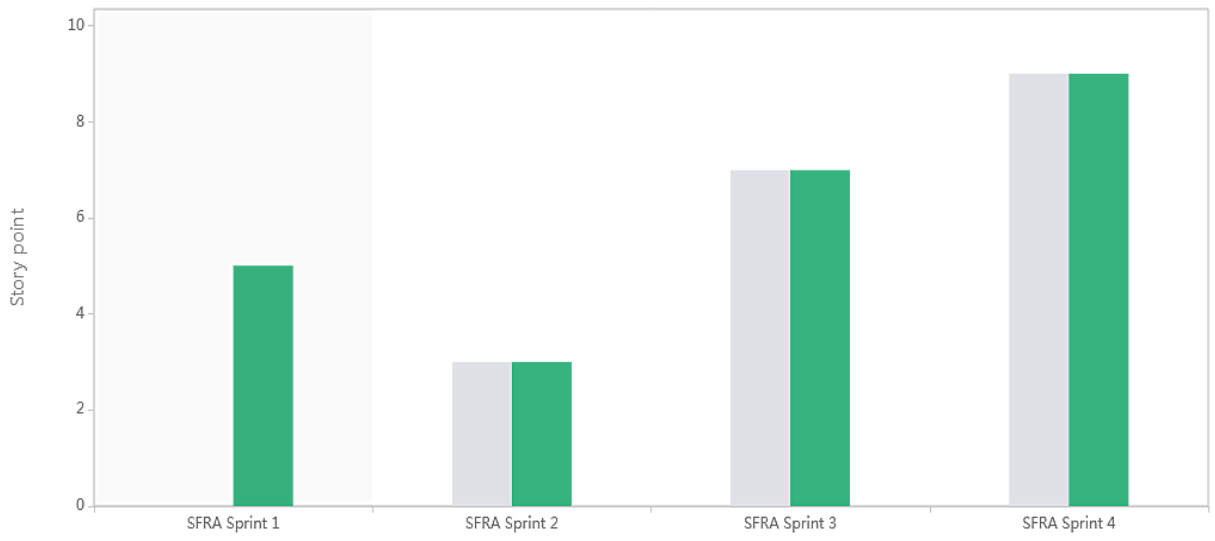
Sprint IV Jira Files



Sprint IV Road Map



Sprint IV Burndown Chart



Sprint IV Velocity Chart

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

Coding is basically the computer language used to develop apps, websites, and software. Without it, we'd have none of the most popular technology we've come to rely on such as Facebook, our smartphones, the browser we choose to view our favorite blogs, or even the blogs themselves an action or process of solving a problem.

Python

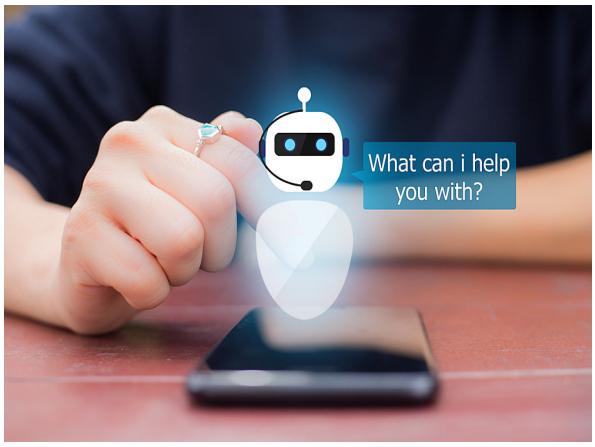
Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming

7.1 Feature 1

- **ChatBot**

Online shopping is not linear. It is easy for customers to get lost in pages, social media channels and crosslinks. E-commerce chatbots can streamline this process for customers and provide an incredible online shopping experience. There are different types of chatbots that can be employed in e-commerce, including customer service bots.

AI-powered bots come with omni-channel messaging support features which help customers communicate with businesses through various channels such as websites, Facebook, etc.



By using IBM Watson Assistant, we can able to build a chatbot and deploy the chatbot with the web application with python code

Code

```
</script>
window.watsonAssistantChatOptions = { integrationID: "614a4315-ff80-4187-8fe4-2fd9b506b723", // The ID of this integration. region: "au-syd", // The region your integration is hosted in.
serviceInstanceID: "9670dcf8-789f-4609-8d7a-6e25c412a9ec", // 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>
```

7.2 Feature 2

➤ **Sendgrid**

Sendgrid is a coud based SMTP provider that allows us to send email without having to maintain email server. It manages all of the technical details, from scaling the infrastructure to ISP outreach and reputation monitering to whitelist service and real time analytics.

In this project, we were using sendgrid to send the order details to the user by integrating a sendgrid service with the app by python code

Code

```
import os
from sendgrid import SendGridAPIClient
from sendgrid helpers.mail import Mail

message=Mail(
      from_email='from_815119106027@smartinternz.com',
      to_email='dailliancedirectioner@gamil.com',
      subject='Smart Fashion Recommender Application',
         html_content='<strong>and easy to do anywhere, even with python</strong>
try:
   sg: = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
response=sg.send(message)
    print(response.status_code)
    print(response.body)
    print(response.headers)
except Exception as e:
     print(e.message)
```

If sendgrid service is not working, we can use Flask Mail

Flask Mail

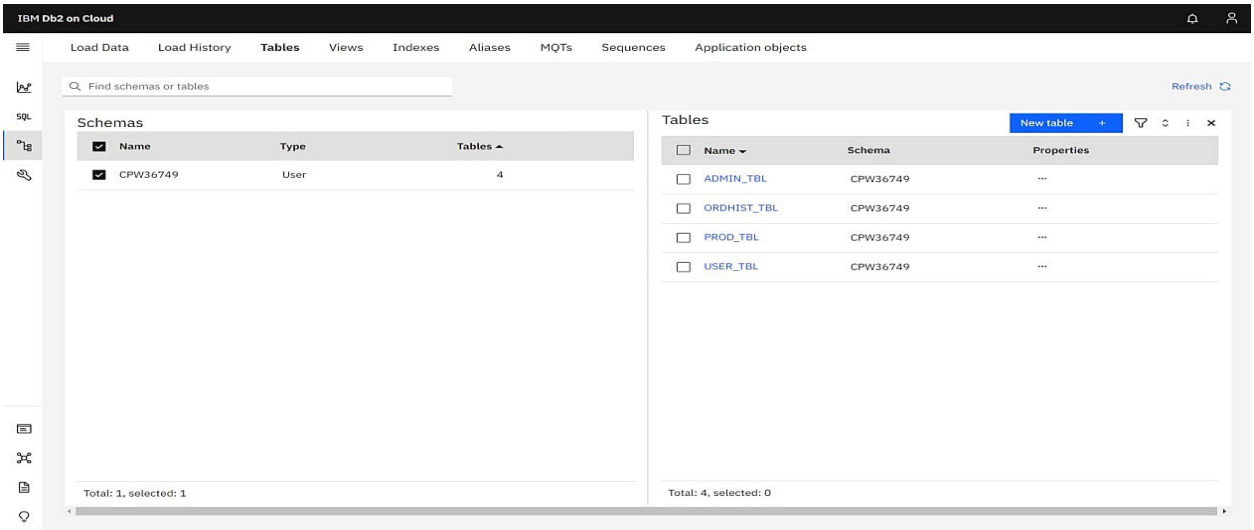
It is a basic function in web application is ability to send email to the users. Flask mail extension provides asimple interface to set up SMTP with flask application and to send messages from your script.

Code

```
1  # importing
2  libraries from flask import Flask
3  from flask_mail import Mail, Message app = Flask( name)
4  mail = Mail(app) # instantiate the mail class
5  # configuration of mail app.config['MAIL_SERVER']='smtp.gmail.com'
   app.config['MAIL_PORT']  =  465  app.config['MAIL_USERNAME']  =
   'yourId@gmail.com'    app.config['MAIL_PASSWORD']    =    '*****'
   app.config['MAIL_USE_TLS']  =  False  app.config['MAIL_USE_SSL']  =
   True mail = Mail(app)
6  # message object mapped to a particular URL '/' @app.route("/")
   def index(): msg = Message(
7                                                    'Hello', sender
   ='yourId@gmail.com',
8                                                       recipients =
   ['receiver'sid@gmail.com']
9                                              )
10 msg.body  =  'Hello  Flask  message  sent  from  Flask-Mail'
   mail.send(msg) return 'Sent'
11 if name == ' main':
12 app.run(debug = True)
13 <div>
```

7.2 Database Schema (if Applicable)

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.



8. TESTING

The process or method of finding error/s in a software application or program so that the application functions according to the end user's requirement is called testing.

8.1 Test Cases

A test case is a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case ID	Feature Type	page	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	BUG ID	Executed By
Login Page TC_OO1	Functional	Home Page	Verify user is able to see Login/Signup popup when user clicked on My account	1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Sing up popup displayed or not	https://attireglitz.myomni.in/	Login/Signup popup should display	Working as expected	Pass			Nisha S
Login Page TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup	1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Singup popup with below UI elements: a. email text box b. password text box c. Login button	https://attireglitz.myomni.in/	Application should show below UI elements: a. email text box b .password text box c. Login button with orange colour	Working as expected	Fail	Steps are not clear to follow	BUG-1234	Meera V
Login Page TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials	1.Enter URL and click go 2.Click on My Account 3.Enter Valid username/email in Email 4.Enter valid password 5.Click on login button	Username: haze@gmail.com password: Hazel2022	User should navigate to user account homepage		pass			Ragaswetha M
Login Page TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter and click go 2.Click on My Account 3.Enter Invalid username/email in Email 4.Enter valid password 5.Click on login button	Username: hazel@gmail.com password: Hazel2022	Application should show 'Incorrect email or password ' validation message.		pass			Nandhini R

Section	TotalCases	Not Tested	Fail	Pass
Login	7	0	0	7
Register	51	0	0	51
Home Page	2	0	0	2
Product Page	3	0	0	3
Order Products	9	0	0	9
FinalReportOutput	4	0	0	4
VersionControl	2	0	0	2

8.2 User Acceptance Testing

User acceptance testing (UAT), also called application testing or end-user testing, is a phase of software development in which the software is tested in the real world by its intended audience.

Defect Analysis

Resolution	Severity1	Severity2	Severity3	Severity4	Subtotal
By Design	5	5	2	3	15
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won'tFix	0	5	2	1	8
Totals	24	14	13	26	77

9.Result

project result means data, reports, deliverables, and any other Know-How developed or produced in the course of development activities performed under any project schedule excluding invention.

9.1 Performance Metrics

Performance metrics are defined as figures and data representative of an organization's actions, abilities, and overall quality. Performance metrics can vary considerably when viewed through different industries. Performance metrics are integral to an organization's success.

			NFT - Risk Assessment						
S. No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes	Risk Score	Justification
1	Smart Fashion Recommender Application	New	Low	No Changes	Moderate		>5 to 10%	ORANGE	As we have seen the changes

NFT - Detailed Test Plan				
S. No	Project Overview	NFT Test approach	Assumptions/Dependencies/Risks	Approvals/Signoff
1	Smart Fashion Recommender Application	Manual testing	Laptop or Mobile with Internet Connection	Nisha S

			End Of Test Report					
S. No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/Signoff
1	Smart Fashion Recommender Application	Manual		Worked as Expected		Use Laptop / Desktop Mode	No Defects	Nisha S

10. ADVANTAGES & DISADVANTAGES

Advantages

- Smart Fashion Recommender Application helps customer easily complete their transactions and also help them if any problems arise. These chatbots are specially designed to lead the customers from the starting which involves browsing the items on the E-commerce website and ending with purchasing to complete the transaction. If you are buying something on the application the chatbot can guide you through the whole process of online shopping.

- It is available to solve customer problems 24/7 whether it is day or night! This means that chatbots can answer customer queries whenever customers have queries which help in increasing customer loyalty.

- Whenever you interact with any chatbots on a company page, you provide basic data such as user preferences, buying habits, sentiments, etc. which can then be analyzed to understand market trends, operational risks, etc. And using this information, the company can solver customer issues much easier and create targeted products.

Disadvantages

- Chatbots are not human and so obviously they cannot interact as a human with customers. They sound too mechanical and can only give answers to problems that they have been programmed with. They cannot answer a customer according to the context and they cannot show any emotions if needed.

- As your user demand and business priorities shift, you'll need to update your bot accordingly, which requires analysis of previous bot conversations to identify common questions your bot receives. If you leave your bot to its own devices, your customers will notice and your customer satisfaction rating will suffer.

- Additionally, it is programmed to handle a specific amount of data, and as you update and edit the data, there can be disruptions to the chatbot model as a whole. This requires ongoing and careful maintenance to make sure you don't create holes in the chatbot interface.

11. CONCLUSION

In Fashion industry where luxury goods can only be bought in a few physical boutiques and one to one customer service is crucial. The Internet changed this dramatically, by giving the customers a smooth but a very detached experience of shopping. This particular problem can be solved by Chatbots. Customers can be provided with personalized services through Fashion chatbot, which can exchange messages, give required suggestions and information.

Bots use the power of conversation to make customers feel comfortable and quelch their need for “instant satisfaction” by tending to their queries immediately. They eliminate the necessity for shoppers to scroll down an endless product page to find what they need. Instead, they use the conversation as a filtering system, bringing the product to the customer. Other than that, chatbots can also be designed to remember previous interactions, purchases to offer personalized recommendations based on the buyer’s preferences and chat history. Having somebody to advise shoppers along the way helps bring them to the final stage of the sales funnel more smoothly.

You might think that customers don’t like sharing data, but providing a name or preferences feels much less intrusive within a conversation. More importantly, it allows you to observe, in real-time, in which part of the conversation users lose interest and fix the flow of the dialog accordingly. A Fashion recommendation chatbot also detects how shoppers react to new products, pricing, discounts... Data that usually gets lost on a static website.

12. FUTURE SCOPE

A chatbot is a strong medium for the online fashion industry. It fills the gap between customers and online retailers and allows them to connect with each other. It is not limited to chatting but also increases your business growth by capturing more new customers. Innovation will likely make conversational AI more accessible and affordable, allowing wider adoption.

Additionally, training will become easier as usage and adoption grow. One of the main issues with today's AI chatbots is that they remain expensive to train. Refined processes will help make sophisticated conversational versions more affordable. Multimodal chatbots to integrate visual media with textual chatbot interfaces.

Multimodal domain knowledge enriched fashion chatbot that understands the semantics of product image and modifies the attributes during back-end retrieval, offers matching suggestions, and generates responses with different modalities. Multimodality to consider extra-rational consumer factors such as attitudes, emotions, likes and dislikes, to provide refined recommendations accordingly.

Create Virtual assistants to mimic a salesperson to replicate the offline shopping experience. Another opportunity is the integration with Augmented Reality (AR) – e.g., in Virtual Fitting Room (VFR) applications. Integration of chatbots to other fashion applications in different points within the consumer journey: since chatbots can provide personalised information for consumers across their journey - e.g., virtually trying a recommended item on, shows potential.

13 Appendix

Source Code

```
from flask_session import Session
from flask import Flask, render_template, redirect, request, session, jsonify
from datetime import datetime
import ibm_db
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
from sendgrid.helpers.mail import To
dsn_hostname = "9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
dsn_uid = "cpw36749"
dsn_pwd = "v3tByUIyAxkVKukX"
dsn_driver = "{{IBM DB2 ODBC DRIVER}}"
dsn_database = "bludb"
dsn_port = "32459"
dsn_protocol = "TCPIP"
dsn_security = "SSL"
dsn = ("DRIVER={0};"
"DATABASE={1};"
"HOSTNAME={2};"
"PORT={3};"
"PROTOCOL={4};"
"UID={5};"
"PWD={6};"
"SECURITY={7};").format(dsn_driver,dsn_database,dsn_hostname,dsn_port,dsn_protocol,dsn_uid,dsn_pwd,dsn_security)
```

```
print(dsn)
try:
  conn = ibm_db.connect(dsn,"","")
  print("success")
except:
  print(ibm_db.conn_errormsg())

app = Flask(__name__)
app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"
Session(app)

@app.route("/", methods=["GET"])
def login():
    return render_template("login.html")



@app.route("/admin", methods=["GET"])
def admin():
    return render_template("admin.html")

@app.route("/best/", methods=["GET"])
def best():
          if not session.get("name"):
           return redirect("/")
          sql = "SELECT * FROM PROD_TBL"
          stmt1 = ibm_db.exec_immediate(conn, sql)
          row = ibm_db.fetch_assoc(stmt1)
          print(row)
          prodname = []
```

```
55          prodprice = []
56          prodimg = []
57          while row != False:
58            print (row["PROD_NAME"])
59            prodname.append(row["PROD_NAME"])
60            prodprice.append(row['PROD_PRICE'])
61            prodimg.append(row['PRO_IMG'])
62            row = ibm_db.fetch_assoc(stmt1)
63           return render_template ( "bestdeals.html"
  ,     len     =     len(prodname),     prodname     =
  prodname,prodprice=prodprice,prodimg=prodimg)
64
65@app.route("/male/", methods=["GET"])
66def male():
67          if not session.get("name"):
68            return redirect("/")
69              sql  =  "SELECT  *  FROM  PROD_TBL  WHERE
  PRO_CAT = 'Male'"
70          stmt1 = ibm_db.exec_immediate(conn, sql)
71          row = ibm_db.fetch_assoc(stmt1)
72          print(row)
73          prodname = []
74          prodprice = []
75          prodimg = []
76          while row != False:
77            print (row["PROD_NAME"])
78            prodname.append(row["PROD_NAME"])
79            prodprice.append(row['PROD_PRICE'])
80            prodimg.append(row['PRO_IMG'])
81            row = ibm_db.fetch_assoc(stmt1)
82           return render_template ( "maleshop.html"
```

```
, len = len(prodname), prodname = prodname,prodprice=prodprice,prodimg=prodimg)

@app.route("/Female/", methods=["GET"])
def female():
        if not session.get("name"):
          return redirect("/")
            sql = "SELECT * FROM PROD_TBL WHERE PRO_CAT = 'Female'"
        stmt1 = ibm_db.exec_immediate(conn, sql)
        row = ibm_db.fetch_assoc(stmt1)
        print(row)
        prodname = []
        prodprice = []
        prodimg = []
        while row != False:
          print (row["PROD_NAME"])
          prodname.append(row["PROD_NAME"])
          prodprice.append(row['PROD_PRICE'])
          prodimg.append(row['PRO_IMG'])
                                                row = ibm_db.fetch_assoc(stmt1)
                                                return render_template ( "Femaleshop.html" , len = len(prodname), prodname = prodname,prodprice=prodprice,prodimg=prodimg)

                                @app.route("/logged/",methods=["POST"])
                                def logged():
                                                user =
```

```
request.form["user"].lower()
106                                         pwd    =
request.form["pwd"]
107                                 logged.mailid  =
user
108                                if user == "" or
pwd == "":
109                                         return
render_template ( "login.html" )
110                                 query = "SELECT
* FROM USER_TBL WHERE username = '"+user+"' AND
password = '"+pwd+"'"
111                                         stmt   =
ibm_db.exec_immediate(conn, query)
112                                         rows   =
ibm_db.fetch_assoc(stmt)
113                                try:
114                                    if len(rows)
== 2:
115
session["name"] = user
116
print('ok')
117                                         return
redirect("/shop/")
118                                  else:
119                                         return
render_template ( "login.html", msg="Wrong username
or password." )
120                                except:
121                                         return
render_template ( "login.html", msg="Wrong username
```

```
or password." )
122
123                                @app.route("/loggedad/",methods=["POST"])
124                                def loggedad():
125                                            user    = request.form["user"]
126                                            pwd     = request.form["pwd"]
127                                if user == "" or pwd == "":
128                                            return render_template ( "login.html" )
129                                query5 = "SELECT * FROM ADMIN_TBL WHERE username = '"+user+"' AND password = '"+pwd+"'"
130                                            stmt5   = ibm_db.exec_immediate(conn, query5)
131                                            rows5   = ibm_db.fetch_assoc(stmt5)
132                                print(rows5)
133                                try:
134                                            session["name"] = user
135                                            if len(rows5) == 2:
136                                            print('ok')
137                                            return redirect("/add/")
138                                else:
139                                            return
```

```
  render_template ( "admin.html", msg="Wrong username
  or password." )
140                                 except:
141                                         return
  render_template ( "admin.html", msg="Wrong username
  or password." )
142
143
144                             @app.route("/shop/"
  , methods=['GET'])
145                             def shop():
146                                         if not
  session.get("name"):
147                                         return
  redirect("/")
148                                         sql =
  "SELECT * FROM PROD_TBL"
149                                         stmt1 =
  ibm_db.exec_immediate(conn, sql)
150                                         row =
  ibm_db.fetch_assoc(stmt1)
151                                         print(row)
152                                         prodname =
  []
153                                         prodprice
  = []
154                                         prodimg =
  []
155                                         while row
  != False:
156                                         print
```

```
(row["PROD_NAME"])
157
prodname.append(row["PROD_NAME"])
158
prodprice.append(row['PROD_PRICE'])
159
prodimg.append(row['PRO_IMG'])
160                                                row =
ibm_db.fetch_assoc(stmt1)
161
162                                              return
render_template ( "index.html" , len =
len(prodname), prodname =
prodname,prodprice=prodprice,prodimg=prodimg)
163
164                                @app.route("/Order/"
, methods=['GET'])
165                                def order():
166                                              if not
session.get("name"):
167                                              return
redirect("/")
168                                              sql =
"SELECT * FROM ORDHIST_TBL Where PUR_MAIL =
'"+logged.mailid+"'"
169                                            stmt2 =
ibm_db.exec_immediate(conn, sql)
170                                              row =
ibm_db.fetch_assoc(stmt2)
171                                           print(row)
172                                            prod_name
```

```
= []
173                                         pur_date =
  []
174                                         pur_mail =
  []
175                                          while row
  != False:
176                                              print
  (row["PROD_NAME"])
177
  prod_name.append(row["PROD_NAME"])
178
  pur_date.append(row['PUR_DATE'])
179
  pur_mail.append(row['PUR_MAIL'])
180                                              row =
  ibm_db.fetch_assoc(stmt2)
181
182                                            return
  render_template    (    "order.html"    ,    len    =
  len(prod_name),             prod_name             =
  prod_name,purc_date=pur_date,purc_mail=pur_mail)
183                              @app.route("/Orderhi
  s/" , methods=['GET'])
184                              def orderhis():
185                                            if not
  session.get("name"):
186                                            return
  redirect("/")
187                                             sql  =
  "SELECT * FROM ORDHIST_TBL"
188                                           stmt2  =
```

```
ibm_db.exec_immediate(conn, sql)
189                                                 row =
ibm_db.fetch_assoc(stmt2)
190                                             print(row)
191                                              prod_name
= []
192                                             pur_date =
[]
193                                             pur_mail =
[]
194                                              while row
!= False:
195                                                 print
(row["PROD_NAME"])
196
prod_name.append(row["PROD_NAME"])
197
pur_date.append(row['PUR_DATE'])
198
pur_mail.append(row['PUR_MAIL'])
199                                                 row =
ibm_db.fetch_assoc(stmt2)
200
201                                             return
render_template ( "orderhis.html" , len =
len(prod_name),          prod_name          =
prod_name,purc_date=pur_date,purc_mail=pur_mail)
202
203
204                                 @app.route("/modal/"
, methods=['GET'])
```

```
205                                def modal():
206                                        pro_name1 =
  request.args.get('pro_name1')
207
  print(pro_name1)
208
  modal.proname1 = pro_name1
209                                            date  =
  datetime.now()
210                                       modal.mailid
  = logged.mailid
211                                     sendmail()
212                                       sql_stmt3 =
  "insert into ORDHIST_TBL values(?, ?, ?)"
213                                          stmt3   =
  ibm_db.prepare(conn, sql_stmt3)
214
  ibm_db.bind_param(stmt3, 1, pro_name1)
215
  ibm_db.bind_param(stmt3, 2, date)
216
  ibm_db.bind_param(stmt3, 3, logged.mailid)
217                                     try:
218
  ibm_db.execute(stmt3)
219                                            return
  render_template("added.html",pro_name1 = pro_name1)
220                                     except:
221
  print(ibm_db.stmt_errormsg())
222
```

```
223
224
225
226                                 @app.route("/add/", methods=["GET"])
227                                 def add():
228                                             if not session.get("name"):
229                                             return redirect("/")
230                                             return render_template("Add_product.html")
231
232
233                                 @app.route("/logout/", methods=["GET"])
234                                 def logout():
235                                         session["name"] = None
236                                             return render_template("login.html")
237
238                                 @app.route("/register/", methods=["GET"])
239                                 def register():
240                                             return render_template("Register.html")
241
242                                 @app.route("/registered/", methods=["POST"])
243                                 def registered():
```

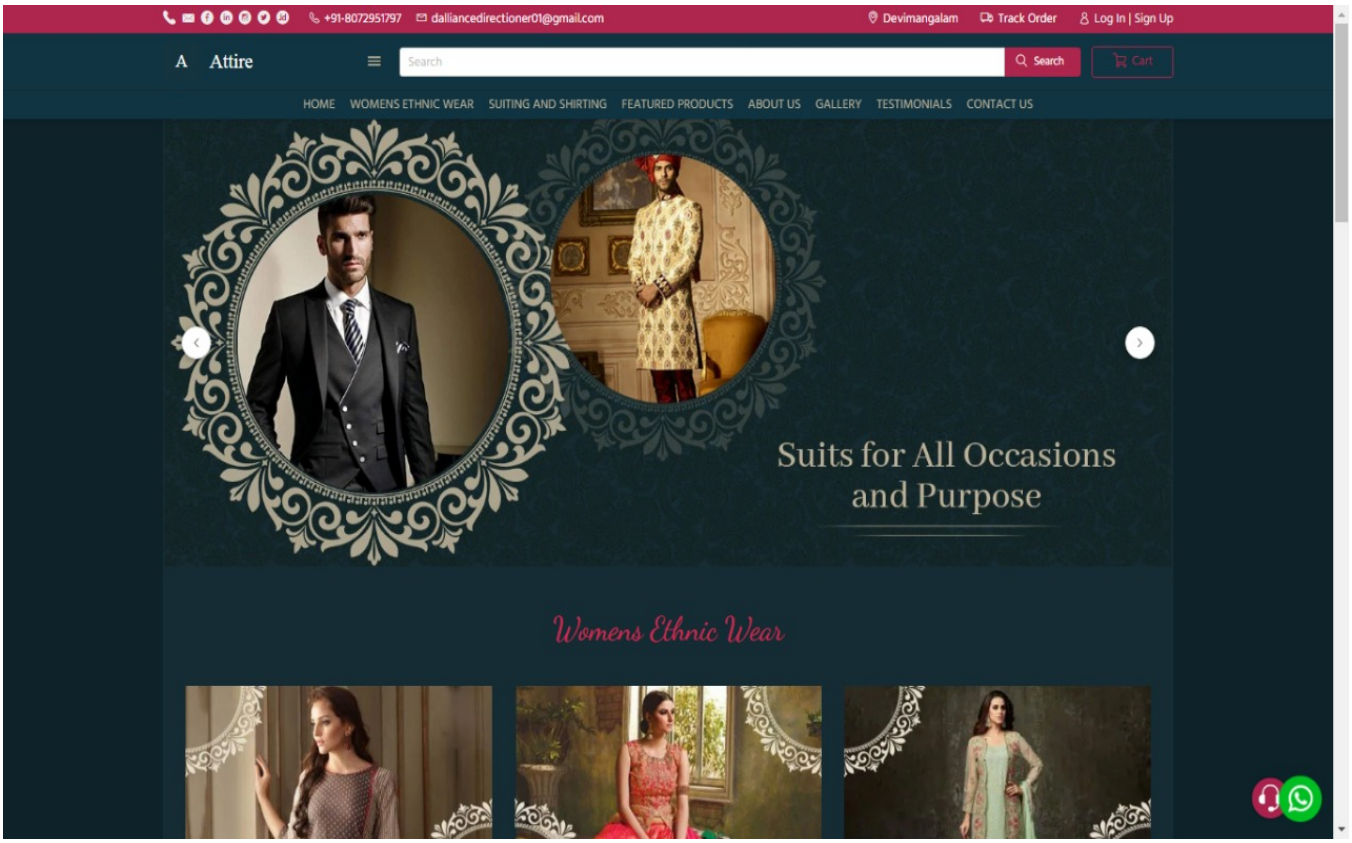
```
                                        username = request.form["username"]
                                        password = request.form["pass"]
                                    sql2 = "insert into USER_TBL values(?,?)"
                                        stmt7 = ibm_db.prepare(conn, sql2)
                                        ibm_db.bind_param(stmt7, 1, username)
                                        ibm_db.bind_param(stmt7, 2, password)
                                    try:
                                        ibm_db.execute(stmt7)
                                        return render_template("login.html",msg = "Added Successfully")
                                    except:
                                        print(ibm_db.stmt_errormsg())
                                        return render_template("Register.html",msg = "tryagin")


                                @app.route('/sendmail')
                                def sendmail():
                                    str
                                        message = Mail(from_email='713319cs124@smartinternz.com',
```

```
262                                                to_emails=
  To(logged.mailid),
263                                            subject='Order
  Confirmed! Fashion Recommender',
264
  html_content='<pre>Thank you for your order! The
  estimated delivery date is based on the handing
  time and the warehouse processing time in certain
  cases, the estimated delivery date will vary.<br>
  You will receive a tracking number by email once
  your package ships.<br> You can check the status of
  your order on our App <br>Find your order
  confirmation below. Thank you again for ordering
  from Glamtique,</pre><br> For any changes to this
  order, contact Order Help Desk<br> <strong>Order
  Item Name :: '+modal.proname1+' </strong><br>If it
  is not you, please contact our
  support.<strong>Thank You</strong>')
265                                                sg      =
  SendGridAPIClient('SG.OrJwFZMBR72cS7mqBrzFJw.T3-
  Wl8wZRdegZUKFrypAc3Plqn6pYl6rRnEbyrJ6IS0')
266                                            response    =
  sg.send(message)
267
  print(response.status_code)
268
  print(response.body)
269
  print(response.headers)
270                                    # except Exception
  as e:
```

```
271                                                                    #
  print(e.message)
272
273                                @app.route("/added/
  ", methods=["POST"])
274                                def added():
275                                        prod_name    =
  request.form["prodname"]
276                                      prod_price    =
  request.form["prodprice"]
277                                       prod_cat     =
  request.form["prodcat"]
278                                       prod_img     =
  request.form["prodimg"]
279                                   sql1   =  "insert
  into PROD_TBL values(?,?,?,?)"
280                                          stmt6     =
  ibm_db.prepare(conn, sql1)
281
  ibm_db.bind_param(stmt6, 1, prod_name)
282
  ibm_db.bind_param(stmt6, 2, prod_price)
283
  ibm_db.bind_param(stmt6, 3, prod_img)
284
  ibm_db.bind_param(stmt6, 4, prod_cat)
285                                  try:
286
  ibm_db.execute(stmt6)
287                                              return
  render_template("Add_product.html",msg    =    "Added
```

```
Successfully")
except:
print(ibm_db.stmt_errormsg())
return render_template("Add_product.html",msg = "tryagin")
if __name__ == "__main__":
app.run(host='0.0.0.0')
```

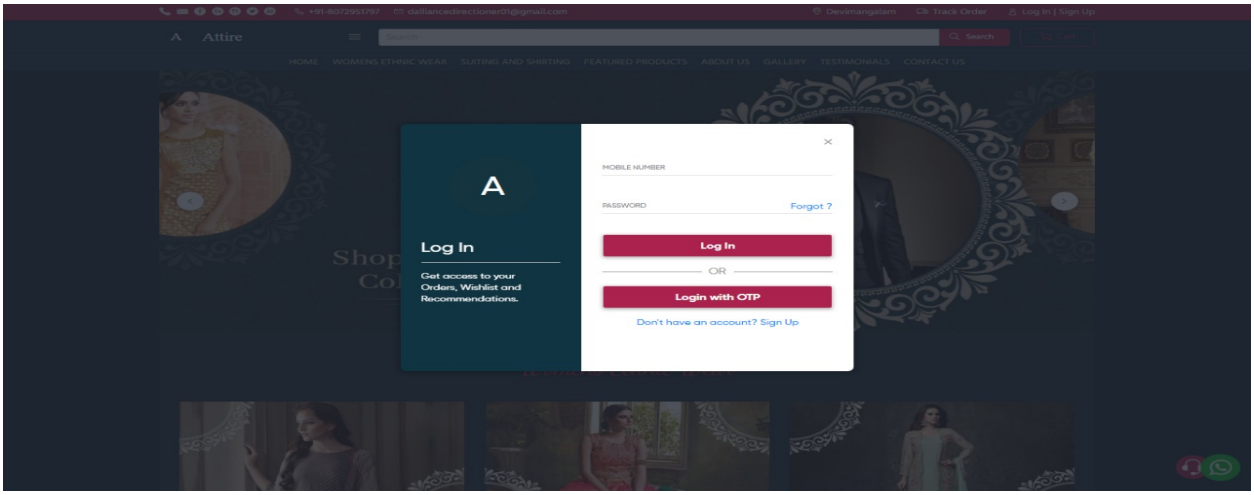


Home Page

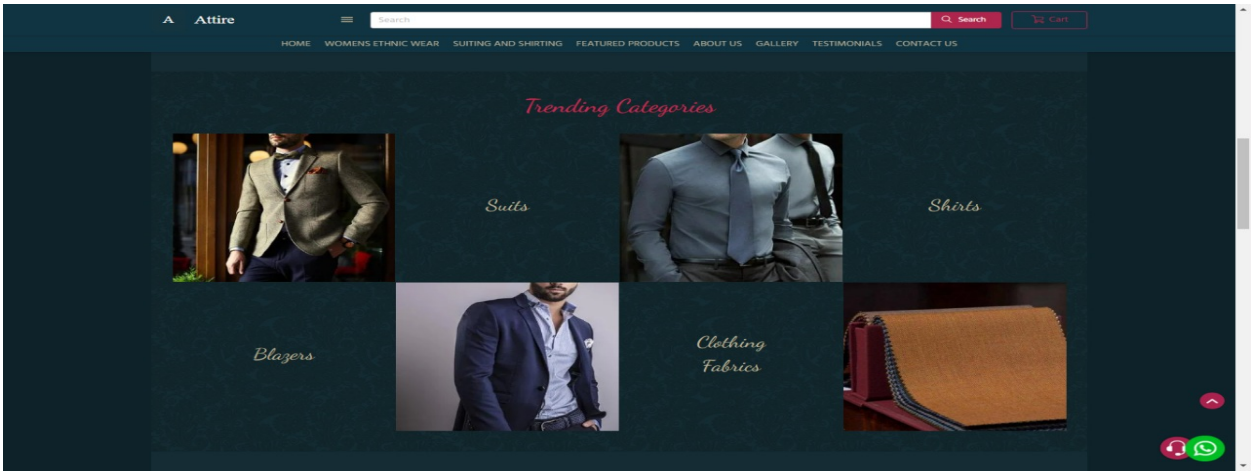
Login.html

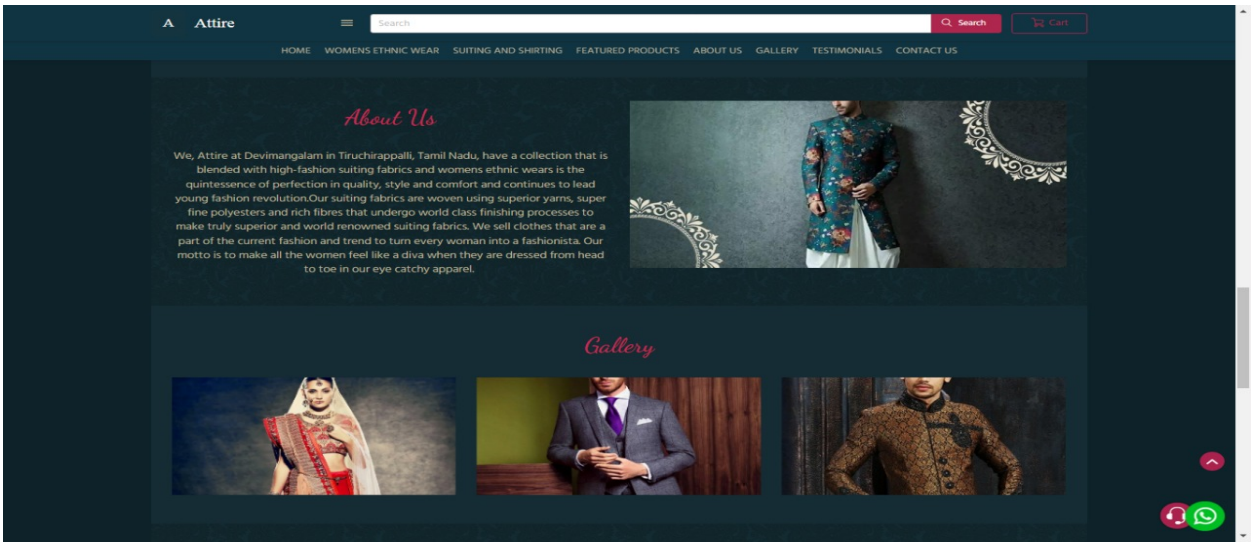
```
1  <html>
2  <head>
3  <meta      name="viewpoint"      content="width=device-width,
   initial-scale=1.0">
4  <title>Retro Walk</title>
5  <link       rel="stylesheet"       href="https://storagedemo-
   madzh.s3.jp-tok.cloud-object-
   storage.appdomain.cloud/Regcss.css">
6  </head>
7  <body>
8  <div class="main">
9  <div class="navbar">
10 <div class="menu">
11 <ul>
12 </ul>
13 </div>
14 </div>
15 <div class="content">
16 <h1> SMART FASHION <br><span>RECOMMENDER</span></h1>
17 <div class="form">
18 <h2>LOGIN</h2>
19 <form action="/Login" method="post">
20 <input    type="text"    name="username"    placeholder="Enter
   Username">
21 <input  type="password"  name="password"  placeholder="Enter
   Password">
22 <button              type="SUBMIT"              class="btnn"><a
   href="#">LOGIN</a></button>
23 <p class="link">DON'T HAVE AN ACCOUNT<br>
24 <a href="/Register">Sign Up </a> HERE </a></p>
25 </form>
26 </div>
27 </div>
28 </body>
29 </html>
```

Login Page

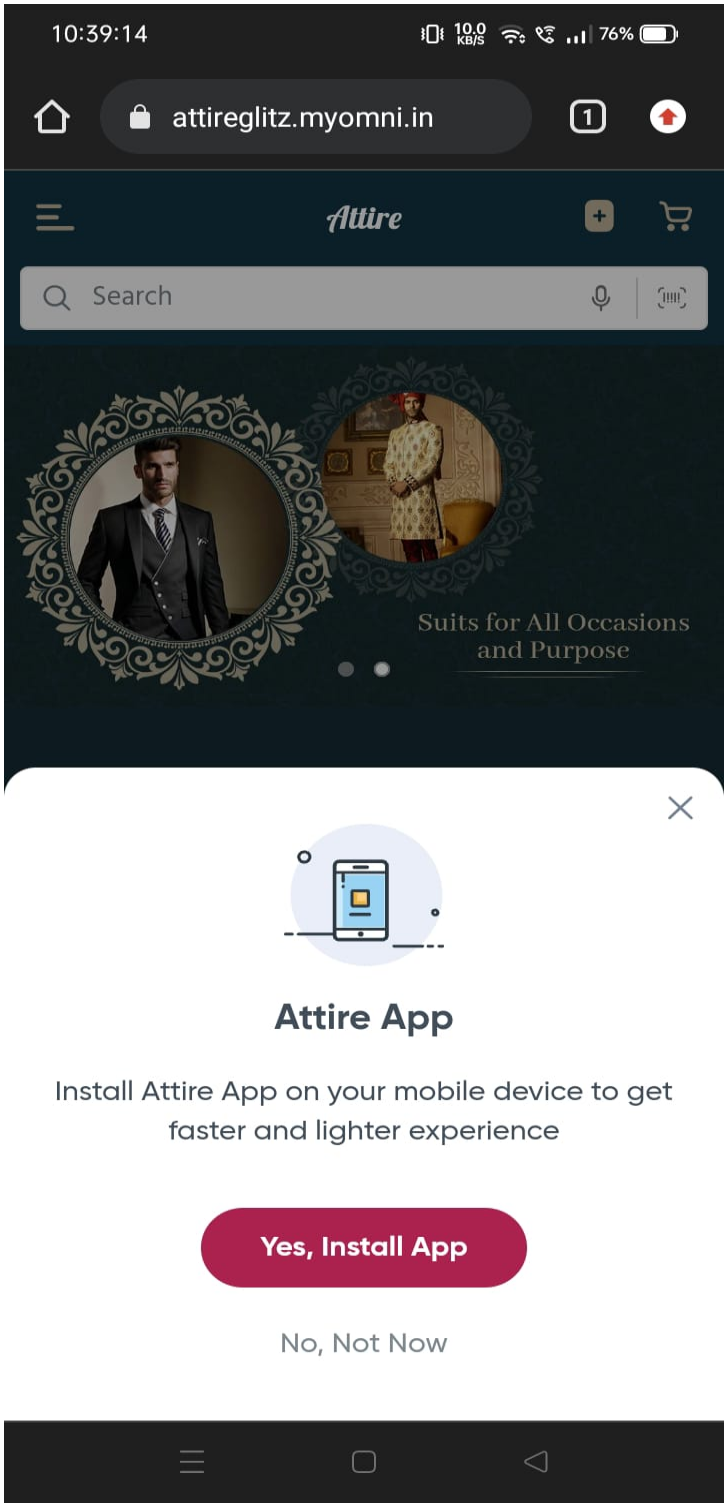


Product Page

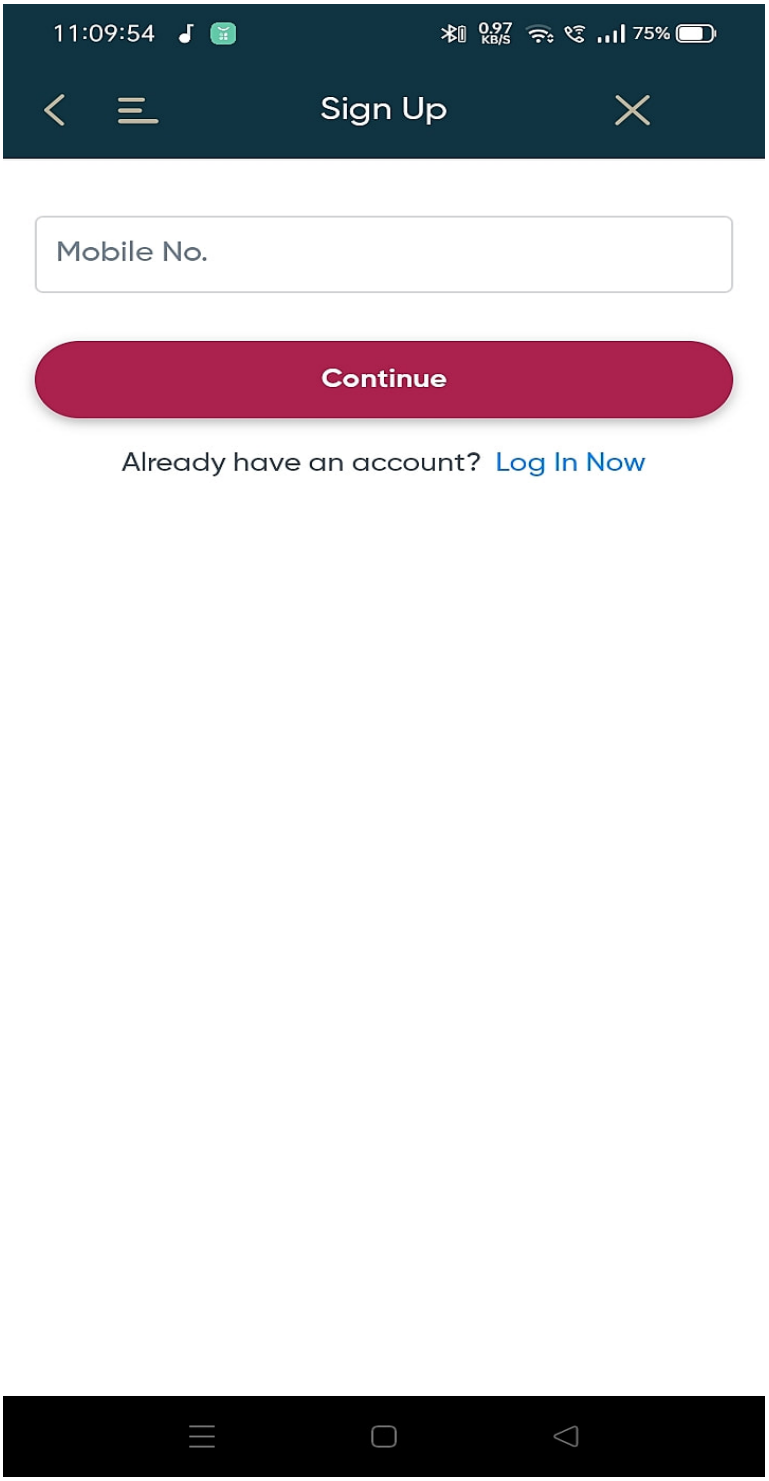




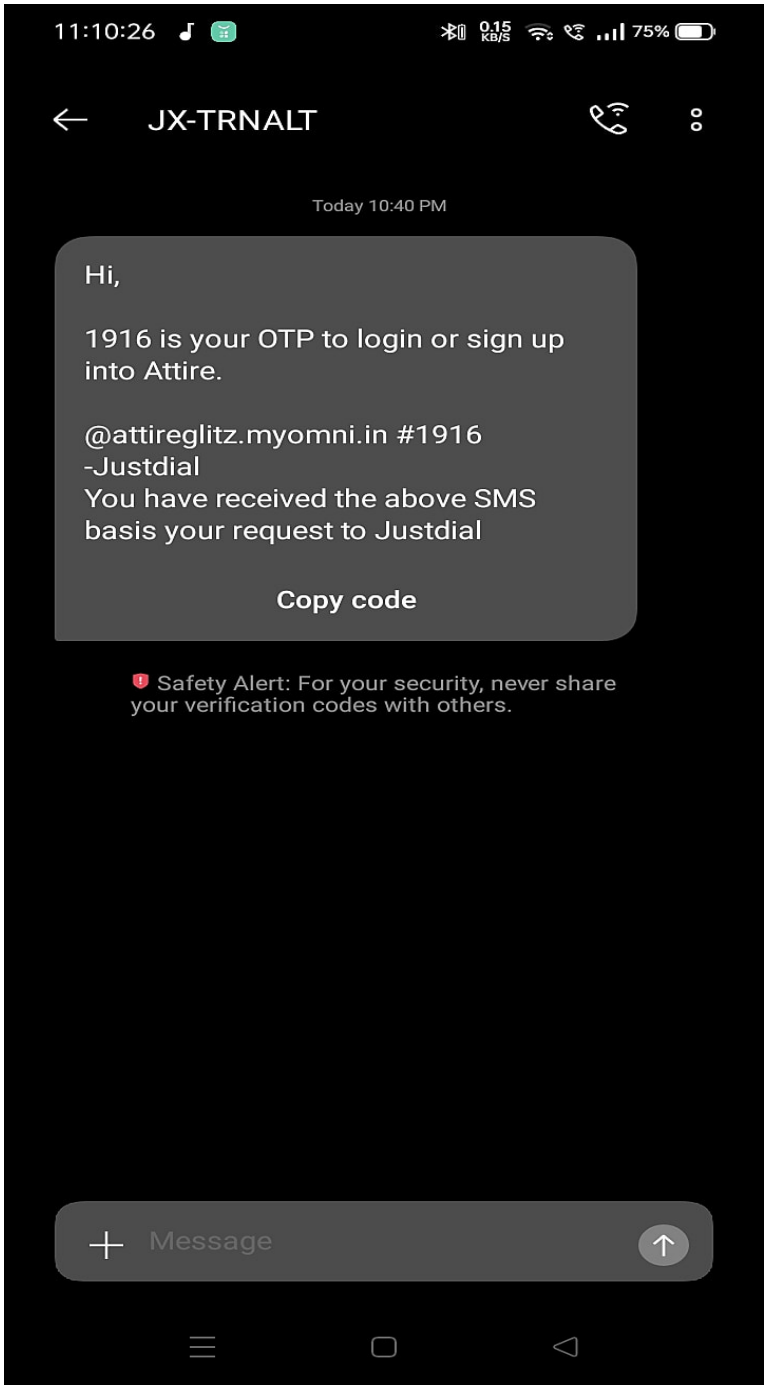
After clicking a link of our web app you can get a popup in chrome page to install our app.



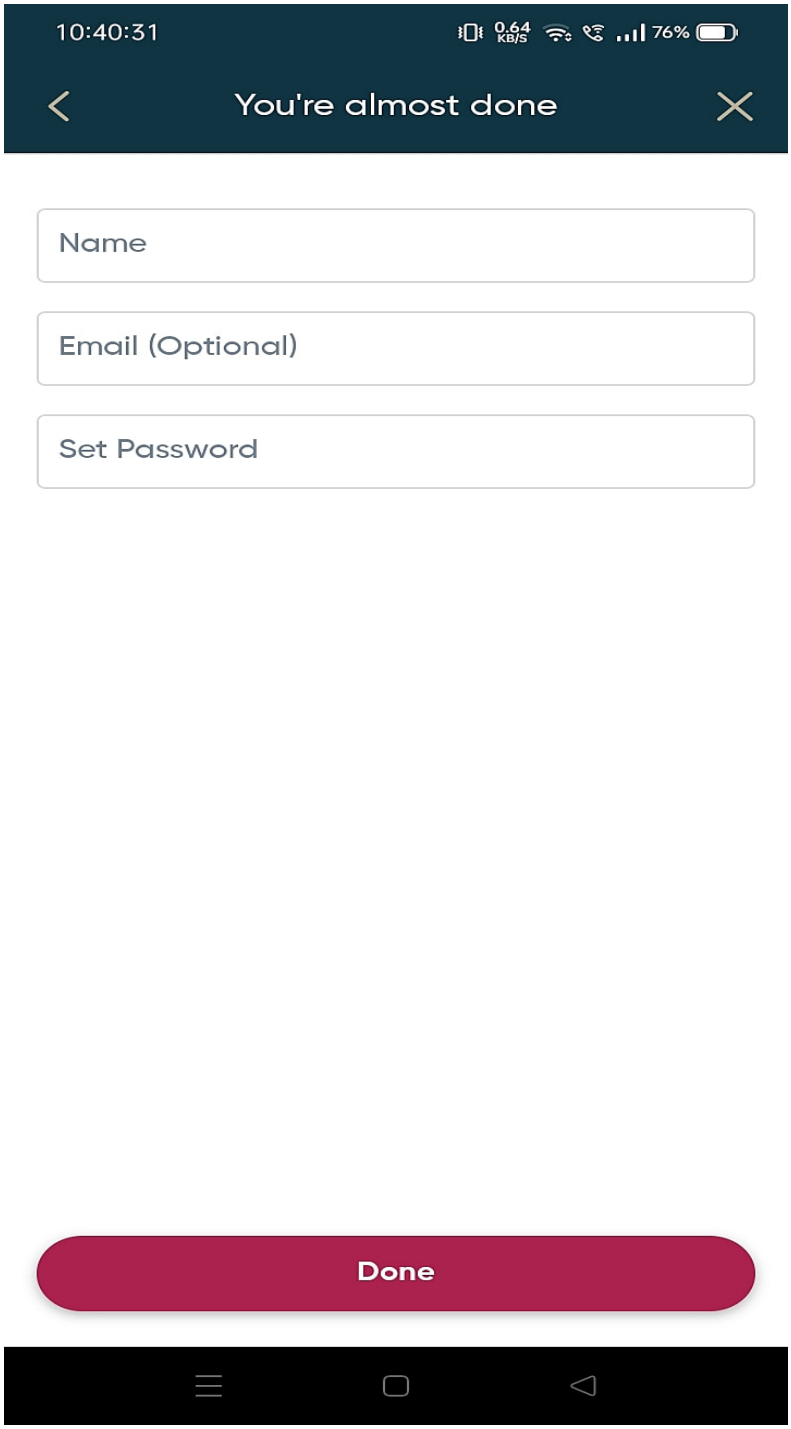
After completing installation process, you need to enter phone number to create an account in our app



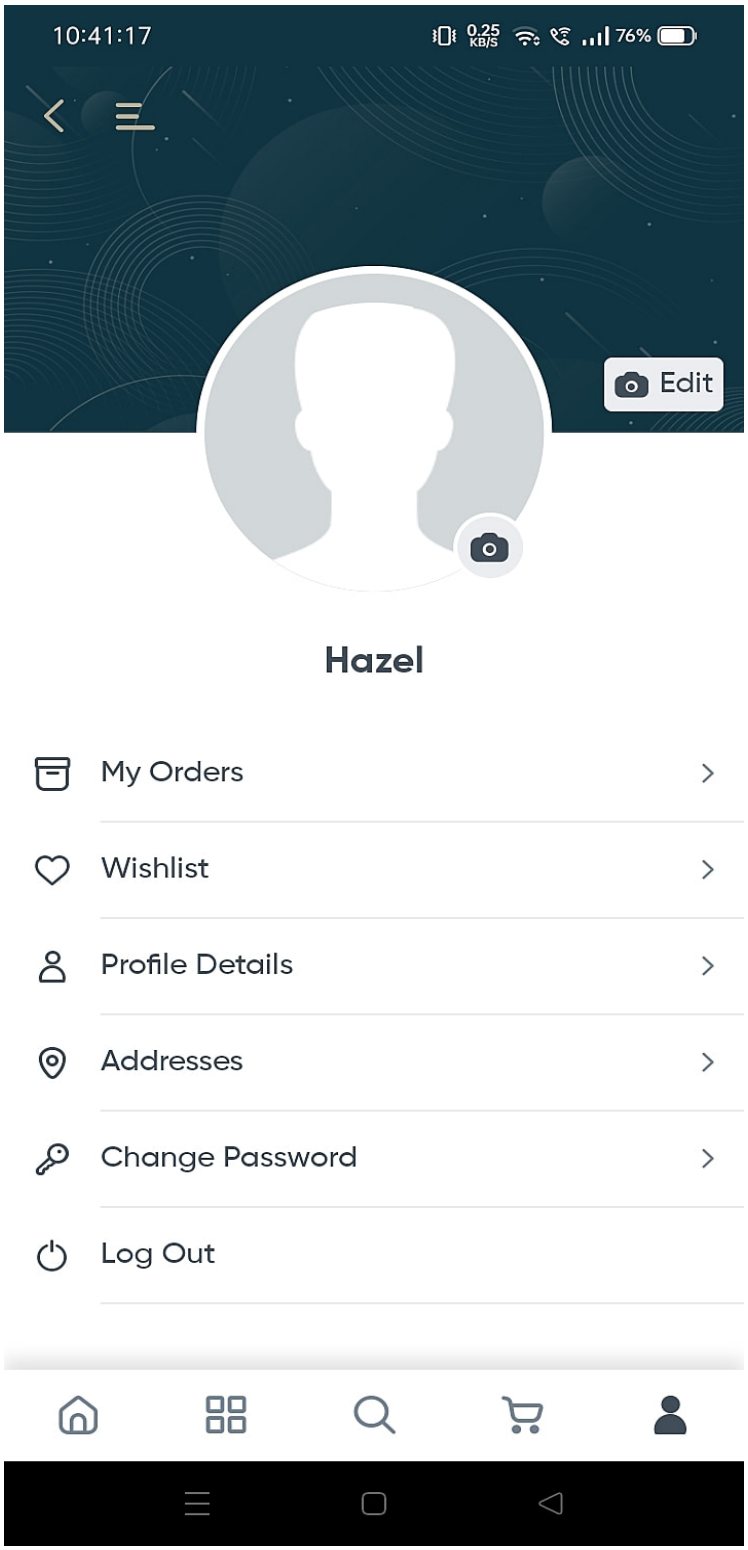
After filling a phone number you will gat a otp through a message and fill the otp to create an account



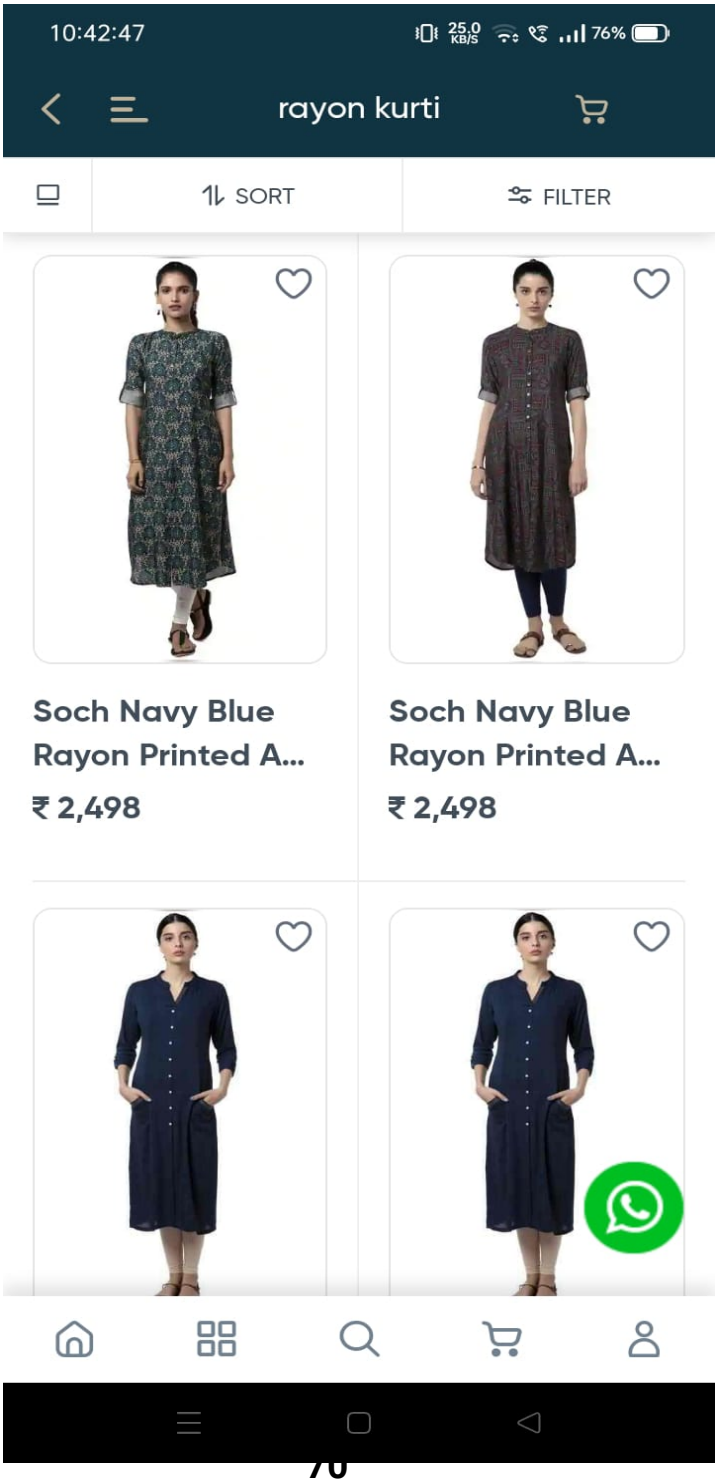
To create an account in our app you need to fill details (Name, Email ID & set password)



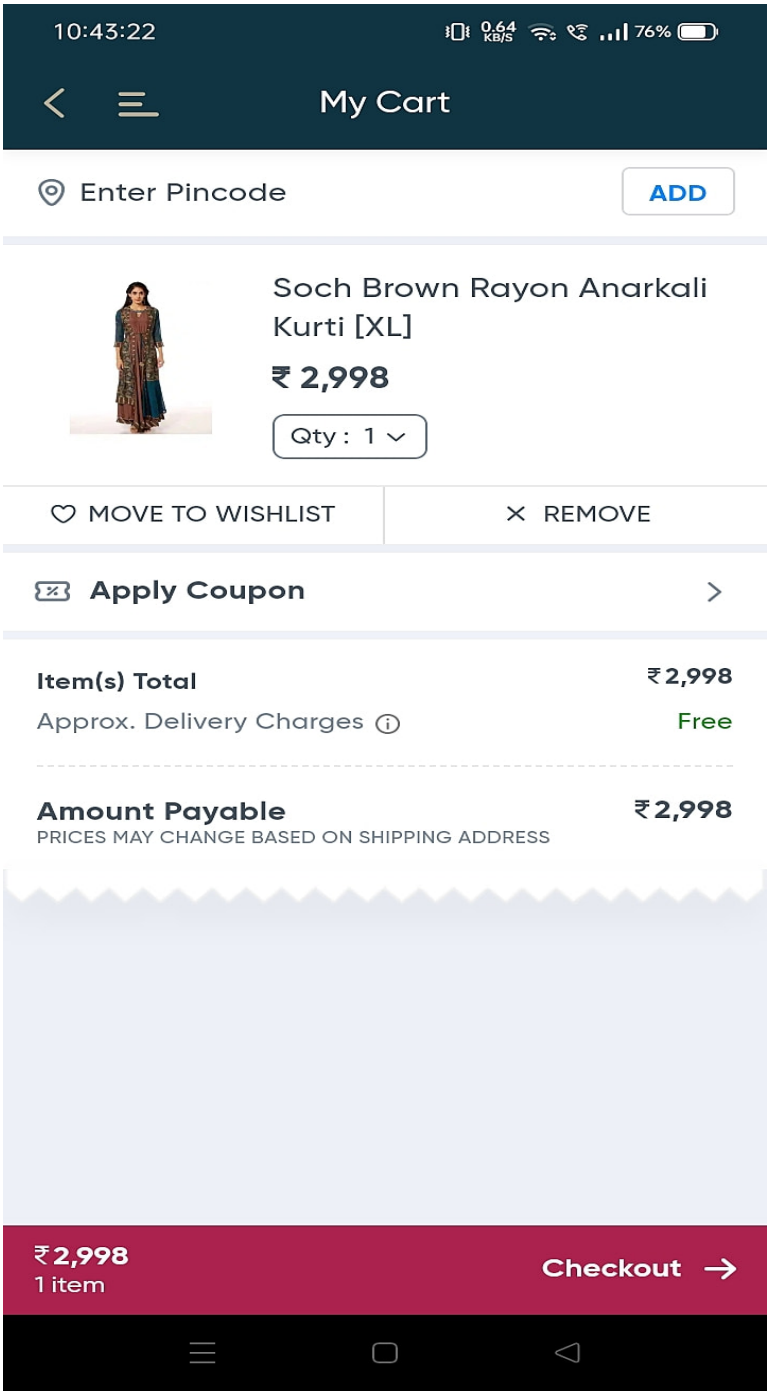
You will get an account page of yours In our web app



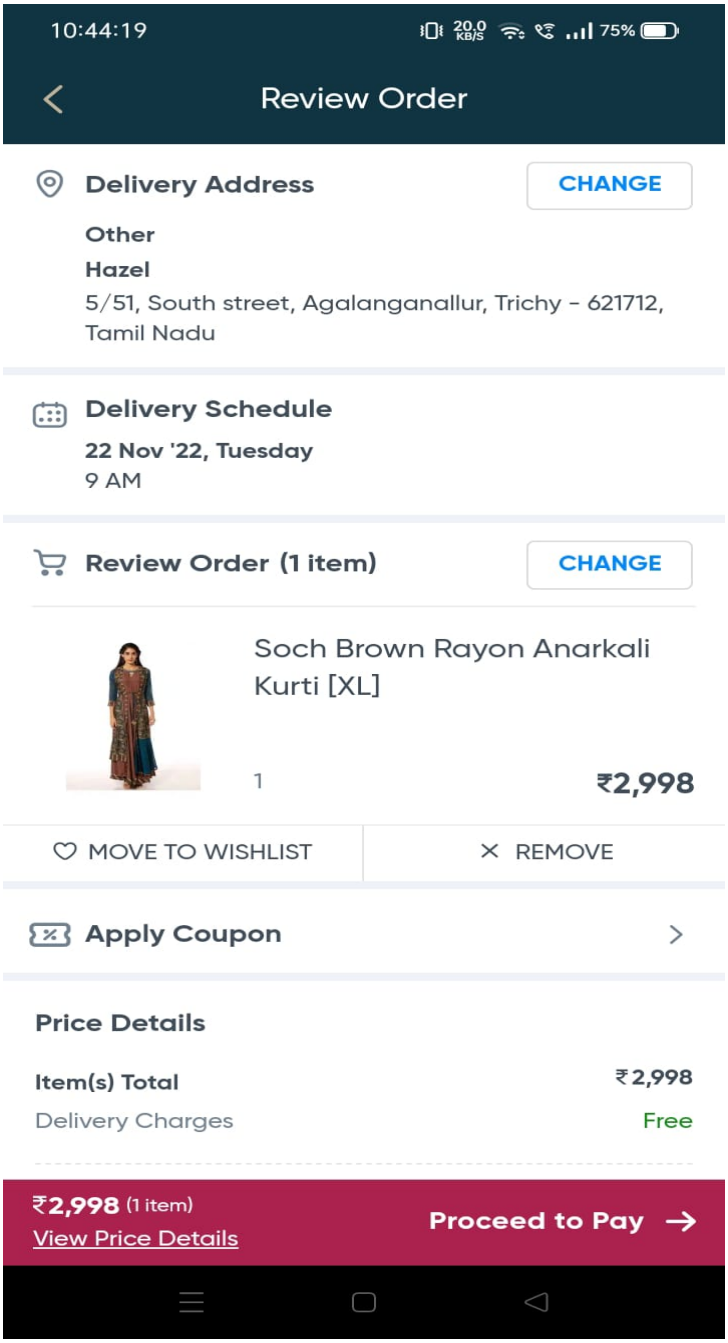
After creating an accont you will get our web app with your page with your access. You can get the products by manual search



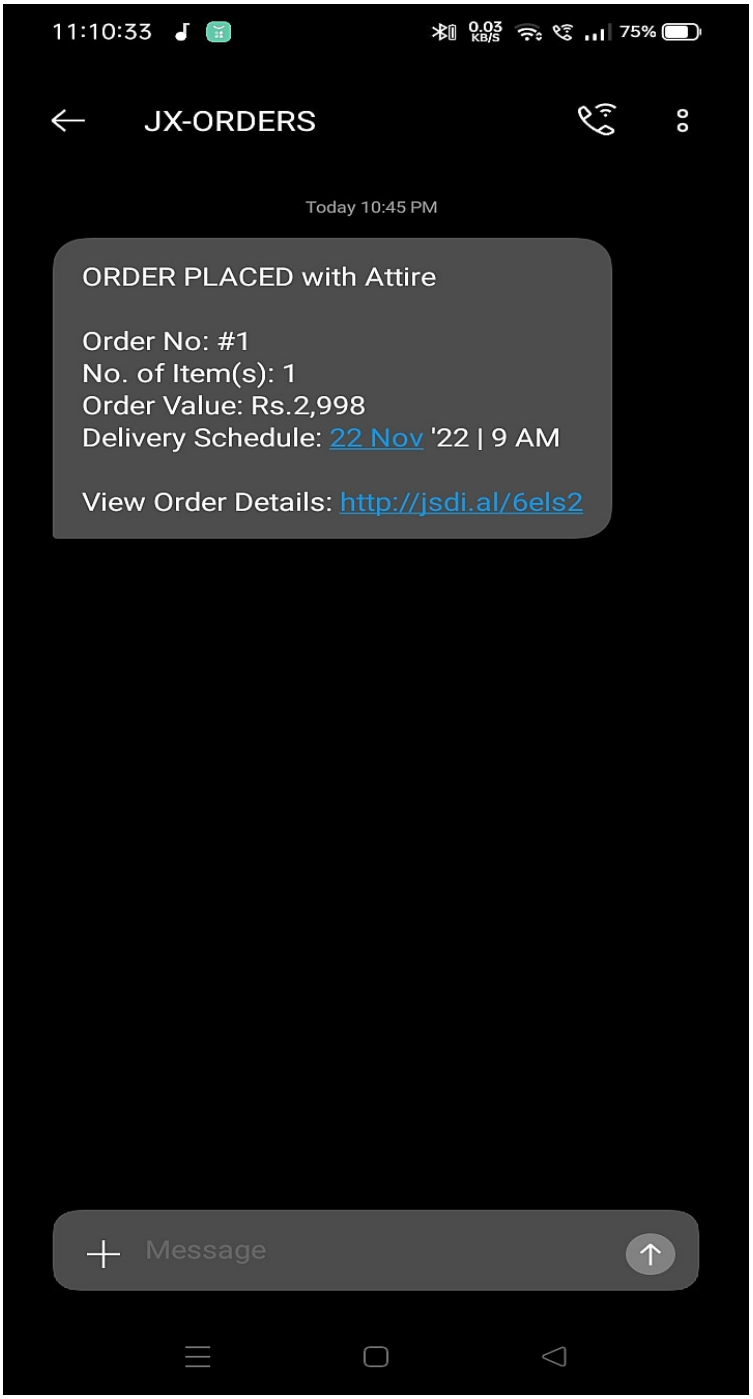
After selecting a particular product based on your need, you have to add that product to cart and checkout the product by filling a address details of yours



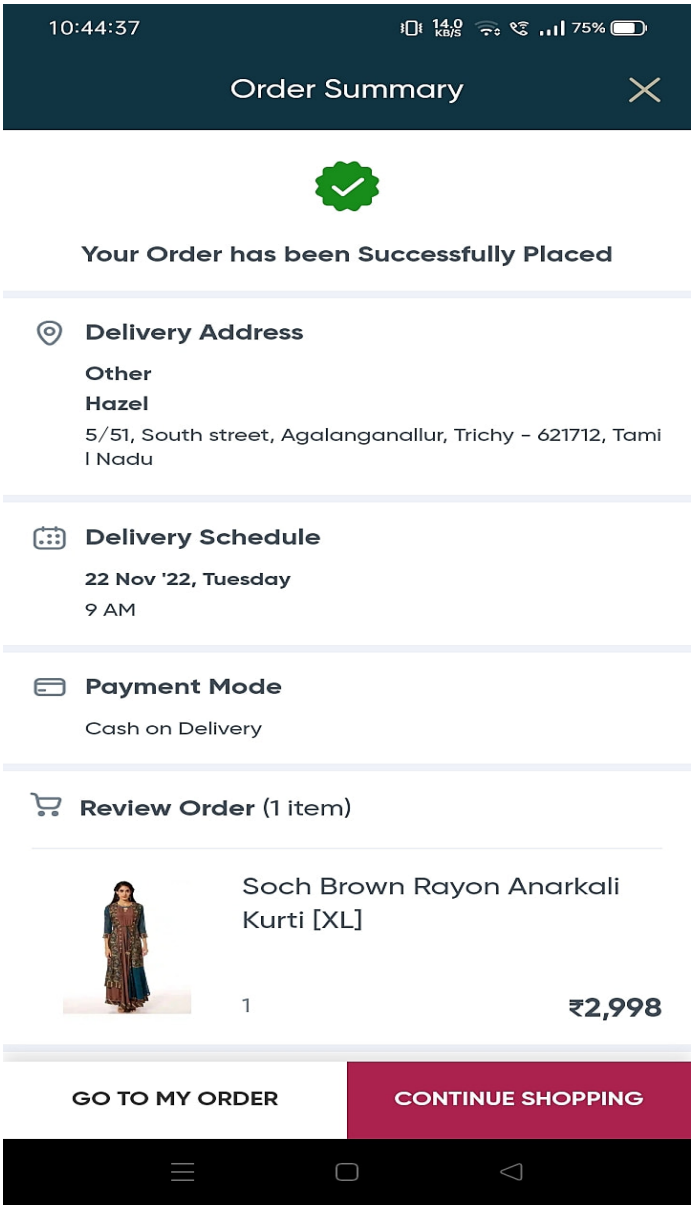
After filling a address details of yours and checkout the product by choosing cash on delivery



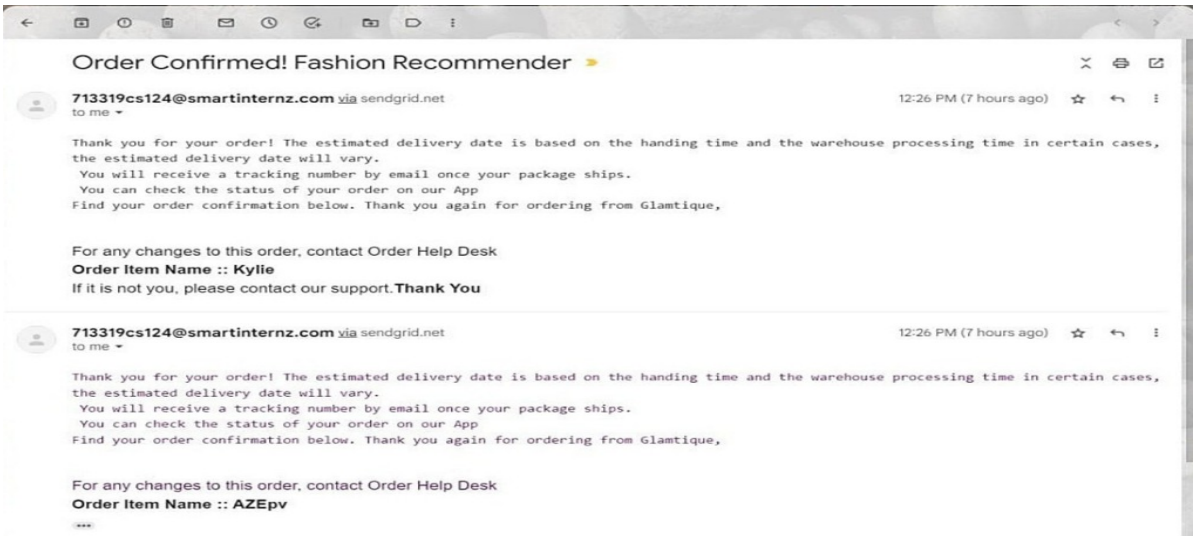
After that you will get a mail and message of your order details to your phone number and mail ID



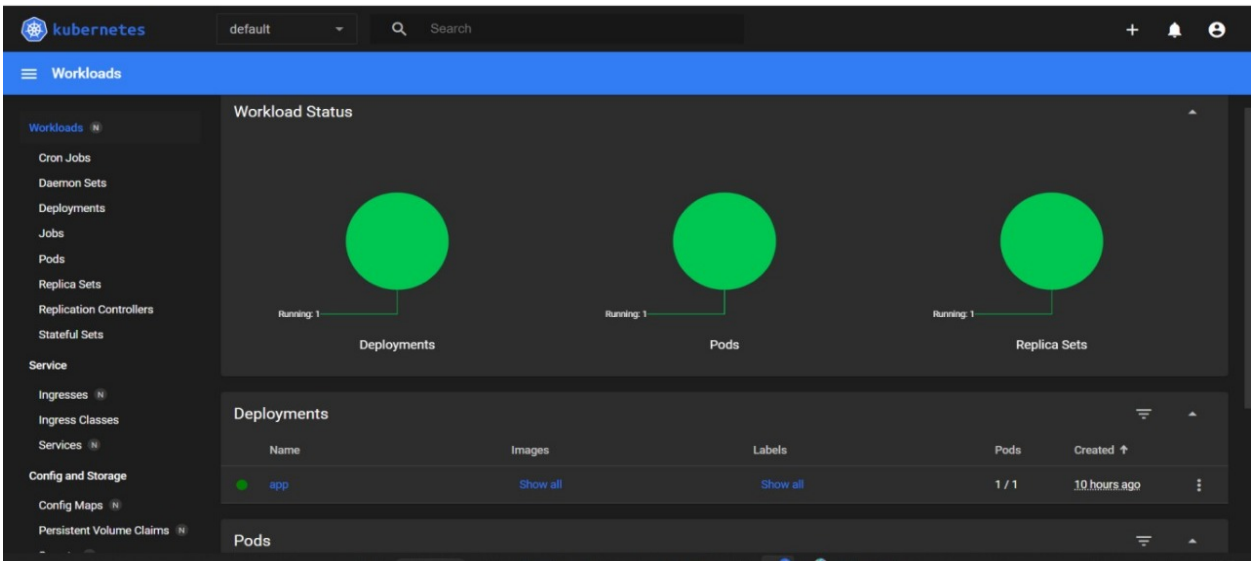
Your will get a order details by clicking link which was send by us to your mailID



By using SendGrid we can able to send a mail of order details to the users



Finally Deploy the app in IBM Cloud (Containerixe the Flask app by using Docker and deploy it to Kubernetes service)



GitHub & Project Demo Link

GitHub Repository Link : https://github.com/IBM-EPBL/IBM-Project-30353-1660144568

Demonstration Vedio

Drive Link : https://drive.google.com/file/d/1zJeUVZ0hq4PK3UmS-PTrWOgWRagSprlU/view?usp=drivesdk

Youtube Link : https://youtu.be/nVIPvCHZrls

SMART FASHION RECOMMENDER APPLICATION

Team ID : PNT2022TMID46176