SMART FASHION RECOMMENDER APPLICATION

REPORT

Submitted by Aakash kumar RMR LEADER Sathish kumar G MEMBER 1 Srinath P MEMBER 2 Shanmugasundaram S MEMBER 3

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING ELECTRONIC AND COMMUNICATION ENGINEERRING UNIVERSITY COLLEGE OF ENGINEERING VILLUPURAM 2 NOV- 2022

ABSTRACT

Fashion applications have seen tremendous growth and are now one of the most used programs in the e-commerce field. The needs of people are continuously evolving, creating room for innovation among the applications. One of the tedious processes and presumably the main activities is choosing what you want to wear. Having an AI program that understands the algorithm of a specific application can be of great aid. We are implementing such a chat bot, which is fed with the

knowledge of the application’s algorithm and helps the user completely from finding their needs to processing the payment and initiating delivery. It works as an advanced filter search that can bring the user what they want with the help of pictorial and named representation. The application also has two main user interfaces - the user and the admin. The users can interact with the chat bot, search for products, order them from the manufacturer or distributor, make payment transactions, track the delivery, and so on. The admin interface enables the user to upload products, find how many products have been bought, supervise the stock availability and interact with the buyer regarding the product as reviews. The rapid progress of computer vision, cloud computing and artificial intelligence combined with the current growing urge for online shopping systems opened an excellent opportunity for the fashion industry. As a result, many studies worldwide are dedicated to modern fashion related applications such as virtual try-on and fashion synthesis. Traditionally, keywords are used to retrieve images, but such methods require a lot of annotations on the image data, which will lead to serious problems such as inconsistent, inaccurate, and incomplete descriptions, and a huge amount of work. However, the accelerated evolution speed of the field makes it hard to track these many research branches in a structured framework. Such hierarchical application-based multi-label classification of studies increases the visibility of current research, promotes the field, provides research directions, and facilitates access to related studies. Project

Report Format

1. INTRODUCTION - 6 3
   1. Project Overview
   2. Purpose
2. LITERATURE SURVEY – 7
   1. Existing problem
   2. References
   3. Problem Statement Definition
3. IDEATION & PROPOSED SOLUTION -11
   1. Empathy Map Canvas
   2. Ideation & Brainstorming
   3. Proposed Solution
   4. Problem Solution fit
4. REQUIREMENT ANALYSIS -18 –
   1. Functional requirement
   2. Non-Functional requirements
5. PROJECT DESIGN -21
   1. Data Flow Diagrams
   2. Solution & Technical Architecture
   3. User Stories
6. PROJECT PLANNING & SCHEDULING -25
   1. Sprint Planning & Estimation
   2. Sprint Delivery Schedule
   3. Reports from JIRA
7. CODING & SOLUTIONING (Explain the features added in the project along with code) -30 4
   1. Feature 1
   2. Feature 2
   3. Database Schema (if Applicable)
8. TESTING -27
   1. Test Cases
   2. User Acceptance Testing
9. RESULTS
   1. Performance Metrics
10. ADVANTAGES & DISADVANTAGES
11. CONCLUSION
12. FUTURE SCOPE
13. APPENDIX Source Code GitHub & Project Demo Link 1.

INTRODUCTION 1.1

Project Overview Instead of searching for products in the search bar and navigating to individual products to find required preferences, this project leverages the use of chatbots to gather all required preferences and recommend products to the user. The solution is implemented in such a way as to improve the interactivity between customers and applications. The chatbot sends messages periodically to notify offers and preferences. For security concerns, this application uses a token to authenticate and authorize users securely. The token has encoded user id and role. Based on the encoded information, access to the resources is restricted to specific users. User is divided base don their roles. The roles comprises of admin and user. Admin is provided access with adding new products, update a product track user details. Users of the application get periodic recommendation via chatbot base don their preference and search. The main purpose of this application is to make process of searching, filtering and ordering of products simple and quickly that enhances the overall user experience. The chatbot is trained by providing different categories of product information and its related details.

* 1. Purpose of the Project Fashion applications have seen tremendous growth and are now one of the most used programs in the e commerce field. The needs of people are continuously evolving, creating room for innovation among the applications. One of the tedious processes and presumably the main activities is choosing what you want to wear. Having an AI program that understands the algorithm of a specific application can be of great aid. We are implementing such a chatbot, which is fed with the knowledge of the application’s algorithm and helps the user completely from finding their needs to processing the payment and initiating delivery. It works as an advanced filter search that can bring the user what they want with the help of pictorial and named representation. The application also has two main user interfaces - the user and the admin. The users can interact with the chatbot, search for products, order them from the manufacturer or distributor, make payment transactions, track the delivery, and so on. The admin interface enables the user to upload products, find how many products have been bought, supervise the stock availability and interact with the buyer regarding the product as reviews. In E-commerce websites, users need to search for products and navigate across screens to view the product, add them to the cart, and order products. The smart fashion recommender application leverages the use of a chat bot to interact with the users, gather information about their preferences, and recommend suitable products to the users. This application has two predefined roles assigned to the users. The roles are customer and admin. The application demands redirection of the user to the appropriate dashboard based on the assigned role. Admin should be able to track the number of different products and admin should be assigned the responsibility to create products with appropriate categories. The user should be able to

mention their preferences using interacting with chat bots. The user must receive a notification on order confirmation/failure. The chat bot must gather feedback from the user at the end of order confirmation. The main objective of this application is to provide better interactivity with the user and to reduce navigating pages to find appropriate produc

## SMART FASHION RECOMMENDER

**Abstract:**

Fashion applications have seen tremendous growth and are now one of the most used programs in the e-commerce field. The needs of people are continuously evolving, creating room for innovation among the applications. One of the tedious processes and presumably the main activities is choosing what you want to wear. Having an AI program that understands the algorithm of a specific application can be of great aid. We are implementing such a chat bot, which is fed with the knowledge of the application’s algorithm and helps the user completely from finding their needs to processing the payment and initiating delivery. It works as an advanced filter search that can bring the user what they want with the help of pictorial and named representation. The application also has two main user interfaces - the user and the admin. The users can interact with the chat bot, search for products, order them from the manufacturer or distributor, make payment transactions, track the delivery, and so on. The admin interface enables the user to upload products, find how many products have been bought, supervise the stock availability and interact with the buyer regarding the product as reviews.

## Introduction

Problem Statement

In E-commerce websites, users need to search for products and navigate across screens to view the product, add them to the cart, and order products. The smart fashion recommender application leverages the use of a chat bot to interact with the users, gather information about their preferences, and recommend suitable products to the users. This application has two predefined roles assigned to the users. The roles are customer and admin. The application demands redirection of the user to the appropriate dashboard based on the assigned role. Admin should be able to track the number of different products and admin should be assigned the responsibility to create products with appropriate categories. The user should be able to mention their preferences using interacting with chat bots. The user must receive a notification on order confirmation/failure. The chat bot must gather feedback from the user at the end of order confirmation. The main objective of this application is to provide better interactivity with the user and to reduce navigating pages to find appropriate products.

Project Description

We have developed a new innovative solution through which you can directly do your online shopping based on your choice without any search. It can be done by using the chat bot. In this project you will be working on two modules:

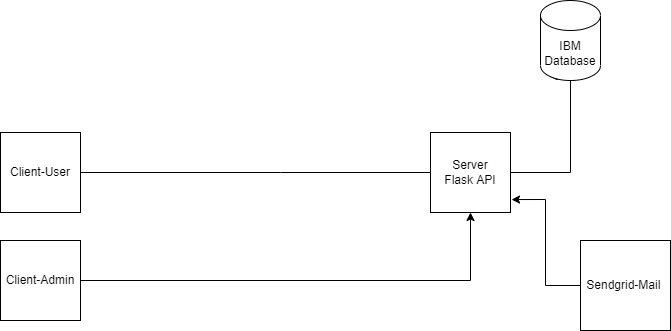
* Admin and
* User

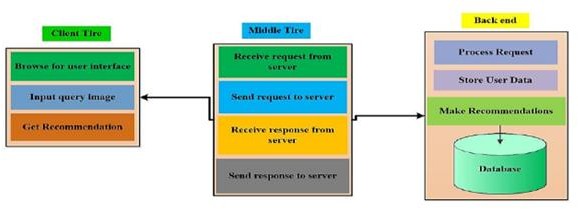
The methodology used in this solution

Instead of searching products in the search bar and navigating to individual products to find required preferences, this project leverages the use of chat bots to gather all required preferences and recommend products to the user. The solution is implemented in such a way as to improve the interactivity between customers and applications. The chat bot sends messages periodically to notify offers and preferences. For security concerns, this application uses a token to authenticate and authorize users securely. The token has encoded user id and role. Based on the encoded information, access to the resources is restricted to specific users.

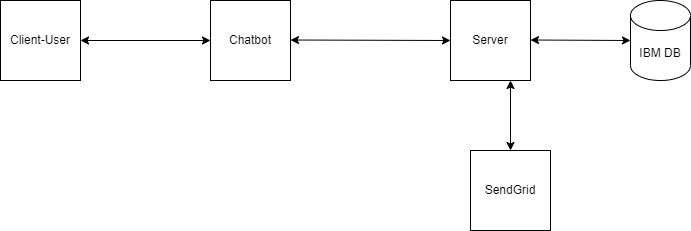
System Architecture:

**Overall Architecture:**

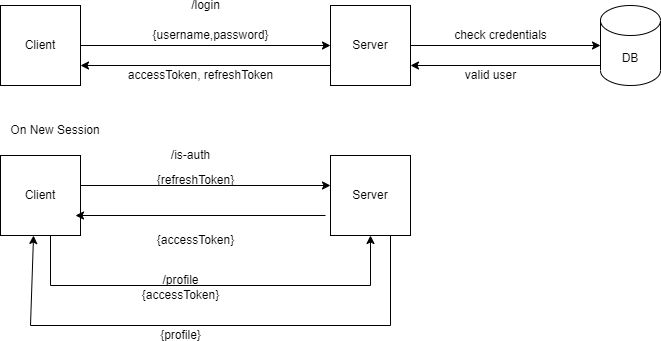




Customer



**Authentication**



## Features:

* Using chat bot we can manage users' choices and orders.
* The chat bot can give recommendations to users based on their interests.
* It can promote the best deals and offers on that day.
* It will store the customer's details and orders in the database.
* The chat bot will send a notification to customers if the order is confirmed.
* Chat bots can also help in collecting customer feedback.

METHODOLOGY PROPOSED:

In traditional e-commerce websites, the users need to search for their required product using a search bar or go through the whole effects of their search. It will take a lot of users' time and it will create a lot of flawed user experiments. This approach will create bad marketing for the product. Later

when the user comes again to purchase the product it will create a bad impression on the user. Even though the product is good the user will not buy the product.

This type of search will create miss matched products when the product has a different name. Let’s say we search for oranges on amazon. Sometimes it will show orange colour or sometimes it will show orange fruit. In recent times, fashion systems have been integrated with artificial intelligence and deep learning. These approaches provide a rich recommendation, but in most cases, it is prone to product mismatch. Even Though the recommender system recommends products based on the user’s preference, this system lacks a chat bot that improves user experience by interacting with users. In most fashion systems, the user needs to navigate across multiple products to find the appropriate product. The users are made to filter products based on the long list of categories present in the system. If this system is integrated with an intelligent bot, it would be able to list out only required categories.

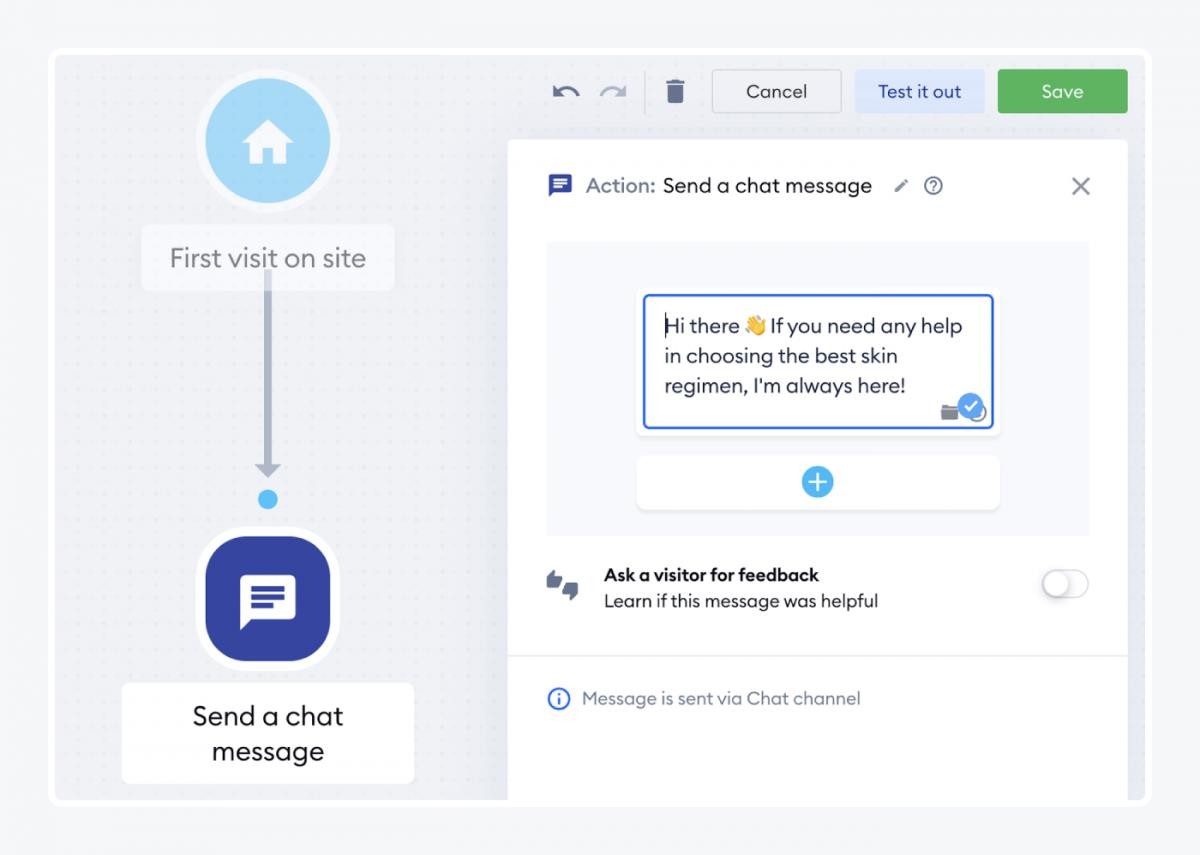
Results and discussions:

The smart fashion recommender system should recommend products based on user input. The user input can be any of the following

* Category
* Subcategory
* Product name
* Product brand
* Colour/Size
* Price Range
* Delivery Date/Time
* Discounts/Deals
* New Arrivals

Based on the user input, chat bot displays products sorted by required fields.





Functional Requirements:

The functional requirements of the application are:

* Redirect users to their respective dashboards based on their roles such as admin and customer
* Allow admin to track sales of individual products
* Allow admin to manage orders made by a particular customer.
* Allow users to interact with the chat bot.
* Manage users' choices and charges using the chat bot.
* Promote the best deals and offers.
* Store customer details and orders.
* Send Notifications to customers if the order is confirmed.
* Collect user feedback.
* Recommend products based on user preference.
* Enable online payment features.
* Generate reports for order summary and order histories.

Non-Functional Requirements

**Performance Requirements**

The system shall be able to handle multiple requests at any given point in time and generate an appropriate response.

* + The response should not take longer than 5 seconds to appear on the client side.
  + The client application should lazy load images of the product to minimize network calls over
  + the network.
  + The responses from the server should be cached on the client side.

Security Requirements

* + Credentials and secrets should be stored securely and should not be leaked.
  + Secured connection HTTPS should be established for transmitting requests and responses
  + between client and server.
  + The system has different roles assigned to a user and every user has access constraints.
  + User access token should be valid for a shorter period and needs to be refreshed
  + periodically.
  + Clients should implement mechanisms to prevent XSS attacks.
  + The server should restrict access to the resources for the particular client domain.

Error Handling

* + The system should handle expected as well as unexpected errors and exceptions to avoid termination of the program.
  + Appropriate error messages should be generated and displayed to the client.

Hardware Requirements

* 8GB RAM
* Intel Core i3
* Laptop/Desktop
* Windows/MAC/Linux OS.

Software Requirements

* Python
* Flask
* Docker
* Kubernetes
* IBM DB

Conclusion

The smart fashion recommender system uses a chat bot as a primary mechanism to interact with users, collect user interest and recommend products periodically. A chat bot is designed to improve user experience by interacting with users. Users need not navigate between multiple pages to find an appropriate product. The system is designed to minimize the efforts taken by customers to search for

the required product. The future enhancements of the chat bot include adding products to the cart, displaying cart items, order history, and payment through the chat bot.

References:

* Samit Chakraborty, Md. Saiful Hoque, Naimur Rahman Jeem-Fashion Recommendation Systems, Models and Methods: A Review
* Samit Chakraborty, Md. Saiful Hoque, S.M. Surid- A Comprehensive review on image-based style prediction and online fashion recommendation
* Qingqing Tu, Le Dong-An Intelligent Personalized Fashion Recommendation System
* M Sridevi, N ManikyaArun, M Shashikala, and E Sudarshan-Personalized fashion recommender system with image-based neural networks



Smart Recommendation

Product Defect

Product Exchange

Periodic Discounts

Facile and Interactive User experience

Shop anytime anywhere

Diverse products

Limited Stocks/ Collections

Shipping Cost

Search and Find products of desire

Products are verified of quality

Quality/Durable Material

Cost efficient

Variety of options available for a product

Product Rating / Review

Customer stratification

Flexible payment methods.

Discounts / Offers

Voguish and trendy product recommendations

Smart and fluid performance

Product advertisements

Will the app have knowledge on what's trending?

How much is it going to cost me?

Will the delivered product be as mentioned?

Empathy Map Canvas

Gain insight and understanding on solving customer problems.

1

Build empathy and keep your focus on the user by putting yourself in their shoes.



Wil it be suitable for me?

Will I find the product I look for?

|  |  |
| --- | --- |
|  |  |
|  |  |

WIll the stock be available for mass purchase?

**PROBLEM STATEMENT**

* Unavailability of chat bots that are interactive enough to navigate the user to do whatever they want.
* The amount of toil a user has to go through to look for a product they desire for.
* Need for a more User-friendly Interface to navigate through.
* Users need to manually select their preferences like size,cost etc…

##### Smart Fashion Recommendation

Fashion applications have seen tremendous growth and are now one of the most used programs in the e- commerce field. The needs of people are continuously evolving, creating room for innovation among the ap**1**p**0**l**m**ic**i**a**nu**t**t**i**e**o**s**ntosp. repare

**1 hour** to collaborate

**2-8 people** recommended

Having an AI program that understands the algorithm of a specific application can be of great aid. We are implementing such a chat

Reduce user navigation

Offers/Discount

High resolution images for each product

**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**

* + 1. **Team gathering**

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

* + 1. **Set the goal**

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

* + 1. **Learn how to use the facilitation tools**

Use the Facilitation Superpowers to run a happy and productive session.

[**Open article**](https://support.mural.co/en/articles/2113740-facilitation-superpowers)

**1**

**Define your problem statement**

Unavailability of chatbots that are interactive enough to navigate the user to do whatever they want.The amount of toil a user has to go through to look for a product they desire for. Need for a more User-friendly Interface. The ma**5**in**m**a**in**im**ute**o**s**f the project is to develop a smart chat-bot that is able to understand the needs of the user and recommend products of desire.

**PROBLEM**

**How might we [your problem statement]?**

**Key rules of brainstorming**

To run an smooth and productive session

**2**

**smart fashion recommendation**

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

**10 minutes**

**Aakashkumar RMR**

user friendly web application

Identify User preferences

Recommend required products

smart chatbo

**Shanmugasundaram S**

**Sathishkumar G**

**TIP**

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

High performance

Product recommendation

Interactive web application

Handle secure payments

**Srinath P**

Reduce user navigation

various preferences to be shown

**3**

**Group ideas**

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

**20 minutes**

smart chatbo

**TIP**

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Well defned product description and its available categories

Integration of intelligent chat-bot

Offers/Discoun

Instant product results based on preferences

**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**

High resolution images for each product

Reduce user navigation

**After you collaborate**

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

**Quick add-ons**

Instant product results based on preferences

1. **Share the mural**

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

1. **Export the mural**

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

User Friendly Web application

**Keep moving forward**

**Strategy blueprint**

Define the components of a new idea or strategy.

[**Open the template**](https://app.mural.co/template/e95f612a-f72a-4772-bc48-545aaa04e0c9/984865a6-0a96-4472-a48d-47639307b3ca)

**Customer experience journey map**

Handle secure payments

Well defned product description and its available categories

Handle secure payments

bot, which is fed with the knowledge

of the application’s algorithm and helps the user completely from

Stay in topic.

Defer judgment.

Encourage wild ideas.

Listen to others.

**Importance**

If each of these

|  |  |
| --- | --- |
|  |  |
|  |  |

Understand customer needs, motivations, and

obstacles for an experience.

High resolution images for each product

[**Open the template**](https://app.mural.co/template/b7114010-3a67-4d63-a51d-6f2cedc9633f/c1b465ab-57af-4624-8faf-ebb312edc0eb)

User Friendly Web application

finding their needs to processing the payment and initiating delivery.

Go for volume. If possible, be visual.

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

**Strengths, weaknesses, opportunities & threats**

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

Well defned product description and its available categories

[**Open the template**](https://app.mural.co/template/6a062671-89ee-4b76-9409-2603d8b098be/ca270343-1d54-4952-9d8c-fbc303ffd0f2)

**TIP**

Secure Authentication

Instant product results based on preferences

various preferences to be shown

Integration of intelligent chat-bot

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.

Offers/Discount

[**Share template feedback**](https://muralco.typeform.com/to/CiqaHVat?typeform-source=app.mural.co)

[**Share template feedback**](https://muralco.typeform.com/to/CiqaHVat?typeform-source=app.mural.co)

**Feasibility**

Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)



**Template**

**Need some inspiration?**

See a finished version of this template to kickstart your work.

[**Open example**](https://app.mural.co/template/e5a93b7b-49f2-48c9-afd7-a635d860eba6/93f1b98d-b2d2-4695-8e85-7e9c0d2fd9b9)

**Project Design Phase-I Proposed Solution Templates**

|  |  |
| --- | --- |
| DATE | 08.10.2002 |
| TEAM ID | PNT2022TMID29308 |
| PROJECT NAME | SMART FASHION RECOMMENDER APPLICATION |
| MAXIMUM MARKS | 2 MARKS |

**PROJECT SOLUTION TEMPLATES**

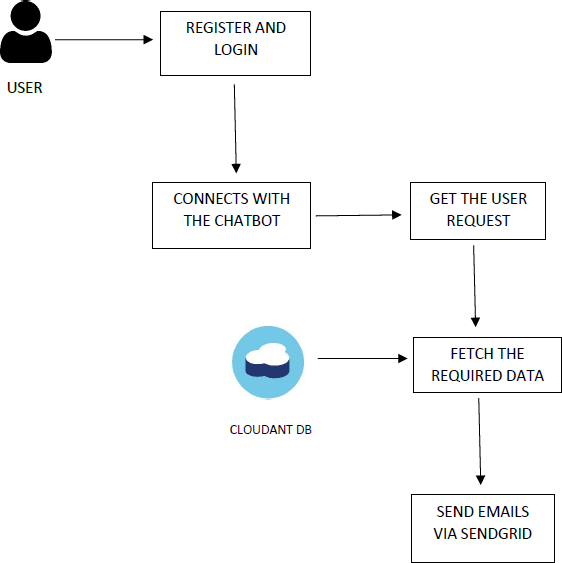
|  |  |  |
| --- | --- | --- |
| S.NO | PARAMETER | DESCRIPTION |
| 1. | PROBLEM STATEMENT | The rapid growth of online market for clothing and fashions consumers are getting increasingly overwhelmed with the volume and variety of production facing too many types of garments, consumers need to try them on repeatedly, which is somewhat time- and energy-consuming. |
| 2. | IDEA/ SOLUTION DESCRIPTION | Smart Fashion Recommender Application can tackle with choice overload by suggesting the most interesting products to the users |
| 3. | NOVERTY/ UNIQUENESS | Instead of searching manually a chatbot will help to find the right product effectively, with this feature user can save time and it is a easy process, chat keep send notification about new collections |
| 4. | SOCIAL IMAPACT  /CUSTOMER SATISTIFICATION | This chatbot helps the users to find the right products easily, the innovations that all levels of business owners can take advantage of. This application used in all fashion markets |
| 5. | BUSINESS MODEL (REVENUE MODEL) |  |

|  |  |  |
| --- | --- | --- |
| 6. | SCALABILITY OF THE SOLUTION | * Bot never runs into errors * Optimized stock database * Established marketing strategy * Responsiveness of the application |

**Project Design Phase-I**

|  |  |
| --- | --- |
| DATE | 08.10.2002 |
| TEAM ID | PNT2022TMID29308 |
| PROJECT NAME | SMART FASHION RECOMMENDER APPLICATION |
| MAXIMUM MARKS | 2 MARKS |

**Solution Architecture**



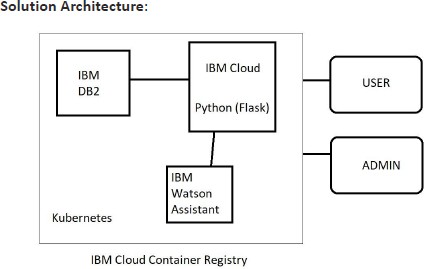
#### RECOMMENDATION FILTERING TECHNIQUES :

The selection of an effective and accurate filtering technique is crucial for developing a successful recommendation system. These are f four widely used recommendation-filtering techniques to build-up the architectures of the recommendation.

* + Content-Based Filtering (CBF) Technique
  + Collaborative Filtering (CF) Technique
  + Hybrid Filtering Technique
  + Hyper personalization Filtering Technique

#### IBM SOLUTION ARCHITECTURE :

Recommendation systems have the potential to explore new opportunities for retailers by enabling them to provide customized recommendations to consumers based on information retrieved from the Internet.



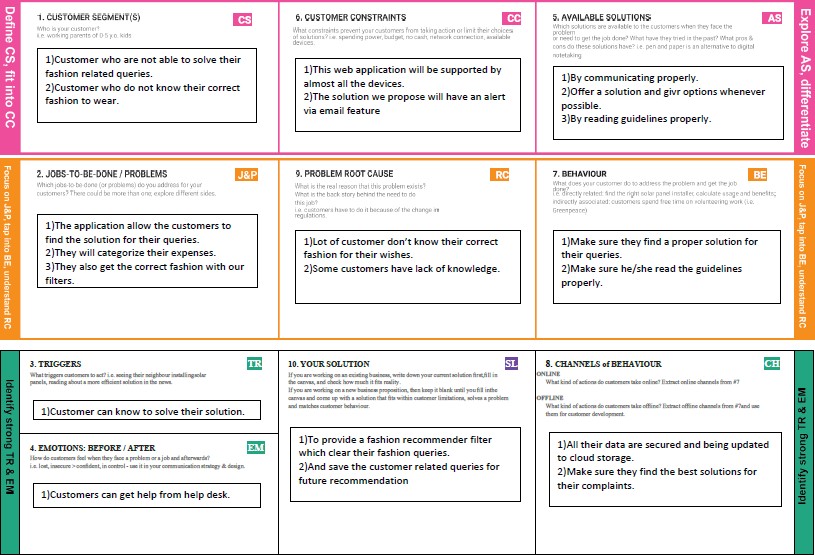
The successful outcome of the recommendation system depends on the relevance of the filtering technique and its compatibility with the proposed model.

-

**Project Design Phase-I**

# Solution Fit Document

|  |  |
| --- | --- |
| DATE | 08.10.2002 |
| TEAM ID | PNT2022TMID29308 |
| PROJECT NAME | SMART FASHION RECOMMENDER APPLICATION |
| MAXIMUM MARKS | 2 MARKS |



Creating a Custome journey Map is a quick way to help you and your team gain a deeper understanding of who you're designing for, aka the stakeholder in your project. The information you add here should be representative of the observations and research you've done about your users. 🔎



**TEAM ID-PNT2022TMID29308**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1 Phases**  High-level steps your user needs to accomplish from start to finish | **ENTICE** | **ENTER** | **ENGAGE** | **EXIT** |
|  | | | | |
| **2 Steps**  Detailed actions your user has to perform | VISIT THE INTERACT BROWSE THE VIEW  WITH THE RECOMMENDED PRODUCT  WEBSITE CHATBOT PRODUCT DESCRIPTION  AND REVIEW | COMPARE PERSONALIZED CONFIRM THE PURCHASING EXPERIENCE ORDER AND  VARIETY OF THE AND BETTER PAYMENT  PRODUCT PRODUCT PRIZES PROCESS | STATUS OF IMPROVEMENT  ORDER THE ORDER INSTANT TEAM  TRACKING TGROUGH NOTIFICATIONS EFFICIENCY  MAIL AND AND CUSTOMER  SMS SATISFACTION | PRODUCT  APPEARS IN PERSONALIZED REAL-TIME  THE POPUP PERSONALIZED DEALS AND  MODE AND RECOMMENDATIONS OFFERS SOLUTIONS  USER PROFILE |
|  | | | | |
| **3 Feelings**  What your user might be thinking and feeling at the moment | MORE NEW EASY RELIABLE EXPERIENCE  PURCHASE AND FUN TO THE  TO CHAT CUSTOMER | CUSTOMER CHATBOT CAN  FEELS GOOD EASY TO GUIDE THE WHEN THEY SEARCH USING CUSTOMER TO ACTUALY FIND RECOMMENDED MAKING GOOD  EXACT SEARCH ENGINE AND SECURE  PRODUCT PRUCHSE | ORDER NOTIFY WHEN ONSISTENCY WITHOUT CUSTOMIZED IN ANSWER HUMAN PRODUCT  HELP RATE | STREAMLIME FEEL FREE  YOUR DAAMAGE PRODUCT  CUTOMER FREE  ONBOARDING PRODUCT ONMLINE  PROCESS SHOPPING |
| PEOPLE ALWAYS  FEELS LIKE TO INCREASE THE FOR COMFORTABLE CUSTOMER'S DISAPPOINTING WITH PERSON DISLIKE THE CUSTOMER COMPARE TO RECOMMENDATION NEEDS  THE CHATBOT | ATBOT DOES NOT TO CREATING  HAVE ANSWER THE PURCHASE  LIMITED MULTIPLE LIST WITHOUT  CUSTOMER  RESPONSE QUERIES PERMISSION | CHABOT CHATBOT  CHATBOT CANNOT ONLY  ARE NOT INTERACT AS A ANSWER  HUMAN HUMAN WITH BASICS  CUSTOMERS QUESTIONS | CHATBOT TO RATE OF  ARE IT REQUIRES CHATBOT IS DIFFICULT CONSTANT HEAVY  MAINTANANCE  TO CREATE BUDGET |
|  | | | | |
| **4 Goals & Motivations**  Problems your user runs into | TO PROVIDE  PROVIDE TO IMPROVE BEST &  24X7 CUSTOMER BETTER  SERVICES SATISFACTION PRODUCT TO  CUSTOMER | ALLOW TO FOCUS HELP ME TO  BUSINESS TO ON OFFERS PURCHASE THE  THE OF THE RECOMMENDED  CUSTOMER PRODUCT  (RATAIL AGENT) PRODUCT | TO MANY OF THE TO FOCUS ON  COMPLETE QUERIES WILL THE CUSTOMER  BE ANSWERED FOR STAYING  DIGITAL EASILY AND UNTIL MINIMUM  LEARNING QUICKLY PURCHASE  TRICKS | MONITOR TO INCREASE TO AVOID CUTOMER THE NEEDS UN- DATA TO OF THE  GAIN CUSTOMER NECESSARY  INSIGHTS TASTE PRODUCT |
|  | | | | |
| **5 Area Of Opportunities**  Potential improvements or  enhancements to the experience | REDUCE TO REFER THE  INCREASES CUSTOMER CUSTOMER  CUSTOMER NEIGHBOURS  ENGAGEMENT SERVICE WHEN THEY  COSTS FEELS GOOD TO  REFER | DEVISE A BALANCE MEET  CONSRVATIONAL AUTOMETION CUSTOMER  MARKETING WITH HUMAN EXPECTATIONS  STRATEGY TOUCH | ACHIEVE IMPROVE TO GENERATE SCALABILTY THEIR MORE SALES OF CUSTOMER AND PROVIDE  SUPPORT EXPERIENCE DISCOUNTS | TO CREATE A TO GAIN TO MAKE DEEPER MORE SALES ONLINE RAPPORT WITH PAYMENT LIKE  WITH DIFFERNT UPI ID AND  CUSTOMER OFFERS SCAN PAY |

T.GOKULSANKAR

V.VIDHYA

A.GUHAN SHANMUGAM

C.ALOK



Aakash

kumar



CUSTOMER JOURNEY MAP

WAY TO GOOD FASHION CHATBOT RECOMMEDATION SYSTEM

🧑

**People**

2–9

⏰

**Time**

30 min

🟢

**Difficulty**

Beginner

P.srinath

S.shanmug

G.sathish

asundarm

kumar. RMR

##### Project Design Phase-II

**Data Flow Diagram & User Stories**

|  |  |
| --- | --- |
| Date | 25 October 2022 |
| Team ID | PNT2022TMID29308 |
| Project Name | SMART FASHION RECOMMENDER  APPLICATION |
| Maximum Marks | 4 Marks |

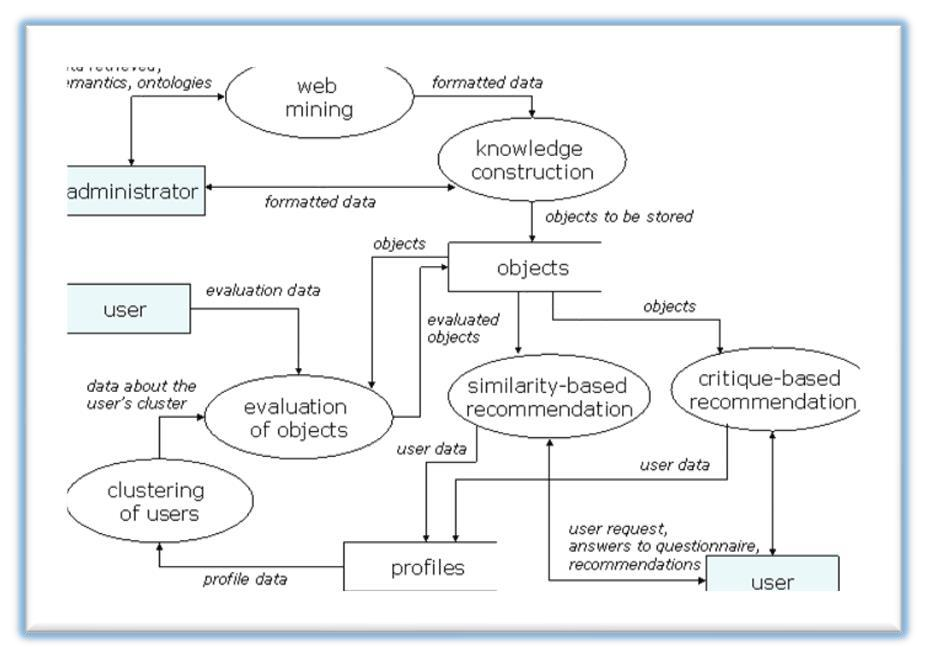
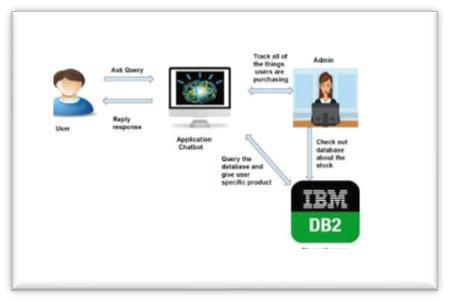
Data Flow Diagrams:

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

Example: [(Simplified)](https://developer.ibm.com/patterns/visualize-unstructured-text/)

**User Flow :**

User Sign up / Login Chatbot Purchasing Product



DFD of Fashion Recommender (Industry Level - 0)

**Work Flow :**

Chatbot IBM Cloud IBM DB2 Watson Assistant

Docker

Container Registry Kubernetes

**User Stories :**

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional Requirement**  **(Epic)** | **User Story Number** | **User Story / Task** | **Acceptance criteria** | **Priority** | **Release** |
| Customer (Mobile user) | Registration | USN-1 | As a user, I can register for the application by  entering my email, password, and confirming my password. | I can access my account / dashboard | High | Sprint-1 |
|  |  | USN-2 | As a user, I will receive confirmation email once I have registered for the application | I can receive confirmation email & click confirm | High | Sprint-1 |
|  |  | USN-3 | As a user, I can register for the application through Facebook | I can register & access the dashboard with Facebook  Login | Low | Sprint-2 |
|  |  | USN-4 | As a user, I can register for the application  through Gmail | I can register &access the  dashboard with Gmail login | Medium | Sprint-1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Login | USN-5 | As a user, I can log into the application by entering email & password | I can login into the application with Gmail  login | High | Sprint-1 |
|  | 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 | Sprint-1 |
| **User Type** | **Functional Requirement**  **(Epic)** | **User Story Number** | **User Story / Task** | **Acceptance criteria** | **Priority** | **Release** |
| Customer (Web user) | Registration | USN-1 | As a user ,I can register for the web page by entering the 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 web-pages | I can register & access the dashboard e  with Gmail login | High | Sprint-1 |
|  |  | USN-3 | As a user, I can registered for the web-page through Email | I can register & access the dashboard with Gmail  Login | Low | Sprint-2 |
|  |  | USN-4 | As a user, I can register for the web-page through Email | I can register & access the dashboard with Gmail  Login | Medium | Sprint-1 |
|  | 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 | Sprint-1 |
|  | 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 | Sprint-1 |
| 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 log into the application with Gmail  login | High | Sprint-1 |
|  | 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 | Sprint-1 |
|  | Service | USN-1 | As a customer Care Executive, I can access  the customer care service page of the application by logging and accessing the page | I can access the service page by logging & accessing the page | High | Sprint-1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Administrator | Login | USN-1 | As a administrator, I can log into the  application by entering my administrator email id & password | I can log into the application with Gmail  application | High | Sprint-1 |
|  | 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 | Sprint-1 |

SUBMITTED BY

TEAM ID - PNT2022TMID29308

TEAM LEADER - Aakash Kumar RMR TEAM MEMBER 1 - Sathish Kumar G TEAM MEMBER 2 - P.Srinath

TEAM MEMBER 3 - S.Shanmugasundaram

**Project Design Phase-II**

##### Solution Requirements (Functional & Non-functional)

|  |  |
| --- | --- |
| Date | 25.10.2022 |
| Team ID | PNT2022TMID29308 |
| Project Name | SMART FASHION RECOMMENDER  APPLICATION |
| Maximum Marks | 4 Marks |

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 / Sign up** | Registration through Form Registration through Gmail  Registration through LinkedIN |
| FR-2 | **User Verification** | Confirmation via Email Confirmation via OTP |
| FR-3 | **Sign In / Login** | Login by using Mobile Number or Email |
| FR-4 | **Profile Details** | Update the Information about Customer Example :-   * Name * Gender * Age * Mobile number * Address |
| FR-5 | **Chatbot (Watson Assistant)** | Get the Information about   * Search Products * View Offers * Discounts * Stock Availability * User Personal Information (FR-4) |
| FR-6 | **Advance Search Capabilities** | Sorting and filtering options |
| FR-7 | **Shopping Cart** | My Cart Button Add-To-Cart-Button  Remove-From-Cart-Button |
| FR-8 | **Checking Item Availability** | Item Availability in rural and urban Locations |
| FR-9 | **Checking The Shipping Status / Tracking The Order Product** | Easily Checking Status availability of ordered Items |
| FR-10 | **Logout** | After the Purchase, user can Logout Or close the application When customer needs. |

Non-functional Requirements:

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

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | The application will be designed for making Good HUMAN – COMPUTER Interaction in such a way,   Any user can easily navigate   User Can easily View and understandable   Comfort while making Place order   Comfort with tracking facilities   Easy and Compact design  These all are about to achieve a defined goal Effectively, Efficiently and Satisfactorily. |
| NFR-2 | **Security** | The application will be Using of “Secure Socket Layer” (SSL) Certificate will provide a More security of the Project and This process will happen while Python Flask to Cloud Connect. This makes user private Information like Baking, Shipping/Home address, email, Phone number etc., will be kept as more  secure. |
| NFR-3 | **Reliability** | Ability of software to perform critical tasks like Collection and Securing customer Data, Providing Gateway Payment to function correctly in a given  Environment, for a Particular amount of time. |
| NFR-4 | **Performance** | It Focus on the loading application as quickly as possible irrespective of the number of user traffic. |
| NFR-5 | **Availability** | The Application will be Available to all users at any given point of time. User can access the chatbot for  raising any queries. |
| NFR-6 | **Scalability** | Chatbot can be very useful during festival season to know about offers and discounts. It will be helpful whenever we make online shopping. |

SUBMITTED BY

TEAM ID - PNT2022TMID29308

TEAM LEADER - Aakash Kumar RMR TEAM MEMBER 1 - Sathish Kumar G

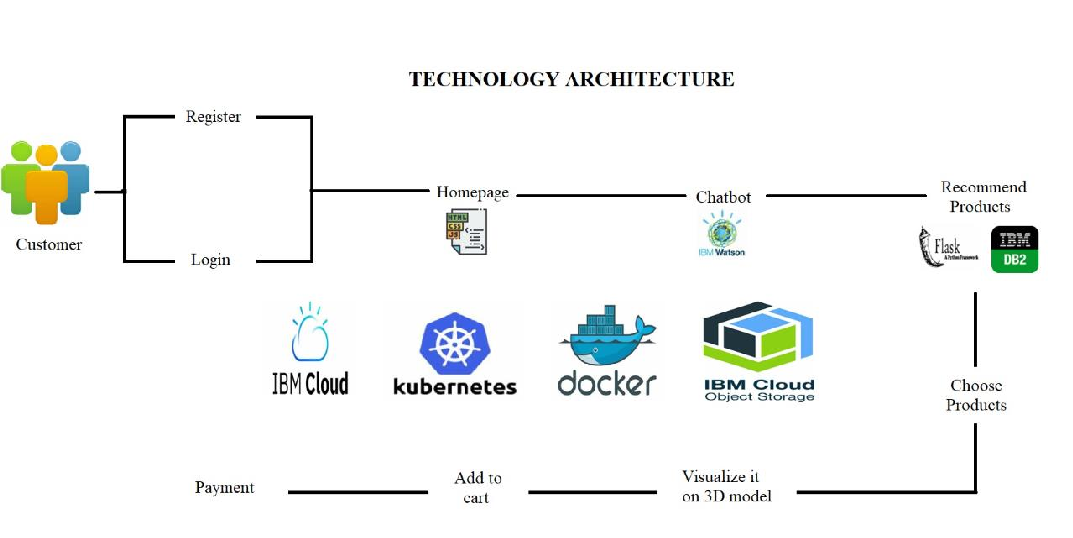
TEAM MEMBER 2 - P.Srinath

TEAM MEMBER 3 - S.Shanmugasundaram

##### Project Design Phase-II Technology Stack (Architecture & Stack)

|  |  |
| --- | --- |
| Date | 15 October 2022 |
| Team ID | PNT2022TMID29308 |
| Project Name | SMART FASHION RECOMMENDER APPLICATION |
| Maximum Marks | 4 Marks |

**Diagram - Architecture :**



##### Way Chatbot:

**FLASK IBM CLOUD IBM DB2 KUBERNETES DOCKER**

##### Table-1 : Components & Technologies:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Technology** | **Description** |
| 1. | User Interface | HTML, CSS, JavaScript / Angular Js  / React Js. | User can interact with the application through Chatbot for good Human-computer interface. |
| 2. | Application Logic-1 | Java Python | The application will have the login/sign up page where the user can login into the main dashboard or they can register into the application. |
| 3. | Application Logic-2 | IBM Watson STT service | The application contains a Chatbot where the user needs to give their details like   * gender, * age * type of product   these were they wish to buy using Watson assistant through chatbot. |
| 4. | Application Logic-3 | IBM Watson Assistant | User’s will get the recommendations based on their interests, can get  the details about offers, discounts and chatbot will send a notification to customers if the order is confirmed. |
| 5. | Database | MySQL, NoSQL, | Customer’s details and order are stored in the database and whenever we can be fetch and retrieve data from database. |
| 6. | Cloud Database | IBM DB2, IBM Cloudant | With use of Database Service on Cloud, user can access all the data stored in the cloud over a network from any device and user’s data are  stored in a well secure manner. |
| 7. | File Storage | IBM Block Storage or Other Storage Service or Local Filesystem | Previously ordered product details and other customer details can be  stored in the IBM Block Storage as the data kept inside are highly protected. |
| 8. | Infrastructure (Server / Cloud) | Local, Cloud Foundry, Kubernetes, Docker | Chatbot with updated services can be deployed in an IBM cloud by using Watson assistant. |

**Table-2: Application Characteristics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristics** | **Technology** | **Description** |
| 1. | Open-Source Frameworks | Python - Flask | * Flask is a web framework in Python is used in the implementation of smart fashion recommender application. |
| 2. | Security Implementations | Container Registry, Kubernetes Cluster. | * This application uses Container Registry in IBM cloud so that the user details are kept as more secure and confidential. * User have to confirm the login while logging in to avoid any misuse of the credentials. |
| 3. | Scalable Architecture | Container Registry, Kubernetes Cluster. | * The Smart Fashion Recommender Application is more useful whenever user’s make online purchase and it’s demand increase at festival season’s to know about the available offers and discounts. |
| 4. | Availability | Docker, Kubernetes Cluster. | * Docker helps to improve the network management so that the application can be accessed at anytime. |
| 5. | Performance | Docker, Kubernetes Cluster. | * The performance of this application is high. * efficient as the network traffic can be easily managed. |

### PROJECT PLANNING PHASE PREPARE MILESTONE AND ACTIVITY LIST

|  |  |
| --- | --- |
| TEAM ID | PNT2022TMID29308 |
| PROJECT NAME | Smart Fashion Recommender Application |

Remaining Task:

|  |  |  |
| --- | --- | --- |
| MILESTONES | ACTIVITY | DESCRIPTION |
| Project development phase | Delivery of Sprint-1,2, 3, 4 | To develop the code and  submit the develop the code after completion of testing |
| Setting-up app environment | Create IBM cloud account | Sign up IBM cloud account |
|  | Create flask project | Getting started with the flask  to create project |
|  | Install IBM cloud cli | Install IBM command line  interface (CLI) |
|  | Docker CLI installation | Installing docker CLI |
|  | Create an account in sendgrid | Create an account in sendgrid  Use service as e-mail  integration to the application for sending emails |
| Implementing web application | Create UI to interact with the application | Create UI   * registration page * login page * view products page * add products page |
|  | Create IBM DB2 and connect with the Python | Create IBM DB2 in IBM  cloud and link with the Python |
| Integrating sendgrid service | Sendgrid integration with the Python | To send emails from the application we need to  integrate the sendgrid services |
| Developing a chat bot | Building a chat bot and  integrate with the application | Build the chat bot and  integrate it to the flask application |
| Deployment of app in IBM cloud | Containerise the app | Create a docker image of the application in addition to  push it to the IBM container registry |
|  | Upload image to IBM  container registry | Upload the image to IBM  container registry |
|  | Deploy in in kubernetes cluster | Once the image is uploaded to IBM container registry deploy the image toward  IBM kubernetes cluster |

Completed Tasks:

|  |  |  |
| --- | --- | --- |
| MILESTONES | ACTIVITY | DESCRIPTION |
| Ideation phase | Literature survey | Literature survey on the  selected project and information gathering |
|  | Empathy map | Prepare empathy map to capture the user pains and gains, prepare a list of  problem statement |
|  | Ideation | Organising the brainstorming session and prioritise the top three ideas  based on feasibility hand importance |
| Project design phase 1 | Proposed solution | Prepare proposed solution document which includes novelty, feasibility of ideas, business model, social  impact, scalability of solution |
|  | Problem solution fit | Prepare problem solution fit  documents |
|  | Solution architecture | Prepare solution architecture  document |
| Project design phase 2 | Customer journey map | Prepare customer journey map to understand the user interactions and experience  with the application |
|  | Functional requirements | Prepare functional and non- functional necessity  document |
|  | Data flow diagram | Prepare data flow diagram  and user stories |
|  | Technology architecture | Draw technology  architecture diagram |
| Project planning phase | Milestones and activity list | Prepare milestones and activity list of the project |
|  | Sprint delivery plan | Prepare spring delivery plan |

# PROJECT PLANNING PHASE

**SPRINT DELIVERY PLAN**

|  |  |
| --- | --- |
| Team ID | PNT2022TMID29308 |
| Project Name | Smart Fashion Recommender Application |

## Product Backlog, Sprint Schedule, Estimation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 cloudaccount. | 2 | High | Ajisha J Amishya  Renjai R J |
| Sprint-1 |  | USN-2 | As a user, I will create a flaskproject | 1 | Low | Babis Dania T Aspiya S |
| Sprint-1 |  | USN-3 | As a user, I will install IBM CloudCLI | 2 | Medium | Ajisha J Amishya Renjai R J |
| Sprint-2 | Setting up App environment | USN-4 | As a user, I can install Docker CLI | 1 | Low | Babis Dania T Aspiya S |
| Sprint-2 |  | USN-5 | As a user, I will Create an accountin sendgrid | 2 | Medium | Ajisha J Amishya  Renjai R J |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-3 | Implementing web application | USN-6 | As a user, I Create UI to interact with the application | 1 | High | Babis Dania T Aspiya S |
| Sprint-3 |  | USN-7 | As a user, I Create IBM DB2 and connect with Python | 3 | High | Ajisha J  Amishya Renjai R J |
| Sprint-3 | Integrating sendgrid service | USN-8 | As a user, I will be integrating sendgridwith python code | 2 | High | Babis Dania T Aspiya S |
| Sprint-3 | Developing a chatbot | USN-9 | As a user, I must build a chatbot and integrate to application | 1 | Medium | Ajisha J Amishya Renjai R J |
| Sprint-4 | Development of App in IBM Cloud | USN-10 | As a user, I will Containerize the App | 1 | Low | Babis Dania T Aspiya S |
| Sprint-4 |  | USN-11 | As a user, I will upload image to IBM Container registry | 2 | Medium | Ajisha J  Amishya Renjai R J |
| Sprint-4 |  | USN-12 | As a user, I will deploy App in Kubernetes cluster | 3 | High | Babis Dania T Aspiya S |
| Sprint-4 | User panel |  | As a user   * Register, Login, Email, Verification * Manual Search * Order placement, Order Details | 3 | High | Ajisha J Amishya  Renjai R J  Babis Dania T Aspiya S |

**Project Tracker, Velocity & Burndown Chart**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 |

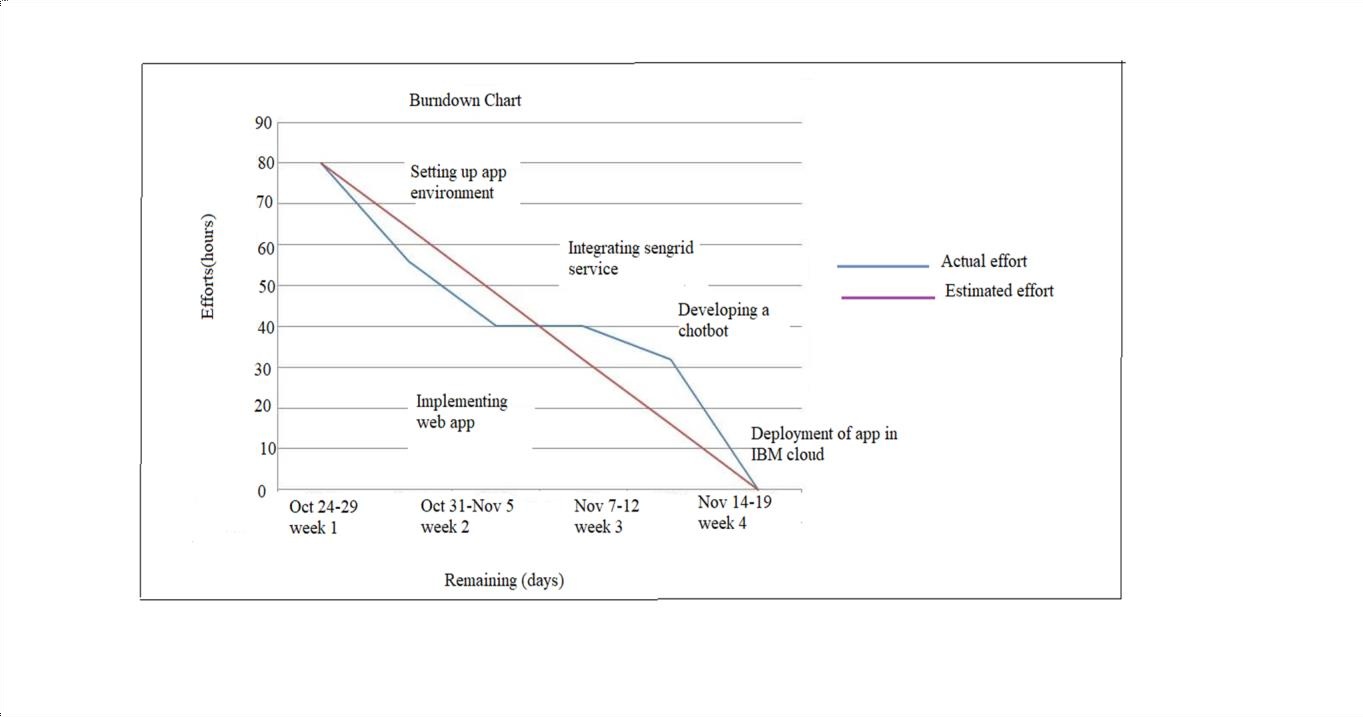
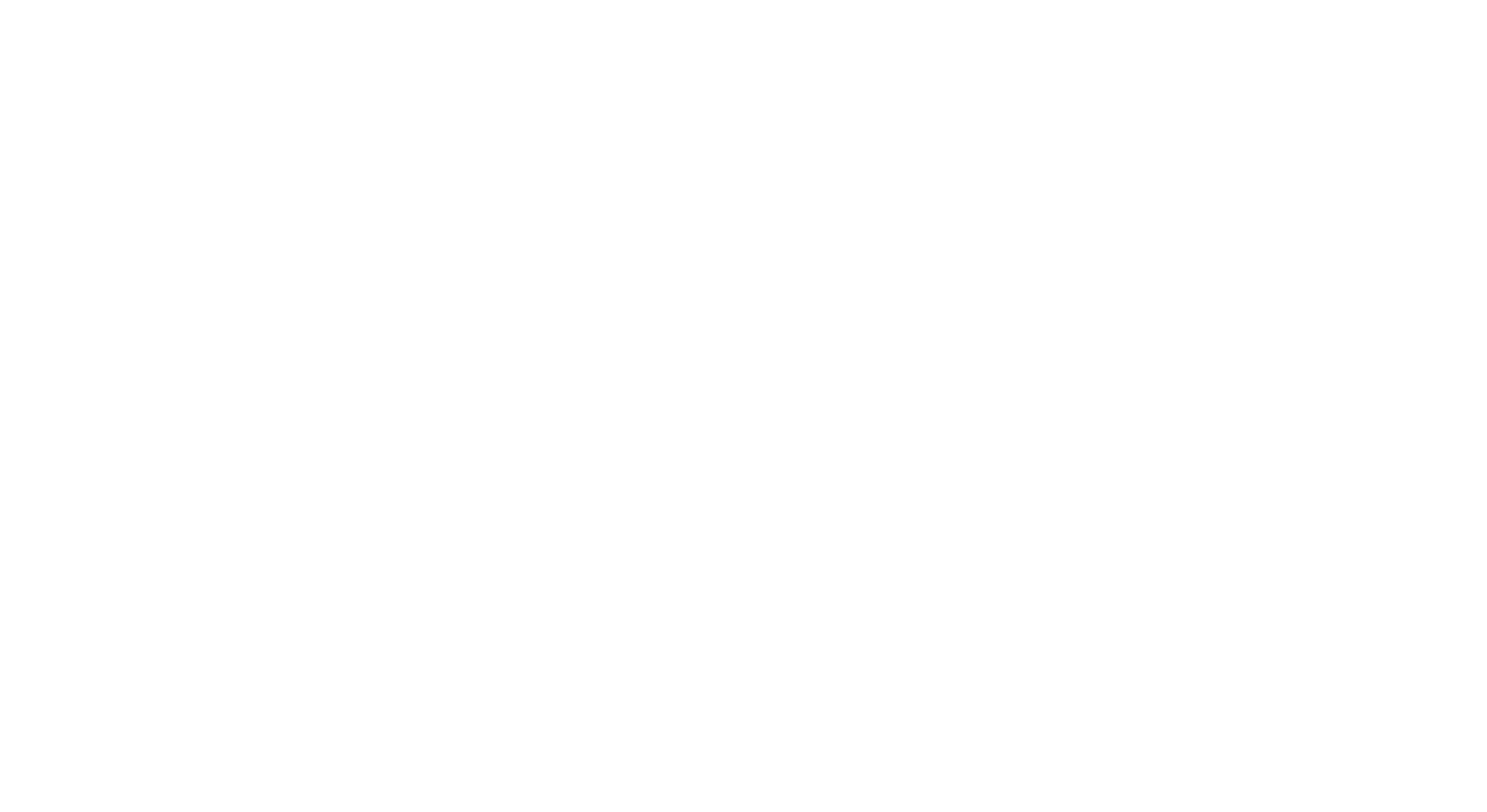
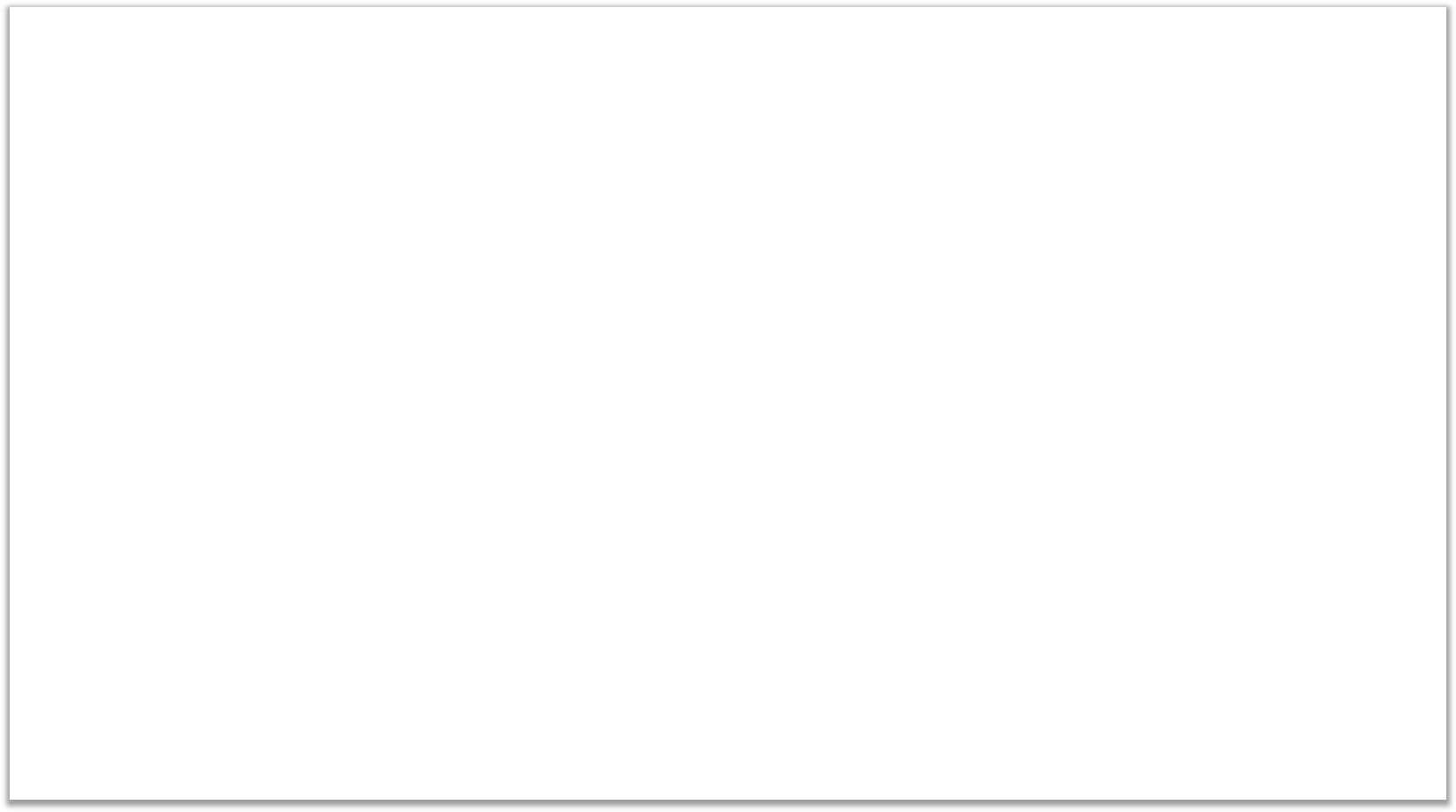
**Velocity**

Imagine we have a 6-day sprint duration, and the velocity of the team is 18(points per sprint). Let’s calculate the team’s average velocity (AV) per iteration unit (story points per day)

AV = Sprint Duration / Velocity AV = 24/6 = 4

**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.



PROJECT DEVELOPMENT PHASE SPRINT - 1

|  |  |
| --- | --- |
| DATE | 3 november 2022 |
| TEAM ID | PNT2022TMID29308 |
| PROJECT NAME | SMART FASHION RECOMMENDER APPLICATION |

**Login, signup and home page UI designs**

HTML CODE:

INITIAL SETUP:

<!doctype html>

|  |
| --- |
| <html lang="en"> |
| <head> |
| <meta charset="utf-8"> |
| <title>SmartFashionRecommender</title> |
| <base href="/">  <meta name="viewport" content="width=device-width, initial-scale=1"> |
| <link rel="icon" type="image/x-icon" href="/assets/logo.png"> |
| <link rel="stylesheet" |
| href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outline |
| d:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" /> |
| <link rel="preconnect" href="https://fonts.googleapis.com">  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin> |
| <link |
| href=["https://fonts.g](https://fonts.googleapis.com/css2?family=Cinzel%2BDecorative%3Awght%407)o[ogleapis.com/css2?family=Cinzel+Decorative:wght@7](https://fonts.googleapis.com/css2?family=Cinzel%2BDecorative%3Awght%407) |
| 00&family=Tangerine:wght@700&display=swap" |
| rel="stylesheet"> |
| </head> |
| <body> |
| <app-root></app-root> |
| </body> |

</html>

LOGIN PAGE

|  |
| --- |
| <section class="container-fluid h-100 login-container"> |
| <div class="row h-100 justify-content-center align-items-center"> |
| <div class="col-12 col-md-6 d-none d-md-block"></div> |
| <div class="col-12 d-flex justify-content-center col-md-6"> |
| <div class="login-form-container p-3"> |
| <h2 class="mb-4 login-text"><img class="login-logo-img" |
| src="/assets/logo.png" alt="logo">Login</h2>  <div> |
| <div> |
| <input type="tel" class="login-input" |
| placeholder="Mobile number"> |
| </div> |
| <div> |
| <input type="password" class="login-input"  placeholder="password"> |
| </div> |
| <p class="text-center" style="font-size: 10px;"> |
| By login, you agree to our Terms of Use and |
| Privacy Policy. |
| </p> |
| <div class="text-center">  <button class="primary-btn" |
| routerLink="/">Login</button> |
| </div> |
| <!-- new account --> |
| <p class="text-center mt-3">New to Fashion <a |
| routerLink="/signup" |
| class="text-decoration-none">Create an  account</a></p> |
| </div> |
| </div> |
| </div> |
| </div> |
| </section> |

SIGNUP CODE:

|  |
| --- |
| <section class="container-fluid h-100 login-container">  <div class="row h-100 justify-content-center align-items-center"> |
| <div class="col-12 col-md-6 d-none d-md-block"></div> |
| <div class="col-12 d-flex justify-content-center col-md-6"> |
| <div class="login-form-container p-3"> |
| <h2 class="mb-4 login-text"><img class="login-logo-img" |
| src="/assets/logo.png" alt="logo">Sign up</h2> |
| <div> |
| <div>  <input type="email" class="login-input" |
| placeholder="Email"> |
| </div> |
| <div> |
| <input type="text" class="login-input" |
| placeholder="User name"> |
| </div>  <div> |
| <input type="tel" class="login-input" |
| placeholder="Mobile number"> |
| </div> |
| <div> |
| <input type="password" class="login-input" |
| placeholder="password">  </div> |
| <p class="text-center" style="font-size: 10px;"> |
| By signup, you agree to our Terms of Use and |
| Privacy Policy. |
| </p> |
| <div class="text-center"> |
| <button class="primary-btn" routerLink="/">Sign  up</button> |
| </div> |
| <!-- new account --> |
| <p class="text-center mt-3">Existing User <a |
| routerLink="/login" |
| class="text-decoration-none">Login</a></p> |
| </div>  </div> |
| </div> |

|  |
| --- |
| </div> |
| </section> |

|  |
| --- |
| /\* You can add global styles to this file, and also import other style |
| files \*/ |
| :root { |

--primary-color: #b0e6dc;

LOGIN , SIGNUP CSS:

|  |
| --- |
| --secondary-color: #c0c2c3; |
| } |
| html, body { |
| height: 100%;  margin: 0; |
| } |
| .cursor-pointer { |
| cursor: pointer; |
| } |
| /\* button styles \*/ |
| .primary-btn { background- color: black; |
| color: white; |
| border: 0; |
| padding: 0.7rem 2.5rem; |
| border-radius: 10px; |
| } |
| .secondary-btn { |
| background-color: white; |
| color: black; |
| border: 1px solid black; |
| padding: 0.7rem 2.5rem; |
| border-radius: 10px; |
| } |
| // login screen and signup screen |
| .login-container { |
| background-image: url("/assets/login-banner.png"); |
| background-repeat: no-repeat; |

|  |
| --- |
| background-size: cover; |
| }  .login-text { |
| font-family: "Cinzel Decorative"; |
| letter-spacing: 0.7rem; |
| } |
| .login-logo-img { |
| width: 4rem; |
| height: 4rem;  } |
| .login-form-container { |
| background: rgba(255, 255, 255, 0.2); |
| border-radius: 16px; |
| box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1); |
| backdrop-filter: blur(5px); |
| -webkit-backdrop-filter: blur(5px); border: 1px solid rgba(255, 255, 255, 0.3); |
| } |
| .login-input { |
| background: rgba(255, 255, 255, 0.2); |
| border-radius: 16px; |
| box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);  border: 1px solid rgba(255, 255, 255, 0.3); |
| margin: 1rem 0; |
| padding: 0 1rem; |
| height: 3rem; |
| width: 25rem; |
| } |
| @media only screen and (max-width: "600px") { |

.login-input { width: 100%;

}

}

HOME PAGE CODE:

|  |
| --- |
| <section class="container-fluid"> |
| <div class="row align-items-center main-con" |
| style="background-color: #b0e6dc;"> |
| <div class="col-12 col-md-6 py-4"> |
| <p class="banner-text">Make your |
| </p>  <p class="banner-text">fashion more |
| </p> |
| <p class="banner-text d-flex align-items-center"> |
| <span class="material-symbols-outlined" |
| style="font-size: 5rem;"> |
| chip\_extraction |
| </span><span>perfect</span>  </p> |
| <div> |

|  |
| --- |
| <button class="primary-btn shop-btn" |
| routerLink="/products">Shop Now</button> |
| </div> |
| </div>  <div class="col-12 col-md-6 text-center"> |
| <img class="w-75 h-75" src="/assets/landing-banner.png" |
| alt="Landing Banner"> |
| </div> |
| </div> |
| <!-- landing icons sections -->  <div class="row justify-content-center main-con shadow"> |
| <div class="col-12 col-md-3 py-2 py-md-4 d-flex |
| align-items-center border-end border-bottom"> |
| <div> |

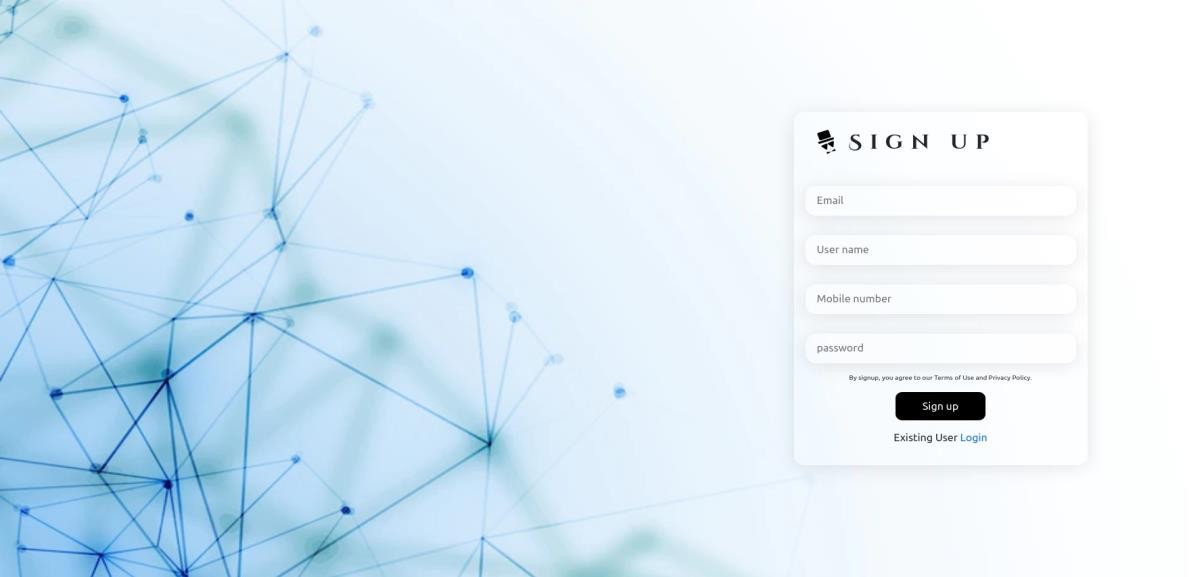
|  |
| --- |
| <span class="material-symbols-outlined land-icon"> |
| chat |
| </span>  </div> |
| <div class="ms-4"> |
| <p class="fw-bold mb-0">CHAT BOT</p> |
| <span>A smart way of finding products</span> |
| </div> |
| </div> |
| <div class="col-12 col-md-3 py-2 py-md-4 d-flex  align-items-center border-end border-bottom"> |
| <div> |
| <span class="material-symbols-outlined land-icon"> |
| rotate\_90\_degrees\_ccw |
| </span> |
| </div> |
| <div class="ms-4">  <p class="fw-bold mb-0">30 DAYS RETURN</p> |
| <span>Simply return within 30 days</span> |
| </div> |
| </div> |
| <div class="col-12 col-md-3 py-2 py-md-4 d-flex |
| align-items-center border-bottom"> |
| <div>  <span class="material-symbols-outlined land-icon"> |
| monetization\_on |
| </span> |
| </div> |
| <div class="ms-4"> |
| <p class="fw-bold mb-0">100% PAYMENT SECURE</p> |
| <span>Best payment gatway</span> |
| </div>  </div> |
| </div> |
| </section> |
| <section class="container-fluid"> |
| <div class="row"> |
| <div class="d-flex justify-content-between"> |

|  |
| --- |
| <h2> TRENDING ITEMS </h2>  <p class="text-end"><a routerLink="/products" |
| class="text-decoration-none">See all ></a> </p> |
| </div> |
| <div class="col-12 p-md-5"> |
| <app-product-card |
| [product]="productDatails"></app-product-card> |
| </div>  </div> |
| </section> |

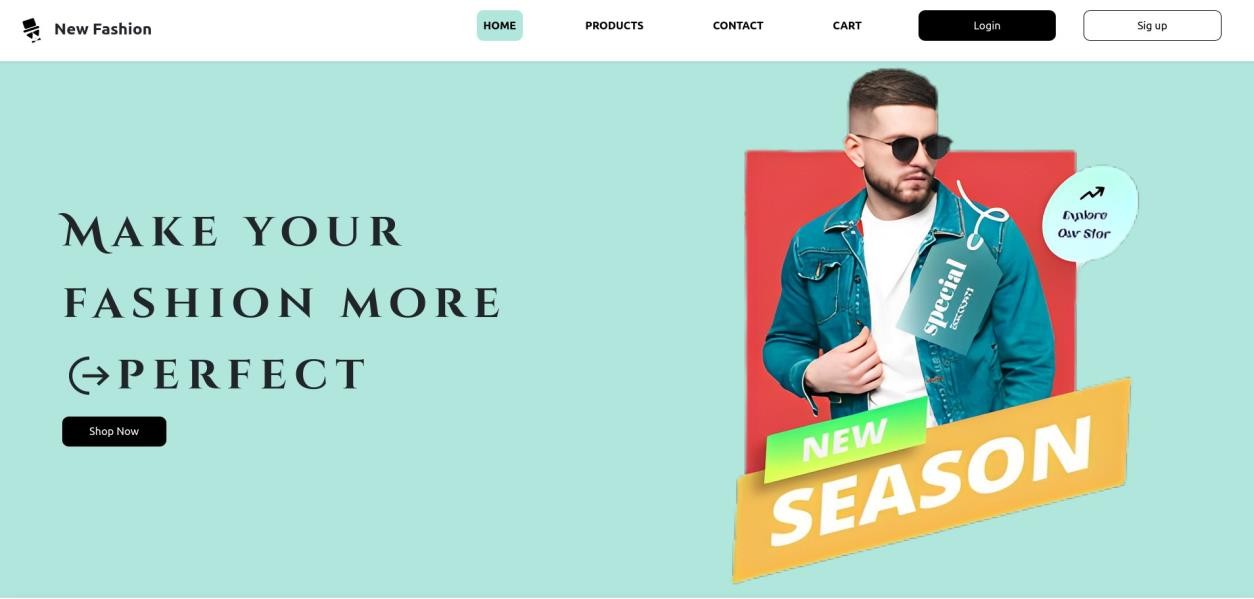
OUTPUT SCREENS: LOGIN SCREEN

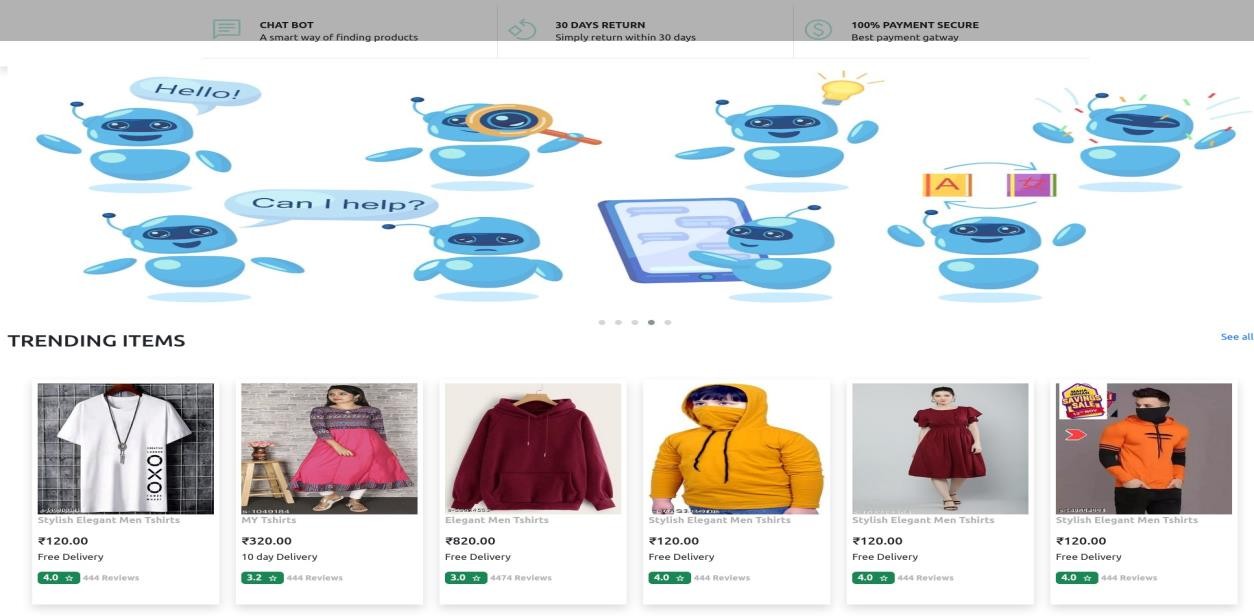


**SIGNUP SCREEN**



HOME PAGE





**PROJECT DEVELOPMENT PHASE SPRINT - 2**

|  |  |
| --- | --- |
| DATE | 05 NOVEMBER 2022 |
| TEAM ID | PNT2022TMID29308 |
| PROJECT NAME | SMART FASHION RECOMMENDER APPLICATION |

PRODUCT LIST PAGE :

|  |
| --- |
| <section class="container-fluid pt-3 pt-md-4"> |
| <!-- search filters --> |
| <div class="row"> |
| <div class="col-12 col-md-6"> |
| <div class="d-flex border">  <select name="category" class="border border-end-0 |
| border-dark py-3"> |
| <option value="all">All</option> |
| <option value="t-shirt">T-shirt</option> |
| <option value="t-shirt">T-shirt</option> |
| <option value="t-shirt">T-shirt</option> |
| </select>  <input type="text" placeholder="search" style="flex: 1;" |
| class="border-0 py-2"> |
| </div> |
| </div> |
| <div class="col-12 col-md-6"> |
| <div class="d-flex justify-content-around |
| justify-content-md-end mt-3 mt-md-0">  <!-- chat bot --> |
| <button class="chat-btn me-md-3">Let's chat</button> |
| <!-- sort by --> |
| <select name="sort" class="border-0 py-3"> |
| <option value="">Sort by</option> |
| <option value="low-high">Price: Low to High</option> |
| <option value="high-low">Price: High to Low</option>  </select> |
| </div> |
| </div> |
| </div> |

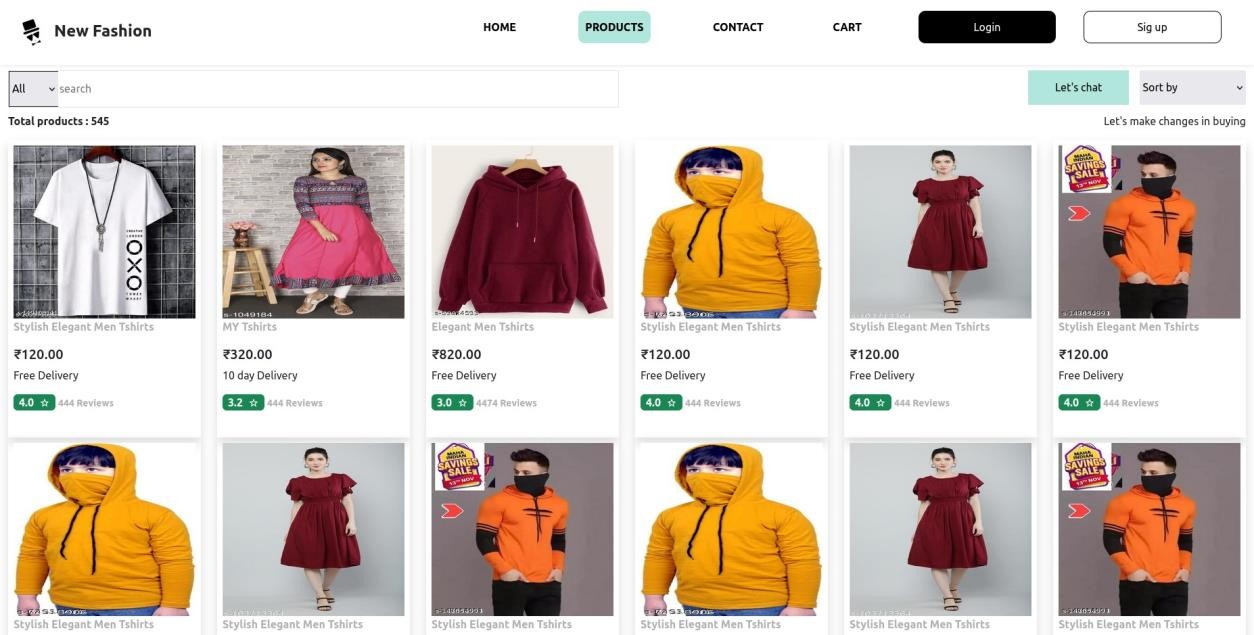
|  |
| --- |
| <!-- product list items --> |
| <div class="row mt-2"> |
| <div class="col-12"> |
| <div class="d-flex justify-content-between"> |
| <p class="fw-bold">Total products : 545</p>  <p class="d-none d-md-block">Let's make changes in |
| buying</p> |
| </div> |
| <app-product-card |
| [product]="productDatails"></app-product-card> |
| </div> |
| <!-- pagination -->  <div class="col-12 mt-5 d-flex justify-content-center"> |
| <nav aria-label="Page navigation example"> |
| <ul class="pagination"> |
| <li class="page-item"><a class="page-link" |
| href="#">1</a></li> |
| <li class="page-item"><a class="page-link" |
| href="#">2</a></li>  <li class="page-item"><a class="page-link" |
| href="#">3</a></li> |
| <li class="page-item"><a class="page-link" |
| href="#">Next</a></li> |
| </ul> |
| </nav> |
| </div>  </div> |
| </section> |

CSS:

|  |
| --- |
| .chat-btn { |
| border: none; |
| padding: 0.7rem 2.5rem; |
| background-color: var(--primary-color);  } |
| .chat-btn:hover { |
| border: 3px solid var(--primary-color); |

|  |
| --- |
| background-color: transparent; |
| } |
| .pagination .page-link { |
| color: black; |
| border: 3px solid var(--primary-color); |

OUTPUT SCREEN



BACKEND API’S:

**Login API**

|  |
| --- |
| from flask import Blueprint,jsonify,g,request |
| import ibm\_db |
| from passlib.hash import sha256\_crypt |
| import jwt |
| from ..lib import validation\_error from ..lib import exception |
| from ..lib import db |
| auth\_bp = Blueprint("auth", name ) |

@auth\_bp.route("/",methods=["GET"]) def check():

print(g.get("db"))

return jsonify({"msg":"hi"})

@auth\_bp.route('/register',methods=['POST'])

def reg(): try:

data = request.get\_json()

name=data['name']

|  |
| --- |
| email=data['email'] |
| password=data['password'] |
| mobile\_no=data['mobileNo'] |
| print(email,password,name,mobile\_no) insert\_sql="INSERT INTO |
| USER(name,email,password,role,mobilenumber) VALUES(?,?,?,?,?)" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,name) |
| ibm\_db.bind\_param(prep\_stmt,2,email) |
| ibm\_db.bind\_param(prep\_stmt,3,sha256\_crypt.encrypt(password)) |
| ibm\_db.bind\_param(prep\_stmt,4,"user") ibm\_db.bind\_param(prep\_stmt,5,mobile\_no) |
| ibm\_db.execute(prep\_stmt) |
| return {"message":'Created'},201 |
| except Exception as e: |
| return exception.handle\_exception(e) |
| @auth\_bp.route('/me',methods=['GET']) |
| def getMe(): |
| try: |
| token = request.headers['Authorization'] |
| if (not token): |

|  |
| --- |
| return validation\_error.throw\_validation("Please login",401) decoded = jwt.decode(token,"secret",algorithms=["HS256"]) |
| select\_sql = "SELECT \* FROM USER WHERE ID=?" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), select\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,decoded['id']) |
| ibm\_db.execute(prep\_stmt) |
| isUser=ibm\_db.fetch\_assoc(prep\_stmt) |
| return isUser except Exception as e: |
| return exception.handle\_exception(e) |
| @auth\_bp.route('/login',methods=['POST']) |
| def auth\_log(): |
| try:  data = request.get\_json() |
| print(data) |
| email=data['email'] |
| password=data['password'] |
| select\_sql = "SELECT \* FROM USER WHERE EMAIL=?" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), select\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,email) |
| ibm\_db.execute(prep\_stmt)  isUser=ibm\_db.fetch\_assoc(prep\_stmt) |
| print(isUser) |
| if not isUser: |
| return validation\_error.throw\_validation("Invalid |
| Credentials",400) |
| if not sha256\_crypt.verify(password,isUser['PASSWORD']): |
| return validation\_error.throw\_validation("Invalid  Credentials",400) |
| encoded\_jwt = |
| jwt.encode({"id":isUser['ID'],"role":isUser['ROLE']},"secret",algorithm |
| ="HS256") |
| isUser["token"] = encoded\_jwt |
| return isUser |
| except Exception as e:  return exception.handle\_exception(e) |

Category API

|  |
| --- |
| from flask import Blueprint,request |
| import ibm\_db  from ..lib import exception |
| from ..lib import db |
| category\_bp = Blueprint("category", name ) |
| @category\_bp.route("/",methods=["GET"]) |
| def get\_category(): |
| try: |
| select\_sql = "SELECT \* FROM CATEGORY WHERE" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), select\_sql) |
| ibm\_db.execute(prep\_stmt) |
| categories=[] category=ibm\_db.fetch\_assoc(prep\_stmt) |
| while(category != False): |
| categories.append(category) |
| category = ibm\_db.fetch\_assoc(prep\_stmt) |
| print(categories) |
| return categories,200 |
| except Exception as e: |
| return exception.handle\_exception(e) |
| @category\_bp.route("/",methods=["POST"]) |
| def add\_category(): |
| try: |
| data = request.get\_json() |
| category = data['category'] |
| insert\_sql="INSERT INTO CATEGORY(category\_name) VALUES(?)" |

|  |
| --- |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) ibm\_db.bind\_param(prep\_stmt,1,category) |
| ibm\_db.execute(prep\_stmt) |
| return {"message":'Created'},201 |
| except Exception as e: |
| return exception.handle\_exception(e) |
| @category\_bp.route("/<id>",methods=["DELETE"]) |
| def get\_category\_id(id): |
| try: |
| print(id) |
| select\_sql = "DELETE FROM CATEGORY WHERE ID=?" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), select\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,id) ibm\_db.execute(prep\_stmt) |
| return {"message":'Deleted'},200 |
| except Exception as e: |
| return exception.handle\_exception(e) |

from flask import Blueprint,request import ibm\_db

from ..lib import exception

from ..lib import db

product\_bp = Blueprint("product", name )

**Product API**

|  |
| --- |
| @product\_bp.route("/",methods=['POST']) |
| def add\_product(): |
| try: |
| data = request.get\_json() name=data['name'] |
| category=data['category'] |
| description = data['description'] |
| stock=data['stock'] |
| specificity = data['specificity'] |
| price = data['price'] |
| brand=data['brand'] insert\_sql="INSERT INTO |
| PRODUCT(product\_name,category,description,stock,specificity,price,brand |
| ) VALUES(?,?,?,?,?,?,?)" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,name) |
| ibm\_db.bind\_param(prep\_stmt,2,category) |
| ibm\_db.bind\_param(prep\_stmt,3,description) ibm\_db.bind\_param(prep\_stmt,4,stock) |
| ibm\_db.bind\_param(prep\_stmt,5,specificity) |
| ibm\_db.bind\_param(prep\_stmt,6,price) |
| ibm\_db.bind\_param(prep\_stmt,7,brand) |
| ibm\_db.execute(prep\_stmt) |
| return {"message":'Created'},201 |
| except Exception as e:  return exception.handle\_exception(e) |
| @product\_bp.route("/",methods=['GET']) |
| def get\_product(): |
| try: |
| select\_sql = "SELECT PRODUCT.ID AS product\_id, |
| category,category\_name,product\_name,description,price,stock,image,brand  ,specificity FROM PRODUCT JOIN CATEGORY ON |
| CATEGORY.ID=PRODUCT.CATEGORY" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), select\_sql) |
| ibm\_db.execute(prep\_stmt) |

|  |
| --- |
| products=[] |
| product=ibm\_db.fetch\_assoc(prep\_stmt) |
| while(product != False): products.append(product) |
| product = ibm\_db.fetch\_assoc(prep\_stmt) |
| print(products) |
| return products or [],200 |

|  |
| --- |
| except Exception as e: |
| return exception.handle\_exception(e) |
| @product\_bp.route("/<id>",methods=['GET']) |
| def get\_product\_id(id): |
| try: |
| select\_sql = "SELECT PRODUCT.ID AS product\_id, |
| category,category\_name,product\_name,description,price,stock,image,brand |
| ,specificity FROM PRODUCT JOIN CATEGORY ON CATEGORY.ID=PRODUCT.CATEGORY |
| WHERE PRODUCT.ID=?" prep\_stmt = ibm\_db.prepare(db.get\_db(), select\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,id) |
| ibm\_db.execute(prep\_stmt) |
| product=ibm\_db.fetch\_assoc(prep\_stmt) |
| print(product) |
| return product or [],200 |
| except Exception as e:  return exception.handle\_exception(e) |
| @product\_bp.route("/<id>",methods=['PUT']) |
| def update\_product(id): |
| try: |
| data = request.get\_json() name=data['name'] |
| category=data['category'] |
| description = data['description'] |
| stock=data['stock'] |
| specificity = data['specificity'] |
| price = data['price'] |

|  |
| --- |
| brand=data['brand']  insert\_sql="UPDATE PRODUCT SET |
| product\_name=?,category=?,description=?,stock=?,specificity=?,price=?,b |
| rand=? WHERE ID=?" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,name) |
| ibm\_db.bind\_param(prep\_stmt,2,category) |
| ibm\_db.bind\_param(prep\_stmt,3,description) ibm\_db.bind\_param(prep\_stmt,4,stock) |
| ibm\_db.bind\_param(prep\_stmt,5,specificity) |
| ibm\_db.bind\_param(prep\_stmt,6,price) |
| ibm\_db.bind\_param(prep\_stmt,7,brand) |
| ibm\_db.bind\_param(prep\_stmt,8,id) |
| ibm\_db.execute(prep\_stmt) |
| return {"message":'Updated'},200 |
| except Exception as e:  return exception.handle\_exception(e) |
| @product\_bp.route("/<id>",methods=['DELETE']) |
| def delete\_product(id): |
| try: |
| insert\_sql="DELETE FROM PRODUCT WHERE ID=?"  prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,id) |
| ibm\_db.execute(prep\_stmt) |
| return {"message":'Deleted'},200 |
| except Exception as e: |
| return exception.handle\_exception(e) |

Cart API

from flask import Blueprint,request import ibm\_db

from ..lib import validation\_error from ..lib.auth import check\_auth

from ..lib import exception from ..lib import db

|  |
| --- |
| cart\_bp = Blueprint("cart", name ) |
| @cart\_bp.route("/",methods=['POST']) |
| def add\_cart(): |
| try: |
| user\_id =check\_auth(request) |
| data=request.get\_json() |
| product=data['product']  select\_sql = "SELECT \* FROM PRODUCT WHERE ID=?" |
| prepare\_select =ibm\_db.prepare(db.get\_db(),select\_sql) |
| ibm\_db.bind\_param(prepare\_select,1,product) |
| ibm\_db.execute(prepare\_select) |
| is\_product = ibm\_db.fetch\_assoc(prepare\_select) |
| print(is\_product) |



|  |
| --- |
| @cart\_bp.route("/",methods=['GET']) |
| def get\_cart(): |
| try: |
| user\_id =check\_auth(request)  insert\_sql="SELECT PRODUCT.ID AS product\_id,cart\_id, |
| category,category\_name,product\_name,description,price,stock,image,brand |
| ,specificity,CART.user as user FROM CART JOIN PRODUCT ON |
| CART.PRODUCT=PRODUCT.ID JOIN CATEGORY ON PRODUCT.CATEGORY = CATEGORY.ID |
| WHERE CART.USER=?" |
| prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,user\_id) |
| ibm\_db.execute(prep\_stmt) |
| products=[] |
| product=ibm\_db.fetch\_assoc(prep\_stmt) |
| while(product != False): |
| products.append(product) |
| product = ibm\_db.fetch\_assoc(prep\_stmt) print(products) |
| return products or [],200 |
| except Exception as e: |
| return exception.handle\_exception(e) |
| @cart\_bp.route("/<product>/<id>",methods=['DELETE']) |
| def delete\_cart(product,id): |
| try: |
| user\_id =check\_auth(request) |
| print(product,id,user\_id) |

|  |
| --- |
| select\_sql = "SELECT \* FROM PRODUCT WHERE ID=?"  prepare\_select =ibm\_db.prepare(db.get\_db(),select\_sql) |
| ibm\_db.bind\_param(prepare\_select,1,product) |
| ibm\_db.execute(prepare\_select) |
| is\_product = ibm\_db.fetch\_assoc(prepare\_select) |
| print(is\_product) |
| if not is\_product: |
| return validation\_error.throw\_validation("No Product found",404) |
| print("ff")  insert\_sql="DELETE FROM CART WHERE CART\_ID=? AND user=?" prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) ibm\_db.bind\_param(prep\_stmt,1,id) ibm\_db.bind\_param(prep\_stmt,2,user\_id) ibm\_db.execute(prep\_stmt)  print("aa")  update\_sql="UPDATE PRODUCT SET stock=? WHERE ID=?" update\_stmt = ibm\_db.prepare(db.get\_db(), update\_sql) ibm\_db.bind\_param(update\_stmt,1,is\_product['STOCK']+1) ibm\_db.bind\_param(update\_stmt,2,product) ibm\_db.execute(update\_stmt)  return {"message":'Deleted'},200 except Exception as e:  return exception.handle\_exception(e) |

**Order API:**

from flask import Blueprint,request import ibm\_db

from ..lib import exception from ..lib import db,auth

|  |
| --- |
| order\_bp = Blueprint("order", name ) |
| @order\_bp.route("/",methods=['POST']) def add\_order(): |
| try: |
| user\_id =auth.check\_auth(request) |
| data=request.get\_json() |
| products=data['products'] |
| insert\_sql="SELECT ORDER\_ID FROM FINAL TABLE (INSERT INTO |
| ORDER(user) VALUES(?))" prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) |
| ibm\_db.bind\_param(prep\_stmt,1,user\_id) |
| ibm\_db.execute(prep\_stmt) |
| order = ibm\_db.fetch\_assoc(prep\_stmt) |
| print(order) |
| for product in products: |

print(product)

insert1\_sql="INSERT INTO ORDERDETAIL(order,product) VALUES(?,?)" prep1\_stmt = ibm\_db.prepare(db.get\_db(), insert1\_sql) ibm\_db.bind\_param(prep1\_stmt,1,order['ORDER\_ID']) ibm\_db.bind\_param(prep1\_stmt,2,product) ibm\_db.execute(prep1\_stmt)

return {"message":'Created'},201 except Exception as e:

return exception.handle\_exception(e)

@order\_bp.route("/<id>",methods=['GET']) def get\_order(id):

try:

insert\_sql="SELECT PRODUCT.ID AS product\_id, category,category\_name,product\_name,description,price,stock,image,brand

,specificity,paid FROM ORDERDETAIL JOIN ORDER ON ORDERDETAIL.ORDER=ORDER.ORDER\_ID JOIN PRODUCT ON ORDERDETAIL.PRODUCT=PRODUCT.ID JOIN CATEGORY ON PRODUCT.CATEGORY = CATEGORY.ID WHERE ORDER.USER=?"

prep\_stmt = ibm\_db.prepare(db.get\_db(), insert\_sql) ibm\_db.bind\_param(prep\_stmt,1,id) ibm\_db.execute(prep\_stmt)

products=[] product=ibm\_db.fetch\_assoc(prep\_stmt) while(product != False):

products.append(product)

product = ibm\_db.fetch\_assoc(prep\_stmt) print(products)

return products or [],200

except Exception as e:

return exception.handle\_exception(e)

PROJECT DEVELOPMENT PHASE SPRINT - 3

|  |  |
| --- | --- |
| DATE | 16 NOVEMBER 2022 |
| TEAM ID | PNT2022TMID29308 |
| PROJECT NAME | SMART FASHION RECOMMENDER APPLICATION |

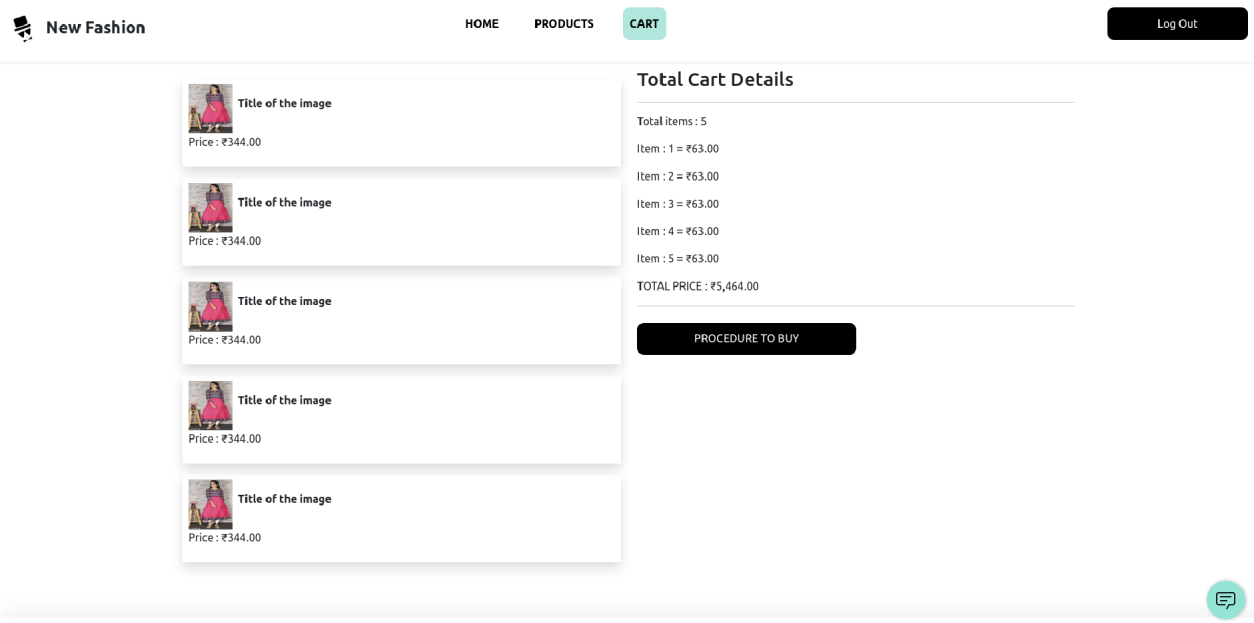
**CART PAGE :**

|  |
| --- |
| <section class="container mt-3 pt-3 pt-md-4"> |
| <div class="row"> |
| <div class="col-12 col-md-6"> |
| <div class="shadow p-2 mt-3" \*ngFor="let t of [1,1,1,1,1]"> |
| <div class="d-flex align-items-center">  <img src="/assets/products/2.jpg" alt="Product |
| image" class="product-img" /> |
| <p class="fw-bold ms-2">Title of the image</p> |
| </div> |
| <p>Price : {{344 | currency:'INR'}}</p> |
| </div> |
| </div>  <div class="col-12 col-md-6"> |
| <h3>Total Cart Details</h3> |
| <hr> |
| <div> |
| <p>Total items : {{5}}</p> |
| <p \*ngFor="let t of [1,1,1,1,1], index as i">Item : {{i |
| + 1}} = {{63 | currency:'INR'}}</p>  <p>TOTAL PRICE : {{5464 | currency:"INR"}}</p> |
| <hr> |
| <div class="col-12 col-md-6"> |
| <button class="primary-btn mt-2 w-100">PROCEDURE TO |
| BUY</button> |
| </div> |
| </div>  </div> |
| </div> |
| </section> |

CART CSS:

|  |
| --- |
| .product-img { |
| width: 4rem; |
| height: 4.5rem; |
| } |

OUTPUT SCREEN



VIEW PRODUCT PAGE HTML

|  |
| --- |
| <section class="container-fluid pt-3 pt-md-4"> |
| <div class="row justify-content-around"> |
| <div class="col-12 col-md-6 col-lg-4"> |
| <img src="/assets/products/1.jpg" alt="product image" |
| class="w-100"> |
| </div>  <div class="col-12 col-md-6 col-lg-4"> |
| <h2>New product</h2> |
| <p class="text-muted"> |
| Women Beige Floral Yoke Design Straight Kurta with |
| Trousers & With Dupatta |
| </p> |
| <div class="d-flex align-items-center">  <div class="d-flex align-items-center bg-success |
| text-white fw-bold px-2 rounded"> |
| <span> |
| {{'4.0'}} |
| </span> |

|  |
| --- |
| <span class="material-symbols-outlined text-white |
| ms-2 fs-6"> |

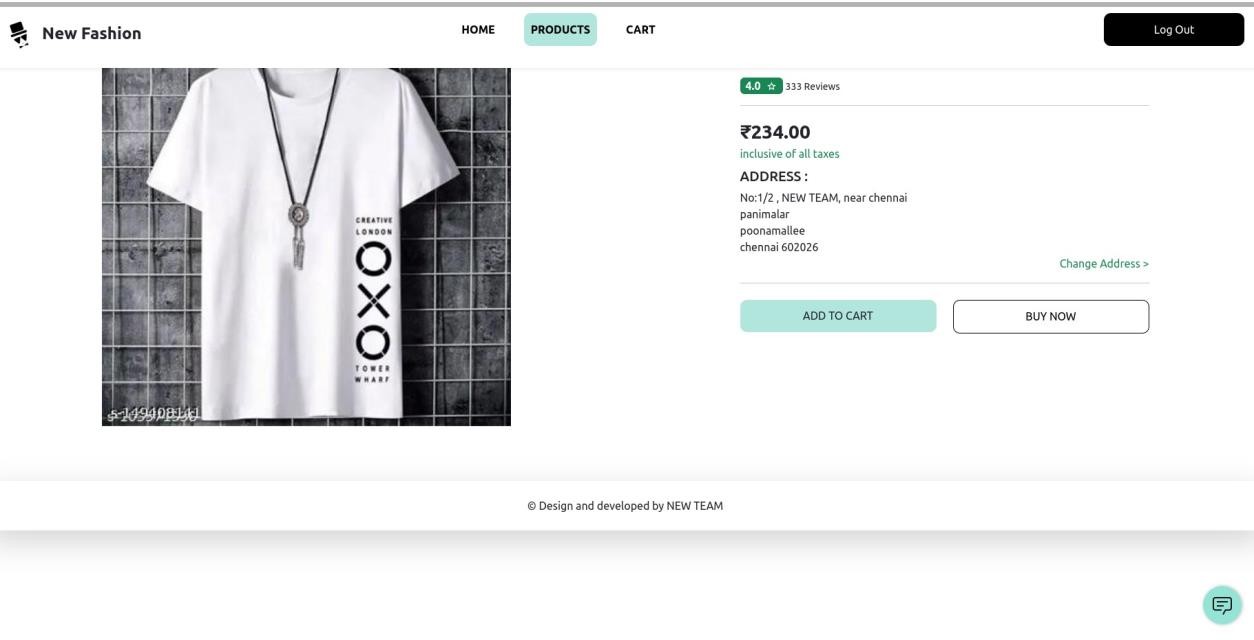
|  |
| --- |
| star |
| </span> |
| </div> |
| <div class="title-text ms-1" style="font-size: 14px;">  {{'333'}} Reviews |
| </div> |
| </div> |
| <hr> |
| <!-- pricedetails --> |
| <div> |
| <p class="mb-0 fw-bold fs-3">{{'234' | currency:  'INR'}}</p> |
| <span class="text-success">inclusive of all taxes</span> |
| </div> |
| <!-- address details --> |
| <div class="mt-2"> |
| <h5>ADDRESS : </h5> |
| <p class="mb-0">No:1/2 , NEW TEAM, near chennai</p>  <p class="mb-0">panimalar</p> |
| <p class="mb-0">poonamallee</p> |
| <p class="mb-0">chennai 602026</p> |
| <p class="text-end text-success cursor-pointer"><a |
| [routerLink]="">Change Address > </a></p> |
| </div> |
| <hr>  <!-- button section --> |
| <div class="row mt-3"> |
| <div class="col-12 col-md-6"> |
| <button class="primary-btn mt-2 text-dark w-100" |
| style="background-color: var(--primary-color);">ADD |
| TO CART</button> |
| </div>  <div class="col-12 col-md-6"> |
| <button class="secondary-btn mt-2 w-100">BUY |
| NOW</button> |
| </div> |
| </div> |

|  |
| --- |
| </div> |
| </div>  </section> |

**VIEW PRODUCT LOGIC**

|  |
| --- |
| // get details |
| async getProductDetails(): Promise<void>{ |
| try { |
| this.productDetails = await |
| this.productServe.getProductById(this.productID); |
| } catch (error) { console.log(error); |
| } |
| } |

OUTPUT SCREEN



**ADMIN PANEL HOME PAGE HTML**

|  |
| --- |
| <div class="container-fluid h-100"> |
| <div class="row h-100 flex-nowrap position-fixed w-100"> |
| <!-- nav bar --> |
| <div class="col-8 col-md-3 col-lg-2 h-100 nav-con" |
| \*ngIf="showNav"> |
| <!-- logo image --> |

|  |
| --- |
| <div class="fw-bold text-dark">  <img class="logo-img" src="/assets/logo.png" /> |
| <span class="d-none d-md-inline">New Fashion</span> |
| <div class="d-md-none"> |
| <span>New Fashion</span> |

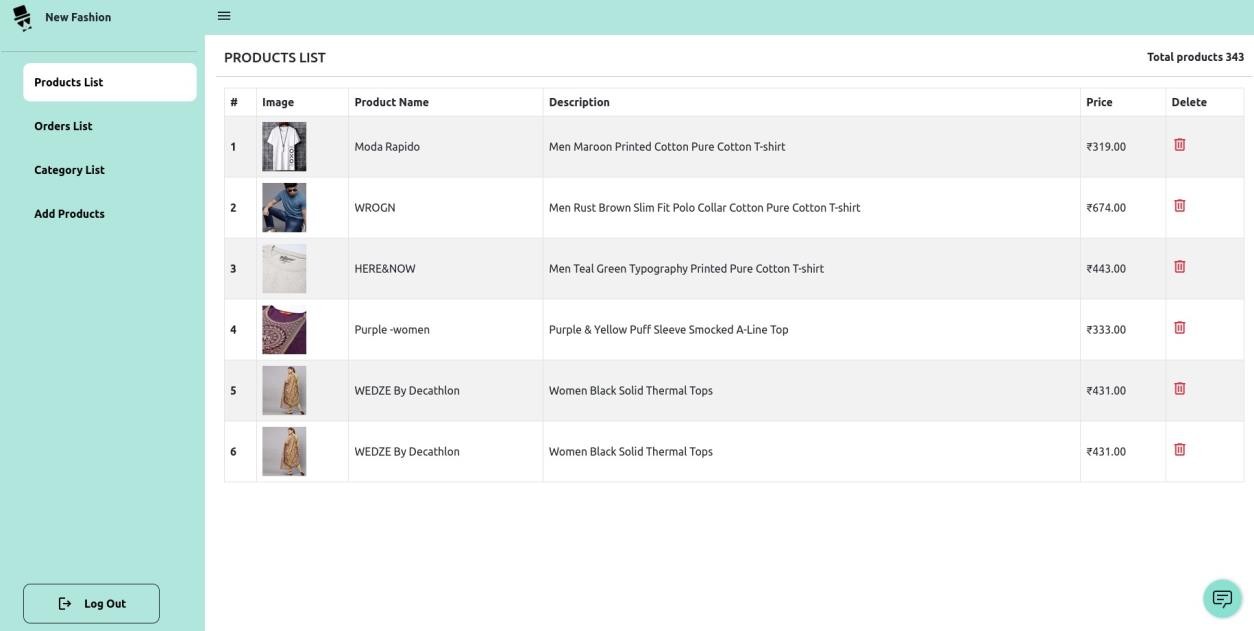
|  |
| --- |
| </div> |
| </div> |
| <hr> |
| <nav class="nav-bar">  <ul> |
| <li class="mt-2"><a class="py-3 ps-3 nav-link" |
| routerLinkActive="active" routerLink="/admin" |
| [routerLinkActiveOptions]="{exact:true}">Products |
| List</a></li> |
| <li class="mt-2"><a class="py-3 ps-3 nav-link" routerLinkActive="active" routerLink="orders" |
| [routerLinkActiveOptions]="{exact:true}">Orders |
| List</a></li> |
| <li class="mt-2"><a class="py-3 ps-3 nav-link" |
| routerLinkActive="active" routerLink="category" |
| [routerLinkActiveOptions]="{exact:true}">Category |
| List</a></li> |
| <li class="mt-2"><a class="py-3 ps-3 nav-link" |
| routerLinkActive="active" routerLink="add-product" |
| [routerLinkActiveOptions]="{exact:true}">Add |
| Products</a></li> |
| <li class="mt-2 position-fixed bottom-0 mb-4"><a  class="py-3 px-3 px-md-5 nav-link border border-dark" |
| (click)="logout()"><span |
| class="material-symbols-outlined" |
| style="vertical-align: bottom;"> |
| logout |
| </span> <span class="ms-3">Log |
| Out</span></a></li>  </ul> |

|  |
| --- |
| </nav> |
| </div> |
| <!-- top nav content --> |
| <div class="col-12 col-md-9 col-lg-10 p-0 flex-fill |
| overflow-scroll"> |
| <div class="d-flex top-nav px-3 py-3 w-100  justify-content-between"> |
| <div class="material-symbols-outlined cursor-pointer" |
| (click)="toggleNav()"> |
| menu |
| </div> |
| <div> |
| profile |
| </div>  </div> |
| <!-- main content of route --> |
| <main class="p-1 p-md-3"> |
| <router-outlet></router-outlet> |
| </main> |
| </div> |
| </div>  </div> |

CSS

|  |
| --- |
| .nav-con, .top-nav { |
| background-color: var(--primary-color); |
| } |
| a { |
| text-decoration: none; color: black; |
| } |
| .nav-bar > ul { |
| list-style: none; |
| font-weight: bold; |
| } |
| .nav-bar a { |
| border-radius: 10px; |
| } |
| .top-nav { |

|  |
| --- |
| position: fixed; |
| top: 0;  } |
| main { |
| margin-top: 60px; |
| } |
| .nav-link:hover, |
| .nav-link.active { |
| background-color: white;  } |
| .logo-img { |
| width: 4rem; |
| height: 4rem; |
| } |

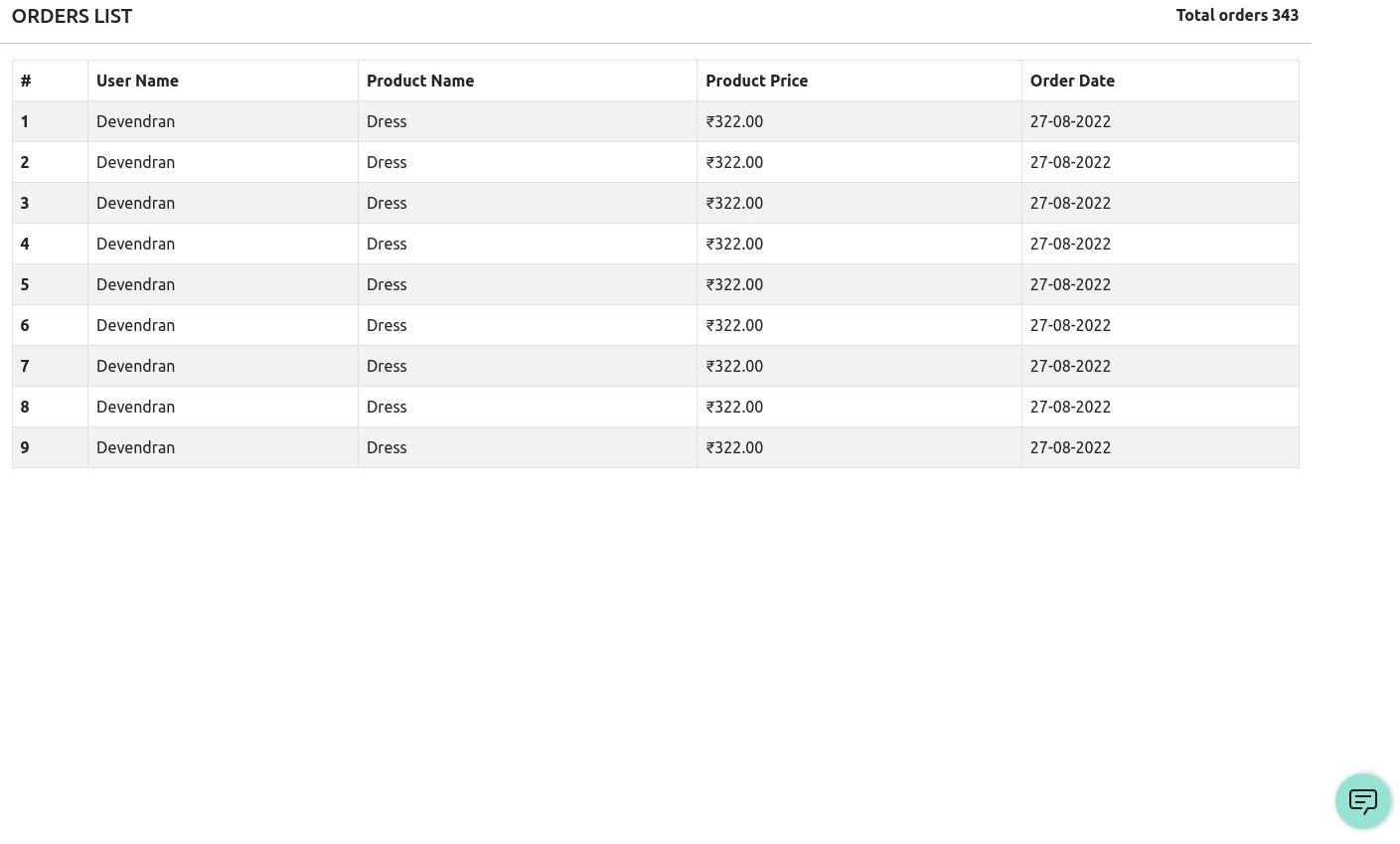
**OUTPUT SCREEN:**

ORDER PAGE HTML PAGE:

|  |
| --- |
| <section class="container">  <div class="row"> |
| <div class="d-flex justify-content-between"> |
| <h5>ORDERS LIST</h5> |
| <p class="fw-bold">Total orders {{'343'}}</p> |
| </div> |

|  |
| --- |
| <hr> |
| <!-- orders list -->  <div class="col-12"> |
| <div class="table-responsive"> |
| <table class="table table-striped table-hover |
| table-bordered"> |
| <thead> |
| <tr> |
| <th scope="col">#</th>  <th scope="col">User Name</th> |
| <th scope="col">Product Name</th> |
| <th scope="col">Product Price</th> |
| <th scope="col">Order Date</th> |
| </tr> |
| </thead> |
| <tbody class="align-middle"> |
| <tr \*ngFor="let t of [1,1,1,1,1,1,1,1,1], index  as i"> |
| <th scope="row">{{i+1}}</th> |
| <td>Devendran</td> |
| <td>Dress</td> |
| <td>{{'322' | currency: 'INR'}}</td> |
| <td>{{'27-08-2022'}}</td> |
| </tr>  </tbody> |
| </table> |
| </div> |
| </div> |
| </div> |
| </section> |

**OUTPUT SCREEN:**



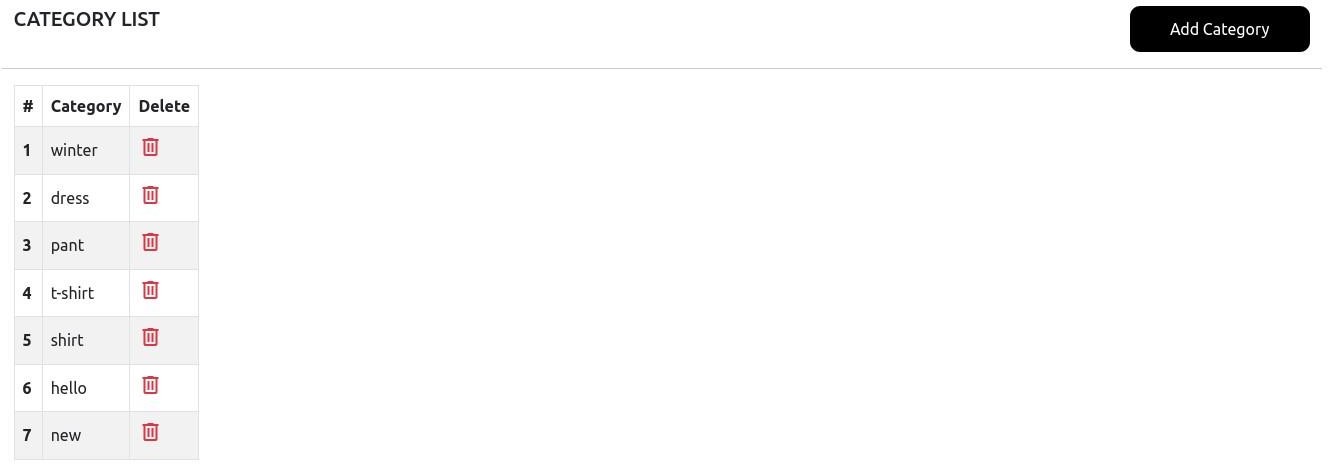
CATEGORY PAGE HTML

|  |
| --- |
| <section class="container"> |
| <div class="row"> |
| <div class="d-flex justify-content-between mb-3"> |
| <h5>CATEGORY LIST</h5>  <div class="position-relative"> |
| <button class="primary-btn" |
| (click)="toggleAddCategory()">Add |

|  |
| --- |
| Category</button> |
| <span class="position-absolute add-category-card shadow |
| border text-success" |
| \*ngIf="successMessage !==  ''">{{successMessage}}</span> |
| <!-- add category --> |
| <form class="position-absolute add-category-card shadow |
| border" \*ngIf="showAddCategory" |
| [formGroup]="addCategory" |
| (ngSubmit)="createCategory()"> |
| <input type="text" placeholder="New Category Name"  class="form-input" formControlName="category"> |
| <div class="text-center"> |
| <button class="secondary-btn" type="submit" |
| style="background-color: var(--primary-color);" |

|  |
| --- |
| [disabled]="this.addCategory.invalid">Create</button> |
| </div>  </form> |
| </div> |
| </div> |
| <hr> |
| <!-- orders list --> |
| <div \*ngIf="categoryList.length === 0"> |
| Loading  <div class="spinner-border text-dark ms-3" |
| style="vertical-align: bottom;" role="status"> |
| <span class="visually-hidden">Loading...</span> |
| </div> |
| </div> |
| <div class="col-12" \*ngIf="categoryList.length > 0"> |
| <div class="table-responsive">  <!-- table --> |
| <table class="table table-striped table-hover |
| table-bordered w-auto"> |
| <thead> |
| <tr> |
| <th scope="col">#</th> |
| <th scope="col">Category</th>  <th scope="col">Delete</th> |
| </tr> |
| </thead> |
| <tbody class="align-middle"> |
| <tr \*ngFor="let t of categoryList, index as i"> |
| <th scope="row">{{i+1}}</th> |
| <td>{{t.CATEGORY\_NAME}}</td> |
| <td>  <span class="material-symbols-outlined |
| cursor-pointer text-danger" |
| (click)="deleteCategory(t.ID)"> |
| delete |
| </span> |
| </td> |
| </tr> |
| </tbody> |

|  |
| --- |
| </table> |
| </div> |
| </div> |
| </div> |
| </section> |

**OUTPUT SCREEN:**

ADD PRODUCT SCREEN HTML:

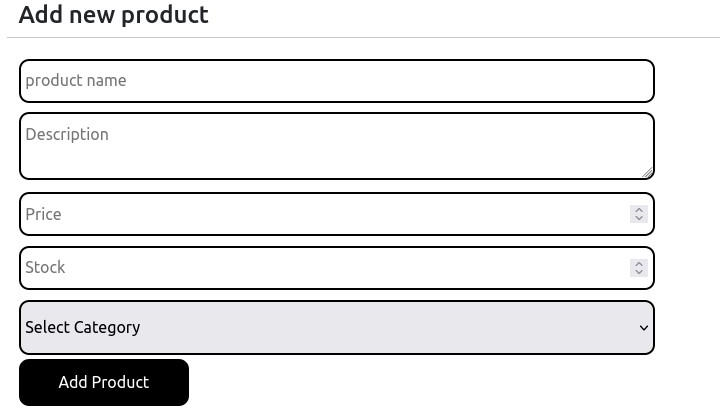
|  |
| --- |
| <section class="container"> |
| <div class="row">  <h4>Add new product</h4> |
| <hr> |
| <form class="col-12 col-md-6" [formGroup]="addProductForm" |
| (ngSubmit)="createProduct()"> |
| <input type="text" class="form-input w-100" |
| placeholder="product name" formControlName="product\_name"> |
| <textarea type="text" class="form-input w-100"  placeholder="Description" formControlName="description"></textarea> |
| <input type="number" class="form-input w-100" |
| placeholder="Price" min="1" formControlName="price"> |
| <input type="number" class="form-input w-100" |
| placeholder="Stock" min="1" formControlName="stock"> |
| <select name="category" class="form-input w-100 py-3" |
| formControlName="category">  <option value="">Select Category</option> |
| <option [value]="c.ID" \*ngFor="let c of |
| categoryList">{{c.CATEGORY\_NAME}}</option> |
| </select> |
| <div> |
| <button class="primary-btn" type="submit">Add |

|  |
| --- |
| Product</button>  </div> |
| </form> |
| <!-- <div class="col-12 col-md-6 text-center"> |
| <p>select product image</p> |
| <img src="/assets/products/4.jpg" alt="Product image" |
| class="product-img"> |
| </div> -->  </div> |
| </section> |

import ibm\_db

conn = None

OUTPUT SCREEN:



**DB CONNECTION**

|  |
| --- |
| def get\_db(): |
| global conn |
| print(conn) if conn == None: |
| conn = |
| ibm\_db.connect("DATABASE=bludb;HOSTNAME=9938aec0-8105-433e-8bf9-0fbb7e4 |
| 83086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32459;SECURIT |
| Y=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=yvp21124;PWD=jN |
| DAkHSrZNopa2oe",'','') |
| return conn |

TOKEN VERIFY

import jwt

from . import validation\_error

def check\_auth(request):

token = request.headers['Authorization'] if (not token):

return validation\_error.throw\_validation("Please login",401)

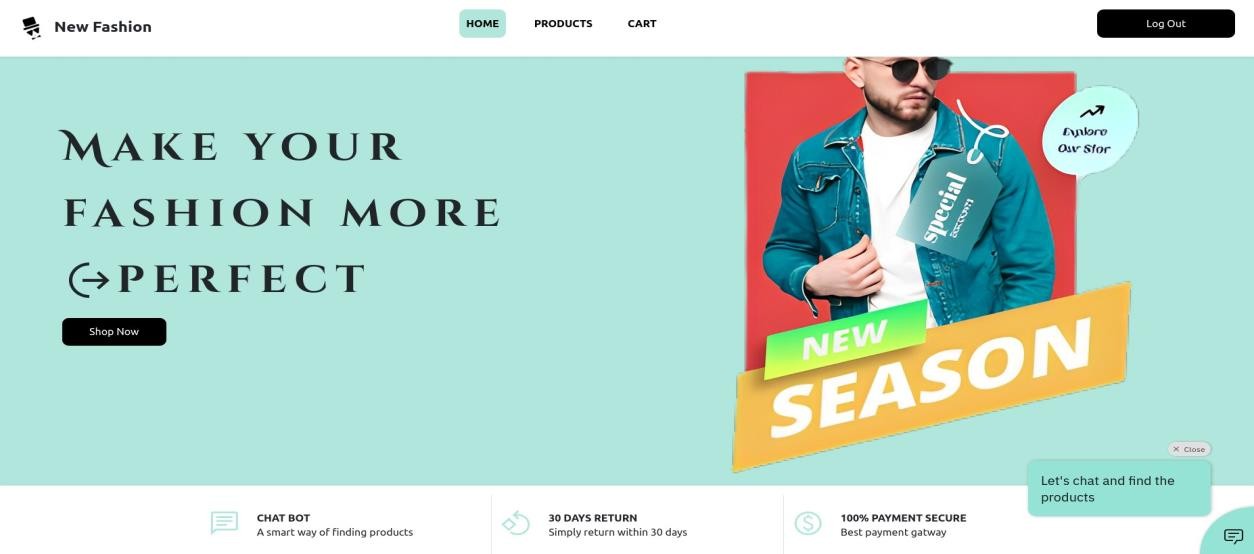
|  |
| --- |
| decoded = jwt.decode(token,"secret",algorithms=["HS256"]) |
| return decoded['id'] |

**PROJECT DEVELOPMENT PHASE SPRINT - 4**

|  |  |
| --- | --- |
| DATE | 17 NOVEMBER 2022 |
| TEAM ID | PNT2022TMID29308 |
| PROJECT NAME | SMART FASHION RECOMMENDER APPLICATION |

CHATBOT INTEGRATION

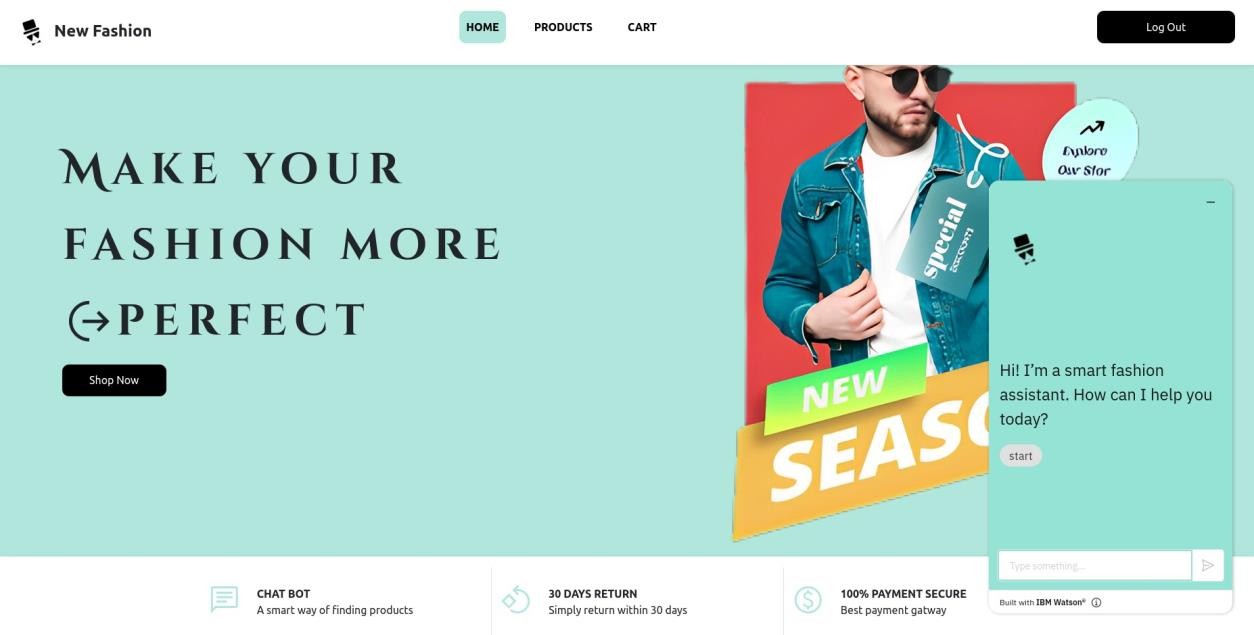
|  |
| --- |
| let body = <HTMLDivElement> document.body; window.watsonAssistantChatOptions = { |
| integrationID: "491873b3-90fe-47a0-867b-484cc805a0b4", // The ID |
| of this integration. |
| region: "au-syd", // The region your integration is hosted in. |
| serviceInstanceID: "65396cc7-c978-42a3-8def-8644b440581d", // |
| The ID of your service instance. |
| onLoad: function (instance: { render: () => void; }) {  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"; |
| body.appendChild(t); |

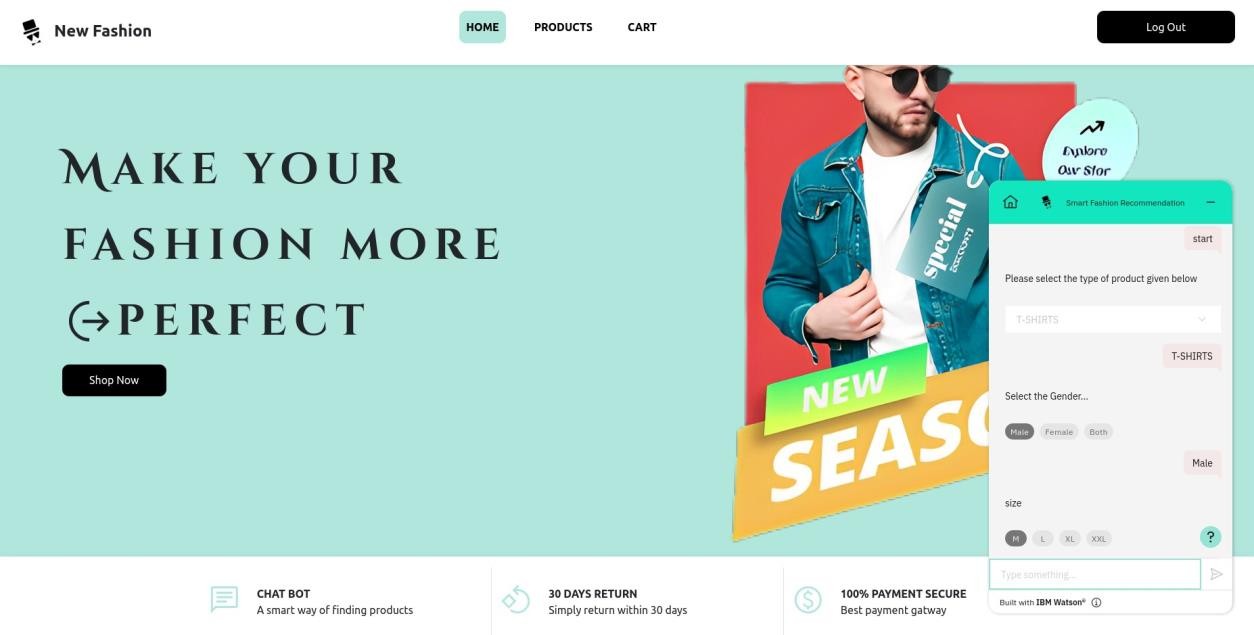


});

**CHATBOT UI DESIGNS**

MAIN CHATBOT OPEN SCREEN



**CHAT SCREEN**

FINDING PRODUCTS USING CHATBOT

