SMART FASHION RECOMMENDER APPLICATION
DOMAIN : CLOUD

IBM – REPORT

UNDER THE GUIDANCE OF

INDUSTRY MENTOR(S) NAME	:	KRISHNA CHAITANYA
FACULTY MENTOR(S) NAME	:	KALPANA C

TEAM ID : PNT2022TMID48658

SUBMITTED BY:

THIYAGARAJAN S	920819104049
THIRUNAVUKKARASAR T	920819104048
PAULSANTHOSHKUMAR J	920819104026
GUHAN P	920819104011

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NPR COLLEGE OF ENGINEERING AND TECHNOLOGY

ANNA UNIVERSITY :: 2019 – 2023

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO.
1.	**INTRODUCTION**	1
	1.1 Project Overview	2
	1.2 Purpose	2
2.	**LITERATURE SURVEY**	
	2.1 Existing problem	3
	2.2 References	5
	2.3 Problem Statement Definition	5
3.	**IDEATION & PROPOSED SOLUTION**	
	3.1 Empathy Map Canvas	7
	3.2 Ideation & Brainstorming	7
	3.3 Proposed Solution	8
	3.4 Problem Solution Fit	9
4.	**REQUIREMENT ANALYSIS**	
	4.1 Functional requirement	10

4.2 Non-Functional requirements 10

5. PROJECT DESIGN

5.1 Data Flow Diagram 11

5.2 Solution & Technical Architecture 11

5.3 User Stories 12

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation 13

6.2 Sprint Delivery Schedule 13

6.3 Reports from JIRA 14

7. CODING & SOLUTIONING

7.1 Feature 1 15

7.2 Feature 2 15

7.3 Data base schema 16

8. TESTING

8.1 Test Cases 17

8.2 User Acceptance Testing 18

9. RESULTS

9.1 Performance Metrics 19

10. ADVANTAGES AND DISADVANTAGES 20

11. CONCLUSI ON 21

12. FUTURE SCOPE 22

13. APPENDIX

SOURCE CODE & OUTPUT 23

GITHUB & PROJECT DEMO LINK 55

1. INTRODUCTION

1.1 PROJECT OVERVIEW

Nowadays, fashion applications and e-commerce are growing more and more, and it also has some problems when finding the customer's wanted product in the web applications. Having a chatbot that understands the algorithm of a specific application can be of great aid. We are implementing such a chat bot in a web application, 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 worksas an advanced filter search that can bring the user what they want with the help of pictorial and named representation by getting simple user information and activities. The application also has two main UI interactions: one is the user panel and the other one is the admin panel. Users can interact with the chat bot to search for products, order them from the manufacturer or distributor through chatbot AI, and it can also make payment transactions, track the delivery, and so on. The admin interface enables the userto upload products' details, user details, orders and find how many products have been bought; supervisethe stock availability; and interact with the buyer regarding the product reviews.

We have come up with a new innovative solution through which you can directly do youronline 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:

1. Admin and
2. User

Admin:

The role of the admin is to check out the database about the stock and have a track of all thethings that the users are purchasing.

User:

The user will login into the website and go through the products available on the website. Instead
of navigating to several screens for booking products online, the user can directly talk to Chat botregarding the products. Get the recommendations based on information provided by the user.

1.2 PURPOSE

a) Using chatbot we can manage user's choices and orders.

b) The chatbot can give recommendations to the users based on their interests.

c) It can promote the best deals and offers on that day.

d) It will store the customer's details and orders in the database.

e) The chatbot will send a notification to customers if the order is confirmed.

2. LITERATURE SURVEY

2.1EXISTING PROBLEM

1. Fashion item representation

Traditional recommender systems such as Collaborative Filtering or Content-Based Filtering have difficulties in the fashion domain due to the sparsity of purchase data, or the insufficient detailabout the visual appearance of the product in category names. Instead, more recent literature has leveraged models that capture a rich representation of fashion items through product images, text descriptions or customer reviews, or videos which are often learned through surrogate tasks like classification or product retrieval. However, learning product representations from such input data requires large datasets to generalize well across different image (or text) styles, attribute variations,etc. Furthermore, constructing a representation that learns which product features customers take most into account when evaluating fashion products is still an open research problem.

2. Fashion item compatibility

Training a model that is able to predict if two fashion items ‘go together,’ or directly combine several products into an outfit, is a challenging task. Different item compatibility signals studied inrecent literature include co-purchase data, outfits composed by professional fashion designers, or combinations found by analyzing what people wear in social media pictures.

3. Personalization and fit

The best fashion product to recommend depends on factors such as the location where the outfitwill be used, the season or occasion, or the cultural and social background of the customer. A challenging task in fashion recommendation systems is how to discover and integrate these disparatefactors. Current research often tackles these tasks by utilizing large-scale social media data.

4. Interpretability and explanation

Most of the existing fashion recommender systems in the literature focus on improving predictive performance, treating the model as a black box. However, deploying accountable and interpretable systems able to explain their recommendations can foster user loyalty in the long termand improve the shopping experience

5. Discovering trends

Being able to forecast consumer preferences is valuable for fashion designers and retailers in order to optimize product-to-market fit, logistics and advertising.

2.2REFERENCES

“A Systematic Study on the Recommender Systems in the E-Commerce”

Electronic commerce or e-commerce includes the service and good exchange through electronicsupport 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 thetraditional 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).

2.3PROBLEM STATEMENT DEFINITION

Problem Statement 1:

The User Needs a way to Find Trending Fashion Clothes so that Here find the All CollectionsProblem Statement 2:

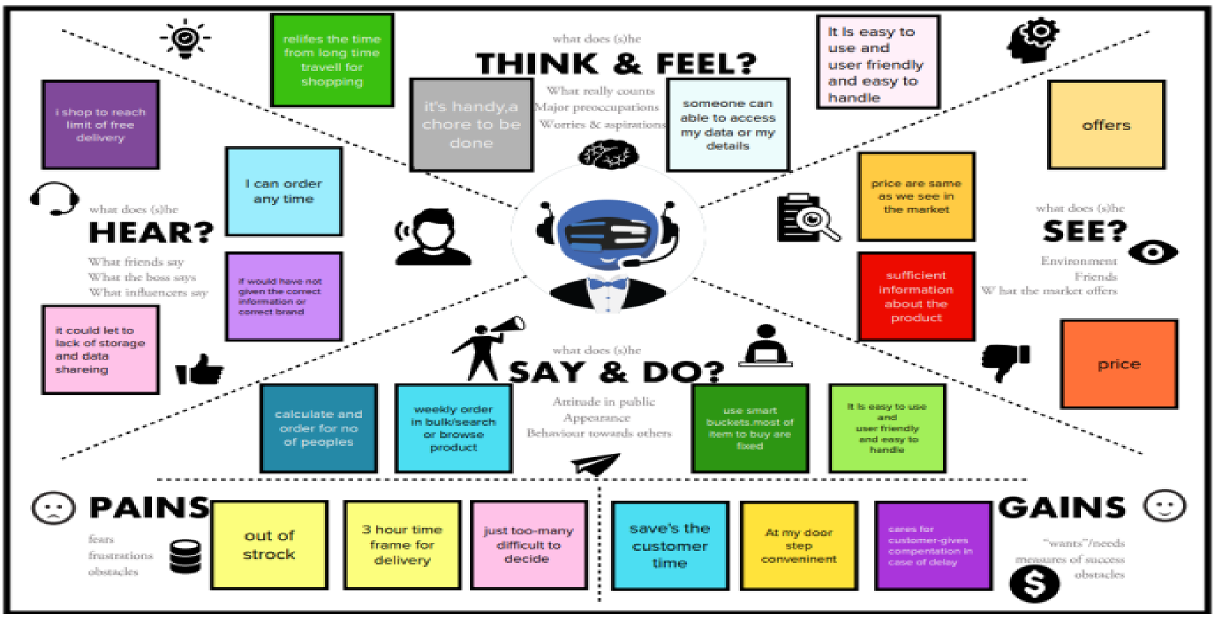
The User Needs a way to Find Offers and Discounts so that Here User easy to find Daily OffersProblem Statement 3:

Chat Bot assistantProblem Statement 4:

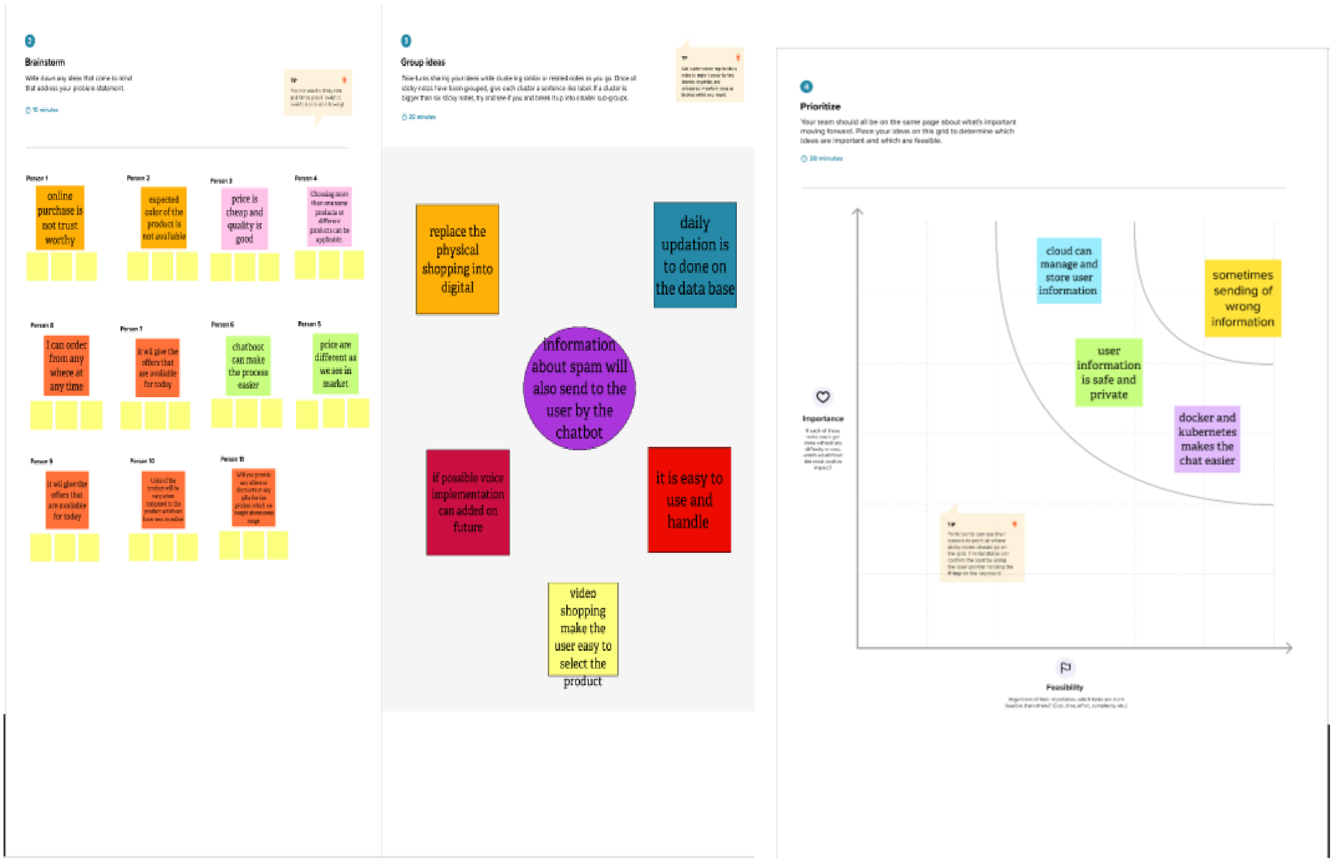
The Sellers Needs a way to struggling to sells products offline so that Here Sellers will Sell Productsvia our application.

3. IDEATION & PROPOSED SOLUTION

3.1EMPATHY MAP CANVAS


relifes the time from long time travell for shopping
what does (s)he
THINK & FEEL?
What really counts
Major preoccupations
Worries & aspirations
it's handy,a chore to be done
someone can able to access my data or my details
It is easy to use and user friendly and easy to handle
i shop to reach limit of free delivery
offers
I can order any time
what does (s)he
HEAR?
What friends say
What the boss says
What influencers say
if would have not given the correct information or correct brand
price are same as we see in the market
what does (s)he
SEE?
Environment
Friends
What the market offers
sufficient information about the product
it could let to lack of storage and data shareing
price
what does (s)he
SAY & DO?
Attitude in public
Appearance
Behaviour towards others
calculate and order for no of peoples
weekly order in bulk/search or browse product
use smart buckets,most of item to buy are fixed
It is easy to use and user friendly and easy to handle
PAINS
fears
frustrations
obstacles
out of strock
3 hour time frame for delivery
just too-many difficult to decide
save's the customer time
At my door step convenient
cares for customer-gives compentation in case of delay
GAINS
"wants"/needs
measures of success
obstacles

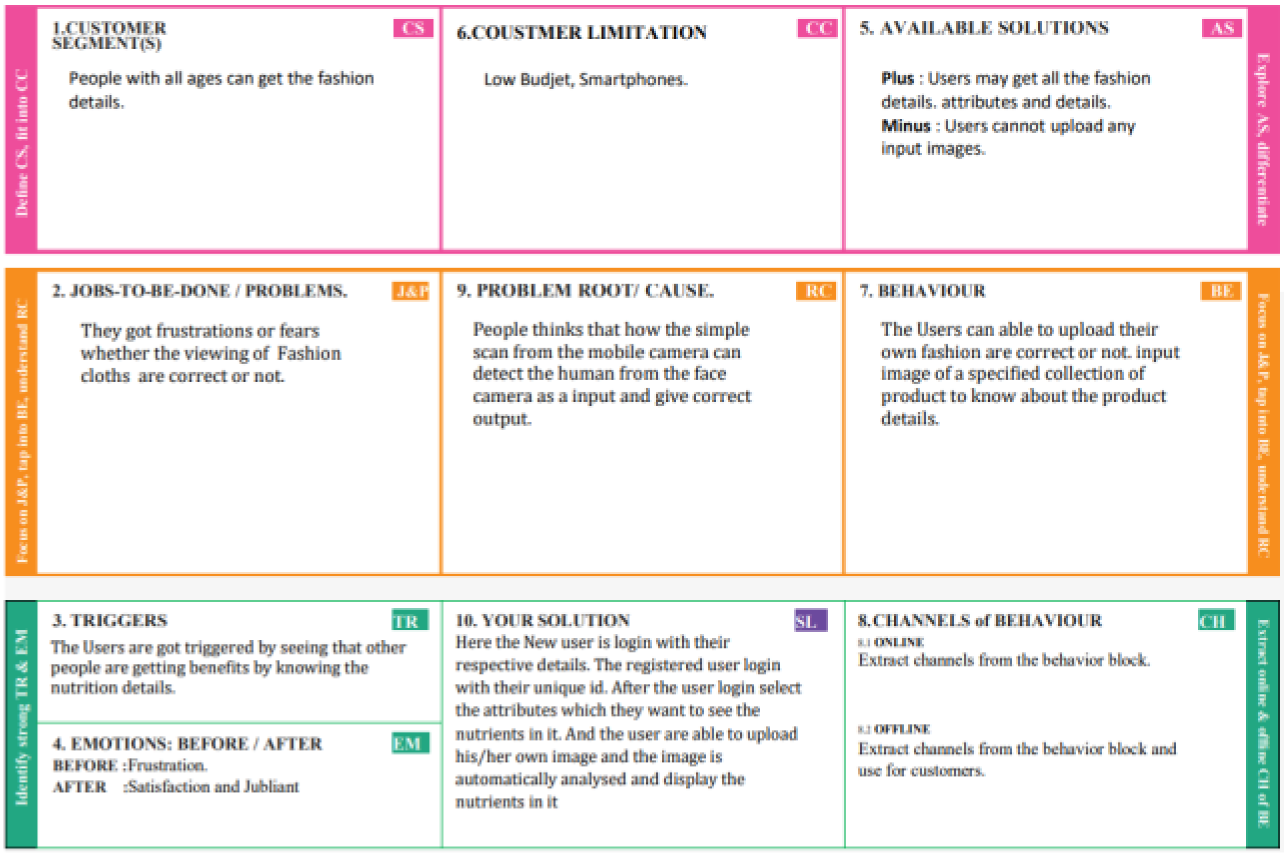
3.2 IDEATION & BRAINSTORMING


Brainstorm
Write down any ideas that come to mind that address your problem statement.
10 minutes
Person 1
online purchase is not trust worthy
Person 2
expected color of the product is not available
Person 3
price is cheap and quality is good
Person 4
Choosing more than one same products or different products can be applicable
Person 8
I can order from any where at any time
Person 7
it wil give the offers that are avaliable for today
Person 6
chatboot can make the process easier
Person 5
price are different as we see in market
Person 9
it wil give the offers that are available for today
Person 10
Person 11
Group ideas
20 minutes
replace the physical shopping into digital
daily updation is to done on the data base
information about spam will also send to the user by the chatbot
if possible voice implementation can added on future
it is easy to use and handle
video shopping make the user easy to select the product
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
cloud can manage and store user information
sometimes sending of wrong information
user information is safe and private
docker and kubernetes makes the chat easier
Importance
Feasibility
TIP

3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Customers feels difficult when Search many websites to find Fashion clothes and accessories.
2.	Idea / Solution description	Customers directly make online shopping based on customer choice without any search.
3.	Novelty / Uniqueness	The customer will talk to Chat Bot regarding the Products. Get the recommendations based on information provided by the user
4.	Social Impact / Customer Satisfaction	The user friendly interface, Assistants form chat bot finding dress makes customer satisfied.
5.	Business Model (Revenue Model)	The chat bot sells our Products to customer. Customers buy our products and generate revenue
6.	Scalability of the Solution	We can easily scalable our Applications by increases the items and products

3.4 PROBLEM SOLUTION FIT



Define CS, fit into CC

1.CUSTOMER SEGMENT(S) CS

People with all ages can get the fashion details.

6.COUSTMER LIMITATION CC

Low Budjet, Smartphones.

5. AVAILABLE SOLUTIONS AS

Plus : Users may get all the fashion details. attributes and details.
Minus : Users cannot upload any input images.

Explore AS, differentiate

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

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

They got frustrations or fears whether the viewing of Fashion cloths are correct or not.

9. PROBLEM ROOT/ CAUSE. RC

People thinks that how the simple scan from the mobile camera can detect the human from the face camera as a input and give correct output.

7. BEHAVIOUR BE

The Users can able to upload their own fashion are correct or not. input image of a specified collection of product to know about the product details.

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

Identify strong TR & EM

3. TRIGGERS TR

The Users are got triggered by seeing that other people are getting benefits by knowing the nutrition details.

4. EMOTIONS: BEFORE / AFTER EM

BEFORE :Frustration.
AFTER :Satisfaction and Jubliant

10. YOUR SOLUTION SL

Here the New user is login with their respective details. The registered user login with their unique id. After the user login select the attributes which they want to see the nutrients in it. And the user are able to upload his/her own image and the image is automatically analysed and display the nutrients in it

8.CHANNELS of BEHAVIOUR CH

8.1 ONLINE
Extract channels from the behavior block.

8.2 OFFLINE
Extract channels from the behavior block and use for customers.

Extract online & offline CH of BE

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	User Interaction	Interact through the Chat Bot
FR-3	Buying Products	Through the chat Bot Recommendation
FR-4	Track Products	Ask the Chat Bot to Track my Orders
FR-5	Return Products	Through the chat Bot
FR_6	New Collections	Recommended from chat Bot

Non-functional Requirements:

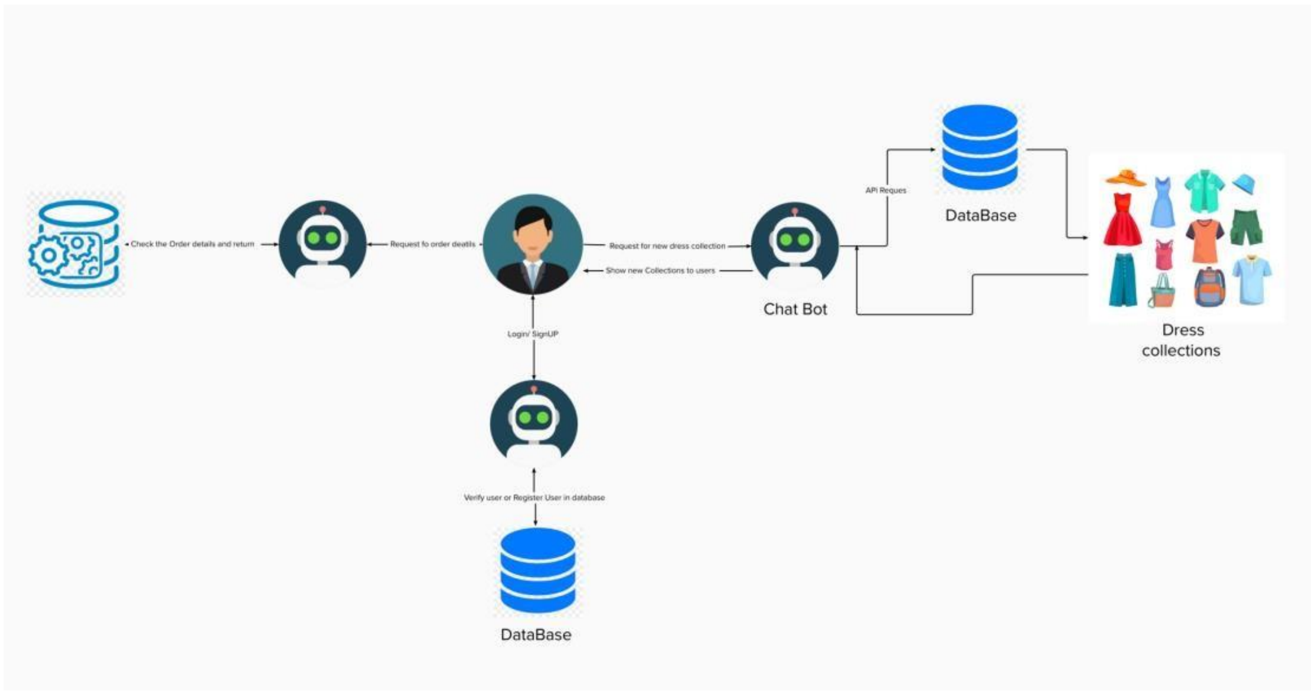
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Using Android or IOS or windows applications.
NFR-2	**Security**	The user data is stored securely in IBM cloud.
NFR-3	**Reliability**	The Quality of the services are trusted.
NFR-4	**Performance**	Its Provide smooth user experience.
NFR-5	**Availability**	The services are available for 24/7.
NFR-6	**Scalability**	Its easy to scalable size of users and products.

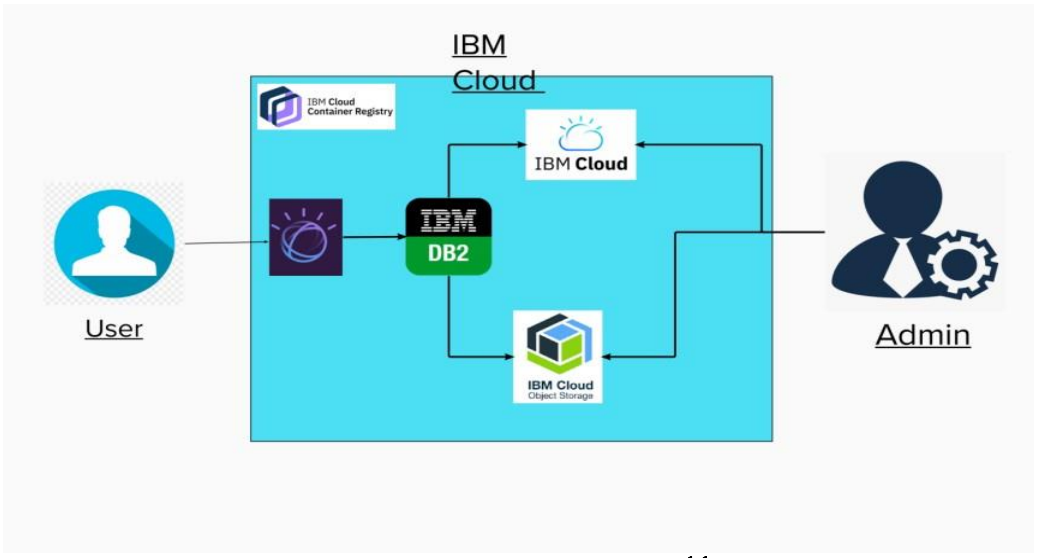
5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

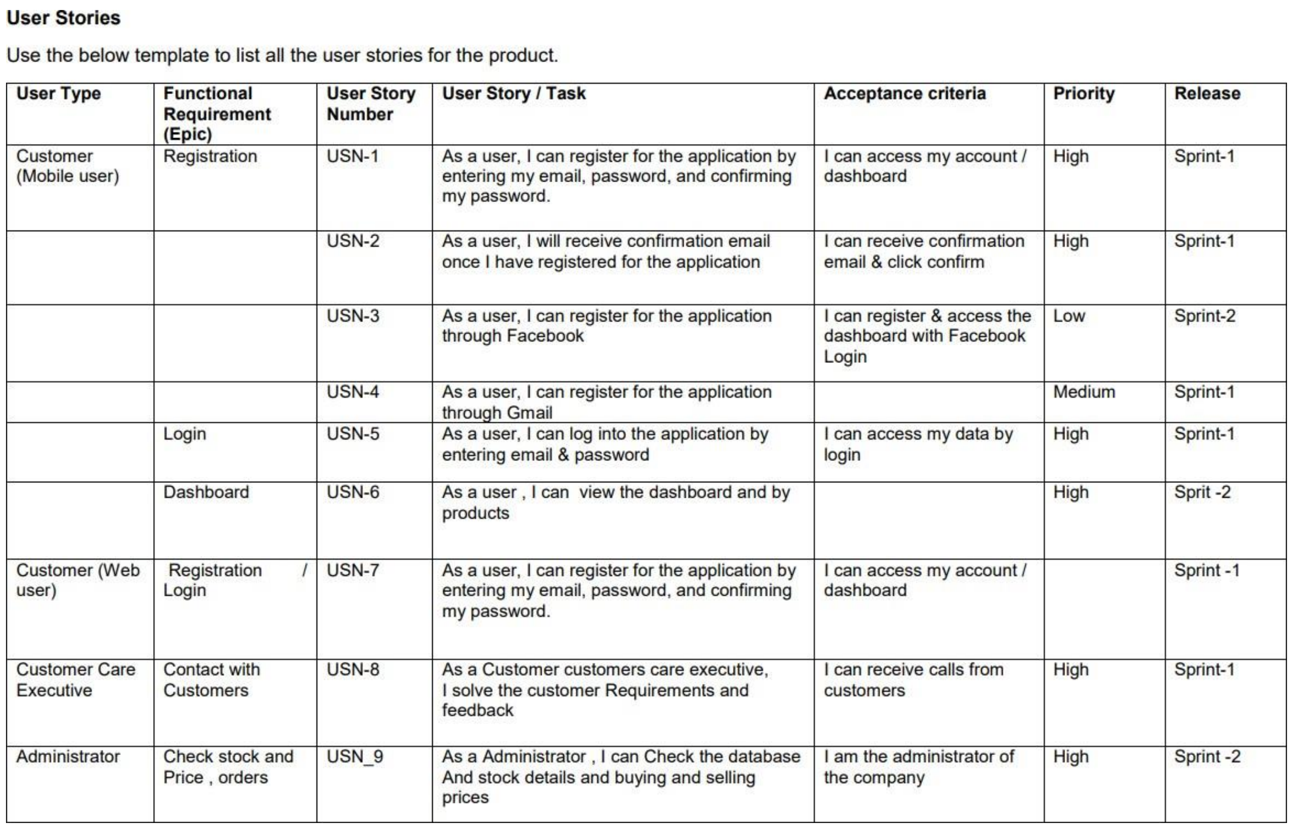
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.


Check the Order details and return
Request fo order deatils
Request for new dress collection
Show new Collections to users
Chat Bot
API Reques
DataBase
Dress collections
Login/ SignUP
Verify user or Register User in database
DataBase

5.2 SOLUTION & TECHNICAL ARCHITECTURE



5.3 USER STORIES



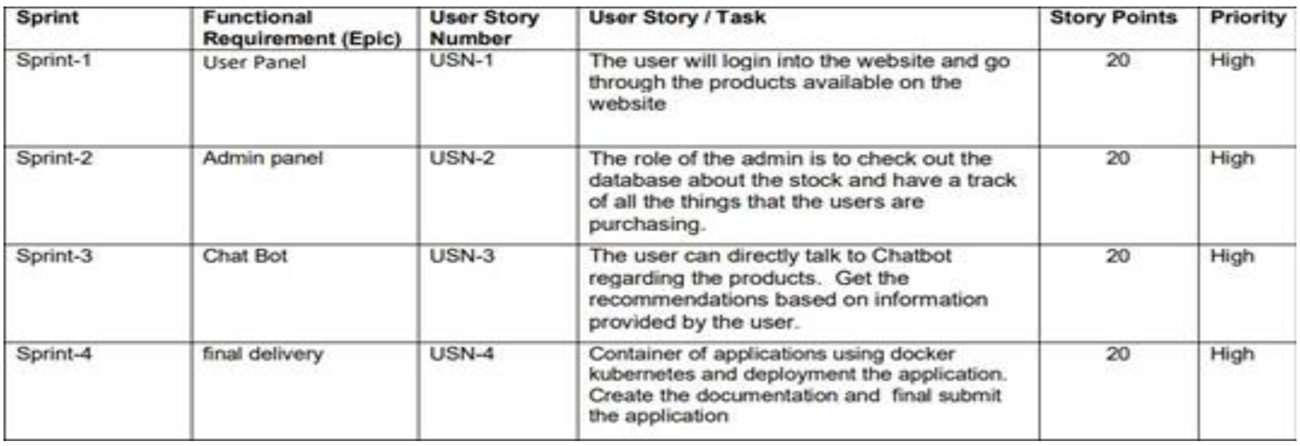
User Stories

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		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access my data by login	High	Sprint-1
	Dashboard	USN-6	As a user , I can view the dashboard and by products		High	Sprit -2
Customer (Web user)	Registration / Login	USN-7	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard		Sprint -1
Customer Care Executive	Contact with Customers	USN-8	As a Customer customers care executive, I solve the customer Requirements and feedback	I can receive calls from customers	High	Sprint-1
Administrator	Check stock and Price , orders	USN_9	As a Administrator , I can Check the database And stock details and buying and selling prices	I am the administrator of the company	High	Sprint -2

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION



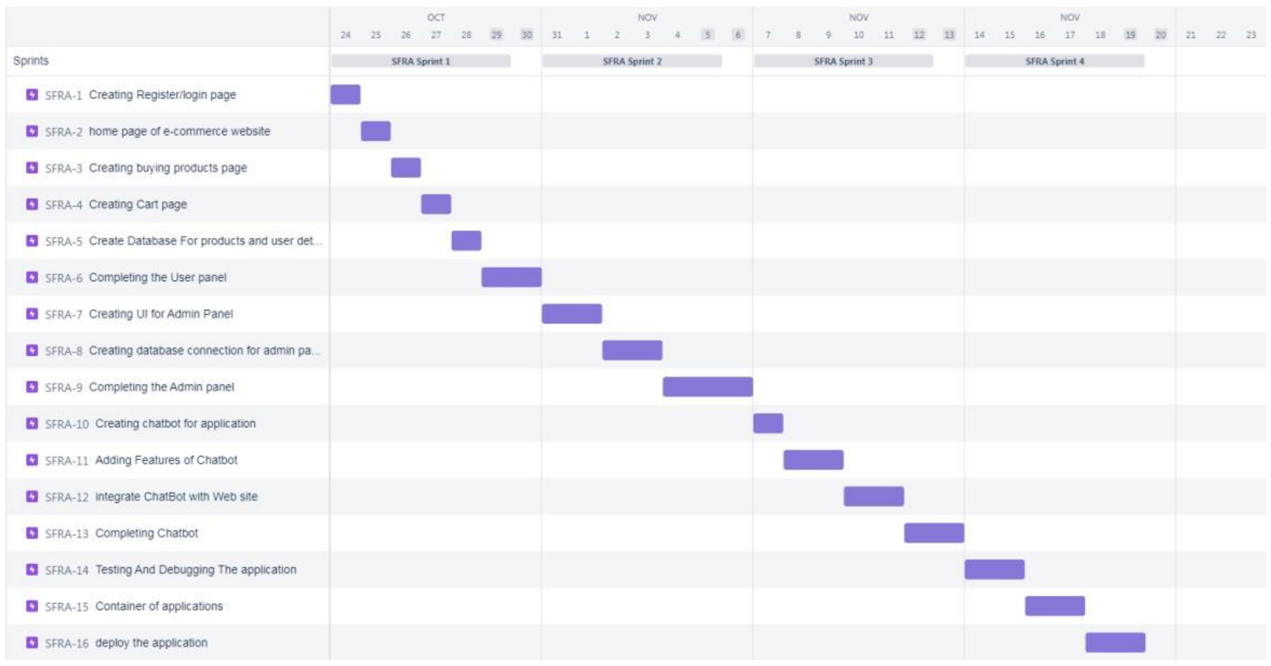
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	User Panel	USN-1	The user will login into the website and go through the products available on the website	20	High
Sprint-2	Admin panel	USN-2	The role of the admin is to check out the database about the stock and have a track of all the things that the users are purchasing.	20	High
Sprint-3	Chat Bot	USN-3	The user can directly talk to Chatbot regarding the products. Get the recommendations based on information provided by the user.	20	High
Sprint-4	final delivery	USN-4	Container of applications using docker kubernetes and deployment the application. Create the documentation and final submit the application	20	High

6.2 SPRINT DELIEVERY SCHEDULE

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	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 REPORTS FROM JIRA

Burndown Chart:



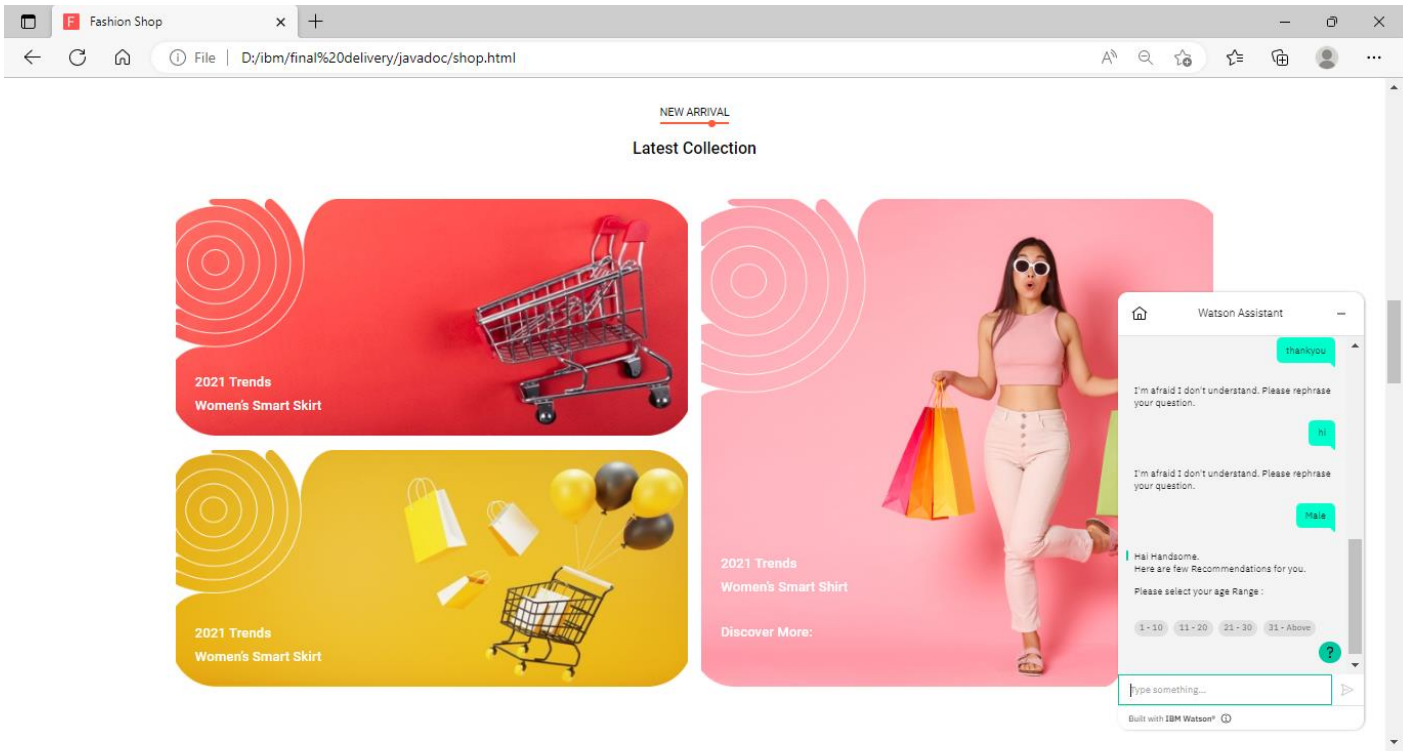
Sprints

SFRA-1 Creating Register/login page
SFRA-2 home page of e-commerce website
SFRA-3 Creating buying products page
SFRA-4 Creating Cart page
SFRA-5 Create Database For products and user det...
SFRA-6 Completing the User panel
SFRA-7 Creating UI for Admin Panel
SFRA-8 Creating database connection for admin pa...
SFRA-9 Completing the Admin panel
SFRA-10 Creating chatbot for application
SFRA-11 Adding Features of Chatbot
SFRA-12 integrate ChatBot with Web site
SFRA-13 Completing Chatbot
SFRA-14 Testing And Debugging The application
SFRA-15 Container of applications
SFRA-16 deploy the application

7.CODING & SOLUTIONING

7.1 FEATURE 1

Using chat bot we can manage user's choices and orders.



thankyou

I'm afraid I don't understand. Please rephrase your question.

hi

I'm afraid I don't understand. Please rephrase your question.

Male

Hai Handsome.
Here are few Recommendations for you.

Please select your age Range :

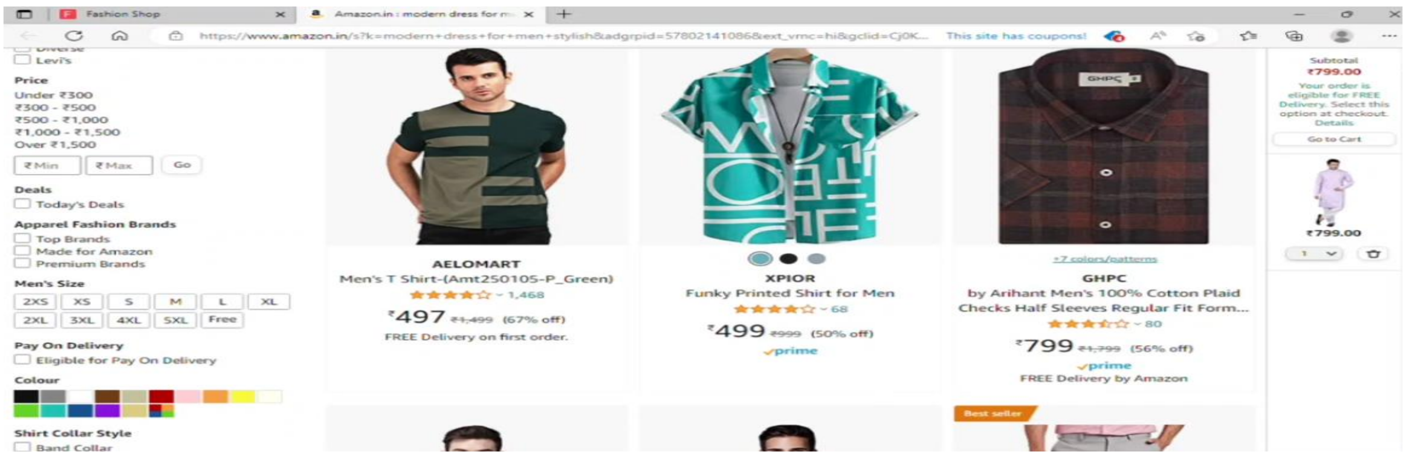
1 - 10 | 11 - 20 | 21 - 30 | 31 - Above

Type something...

Built with IBM Watson®

7.2 FEATURE 2

Chat Bot promote the best deals and offers on that day.


Fashion Shop
Amazon.in : modern dress for m...
https://www.amazon.in/s?k=modern+dress+for+men+stylish&adgrpid=57802141086&ext_vmc=hi&gclid=CjOK...
This site has coupons!
Go
Go to Cart
Best seller

Levi's

Price
Under ₹300
₹300 - ₹500
₹500 - ₹1,000
₹1,000 - ₹1,500
Over ₹1,500

₹ Min | ₹ Max

Deals
Today's Deals

Apparel Fashion Brands
Top Brands
Made for Amazon
Premium Brands

Men's Size
2XS | XS | S | M | L | XL
2XL | 3XL | 4XL | 5XL | Free

Pay On Delivery
Eligible for Pay On Delivery

Colour

Shirt Collar Style
Band Collar

AELOMART
Men's T Shirt-(Amt250105-P_Green)
1,468
₹497 ₹1,499 (67% off)
FREE Delivery on first order.

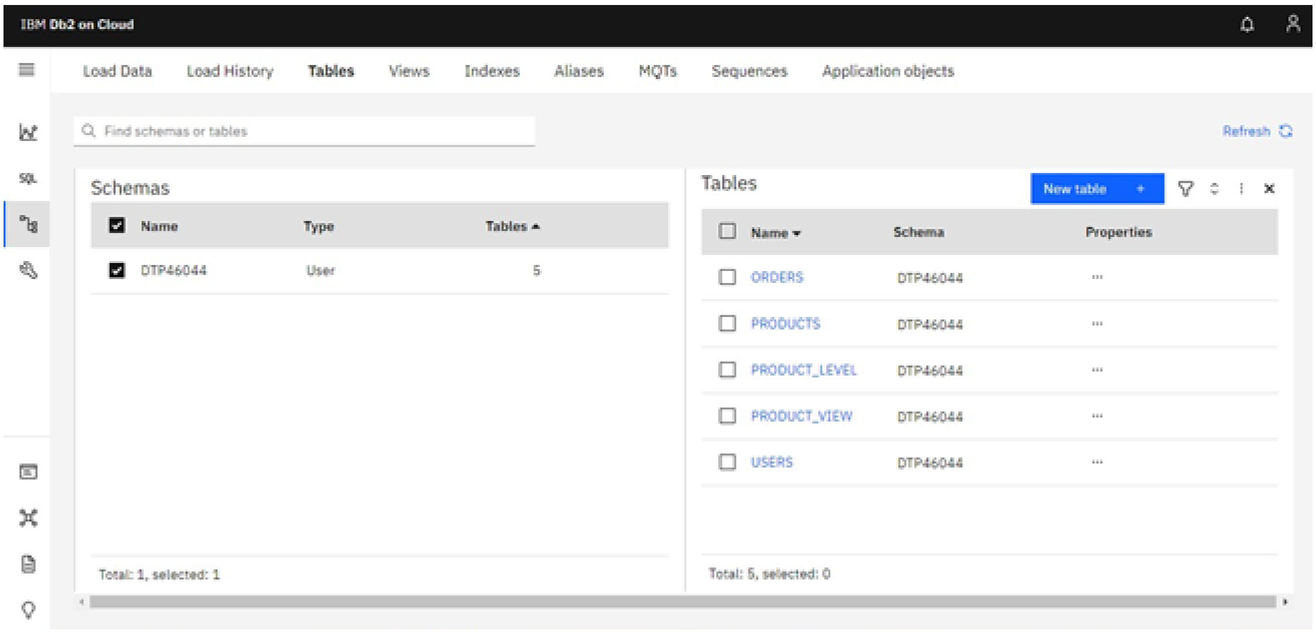
XPIOR
Funky Printed Shirt for Men
68
₹499 ₹999 (50% off)
prime

+7 colors/patterns
GHPC
by Arihant Men's 100% Cotton Plaid Checks Half Sleeves Regular Fit Form...
80
₹799 ₹1,799 (56% off)
prime
FREE Delivery by Amazon

Subtotal
₹799.00
Your order is eligible for FREE Delivery. Select this option at checkout. Details

₹799.00

7.3 DATABASE SCHEMA



Schemas

Name	Type	Tables
DTP46044	User	5

Total: 1, selected: 1

Tables

Name	Schema	Properties
ORDERS	DTP46044	...
PRODUCTS	DTP46044	...
PRODUCT_LEVEL	DTP46044	...
PRODUCT_VIEW	DTP46044	...
USERS	DTP46044	...

Total: 5, selected: 0

8.TESTING

8.1 TEST CASES

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

Section	Total Cases	Not Tested	Fail	Pass
Login	5	0	0	5
Register	7	0	0	7
Home Page	2	0	0	2
Order page	3	0	0	3
Order products	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

8.2 USER ACCEPTANCE TESTING

Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the SmartFashion Recommender Application project at the time of the release to User Acceptance Testing (UAT).

Defect Analysis

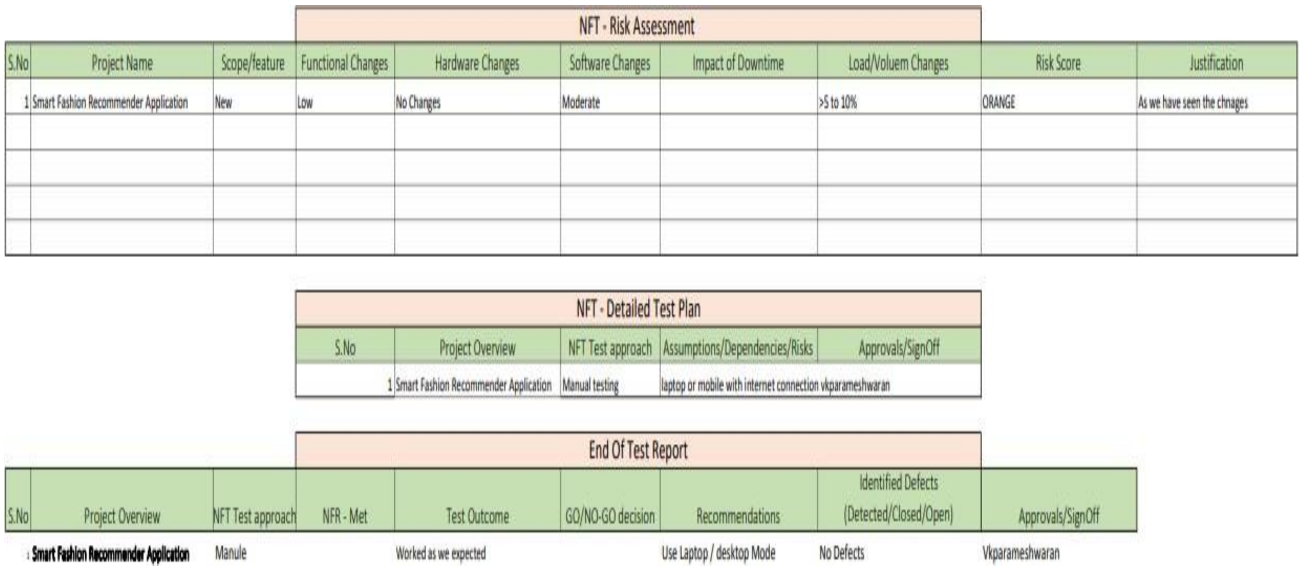
This report shows the number of resolved or closed bugs at each severity level, and how theywere resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	5	2	3	21
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't Fix	0	5	2	1	8
Totals	24	14	13	26	77

7. RESULT

9.1 PERFORMANCE METRICS

Project team shall fill the following information in model performance testing.



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

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	vkparameshwaran

			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	Manule		Worked as we expected		Use Laptop / desktop Mode	No Defects	Vkparameshwaran

8. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- Its helps to user Shopping with Assistant
- Its helps to user manage there order list
- Its helps to user shopping at home

DISADVANTAGES:

- User have fear about online shopping
- User have sometimes received wrong items
- User have fear about online payment

9. CONCLUSION

Recommendation systems have the potential to explore new opportunities for retailers by enabling them to provide customized recommendations to consumers based on informationretrieved from the Internet. They help consumers to instantly find the products and services that closely match with their choices. Moreover, different stat-of-the-art algorithms have been developed to recommend products based on users' interactions with their social groups. Therefore, research on embedding social media images within fashion recommendation systems has gained huge popularity in recent times. This paper presented a review of the fashion recommendation systems, algorithmic models and filtering techniques based on the academic articles related to this topic. The technical aspects, strengths and weaknesses of the filtering techniques have been discussed elaborately, which will help future researchers gain an in-depth understanding of fashion recommender systems. However, the proposed prototypes should be tested in commercial applications to understand their feasibility and accuracy in the retail market, because inaccurate recommendations can produce a negative impact on a customer. Moreover, future research shouldconcentrate on including time series analysis and accurate categorization of product images basedon the variation in color, trend and clothing style in order to develop an effective recommendationsystem. The proposed model will follow brand specific personalization campaigns and hence it will ensure highly curated and tailored offerings for users. Hence, this research will be highly beneficial for researchers interested in using augmented and virtual reality features to develop recommendation systems.

10. FUTURE SCOPE

There has been significant progress recently in fashion recommendation system research, which will benefit both consumers and retailers soon. The use of product and user images, textual content, demographic history, and cultural information is crucial in developing recommendation frameworks. Product attributes and clothing style matching are common features of collaborative and content-based filtering techniques. Researchers can develop more sophisticated hyper personalized filtering techniques considering the correlation between consumers' clothing styles and personalities. The methods based on employing a scoring system for quantifying each product attribute will be helpful in increasing the precision of the model. The use of virtual sales advisers in an online shopping portal would provide consumers with a real time offline shopping experience. Retailers can collect the data on users' purchase history and product reviews from the recommendation system and subsequently use them in style prediction for the upcoming seasons. The integration of different domain information strengthens the deep learning paradigm by enabling the detection of design component variation, which improves the performance of the recommendation system in the long run. Deep learning approaches should be more frequently used to quickly explore fashion items from different online databases to provide prompt recommendations to users or consumers.

13. APPENDIX

13.1 SOURCE CODE

Home code:

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
   <!-- ====== Favicon ====== -->
   <link rel="shortcut icon" href="images/favicon-32x32.png" type="image/png" />
   <!-- ====== Boxicons ====== -->
   <link href="https://unpkg.com/boxicons@2.0.9/css/boxicons.min.css" rel="stylesheet" />
   <!-- ====== Swiper CSS ====== -->
   <link rel="stylesheet" href="https://unpkg.com/swiper/swiper-bundle.min.css" />
   <!-- ====== Custom CSS ====== -->
    <link rel="stylesheet" href="css/styles.css" />
   <title>Fashion Shop</title>
</head>
<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>
<body>
   <!-- ====== Header ====== -->
   <header class="header ">
      <!-- ====== Navigation ====== -->
      <nav class="navbar ">
         <div class="row container d-flex ">
            <div class="logo ">
               <img src="./images/logo.png " alt=" " />
            </div>

            <div class="nav-list d-flex ">
                <a href=" ">Home</a>
                <a href="./shop.html ">Shop</a>
                <a href=" ">Pages</a>
                <a href=" ">About Us</a>
                <a href=" ">Lookups</a>
                <div class="close ">
                    <i class="bx bx-x "></i>
                </div>
                <a class="user-link ">Login</a>
            </div>

            <div class="icons d-flex ">
                <div class="icon d-flex "><i class="bx bx-search "></i></div>
                <div class="icon user-icon d-flex ">
                    <i class="bx bx-user "></i>
                </div>
                <div class="icon d-flex ">
                    <i class="bx bx-bell "></i>
                    <span></span>
                </div>
            </div>

            <!-- Hamburger -->
            <div class="hamburger ">
                <i class="bx bx-menu-alt-right "></i>
            </div>
        </div>
    </nav>

    <!-- ====== Hero Area ====== -->
    <div class="hero ">
        <div class="row container d-flex ">
            <div class="col ">
                <span class="subtitle ">Limited Time Only For Winter</span>
                <h1>fash<span class="i ">i</span>on</h1>
                <p>LOOK YOUR BEST ON YOUR BEST DAY</p>

                <button class="btn ">Explore Now!</button>
            </div>
            <img src="./images/woman-in-cart.png " alt=" " />
        </div>
    </div>
</header>
<!-- ====== Collection ====== -->
<section class="section collection ">
    <div class="title ">
```

```
   <span>COLLECTION</span>
   <h2>Our Top Collection</h2>
</div>
<div class="filters d-flex ">
   <div data-filter="Jewellery ">Jewellery</div>
   <div data-filter="Accessories ">Accessories</div>
   <div data-filter="Dresses ">Dresses</div>
   <div data-filter="Footwear ">Footwear</div>
</div>
<!-- ====== Login and Signup Form ====== -->
<div class="user-form ">
   <div class="close-form d-flex "><i class="bx bx-x "></i></div>
   <div class="form-wrapper container ">
      <div class="user login ">
         <div class="img-box ">
            <img src="./images/login.svg " alt=" " />
         </div>
         <div class="form-box ">
            <div class="top ">
               <p>
                  Not a member?
                  <span data-id="#ff0066 ">Register now</span>
               </p>
            </div>
            <form action=" ">
               <div class="form-control ">
                  <h2>Hello Again!</h2>
                  <p>Welcome back you've been missed.</p>
                  <input type="text " placeholder="Enter Username " />
                  <div>
                     <input type="password " placeholder="Password " />
                     <div class="icon form-icon ">
                        <!-- <img src="./images/eye.svg " alt=" " /> -->
                     </div>
                  </div>
                  <span>Recovery Password</span>
                  <button type="submit" onclick="">login</button>
               </div>
               <div class="form-control ">
                  <p>Or continue with</p>
                  <div class="icons ">
                     <div class="icon ">
                        <img src="./images/search.svg " alt=" " />
                     </div>
                     <div class="icon ">
                        <img src="./images/apple.svg " alt=" " />
                     </div>
                     <div class="icon ">
                        <img src="./images/facebook.svg " alt=" " />
```

```
                    </div>
                    <div class="icon ">
                      <img src="./images/github.svg " alt=" " />
                    </div>
                  </div>
                </div>
              </form>
            </div>
          </div>
```
Register login
```
          <!-- Register -->
          <div class="user signup ">
            <div class="form-box ">
              <div class="top ">
                <p>
                  Already a member?
                  <span data-id="#1a1aff ">Login now</span>
                </p>
              </div>
              <form action=" ">
                <div class="form-control ">
                  <h2>Welcome javadoc!</h2>
                  <p>It's good to have you.</p>
                  <input type="email " placeholder="Enter Email " />
                  <div>
                    <input type="password " placeholder="Password " />
                    <div class="icon form-icon ">
                      <img src="./images/eye.svg " alt=" " />
                    </div>
                  </div>
                  <div>
                    <input type="password " placeholder="Confirm Password " />
                    <div class="icon form-icon ">
                      <img src="./images/eye.svg " alt=" " />
                    </div>
                  </div>
                  <button type="submit" onclick="alert('successfully Register')">Register</button>
                </div>
                <div class="form-control ">
                  <p>Or continue with</p>
                  <div class="icons ">
                    <div class="icon ">
                      <img src="./images/search.svg " alt=" " />
                    </div>
                    <div class="icon ">
                      <img src="./images/apple.svg " alt=" " />
                    </div>
                    <div class="icon ">
                      <img src="./images/facebook.svg " alt=" " />
                    </div>
                    <div class="icon ">
                      <img src="./images/github.svg " alt=" " />
                    </div>
                  </div>
                </div>
              </form>
            </div>
            <div class="img-box ">
              <img src="./images/trial.svg " alt=" " />
            </div>
          </div>
        </div>
      </div>
      <!-- ====== SwiperJs ====== -->
      <script src="https://unpkg.com/swiper/swiper-bundle.min.js "></script>

      <!-- ====== Custom Script ====== -->
      <script src="./js/product.js "></script>
      <script src="./js/main.js "></script>
</body>
</html>
```

Shope code

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <!-- ====== Favicon ====== -->
  <link rel="shortcut icon" href="images/favicon-32x32.png" type="image/png" />
  <!-- ====== Boxicons ====== -->
  <link href="https://unpkg.com/boxicons@2.0.9/css/boxicons.min.css" rel="stylesheet" />
  <!-- ====== Swiper CSS ====== -->
  <link rel="stylesheet" href="https://unpkg.com/swiper/swiper-bundle.min.css" />
  <!-- ====== Custom CSS ====== -->
  <link rel="stylesheet" href="css/styles.css" />
  <title>Fashion Shop</title>
</head>
<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>

<body>
   <!-- ====== Collection ====== -->
   <section class="section collection">
      <div class="title">
         <span>COLLECTION</span>
         <h2>Our Top Collection</h2>
      </div>
      <div class="filters d-flex">
         <div data-filter="Jewellery">Jewellery</div>
         <div data-filter="Accessories">Accessories</div>
         <div data-filter="Dresses">Dresses</div>
         <div data-filter="Footwear">Footwear</div>
      </div>

      <div class="products container">
         <div class="swiper mySwiper">
            <div class="swiper-wrapper" id="products">
               <div class="swiper-slide">
                  <div class="product">
                     <div class="top d-flex">
                        <img src="./images/product-1.png" alt="" />
                        <div class="icon d-flex">
                           <i class="bx bxs-heart"></i>
                        </div>
                     </div>
                     <div class="bottom">
                        <h4>Nike Air Men’s Hoodie - Imported Hoodie Red</h4>
                        <div class="d-flex">
                           <div class="price">₹550</div>
                           <div class="rating">
                              <i class="bx bxs-star"></i>
                              <i class="bx bxs-star"></i>
                              <i class="bx bxs-star"></i>
                              <i class="bx bxs-star"></i>
                              <i class="bx bxs-star"></i>
                           </div>
                        </div>
                     </div>
                  </div>
               </div>
```

```
    </div>
  </div>
  <div class="products container">
    <div class="swiper mySwiper">
      <div class="swiper-wrapper" id="products">
        <div class="swiper-slide">
          <div class="product">
            <div class="top d-flex">
              <img src="./images/product-2.png" alt="" />
              <div class="icon d-flex">
                <i class="bx bxs-heart"></i>
              </div>
            </div>
            <div class="bottom">
              <h4>skirt women's - ladies's suit blue</h4>
              <div class="d-flex">
                <div class="price">₹950</div>
                <div class="rating">
                  <i class="bx bxs-star"></i>
                  <i class="bx bxs-star"></i>
                  <i class="bx bxs-star"></i>
                  <i class="bx bxs-star"></i>
                  <i class="bx bxs-star"></i>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
    <div class="products container">
      <div class="swiper mySwiper">
        <div class="swiper-wrapper" id="products">
          <div class="swiper-slide">
            <div class="product">
              <div class="top d-flex">
                <img src="./images/product-3.png" alt="" />
                <div class="icon d-flex">
                  <i class="bx bxs-heart"></i>
                </div>
              </div>
              <div class="bottom">
                <h4>Apple i watch - silver rubber strap</h4>
                <div class="d-flex">
                  <div class="price">₹4150</div>
                  <div class="rating">
                    <i class="bx bxs-star"></i>
                    <i class="bx bxs-star"></i>
                    <i class="bx bxs-star"></i>
```

```
                      <i class="bx bxs-star"></i>
                      <i class="bx bxs-star"></i>
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
        <div class="products container">
          <div class="swiper mySwiper">
            <div class="swiper-wrapper" id="products">
              <div class="swiper-slide">
                <div class="product">
                  <div class="top d-flex">
                    <img src="./images/product-4.png" alt="" />
                    <div class="icon d-flex">
                      <i class="bx bxs-heart"></i>
                    </div>
                  </div>
                  <div class="bottom">
                    <h4>Air Men’s Cooler's- Multi colour</h4>
                    <div class="d-flex">
                      <div class="price">₹150</div>
                      <div class="rating">
                        <i class="bx bxs-star"></i>
                        <i class="bx bxs-star"></i>
                        <i class="bx bxs-star"></i>
                        <i class="bx bxs-star"></i>
                        <i class="bx bxs-star"></i>
                      </div>
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>
          <div class="pagination">
            <div class="custom-pagination"></div>
          </div>
        </div>
</section>

<!-- ====== New Arrival ====== -->
<section class="section new-arrival">
  <div class="title">
    <span>NEW ARRIVAL</span>
    <h2>Latest Collection</h2>
  </div>
  <div class="row container">
    <div class="col col-1">
      <img src="./images/poster-1.png" alt="" />
      <h3>
        2021 Trends <br /> Women’s Smart Skirt
      </h3>
    </div>
    <div class="col col-2">
      <img src="./images/poster-2.png" alt="" />
      <h3>
        2021 Trends <br /> Women’s Smart Skirt
      </h3>
    </div>
    <div class="col col-3">
      <img src="./images/poster-3.png" alt="" />
      <h3>
        2021 Trends <br /> Women’s Smart Shirt <br />
        <span>Discover More:</span>
      </h3>
    </div>
  </div>
</section>

<!-- ====== Categories ====== -->
<section class="section categories">
  <div class="title">
    <span>CATEGORIES</span>
    <h2>2021 Latest Collection</h2>
  </div>

  <div class="products container">
    <div class="product">
      <div class="top d-flex">
        <img src="./images/product-5.png" alt="" />
        <div class="icon d-flex">
          <i class="bx bxs-heart"></i>
        </div>
      </div>
      <div class="bottom">
        <div class="d-flex">
          <h4>Printed Shirt - Cotten Material</h4>
          <a href="" class="btn cart-btn">Add to Cart</a>
        </div>
        <div class="d-flex">
          <div class="price">₹750</div>
          <div class="rating">
            <i class="bx bxs-star"></i>
            <i class="bx bxs-star"></i>
```

```
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
            </div>
          </div>
        </div>
      </div>
    </div>
    <div class="products container">
      <div class="product">
        <div class="top d-flex">
          <img src="./images/product-6.png" alt="" />
          <div class="icon d-flex">
            <i class="bx bxs-heart"></i>
          </div>
        </div>
        <div class="bottom">
          <div class="d-flex">
            <h4>Sony Headphones - for Gameing</h4>
            <a href="" class="btn cart-btn">Add to Cart</a>
          </div>
          <div class="d-flex">
            <div class="price">₹5450</div>
            <div class="rating">
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
            </div>
          </div>
        </div>
      </div>
    </div>
    <div class="products container">
      <div class="product">
        <div class="top d-flex">
          <img src="./images/product-7.png" alt="" />
          <div class="icon d-flex">
            <i class="bx bxs-heart"></i>
          </div>
        </div>
        <div class="bottom">
          <div class="d-flex">
            <h4>Handmade Bags - for Office </h4>
            <a href="" class="btn cart-btn">Add to Cart</a>
          </div>
          <div class="d-flex">
            <div class="price">₹2150</div>
            <div class="rating">
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
            </div>
          </div>
        </div>
      </div>
    </div>
    <div class="products container">
      <div class="product">
        <div class="top d-flex">
          <img src="./images/product-8.png" alt="" />
          <div class="icon d-flex">
            <i class="bx bxs-heart"></i>
          </div>
        </div>
        <div class="bottom">
          <div class="d-flex">
            <h4>Football Boots - with sharp needles</h4>
            <a href="" class="btn cart-btn">Add to Cart</a>
          </div>
          <div class="d-flex">
            <div class="price">₹950</div>
            <div class="rating">
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
            </div>
          </div>
        </div>
      </div>
    </div>
    <div class="products container">
      <div class="product">
        <div class="top d-flex">
          <img src="./images/product-9.png" alt="" />
          <div class="icon d-flex">
            <i class="bx bxs-heart"></i>
          </div>
        </div>
        <div class="bottom">
          <div class="d-flex">
            <h4>Modern Sofas heals - Merun Red</h4>
            <a href="" class="btn cart-btn">Add to Cart</a>
          </div>
          <div class="d-flex">
            <div class="price">₹1150</div>
            <div class="rating">
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
            </div>
          </div>
        </div>
      </div>
    </div>
    <div class="products container">
      <div class="product">
        <div class="top d-flex">
          <img src="./images/product-10.png" alt="" />
          <div class="icon d-flex">
            <i class="bx bxs-heart"></i>
          </div>
        </div>
        <div class="bottom">
          <div class="d-flex">
            <h4>logitech - Gameing mouse </h4>
            <a href="" class="btn cart-btn">Add to Cart</a>
          </div>
          <div class="d-flex">
            <div class="price">₹1150</div>
            <div class="rating">
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
              <i class="bx bxs-star"></i>
            </div>
          </div>
        </div>
      </div>
    </div>
    <div class="products container">
      <div class="product">
        <div class="top d-flex">
          <img src="./images/product-11.png" alt="" />
          <div class="icon d-flex">
            <i class="bx bxs-heart"></i>
          </div>
        </div>
        <div class="bottom">
```

```
            <div class="d-flex">
              <h4>Realme Airpods - hQ 001(version-5)</h4>
              <a href="" class="btn cart-btn">Add to Cart</a>
            </div>
            <div class="d-flex">
              <div class="price">₹1150</div>
              <div class="rating">
                <i class="bx bxs-star"></i>
                <i class="bx bxs-star"></i>
                <i class="bx bxs-star"></i>
                <i class="bx bxs-star"></i>
                <i class="bx bxs-star"></i>
              </div>
            </div>
          </div>
        </div>
      </div>
      <div class="products container">
        <div class="product">
          <div class="top d-flex">
            <img src="./images/product-12.png" alt="" />
            <div class="icon d-flex">
              <i class="bx bxs-heart"></i>
            </div>
          </div>
          <div class="bottom">
            <div class="d-flex">
              <h4>Samsaung smart watch - Imported Watch blue</h4>
              <a href="" class="btn cart-btn">Add to Cart</a>
            </div>
            <div class="d-flex">
              <div class="price">₹1480</div>
              <div class="rating">
                <i class="bx bxs-star"></i>
                <i class="bx bxs-star"></i>
                <i class="bx bxs-star"></i>
                <i class="bx bxs-star"></i>
                <i class="bx bxs-star"></i>
              </div>
            </div>
          </div>
        </div>
      </div>

      <div class="button d-flex">
        <a class="btn loadmore" link="load"
href="https://www.amazon.in/s?k=modern+dress+for+men+stylish&adgrpid=57802141086&ext_vrnc=hi&gclid=C
j0KCQjwk5ibBhDqARIsACzmgLTiBWaJOREQSvwn2pz6XFw5CqXLzAVTgFeCu3PADM95KJHG_JEp4aoaAroiEALw_wc
B&hvadid=380037112028&hvdev=c&hvlocphy=1007811&hvnetw=g&hvqmt=b&hvrand=6622843879145823838
```

```
&hvtargid=kwd-934297070151&hydadcr=23434_1935760&tag=googinhydr1-21&ref=pd_sl_4y92ytkove_b">Load
More</a>
      </div>
   </section>
   </header>
</body>

</html>
```

FLUSK CODE:

```
from flask import Flask, render_template, flash, redirect, url_for, session, request, logging
from sqlalchemy import sqlalchemy
from wtforms import Form, StringField, TextAreaField, PasswordField, validators,
SelectField
from passlib.hash import sha256_crypt
from functools import wraps
from flask_uploads import UploadSet, configure_uploads, IMAGES
import timeit
import datetime
from flask_mail import Mail, Message
import os
from wtforms import EmailField
import sendgrid
import os



db = sqlalchemy.create_engine('ibm_db_sa://#/bludb')
bcrypt =Bcrypt(app)
metadata = MetaData()

app = Flask(__name__)
app.secret_key = os.urandom(24)
app.config['UPLOADED_PHOTOS_DEST'] = 'static/image/product'
photos = UploadSet('photos', IMAGES)
configure_uploads(app, photos)



db.init_app(app)

def sendemail():
    sg = sendgrid.SendGridAPIClient(api_key=os.environ.get('SENDGRID_API_KEY'))
    data = {
```

```
    "personalizations": [
        {
        "to": [
            {
            "email": "hari123@gmail.com"
            }
        ],
        "subject": "javadoc.Thankyou for purchasing"
        }
    ],
    "from": {
        "email": "javadocshopping@gmail.com"
    },
    "content": [
        {
        "type": "text/plain",
        "value": "Oder successfull your order arive within 5 working days"
        }
    ]
    }
    response = sg.client.mail.send.post(request_body=data)
    print(response.status_code)
    print(response.body)
    print(response.headers)


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

    return wrap


def not_logged_in(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if 'logged_in' in session:
            return redirect(url_for('index'))
        else:
            return f(*args, *kwargs)

    return wrap


def is_admin_logged_in(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if 'admin_logged_in' in session:
            return f(*args, *kwargs)
        else:
            return redirect(url_for('admin_login'))

    return wrap


def not_admin_logged_in(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if 'admin_logged_in' in session:
            return redirect(url_for('admin'))
        else:
            return f(*args, *kwargs)

    return wrap


def wrappers(func, *args, **kwargs):
    def wrapped():
        return func(*args, **kwargs)

    return wrapped


def content_based_filtering(product_id):
    cur = db.connection.cursor()
    cur.execute("SELECT * FROM products WHERE id=%s", (product_id,))  # getting id
row
    data = cur.fetchone()  # get row info
    data_cat = data['category']  # get id category ex shirt
    print('Showing result for Product Id: ' + product_id)
    category_matched = cur.execute("SELECT * FROM products WHERE category=%s",
(data_cat,))  # get all shirt category
    print('Total product matched: ' + str(category_matched))
    cat_product = cur.fetchall()  # get all row
    cur.execute("SELECT * FROM product_level WHERE product_id=%s", (product_id,))
# id level info
    id_level = cur.fetchone()
    recommend_id = []
    cate_level = ['v_shape', 'polo', 'clean_text', 'design', 'leather', 'color', 'formal', 'converse',
'loafer', 'hook',
                  'chain']
    for product_f in cat_product:
        cur.execute("SELECT * FROM product_level WHERE product_id=%s",
(product_f['id'],))
        f_level = cur.fetchone()
        match_score = 0
        if f_level['product_id'] != int(product_id):
            for cat_level in cate_level:
                if f_level[cat_level] == id_level[cat_level]:
                    match_score += 1
            if match_score == 11:
                recommend_id.append(f_level['product_id'])
    print('Total recommendation found: ' + str(recommend_id))
    if recommend_id:
        cur = db.connection.cursor()
        placeholders = ','.join((str(n) for n in recommend_id))
        query = 'SELECT * FROM products WHERE id IN (%s)' % placeholders
        cur.execute(query)
        recommend_list = cur.fetchall()
        return recommend_list, recommend_id, category_matched, product_id
    else:
        return ''


@app.route('/')
@is_logged_in
def index():
    form = OrderForm(request.form)
    # Create cursor
    cur = db.connection.cursor()
    # Get message
    values = 'tshirt'
    cur.execute("SELECT * FROM products WHERE category=%s ORDER BY RAND() 
LIMIT 4", (values,))
    tshirt = cur.fetchall()
    values = 'wallet'
    cur.execute("SELECT * FROM products WHERE category=%s ORDER BY RAND() 
LIMIT 4", (values,))
    wallet = cur.fetchall()
    values = 'belt'
    cur.execute("SELECT * FROM products WHERE category=%s ORDER BY RAND() 
LIMIT 4", (values,))
    belt = cur.fetchall()
    values = 'shoes'
    cur.execute("SELECT * FROM products WHERE category=%s ORDER BY RAND() 
LIMIT 4", (values,))
    shoes = cur.fetchall()
    # Close Connection
    cur.close()
    return render_template('home.html', tshirt=tshirt, wallet=wallet, belt=belt, shoes=shoes, 
form=form)


class LoginForm(Form):  # Create Login Form
    username = StringField('', [validators.length(min=1)],
                           render_kw={'autofocus': True, 'placeholder': 'Username'})
    password = PasswordField('', [validators.length(min=3)],
                             render_kw={'placeholder': 'Password'})


# User Login
@app.route('/login', methods=['GET', 'POST'])
@not_logged_in
def login():
    form = LoginForm(request.form)
    if request.method == 'POST' and form.validate():
        # GEt user form
        username = form.username.data
        # password_candidate = request.form['password']
        password_candidate = form.password.data
        # Create cursor
        cur = db.connection.cursor()

        # Get user by username
        result = cur.execute("SELECT * FROM users WHERE username=%s", [username])

        if result > 0:
            # Get stored value
            data = cur.fetchone()
            password = data['password']
            uid = data['id']
            name = data['name']

            # Compare password
            if sha256_crypt.verify(password_candidate, password):
                # passed
                session['logged_in'] = True
                session['uid'] = uid
                session['s_name'] = name
                x = '1'
                cur.execute("UPDATE users SET online=%s WHERE id=%s", (x, uid))

                return redirect(url_for('index'))

            else:
                flash('Incorrect password', 'danger')
                return render_template('login.html', form=form)

        else:
            flash('Username not found', 'danger')
            # Close connection
            cur.close()
            return render_template('login.html', form=form)
    return render_template('login.html', form=form)


@app.route('/out')
def logout():
    if 'uid' in session:
        # Create cursor
        cur = db.connection.cursor()
        uid = session['uid']
        x = '0'
        cur.execute("UPDATE users SET online=%s WHERE id=%s", (x, uid))
        session.clear()
        flash('You are logged out', 'success')
        return redirect(url_for('index'))
    return redirect(url_for('login'))


class RegisterForm(Form):
    name = StringField('', [validators.length(min=3, max=50)],
                       render_kw={'autofocus': True, 'placeholder': 'Full Name'})
    username = StringField('', [validators.length(min=3, max=25)],
render_kw={'placeholder': 'Username'})
    email = EmailField('', [validators.DataRequired(), validators.Email(),
validators.length(min=4, max=25)],
                       render_kw={'placeholder': 'Email'})
    password = PasswordField('', [validators.length(min=3)],
                             render_kw={'placeholder': 'Password'})
    mobile = StringField('', [validators.length(min=11, max=15)],
render_kw={'placeholder': 'Mobile'})


@app.route('/register', methods=['GET', 'POST'])
@not_logged_in
def register():
    form = RegisterForm(request.form)
    if request.method == 'POST' and form.validate():
        name = form.name.data
        email = form.email.data
        username = form.username.data
        password = sha256_crypt.encrypt(str(form.password.data))
        mobile = form.mobile.data
        # insert_sql = "INSERT INTO  users VALUES (?, ?, ?)"
        # prep_stmt = ibm_db.prepare(conn, insert_sql)
        # ibm_db.bind_param(prep_stmt, 1, username)
        # ibm_db.bind_param(prep_stmt, 2, email)
        # ibm_db.bind_param(prep_stmt, 3, password)
        # ibm_db.execute(prep_stmt)
        # Create Cursor
        cur = db.connection.cursor()
        cur.execute("INSERT INTO users(name, email, username, password, mobile)
VALUES(%s, %s, %s, %s, %s)",
                    (name, email, username, password, mobile))

        # Commit cursor
        db.connection.commit()

        # Close Connection
        cur.close()

        flash('You are now registered and can login', 'success')

        return redirect(url_for('index'))
    return render_template('register.html', form=form)


class MessageForm(Form):  # Create Message Form
    body = StringField('', [validators.length(min=1)], render_kw={'autofocus': True})


@app.route('/chatting/<string:id>', methods=['GET', 'POST'])
def chatting(id):
    if 'uid' in session:
        form = MessageForm(request.form)
        # Create cursor
        cur = db.connection.cursor()

        # lid name
        get_result = cur.execute("SELECT * FROM users WHERE id=%s", [id])
        l_data = cur.fetchone()
        if get_result > 0:
            session['name'] = l_data['name']
            uid = session['uid']
            session['lid'] = id

            if request.method == 'POST' and form.validate():
                txt_body = form.body.data
                # Create cursor
                cur = db.connection.cursor()
                cur.execute("INSERT INTO messages(body, msg_by, msg_to) VALUES(%s,
%s, %s)",
                            (txt_body, id, uid))
                # Commit cursor
                db.connection.commit()

            # Get users
            cur.execute("SELECT * FROM users")
            users = cur.fetchall()

            # Close Connection
            cur.close()
            return render_template('chat_room.html', users=users, form=form)
        else:
            flash('No permission!', 'danger')
            return redirect(url_for('index'))
    else:
        return redirect(url_for('login'))


@app.route('/chats', methods=['GET', 'POST'])
def chats():
    if 'lid' in session:
        id = session['lid']
        uid = session['uid']
        # Create cursor
        cur = db.connection.cursor()
        # Get message
        cur.execute("SELECT * FROM messages WHERE (msg_by=%s AND msg_to=%s)
OR (msg_by=%s AND msg_to=%s) "
                    "ORDER BY id ASC", (id, uid, uid, id))
        chats = cur.fetchall()
        # Close Connection
        cur.close()
        return render_template('chats.html', chats=chats, )
    return redirect(url_for('login'))


class OrderForm(Form):  # Create Order Form
    name = StringField('', [validators.length(min=1), validators.DataRequired()],
                       render_kw={'autofocus': True, 'placeholder': 'Full Name'})
    mobile_num = StringField('', [validators.length(min=1), validators.DataRequired()],
                             render_kw={'autofocus': True, 'placeholder': 'Mobile'})
    quantity = SelectField('', [validators.DataRequired()],
                           choices=[('1', '1'), ('2', '2'), ('3', '3'), ('4', '4'), ('5', '5')])
    order_place = StringField('', [validators.length(min=1), validators.DataRequired()],
                              render_kw={'placeholder': 'Order Place'})


@app.route('/tshirt', methods=['GET', 'POST'])
def tshirt():
    form = OrderForm(request.form)
    # Create cursor
    cur = db.connection.cursor()
    # Get message
    values = 'tshirt'
    cur.execute("SELECT * FROM products WHERE category=%s ORDER BY id ASC",
(values,))
    products = cur.fetchall()
    # Close Connection
    cur.close()
    if request.method == 'POST' and form.validate():
        name = form.name.data
        mobile = form.mobile_num.data
        order_place = form.order_place.data
        quantity = form.quantity.data
        pid = request.args['order']
        now = datetime.datetime.now()
        week = datetime.timedelta(days=7)
        delivery_date = now + week
        now_time = delivery_date.strftime("%y-%m-%d %H:%M:%S")
        # Create Cursor
        curs = db.connection.cursor()
        if 'uid' in session:
            uid = session['uid']
            curs.execute("INSERT INTO orders(uid, pid, ofname, mobile, oplace, quantity,
ddate) "
                         "VALUES(%s, %s, %s, %s, %s, %s, %s)",
                         (uid, pid, name, mobile, order_place, quantity, now_time))
        else:
            curs.execute("INSERT INTO orders(pid, ofname, mobile, oplace, quantity, ddate)
"
```

```
                    "VALUES(%s, %s, %s, %s, %s, %s)",
                    (pid, name, mobile, order_place, quantity, now_time))
        # Commit cursor
        db.connection.commit()

        # Close Connection
        cur.close()
        sendemail()

        flash('Order successful', 'success')
        return render_template('tshirt.html', tshirt=products, form=form)
    if 'view' in request.args:
        product_id = request.args['view']
        curso = db.connection.cursor()
        curso.execute("SELECT * FROM products WHERE id=%s", (product_id,))
        product = curso.fetchall()
        x = content_based_filtering(product_id)
        wrappered = wrappers(content_based_filtering, product_id)
        execution_time = timeit.timeit(wrappered, number=0)
        # print('Execution time: ' + str(execution_time) + ' usec')
        if 'uid' in session:
            uid = session['uid']
            # Create cursor
            cur = db.connection.cursor()
            cur.execute("SELECT * FROM product_view WHERE user_id=%s AND
product_id=%s", (uid, product_id))
            result = cur.fetchall()
            if result:
                now = datetime.datetime.now()
                now_time = now.strftime("%y-%m-%d %H:%M:%S")
                cur.execute("UPDATE product_view SET date=%s WHERE user_id=%s AND
product_id=%s",
                            (now_time, uid, product_id))
            else:
                cur.execute("INSERT INTO product_view(user_id, product_id) VALUES(%s,
%s)", (uid, product_id))
                db.connection.commit()
        return render_template('view_product.html', x=x, tshirts=product)
    elif 'order' in request.args:
        product_id = request.args['order']
        curso = db.connection.cursor()
        curso.execute("SELECT * FROM products WHERE id=%s", (product_id,))
        product = curso.fetchall()
        x = content_based_filtering(product_id)
        return render_template('order_product.html', x=x, tshirts=product, form=form)
    return render_template('tshirt.html', tshirt=products, form=form)


@app.route('/wallet', methods=['GET', 'POST'])
def wallet():
    form = OrderForm(request.form)
    # Create cursor
    cur = db.connection.cursor()
    # Get message
    values = 'wallet'
    cur.execute("SELECT * FROM products WHERE category=%s ORDER BY id ASC",
(values,))
    products = cur.fetchall()
    # Close Connection
    cur.close()

    if request.method == 'POST' and form.validate():
        name = form.name.data
        mobile = form.mobile_num.data
        order_place = form.order_place.data
        quantity = form.quantity.data
        pid = request.args['order']

        now = datetime.datetime.now()
        week = datetime.timedelta(days=7)
        delivery_date = now + week
        now_time = delivery_date.strftime("%y-%m-%d %H:%M:%S")
        # Create Cursor
        curs = db.connection.cursor()
        if 'uid' in session:
            uid = session['uid']
            curs.execute("INSERT INTO orders(uid, pid, ofname, mobile, oplace, quantity,
ddate) "
                         "VALUES(%s, %s, %s, %s, %s, %s, %s)",
                         (uid, pid, name, mobile, order_place, quantity, now_time))
        else:
            curs.execute("INSERT INTO orders(pid, ofname, mobile, oplace, quantity, ddate)
"
                    "VALUES(%s, %s, %s, %s, %s, %s)",
                    (pid, name, mobile, order_place, quantity, now_time))
        # Commit cursor
        db.connection.commit()
        # Close Connection

        cur.close()
        # sendemail()
        sendemail()

        flash('Order successful', 'success')
        return render_template('wallet.html', wallet=products, form=form)
    if 'view' in request.args:
        q = request.args['view']
        product_id = q
        x = content_based_filtering(product_id)
        curso = db.connection.cursor()
        curso.execute("SELECT * FROM products WHERE id=%s", (q,))
        products = curso.fetchall()
        return render_template('view_product.html', x=x, tshirts=products)
    elif 'order' in request.args:
        product_id = request.args['order']
        curso = db.connection.cursor()
        curso.execute("SELECT * FROM products WHERE id=%s", (product_id,))
        product = curso.fetchall()
        x = content_based_filtering(product_id)
        return render_template('order_product.html', x=x, tshirts=product, form=form)
    return render_template('wallet.html', wallet=products, form=form)


@app.route('/belt', methods=['GET', 'POST'])
def belt():
    form = OrderForm(request.form)
    # Create cursor
    cur = db.connection.cursor()
    # Get message
    values = 'belt'
    cur.execute("SELECT * FROM products WHERE category=%s ORDER BY id ASC",
(values,))
    products = cur.fetchall()
    # Close Connection
    cur.close()

    if request.method == 'POST' and form.validate():
        name = form.name.data
        mobile = form.mobile_num.data
        order_place = form.order_place.data
        quantity = form.quantity.data
        pid = request.args['order']
        now = datetime.datetime.now()
        week = datetime.timedelta(days=7)
        delivery_date = now + week
        now_time = delivery_date.strftime("%y-%m-%d %H:%M:%S")
        # Create Cursor
        curs = db.connection.cursor()
        if 'uid' in session:
            uid = session['uid']
            curs.execute("INSERT INTO orders(uid, pid, ofname, mobile, oplace, quantity,
ddate) "
                         "VALUES(%s, %s, %s, %s, %s, %s, %s)",
                         (uid, pid, name, mobile, order_place, quantity, now_time))
        else:
            curs.execute("INSERT INTO orders(pid, ofname, mobile, oplace, quantity, ddate)
"
                         "VALUES(%s, %s, %s, %s, %s, %s)",
                         (pid, name, mobile, order_place, quantity, now_time))

        # Commit cursor
        db.connection.commit()

        # Close Connection
        cur.close()
        sendemail()
        flash('Order successful', 'success')
        return render_template('belt.html', belt=products, form=form)
    if 'view' in request.args:
        q = request.args['view']
        product_id = q
        x = content_based_filtering(product_id)
        curso = db.connection.cursor()
        curso.execute("SELECT * FROM products WHERE id=%s", (q,))
        products = curso.fetchall()
        return render_template('view_product.html', x=x, tshirts=products)
    elif 'order' in request.args:
        product_id = request.args['order']
        curso = db.connection.cursor()
        curso.execute("SELECT * FROM products WHERE id=%s", (product_id,))
        product = curso.fetchall()
        x = content_based_filtering(product_id)
        return render_template('order_product.html', x=x, tshirts=product, form=form)
    return render_template('belt.html', belt=products, form=form)


@app.route('/shoes', methods=['GET', 'POST'])
def shoes():
    form = OrderForm(request.form)
    # Create cursor
    cur = db.connection.cursor()
    # Get message
    values = 'shoes'
    cur.execute("SELECT * FROM products WHERE category=%s ORDER BY id ASC",
(values,))
    products = cur.fetchall()
    # Close Connection
    cur.close()

    if request.method == 'POST' and form.validate():
        name = form.name.data
        mobile = form.mobile_num.data
        order_place = form.order_place.data
        quantity = form.quantity.data
        pid = request.args['order']
        now = datetime.datetime.now()
        week = datetime.timedelta(days=7)
        delivery_date = now + week
        now_time = delivery_date.strftime("%y-%m-%d %H:%M:%S")
        # Create Cursor
        curs = db.connection.cursor()
        if 'uid' in session:
            uid = session['uid']
            curs.execute("INSERT INTO orders(uid, pid, ofname, mobile, oplace, quantity,
ddate) "
```

```
                    "VALUES(%s, %s, %s, %s, %s, %s, %s)",
                    (uid, pid, name, mobile, order_place, quantity, now_time))
        else:
            curs.execute("INSERT INTO orders(pid, ofname, mobile, oplace, quantity, ddate)
"
                    "VALUES(%s, %s, %s, %s, %s, %s)",
                    (pid, name, mobile, order_place, quantity, now_time))
        # Commit cursor
        db.connection.commit()
        # Close Connection
        cur.close()
        sendemail()

        flash('Order successful', 'success')
        return render_template('shoes.html', shoes=products, form=form)
    if 'view' in request.args:
        q = request.args['view']
        product_id = q
        x = content_based_filtering(product_id)
        curso = db.connection.cursor()
        curso.execute("SELECT * FROM products WHERE id=%s", (q,))
        products = curso.fetchall()
        return render_template('view_product.html', x=x, tshirts=products)
    elif 'order' in request.args:
        product_id = request.args['order']
        curso = db.connection.cursor()
        curso.execute("SELECT * FROM products WHERE id=%s", (product_id,))
        product = curso.fetchall()
        x = content_based_filtering(product_id)
        return render_template('order_product.html', x=x, tshirts=product, form=form)
    return render_template('shoes.html', shoes=products, form=form)


@app.route('/admin_login', methods=['GET', 'POST'])
@not_admin_logged_in
def admin_login():
    if request.method == 'POST':
        # GEt user form
        username = request.form['email']
        password_candidate = request.form['password']
        # Create cursor
        cur = db.connection.cursor()

        # Get user by username
        result = cur.execute("SELECT * FROM admin WHERE email=%s", [username])

        if result > 0:
            # Get stored value
            data = cur.fetchone()
            password = data['password']
            uid = data['id']
            name = data['firstName']

            # Compare password
            if (password):
                # passed
                session['admin_logged_in'] = True
                session['admin_uid'] = uid
                session['admin_name'] = name

                return redirect(url_for('admin'))

            else:
                flash('Incorrect password', 'danger')
                return render_template('pages/login.html')

        else:
            flash('Username not found', 'danger')
            # Close connection
            cur.close()
            return render_template('pages/login.html')
    return render_template('pages/login.html')


@app.route('/admin_out')
def admin_logout():
    if 'admin_logged_in' in session:
        session.clear()
        return redirect(url_for('admin_login'))
    return redirect(url_for('admin'))
```

```
@app.route('/admin')
@is_admin_logged_in
def admin():
    curso = db.connection.cursor()
    num_rows = curso.execute("SELECT * FROM products")
    result = curso.fetchall()
    order_rows = curso.execute("SELECT * FROM orders")
    users_rows = curso.execute("SELECT * FROM users")
    return render_template('pages/index.html', result=result, row=num_rows,
order_rows=order_rows,
                           users_rows=users_rows)


@app.route('/orders')
@is_admin_logged_in
def orders():
    curso = db.connection.cursor()
    num_rows = curso.execute("SELECT * FROM products")
    order_rows = curso.execute("SELECT * FROM orders")
    result = curso.fetchall()
    users_rows = curso.execute("SELECT * FROM users")
    return render_template('pages/all_orders.html', result=result, row=num_rows,
order_rows=order_rows,
                           users_rows=users_rows)


@app.route('/users')
@is_admin_logged_in
def users():
    curso = db.connection.cursor()
    num_rows = curso.execute("SELECT * FROM products")
    order_rows = curso.execute("SELECT * FROM orders")
    users_rows = curso.execute("SELECT * FROM users")
    result = curso.fetchall()
    return render_template('pages/all_users.html', result=result, row=num_rows,
order_rows=order_rows,
                           users_rows=users_rows)


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

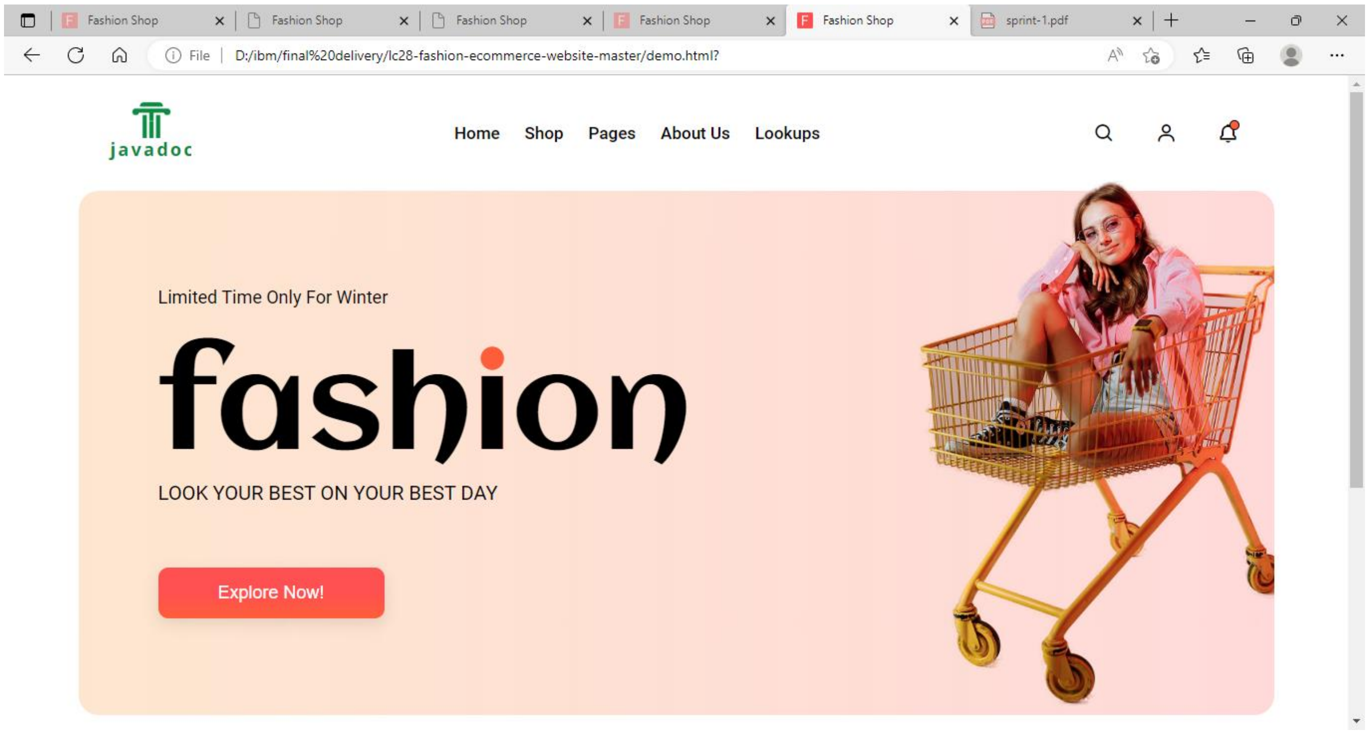
```
@is_admin_logged_in
def admin_add_product():
    if request.method == 'POST':
        name = request.form.get('name')
        price = request.form['price']
        description = request.form['description']
        available = request.form['available']
        category = request.form['category']
        item = request.form['item']
        code = request.form['code']
        file = request.files['picture']
        if name and price and description and available and category and item and code and
file:
            pic = file.filename
            photo = pic.replace("'", "")
            picture = photo.replace(" ", "_")
            if picture.lower().endswith(('.png', '.jpg', '.jpeg')):
                save_photo = photos.save(file, folder=category)
                if save_photo:
                    # Create Cursor
                    curs = db.connection.cursor()
                    curs.execute("INSERT INTO
products(pName,price,description,available,category,item,pCode,picture)"
                                 "VALUES(%s, %s, %s, %s, %s, %s, %s, %s)",
                                 (name, price, description, available, category, item, code, picture))
                    db.connection.commit()
                    product_id = curs.lastrowid
                    curs.execute("INSERT INTO product_level(product_id)" "VALUES(%s)",
[product_id])
                    if category == 'tshirt':
                        level = request.form.getlist('tshirt')
                        for lev in level:
                            yes = 'yes'
                            query = 'UPDATE product_level SET {field}=%s WHERE
product_id=%s'.format(field=lev)
                            curs.execute(query, (yes, product_id))
                            # Commit cursor
                            db.connection.commit()
                    elif category == 'wallet':
                        level = request.form.getlist('wallet')
                        for lev in level:
```

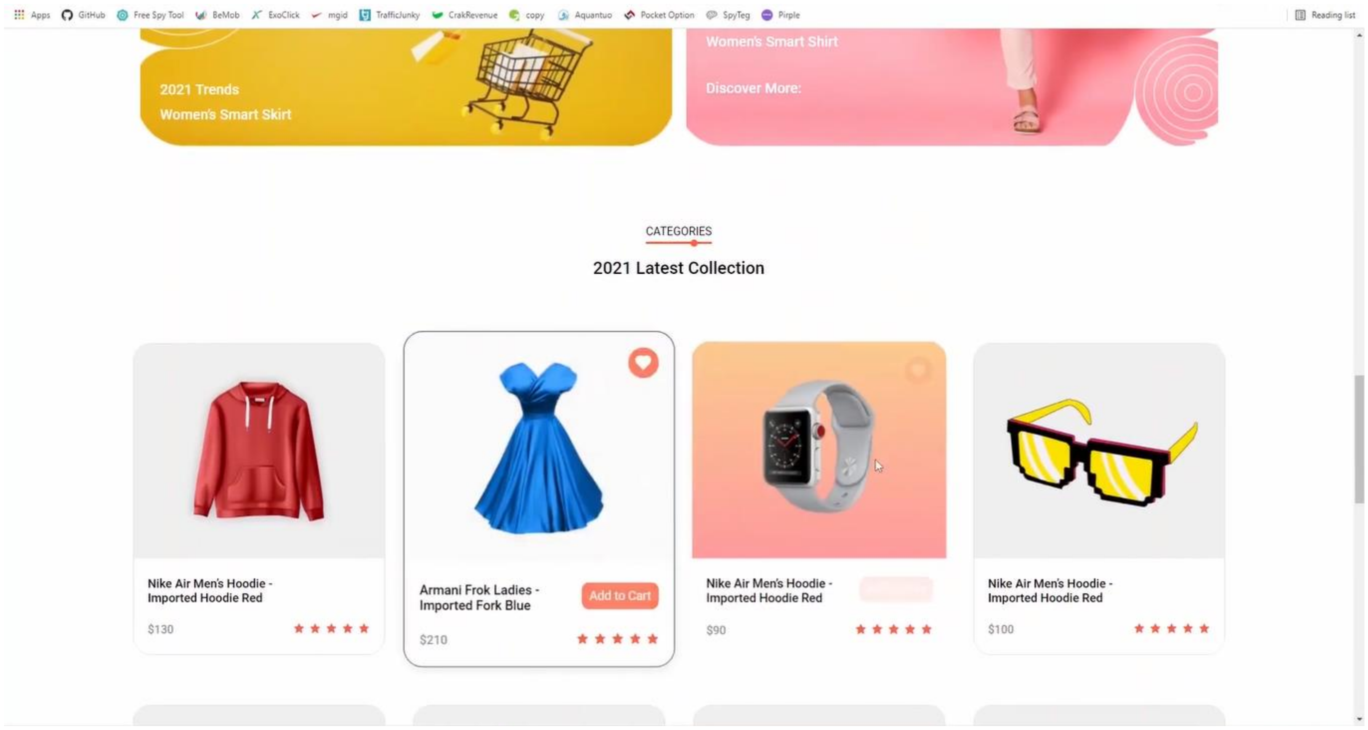
```
                    yes = 'yes'
                    query = 'UPDATE product_level SET {field}=%s WHERE
product_id=%s'.format(field=lev)
                    curs.execute(query, (yes, product_id))
                    # Commit cursor
                    db.connection.commit()
                elif category == 'belt':
                    level = request.form.getlist('belt')
                    for lev in level:
                        yes = 'yes'
                        query = 'UPDATE product_level SET {field}=%s WHERE
product_id=%s'.format(field=lev)
                        curs.execute(query, (yes, product_id))
                        # Commit cursor
                        db.connection.commit()
                elif category == 'shoes':
                    level = request.form.getlist('shoes')
                    for lev in level:
                        yes = 'yes'
                        query = 'UPDATE product_level SET {field}=%s WHERE
product_id=%s'.format(field=lev)
                        curs.execute(query, (yes, product_id))
                        # Commit cursor
                        db.connection.commit()
                else:
                    flash('Product level not fund', 'danger')
                    return redirect(url_for('admin_add_product'))
                # Close Connection
                curs.close()

                flash('Product added successful', 'success')
                return redirect(url_for('admin_add_product'))
            else:
                flash('Picture not save', 'danger')
                return redirect(url_for('admin_add_product'))
        else:
            flash('File not supported', 'danger')
            return redirect(url_for('admin_add_product'))
    else:
        flash('Please fill up all form', 'danger')
        return redirect(url_for('admin_add_product'))
else:
    return render_template('pages/add_product.html')


@app.route('/edit_product', methods=['POST', 'GET'])
@is_admin_logged_in
def edit_product():
if 'id' in request.args:
    product_id = request.args['id']
    curso = db.connection.cursor()
    res = curso.execute("SELECT * FROM products WHERE id=%s", (product_id,))
    product = curso.fetchall()
    curso.execute("SELECT * FROM product_level WHERE product_id=%s",
(product_id,))
    product_level = curso.fetchall()
    if res:
        if request.method == 'POST':
            name = request.form.get('name')
            price = request.form['price']
            description = request.form['description']
            available = request.form['available']
            category = request.form['category']
            item = request.form['item']
            code = request.form['code']
            file = request.files['picture']
            # Create Cursor
            if name and price and description and available and category and item and code
and file:
                pic = file.filename
                photo = pic.replace("'", "")
                picture = photo.replace(" ", "")
                if picture.lower().endswith(('.png', '.jpg', '.jpeg')):
                    file.filename = picture
                    save_photo = photos.save(file, folder=category)
                    if save_photo:
                        # Create Cursor
                        cur = db.connection.cursor()
                        exe = curso.execute(
                            "UPDATE products SET pName=%s, price=%s, description=%s,
available=%s, category=%s, item=%s, pCode=%s, picture=%s WHERE id=%s",
                            (name, price, description, available, category, item, code, picture,
product_id))
                    if exe:
                        if category == 'tshirt':
                            level = request.form.getlist('tshirt')
                            for lev in level:
                                yes = 'yes'
                                query = 'UPDATE product_level SET {field}=%s WHERE
product_id=%s'.format(
                                    field=lev)
                                cur.execute(query, (yes, product_id))
                                # Commit cursor
                                db.connection.commit()
                        elif category == 'wallet':
                            level = request.form.getlist('wallet')
                            for lev in level:
                                yes = 'yes'
                                query = 'UPDATE product_level SET {field}=%s WHERE
product_id=%s'.format(
                                    field=lev)
                                cur.execute(query, (yes, product_id))
                                # Commit cursor
                                db.connection.commit()
                        elif category == 'belt':
                            level = request.form.getlist('belt')
                            for lev in level:
                                yes = 'yes'
                                query = 'UPDATE product_level SET {field}=%s WHERE
product_id=%s'.format(
                                    field=lev)
                                cur.execute(query, (yes, product_id))
                                # Commit cursor
                                db.connection.commit()
                        elif category == 'shoes':
                            level = request.form.getlist('shoes')
                            for lev in level:
                                yes = 'yes'
                                query = 'UPDATE product_level SET {field}=%s WHERE
product_id=%s'.format(
                                    field=lev)
                                cur.execute(query, (yes, product_id))
                                # Commit cursor
```

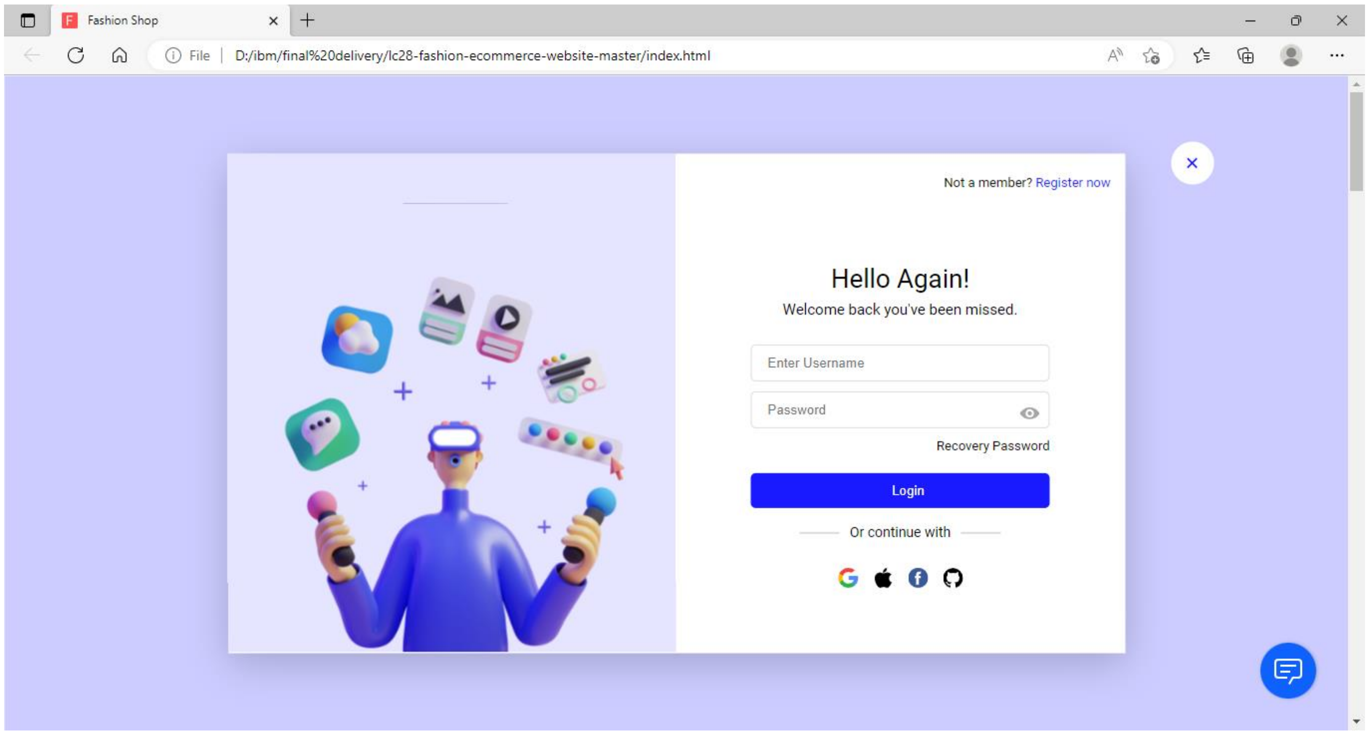
```
                              db.connection.commit()
                          else:
                              flash('Product level not fund', 'danger')
                              return redirect(url_for('admin_add_product'))
                          flash('Product updated', 'success')
                          return redirect(url_for('edit_product'))
                      else:
                          flash('Data updated', 'success')
                          return redirect(url_for('edit_product'))
                  else:
                      flash('Pic not upload', 'danger')
                      return render_template('pages/edit_product.html', product=product,
                                             product_level=product_level)
              else:
                  flash('File not support', 'danger')
                  return render_template('pages/edit_product.html', product=product,
                                         product_level=product_level)
          else:
              flash('Fill all field', 'danger')
              return render_template('pages/edit_product.html', product=product,
                                     product_level=product_level)
      else:
          return render_template('pages/edit_product.html', product=product,
product_level=product_level)
    else:
        return redirect(url_for('admin_login'))
  else:
    return redirect(url_for('admin_login'))


@app.route('/search', methods=['POST', 'GET'])
def search():
  form = OrderForm(request.form)
  if 'q' in request.args:
    q = request.args['q']
    # Create cursor
    cur = db.connection.cursor()
    # Get message
    query_string = "SELECT * FROM products WHERE pName LIKE %s ORDER BY
id ASC"
    cur.execute(query_string, ('%' + q + '%',))
```

```
        products = cur.fetchall()
        # Close Connection
        cur.close()
        flash('Showing result for: ' + q, 'success')
        return render_template('search.html', products=products, form=form)
    else:
        flash('Search again', 'danger')
        return render_template('search.html')


@app.route('/profile')
@is_logged_in
def profile():
    if 'user' in request.args:
        q = request.args['user']
        curso = db.connection.cursor()
        curso.execute("SELECT * FROM users WHERE id=%s", (q,))
        result = curso.fetchone()
        if result:
            if result['id'] == session['uid']:
                curso.execute("SELECT * FROM orders WHERE uid=%s ORDER BY id
ASC", (session['uid'],))
                res = curso.fetchall()
                return render_template('profile.html', result=res)
            else:
                flash('Unauthorised', 'danger')
                return redirect(url_for('login'))
        else:
            flash('Unauthorised! Please login', 'danger')
            return redirect(url_for('login'))
    else:
        flash('Unauthorised', 'danger')
        return redirect(url_for('login'))


class UpdateRegisterForm(Form):
    name = StringField('Full Name', [validators.length(min=3, max=50)],
                       render_kw={'autofocus': True, 'placeholder': 'Full Name'})
    email = EmailField('Email', [validators.DataRequired(), validators.Email(),
validators.length(min=4, max=25)],
                       render_kw={'placeholder': 'Email'})
    password = PasswordField('Password', [validators.length(min=3)],
                             render_kw={'placeholder': 'Password'})
    mobile = StringField('Mobile', [validators.length(min=11, max=15)],
render_kw={'placeholder': 'Mobile'})


@app.route('/settings', methods=['POST', 'GET'])
@is_logged_in
def settings():
    form = UpdateRegisterForm(request.form)
    if 'user' in request.args:
        q = request.args['user']
        curso = db.connection.cursor()
        curso.execute("SELECT * FROM users WHERE id=%s", (q,))
        result = curso.fetchone()
        if result:
            if result['id'] == session['uid']:
                if request.method == 'POST' and form.validate():
                    name = form.name.data
                    email = form.email.data
                    password = sha256_crypt.encrypt(str(form.password.data))
                    mobile = form.mobile.data

                    # Create Cursor
                    cur = db.connection.cursor()
                    exe = cur.execute("UPDATE users SET name=%s, email=%s, password=%s,
mobile=%s WHERE id=%s",
                                      (name, email, password, mobile, q))
                    if exe:
                        flash('Profile updated', 'success')
                        return render_template('user_settings.html', result=result, form=form)
                    else:
                        flash('Profile not updated', 'danger')
                return render_template('user_settings.html', result=result, form=form)
            else:
                flash('Unauthorised', 'danger')
                return redirect(url_for('login'))
        else:
            flash('Unauthorised! Please login', 'danger')
            return redirect(url_for('login'))
    else:
        flash('Unauthorised', 'danger')
        return redirect(url_for('login'))


class DeveloperForm(Form):  #
    id = StringField('', [validators.length(min=1)],
                     render_kw={'placeholder': 'Input a product id...'})


@app.route('/developer', methods=['POST', 'GET'])
def developer():
    form = DeveloperForm(request.form)
    if request.method == 'POST' and form.validate():
        q = form.id.data
        curso = db.connection.cursor()
        result = curso.execute("SELECT * FROM products WHERE id=%s", (q,))
        if result > 0:
            x = content_based_filtering(q)
            wrappered = wrappers(content_based_filtering, q)
            execution_time = timeit.timeit(wrappered, number=0)
            seconds = ((execution_time / 1000) % 60)
            return render_template('developer.html', form=form, x=x,
execution_time=seconds)
        else:
            nothing = 'Nothing found'
            return render_template('developer.html', form=form, nothing=nothing)
    else:
        return render_template('developer.html', form=form)


if __name__ == '__main__':
    app.run(debug=True)
```

OUTPUT:


Fashion Shop
sprint-1.pdf
File
D:/ibm/final%20delivery/lc28-fashion-ecommerce-website-master/demo.html?
javadoc
Home
Shop
Pages
About Us
Lookups
Limited Time Only For Winter
fashion
LOOK YOUR BEST ON YOUR BEST DAY
Explore Now!


Apps
GitHub
Free Spy Tool
BeMob
ExoClick
mgid
TrafficJunky
CrakRevenue
copy
Aquantuo
Pocket Option
SpyTeg
Pirple
Reading list
2021 Trends
Women's Smart Skirt
Women's Smart Shirt
Discover More:
CATEGORIES
2021 Latest Collection
Nike Air Men's Hoodie - Imported Hoodie Red
$130
Armani Frok Ladies - Imported Fork Blue
Add to Cart
$210
Nike Air Men's Hoodie - Imported Hoodie Red
$90
Nike Air Men's Hoodie - Imported Hoodie Red
$100

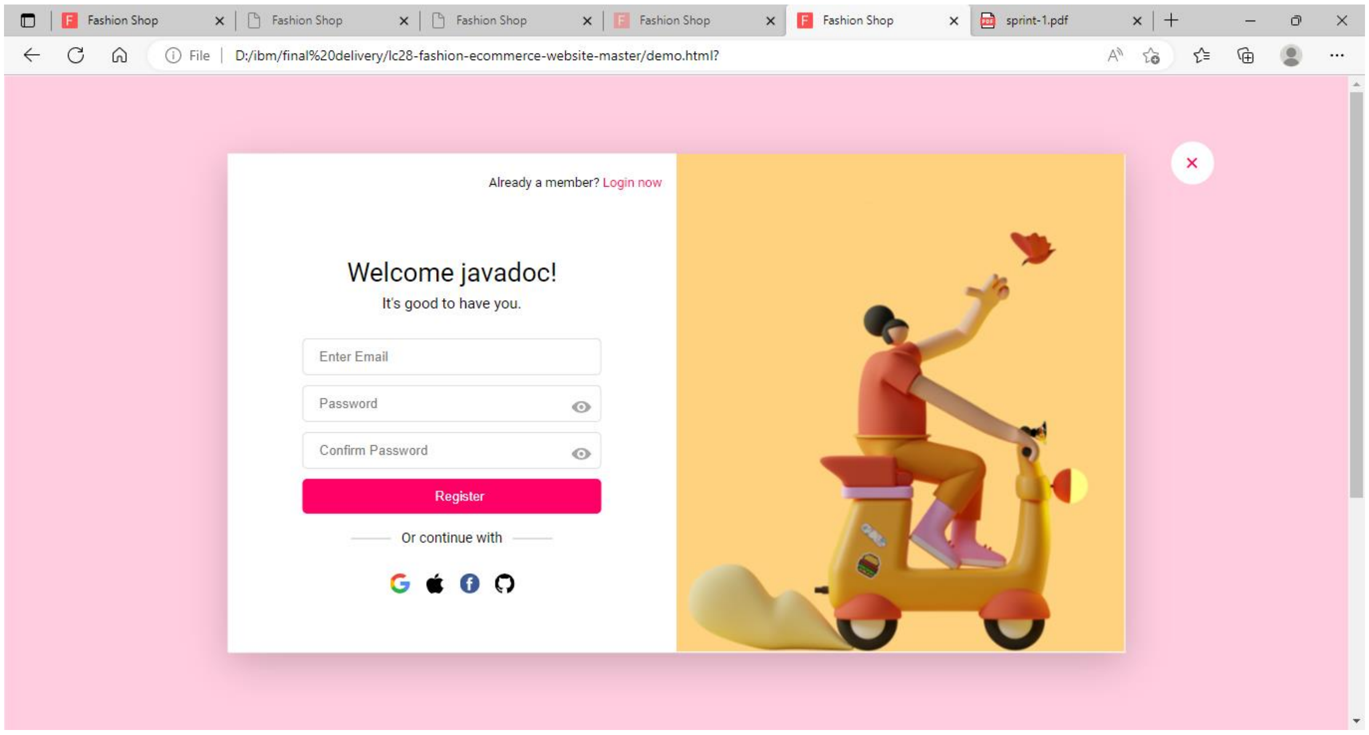

Fashion Shop

D:/ibm/final%20delivery/lc28-fashion-ecommerce-website-master/index.html

Not a member? Register now

Hello Again!

Welcome back you've been missed.

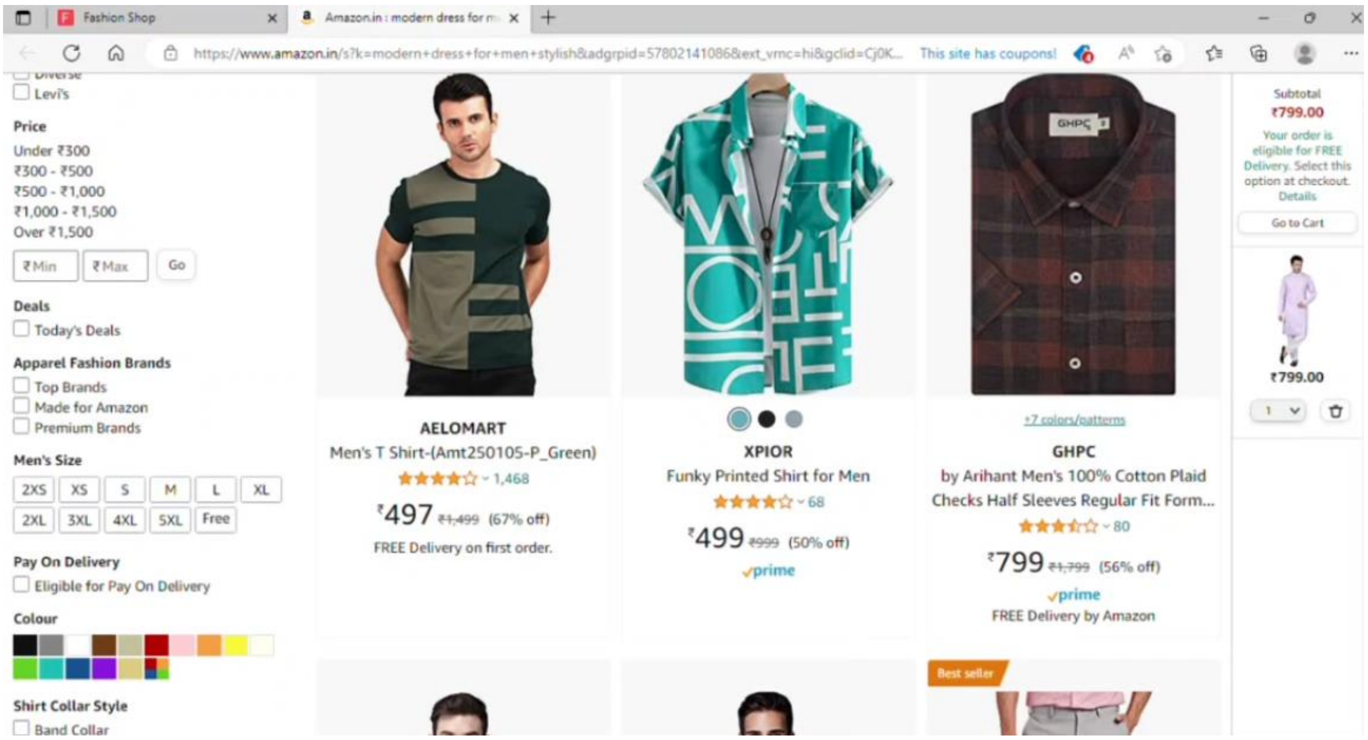

Fashion Shop | Fashion Shop | Fashion Shop | Fashion Shop | Fashion Shop | sprint-1.pdf

D:/ibm/final%20delivery/lc28-fashion-ecommerce-website-master/demo.html?

Already a member? Login now

Welcome javadoc!

It's good to have you.

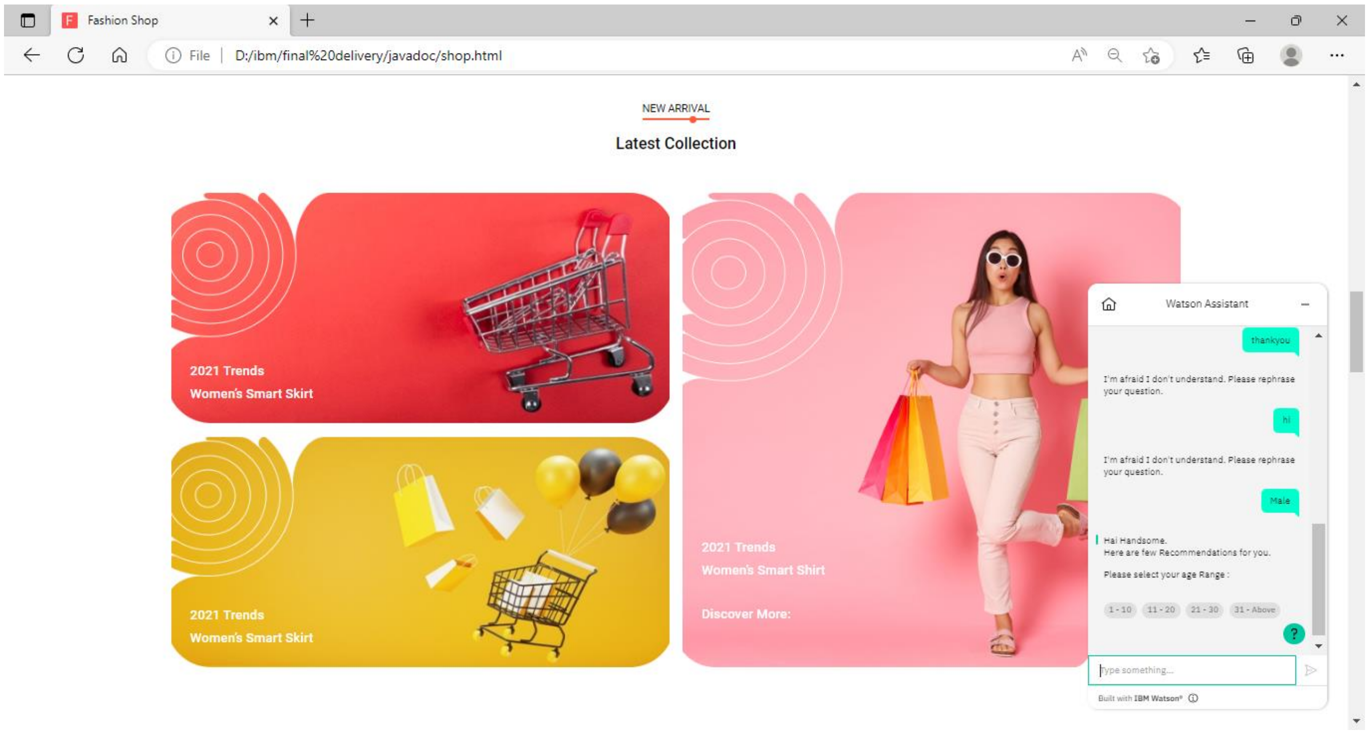


Your order is eligible for FREE Delivery. Select this option at checkout. Details

AELOMART
Men's T Shirt-(Amt250105-P_Green)
1,468
₹497 ~~₹1,499~~ (67% off)
FREE Delivery on first order.

XPIOR
Funky Printed Shirt for Men
68
₹499 ~~₹999~~ (50% off)
prime

+7 colors/patterns
GHPC
by Arihant Men's 100% Cotton Plaid Checks Half Sleeves Regular Fit Form...
80
₹799 ~~₹1,799~~ (56% off)
prime
FREE Delivery by Amazon



thankyou

I'm afraid I don't understand. Please rephrase your question.

hi

I'm afraid I don't understand. Please rephrase your question.

Male

Hai Handsome.
Here are few Recommendations for you.

Please select your age Range :

GITHUB LINK: https://github.com/IBM-EPBL/IBM-Project-46347-1660745898

DEMO VIDEO LINK: https://www.youtube.com/watch?v=h1mw3h906Gk