Rajalakshmi Institute Of Technology

PROJECT

SMART FASHION RECOMMENDER APPLICATION

DONE BY

TEAM ID: PNT2022TMID26539

SANTOSH V (211719106073)

SACHIN PJ (21171906066)

PRATHEEV J (211719106059)

SRIVARSHAN R (211719106083)

TABLE OF CONTENT

1. INTRODUCTION

1.1 Project Overview :

E-commerce and fashion apps are expanding in popularity right now. Additionally, it has certain issues with locating the desired goods for the user in the web apps. It can be very helpful to have a chatbot that comprehends the algorithm of a certain application. We are integrating a chatbot like this into a web application, which is supplied with the algorithmic knowledge of the application. It entirely assists the user from identifying their needs to processing payments and starting deliveries. By gathering basic user information and behaviours, it functions as an advanced filter search that may give the user what they want with the use of visual and naming representation. Additionally, there are two main UI interactions in the application: one is the user panel, and the otherPurpose :

Contrary to other sectors, fashion advice shouldn't be exclusively focused on a customer's preferences and past behaviour. The complexity of developing a fashion suggestion system is increased by numerous external elements, many of which are emotional. The general public's perceptions must be taken into consideration, along with fashion, clothing, and trend guidelines.

2. LITERATURE SURVEY

2.1 Existing problem :

In traditional e-commerce websites, visitors must utilise a search box to find the goods they need or scroll through all of the results of their search. It will require a lot of user time, and many user trials will be flawed as a result. This strategy will result in poor product marketing. It will leave a terrible impression on the user when they return later to buy the product. Despite the fact that the product is excellent, the user When a product has a different name, this type of search will generate mismatched results. Consider a hypothetical Amazon search for oranges. It will occasionally display orange fruit or an orange tint. Deep learning and artificial intelligence have recently been merged with fashion systems. Although these methods offer rich recommendations, they are typically prone to product mismatch. Even The recommender system suggests products based on the user's preferences, but it is missing a chat bot that enhances user experience by communicating with people. To discover the right product in the majority of fashion systems, the user must search through a variety of products. Users are required to filter products using the extensive variety of categories available.

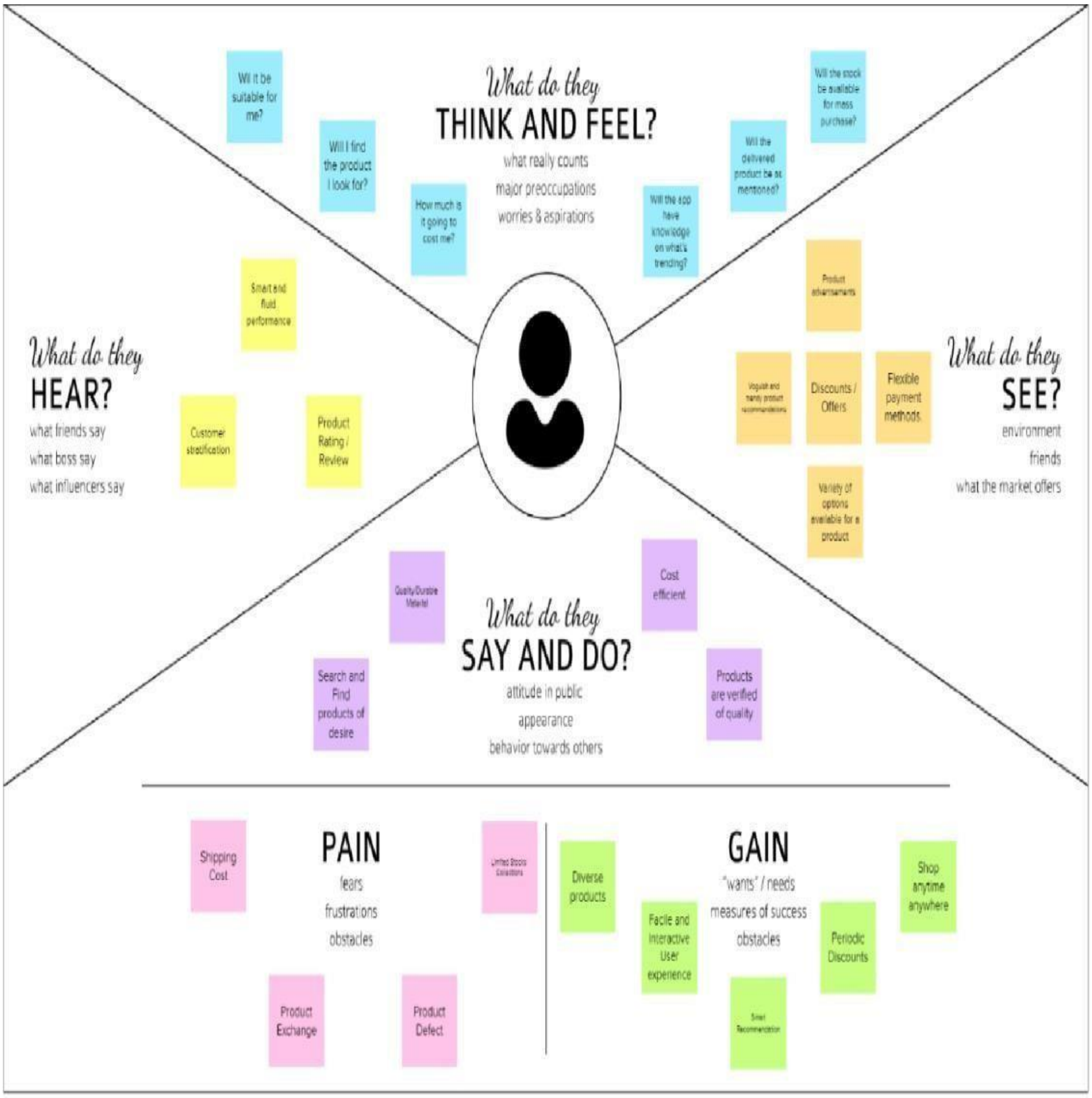
2.2 Problem Statement Definition:

In order to browse, add to basket, and place orders on e-commerce websites, customers must search for products and move between screens. The smart fashion recommender programme makes use of a chat bot to communicate with users, learn about their preferences, and provide appropriate product recommendations. The users of this programme are assigned to one of two predetermined roles. Customer and administrator are the roles. According to the designated role, the application requires that the user be forwarded to the proper dashboard. The quantity of various products and admins should be tracked

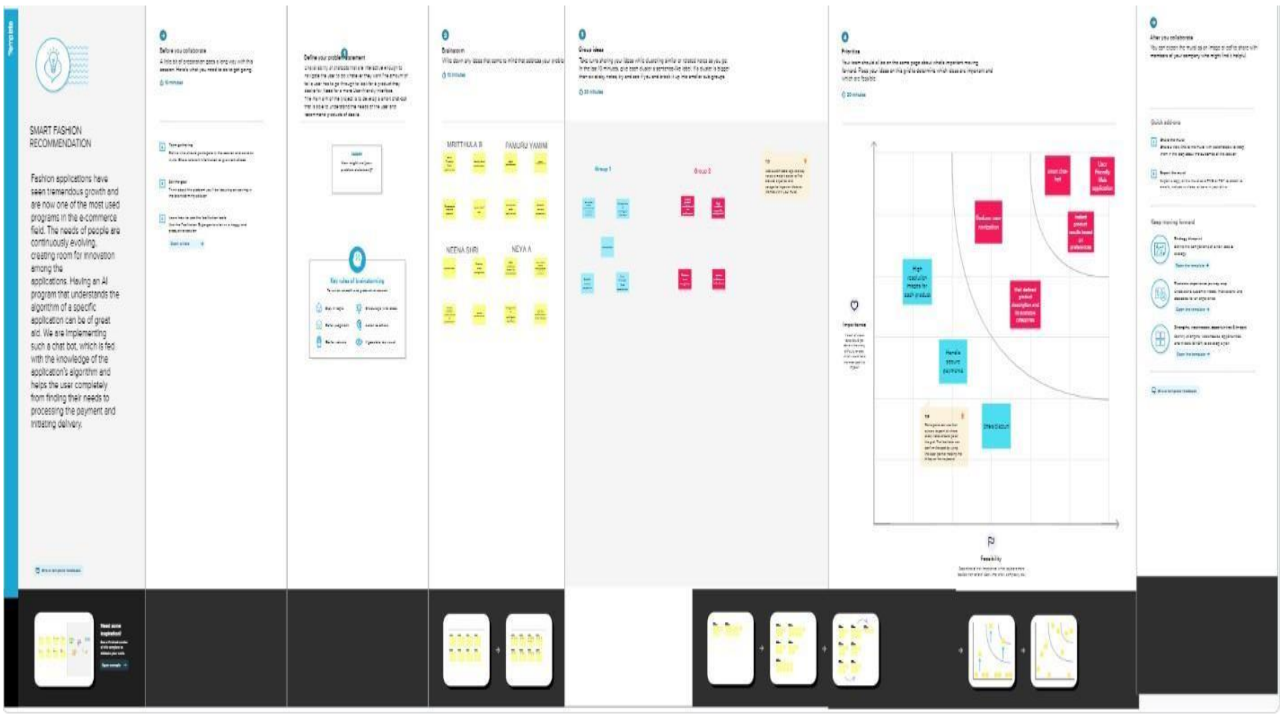
should be tasked with developing products that fall into the right categories. Through chat bot interaction, the user should be able to express their preferences. On order confirmation or failure, the user must be notified. At the conclusion of order confirmation, the chatbot needs to get user input. The major goals of this programme are to improve user interaction and minimise page-scrolling in order to locate the right products.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



SMART FASHION RECOMMENDATION

Fashion applications have seen tremendous growth and are now one of the most used programs in the e-commerce field. The needs of people are continuously evolving, creating room for innovation among the applications. Having an AI program that understands the algorithm of a specific application can be of great aid. We are Implementing such a chat bot, which is fed with the knowledge of the application's algorithm and helps the user completely from finding their needs to processing the payment and initiating delivery.

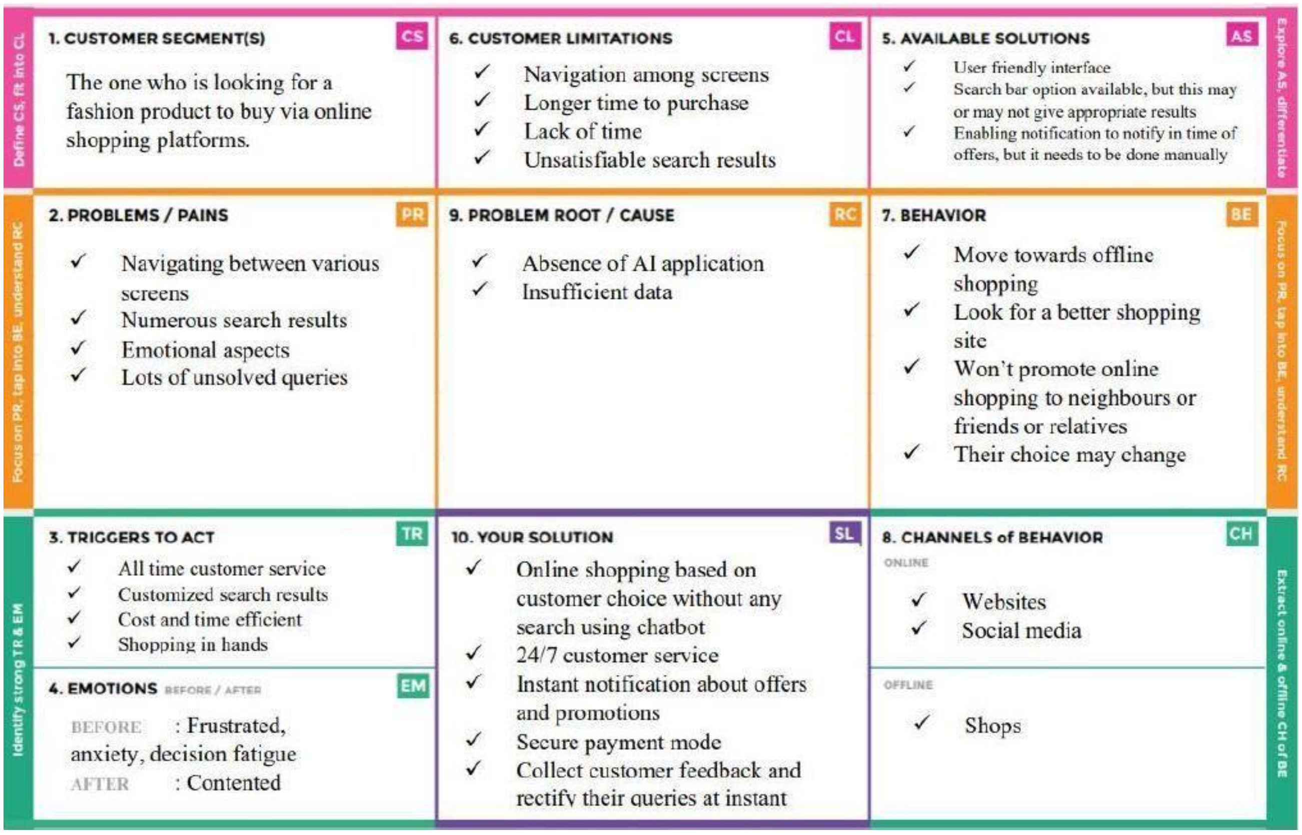
3.3 Proposed Solution

S.No.		**Parameter**	**Description**
1.		Problem Statement (Problem to be solved)	In normal Online Shopping: • Human mistake can happen in many different ways. • Negative customer feedback • There isn't a good enough way to properly direct customers. • Poor sales. • Poor customer relationships

2.		Idea / Solution description	• Lessen the possibility of human error. • Gather frank and insightful client feedback. • Lead customers in the direction of a purchase. • Converts leads to sales (aka, boost conversion). develop more effective customer interactions
3.		Novelty / Uniqueness	• Raise the percentage of calls and chats that are successfully handled by using the most recent BERTbased natural language understanding (NLU) models, which can correctly and effectively discern intent and context in use cases with more complex use cases..
4.		Social Impact / Customer Satisfaction	• • From little to major complaints, how complaints are handled may be one of the most important components in gaining client satisfaction. In order to maintain the customer's satisfaction, complaints must be handled as quickly as feasible.
5.		Business Model (Revenue Model)	• This application can be created at a low price while still delivering high performance.

6.		Scalability of the Solution	This can be made into a scalable product by using sensors, transmitting the data over wireless sensor networks, and analysing the data. Using chat bots, operations are carried out on the cloud.

3.4 Problem Solution fit



Define CS, fit into CL

1. CUSTOMER SEGMENT(S) CS

The one who is looking for a fashion product to buy via online shopping platforms.

6. CUSTOMER LIMITATIONS CL

- ✓ Navigation among screens
- ✓ Longer time to purchase
- ✓ Lack of time
- ✓ Unsatisfiable search results

5. AVAILABLE SOLUTIONS AS

- ✓ User friendly interface
- ✓ Search bar option available, but this may or may not give appropriate results
- ✓ Enabling notification to notify in time of offers, but it needs to be done manually

Explore AS, differentiate

Focus on PR, tap into BE, understand RC

2. PROBLEMS / PAINS PR

- ✓ Navigating between various screens
- ✓ Numerous search results
- ✓ Emotional aspects
- ✓ Lots of unsolved queries

9. PROBLEM ROOT / CAUSE RC

- ✓ Absence of AI application
- ✓ Insufficient data

7. BEHAVIOR BE

- ✓ Move towards offline shopping
- ✓ Look for a better shopping site
- ✓ Won't promote online shopping to neighbours or friends or relatives
- ✓ Their choice may change

Focus on PR, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TO ACT TR

- ✓ All time customer service
- ✓ Customized search results
- ✓ Cost and time efficient
- ✓ Shopping in hands

4. EMOTIONS BEFORE / AFTER EM

BEFORE : Frustrated, anxiety, decision fatigue
AFTER : Contented

10. YOUR SOLUTION SL

- ✓ Online shopping based on customer choice without any search using chatbot
- ✓ 24/7 customer service
- ✓ Instant notification about offers and promotions
- ✓ Secure payment mode
- ✓ Collect customer feedback and rectify their queries at instant

8. CHANNELS of BEHAVIOR CH

ONLINE

- ✓ Websites
- ✓ Social media

OFFLINE

- ✓ Shops

Extract online & offline CH of BE

4. REQUIREMENT ANALYSIS

2.3 Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Registration	Registration requires some user information and can be completed using a cell number or Gmail.
FR-2	Login	Only the user id and password provided during registration are used to log in.
FR-3	Delivery confirmation	confirmation through phone and email
FR-4	Assistance	Bot is integrated with the application to make the usability simple

2.4 Non-Functional requirements

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

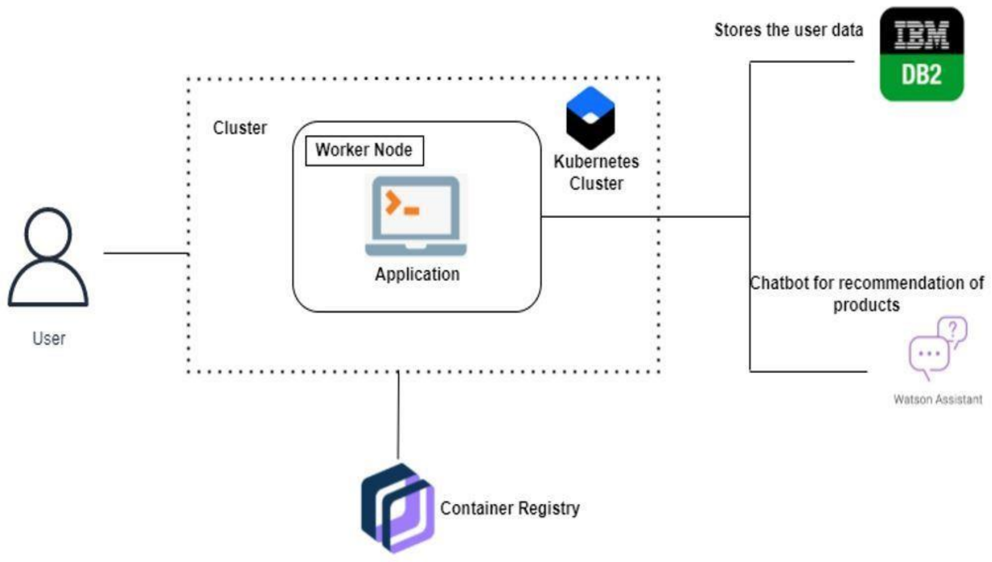
NFR-4	Performance	The system shall be able to handle multiple requests at any given point in time and generate an appropriate response.

5. PROJECT DESIGN

5.1 Data Flow Diagrams:

FR No.	Non-Functional Requirement	Description
NF R-1	**Usability**	a chatbot and user-friendly UI for effective usability
NF R-2	**Security**	A protected connection Requests and answers should be transmitted via HTTPS.
NF R-3	**Reliability**	To prevent programme termination, the system should manage expected as well as unexpected failures and exceptions.

5.2 Solution & Technical Architecture:


Stores the user data
IBM
DB2
Cluster
Worker Node
Application
Kubernetes Cluster
User
Chatbot for recommendation of products
Watson Assistant
Container Registry

6 PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation & Sprint Delivery Schedule & Reports from JIRA

Date	30 October 2022
Team ID	PNT2022TMID26539
Project Name	Smart Fashion Recommender Application

Remaining tasks (Milestones & Activities) to be completed

Milestones	Activities	Description
Project Development Phase	Delivery of Sprint – 1,2,3,4	To develop the code and submit the developed code by testing it
Setting up App environment	Create IBM Cloud account	Signup for an IBM Cloud account
	Create flask project	Getting started with Flask to create project
	Install IBM Cloud CLI	Install IBM Command Line Interface
	Docker CLI Installation	Installing Docker CLI on laptop
	Create an account in sendgrid	Create an account in sendgrid. Use the service as email integration to our application for sending emails
Implementing web Application	Create UI to interact with Application	Create UI • Registration page • Login page • View products page • Add products page
	Create IBM DB2 & connect with python	Create IBM DB2 service in IBM Cloud and connect with python code with DB
Integrating sendgrid service	Sendgrid integration with python	To send emails form the application we need to integrate the Sendgrid service
Developing a chatbot	Building a chatbot and Integrate to application	Build the chatbot and Integrate it to the flask application
Deployment of App in IBM Cloud	Containerize the App	Create a docker image of your application and push it to the IBM container registry
	Upload image to IBM container registry	Upload the image to IBM container registry

	Deploy in kubernetes cluster	Once the image is uploaded to IBM Container registry deploy the image to IBM Kebernetes cluster

Finished tasks (Milestones & Activities)

Milestones	Activities	Description
Ideation Phase	Literature Survey	Literature survey on the selected project & information gathering
	Empathy Map	Prepare Empathy map to capture the user Panis & Gains, prepare list of problem statement
	Ideation	Organizing the brainstroming session and prioritise the top 3 ideas based on feasibility & Importance
Project Design Phase I	Proposed Solution	Prepare proposed solution document which includes novelty, feasibility of ideas, business model, social impact, Scalability of solution
	Problem Solution Fit	Prepare problem solution fit document
	Solution Architecture	Prepare solution architecture document
Project Design Phase II	Customer Journey	Prepare customer journey map to understand the user interactions & experience with the application
	Functional requirement	Prepare functional & non functional requirement document
	Data Flow Diagram	Prepare Data Flow Diagram and user stories
	Technology architecture	Draw the technology architecture diagram
Project Planning Phase	Milestones & Activity list	Prepare milestones and activity list of the project
	Sprint Delivery Plan	Prepare sprint delivery plan

Product Backlog, Sprint Schedule, Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story points	Priority	Team Members
Sprint-1	Setting up App environment	USN-1	As a user, I can register in ICTA Academy and create IBM cloud account.	2	High	Santosh V Pratheev j
Sprint-1		USN-2	As a user, I will create a flask project	1	Low	Sachin pj Srivarshan R
Sprint-1		USN-3	As a user, I will install IBM Cloud CLI	2	Medium	Santosh v Sachin Pj
Sprint-2	Setting up App environment	USN-4	As a user, I can install Docker CLI	1	Low	Pratheev j Srivarshan R
Sprint-2		USN-5	As a user, I will Create an account in sendgrid	2	Medium	Pratheev j Santosh V
Sprint-3	Implementing web application	USN -6	As a user, I Create UI to interact with the application	1	High	Santosh V Sachin pj
Sprint-3		USN -7	As a user, I Create IBM DB2 and connect with Python		High	Pratheevj
Sprint-3	Integrating sendgrid service	USN -8	As a user, I will integrating sendgrid with python code	2	High	Srivarshan R
Sprint-3	Developing a chatbot	USN -9	As a user, I have to build a chatbot and Integrate to application	1	Medium	Santhosh V
Sprint -4	Development of App in IBM Cloud	USN -10	As a user, I will Containerize the App	1	Low	Sachin pj
Sprint -4		USN -11	As a user, I will upload image to IBM Container registry	2	Medium	Pratheev j
Sprint -4		USN -12	As a user, I will deploy App in Kebernetes cluster	3	High	Pratheev j

Sprint -4	User panel		As a user ● Register, Login, Email, Verification ● Manual Search ● Order placement, Order Details	3	High	Santosh v Srivarshan R

Project Tracker, Velocity & Burndown Chart

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

Velocity

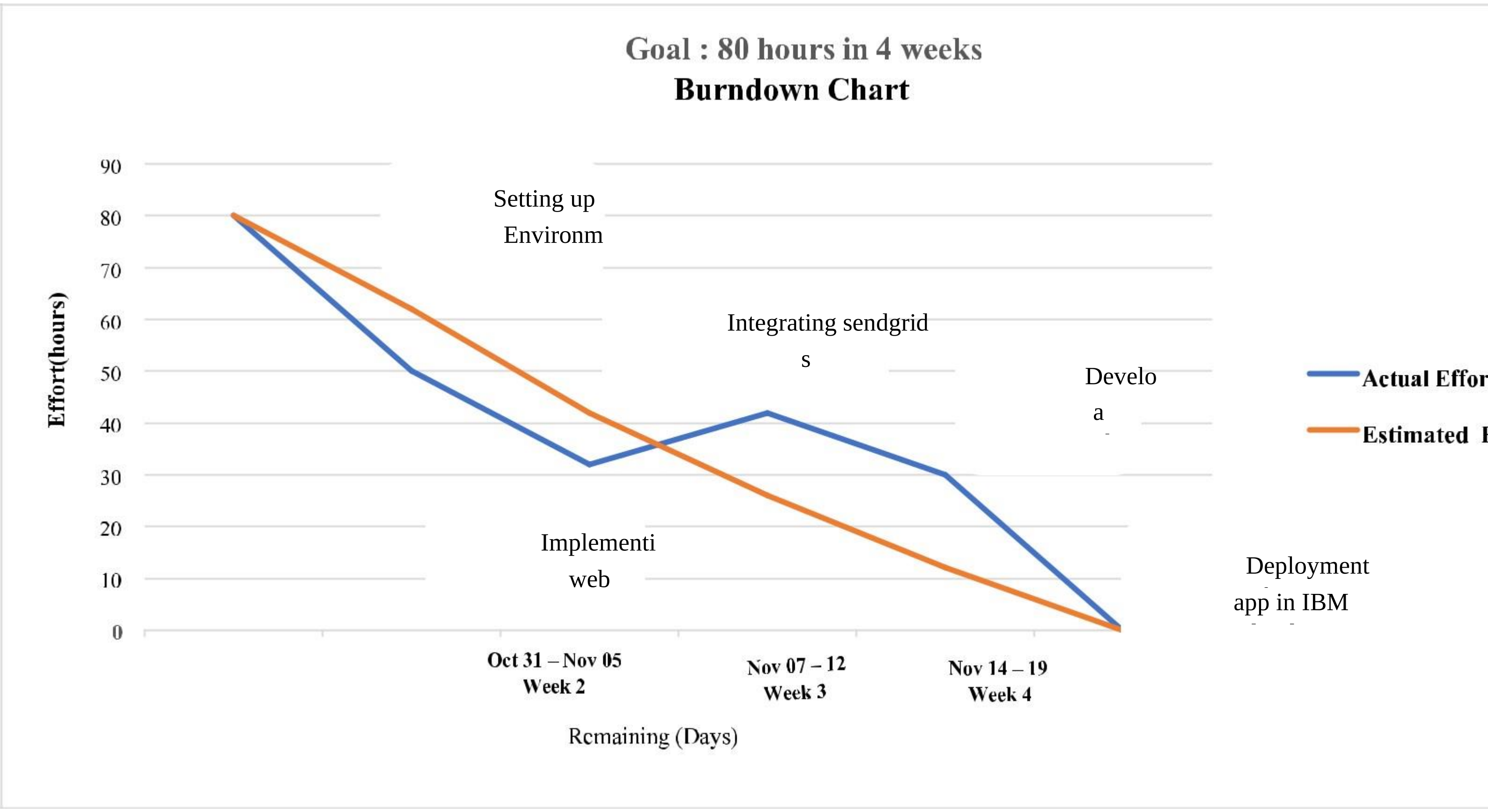
Imagine we have a 6-day sprint duration, and the velocity of the team is 18(points per sprint). Let’s calculate the team’s average velocity (AV) per iteration unit (story points per day)

AV = Sprint Duration / Velocity

AV=24/6=4

Burndown Chart

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

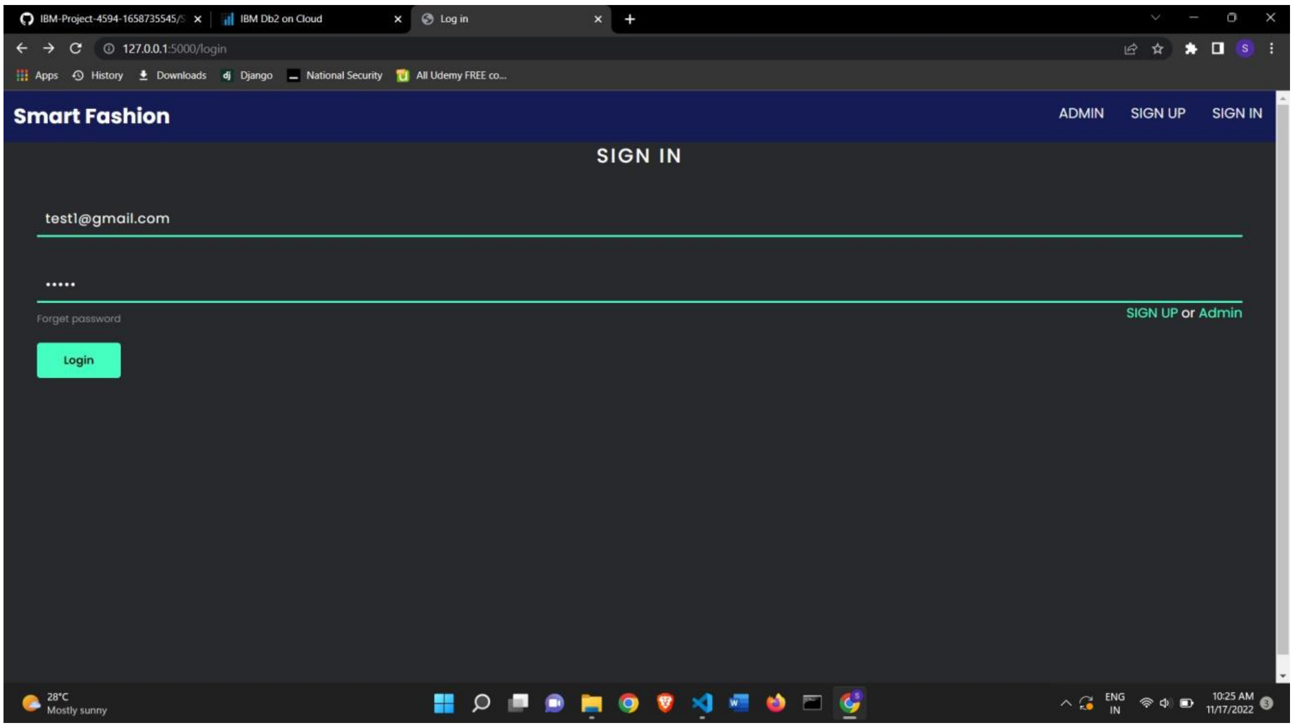


7 TESTING

7.1 Test Cases

TEST CASE ID	Test Description	Input	Excepted result	Remarks
Test case – 1	Enter a field	User detail	Successfully registered	Pass
Test case – 2	Enter username and password	User details	Redirect to main page	Pass
Test case – 3	Username and password are empty	User details	Username and password are still empty	Fail

8 RESULTS


Smart Fashion

ADMIN SIGN UP SIGN IN

SIGN IN

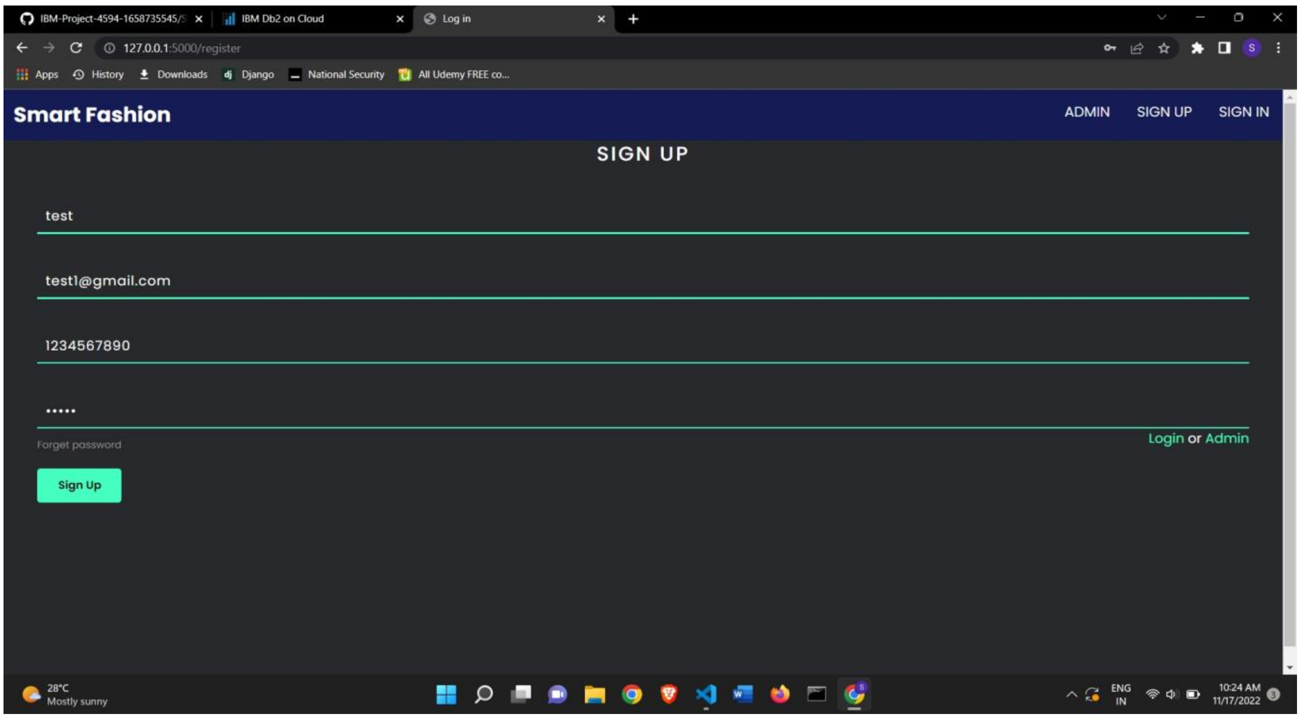
test1@gmail.com

•••••

Forget password

SIGN UP or Admin

Login


Smart Fashion

ADMIN SIGN UP SIGN IN

SIGN UP

test

test1@gmail.com

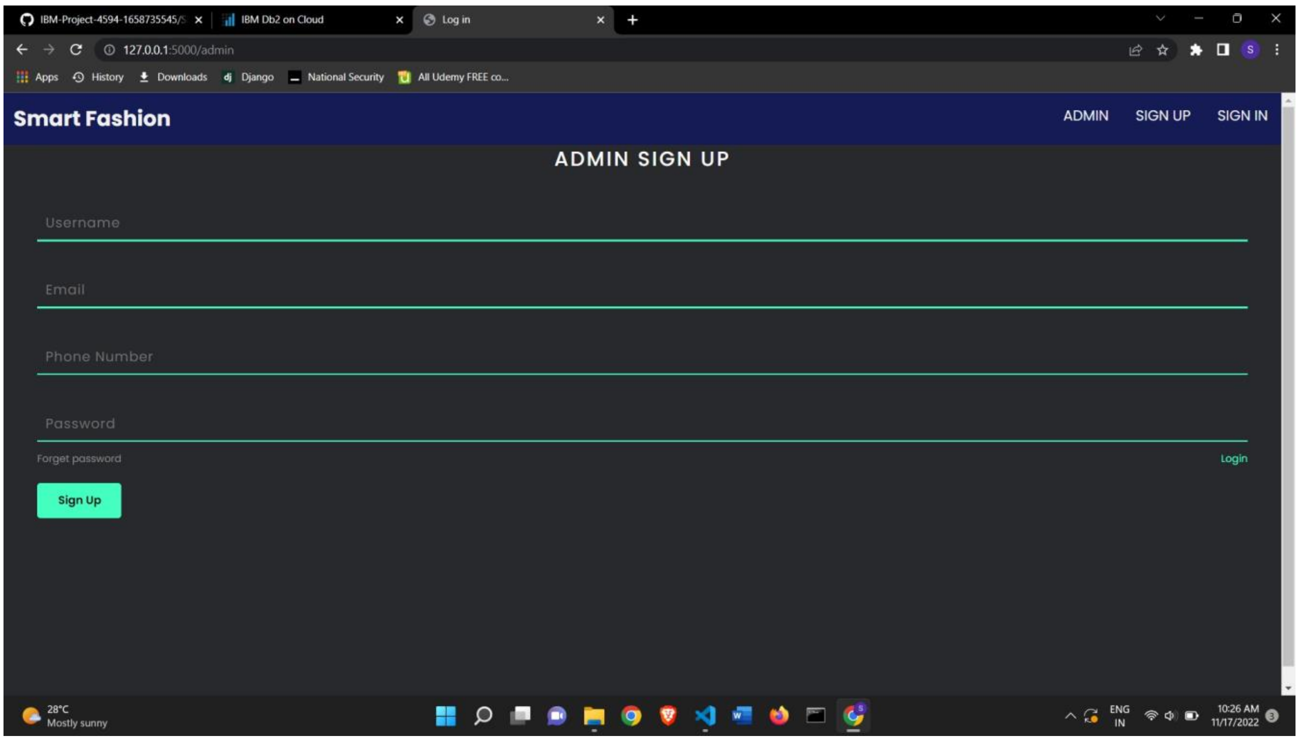
1234567890

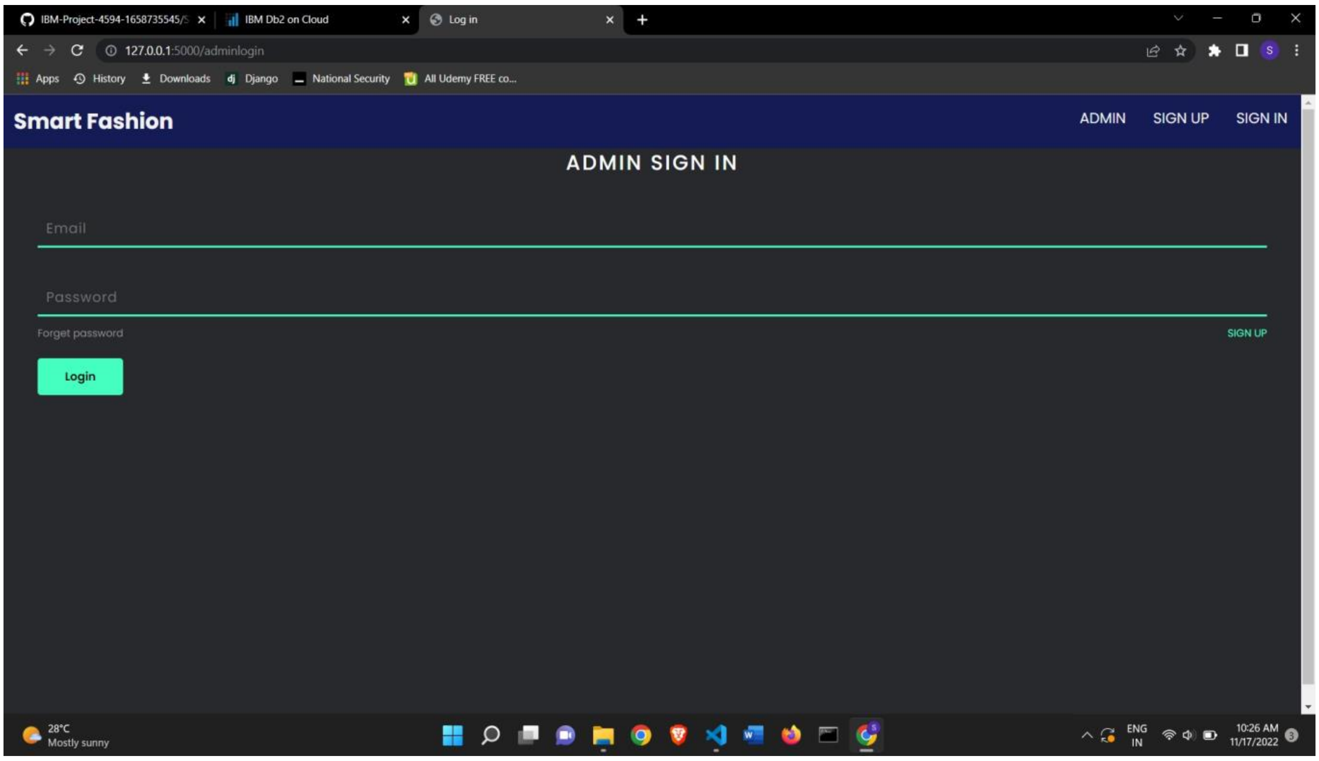
•••••

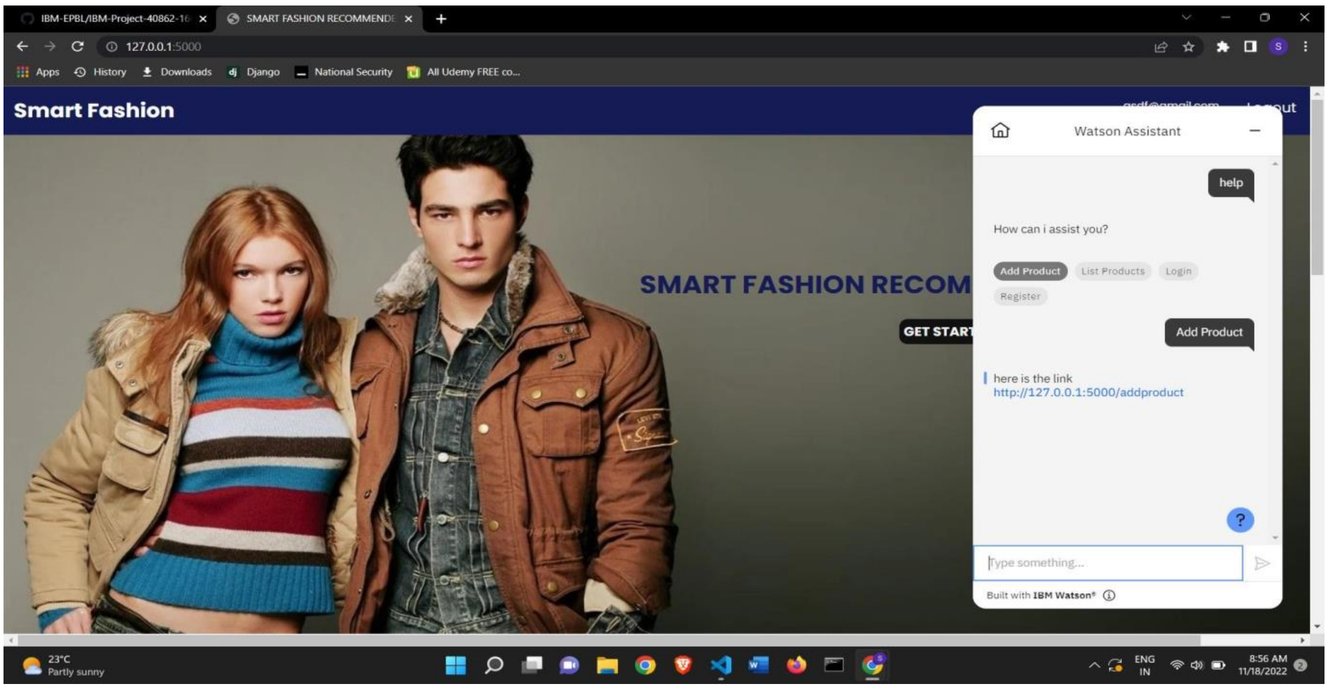
Forget password

Login or Admin

Sign Up


IBM-Project-4594-1658735545/S
IBM Db2 on Cloud
Log in
127.0.0.1:5000/admin
Apps
History
Downloads
Django
National Security
All Udemy FREE co...
Smart Fashion
ADMIN
SIGN UP
SIGN IN
ADMIN SIGN UP
Username
Email
Phone Number
Password
Forget password
Login
Sign Up
28°C
Mostly sunny
ENG
IN
10:26 AM
11/17/2022


IBM-Project-4594-1658735545/S
IBM Db2 on Cloud
Log in
127.0.0.1:5000/adminlogin
Apps
History
Downloads
Django
National Security
All Udemy FREE co...
Smart Fashion
ADMIN
SIGN UP
SIGN IN
ADMIN SIGN IN
Email
Password
Forget password
SIGN UP
Login
28°C
Mostly sunny
ENG
IN
10:26 AM
11/17/2022



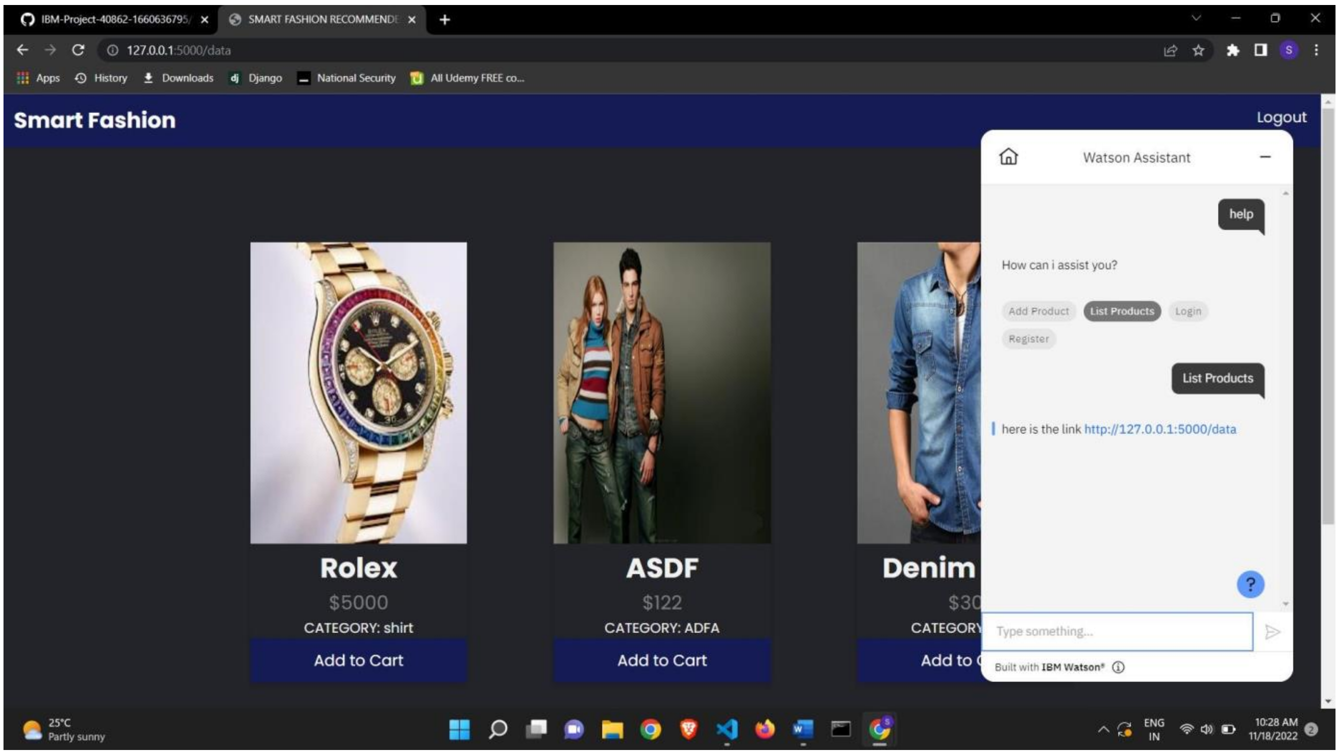
help

How can i assist you?

Add Product · List Products · Login · Register

Add Product

here is the link http://127.0.0.1:5000/addproduct



Rolex
$5000
CATEGORY: shirt
Add to Cart

ASDF
$122
CATEGORY: ADFA
Add to Cart

Denim
$30
CATEGORY
Add to C

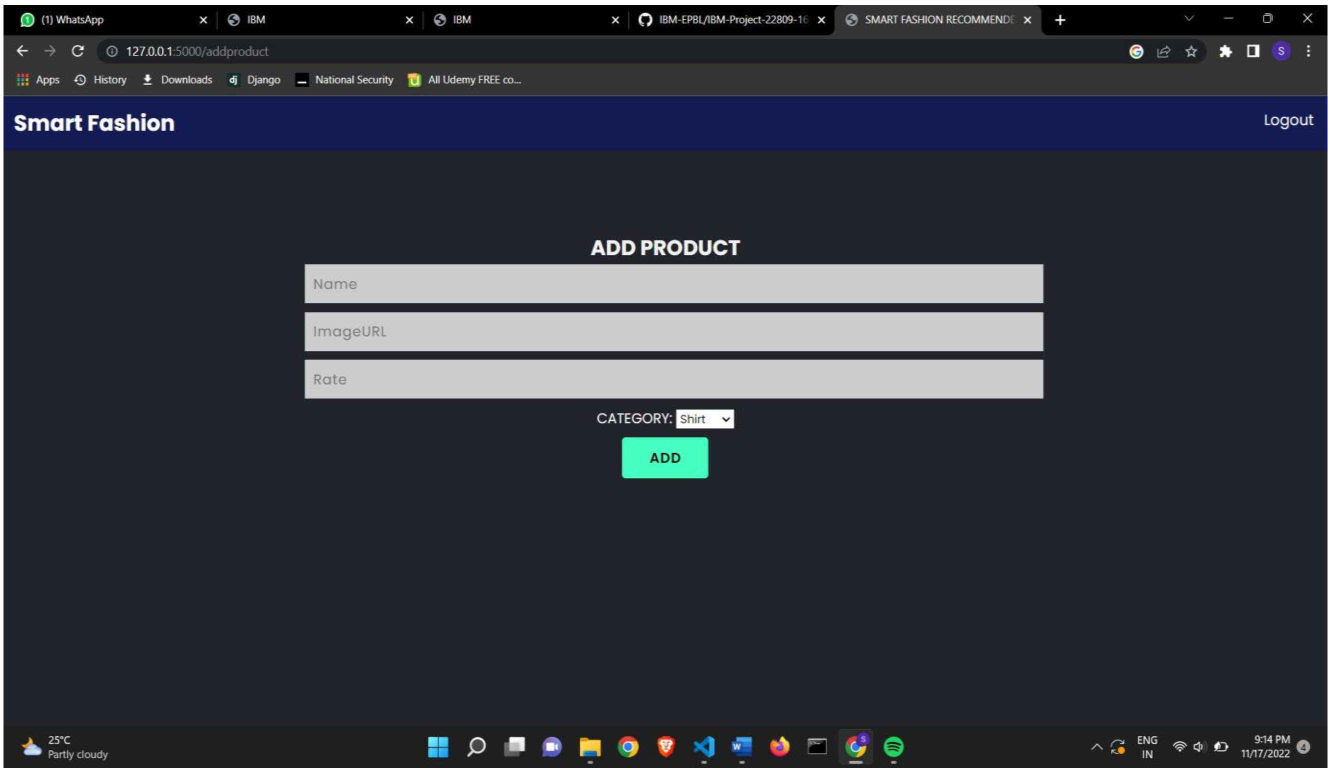
help

How can i assist you?

Add Product · List Products · Login · Register

List Products

here is the link http://127.0.0.1:5000/data



ADD PRODUCT

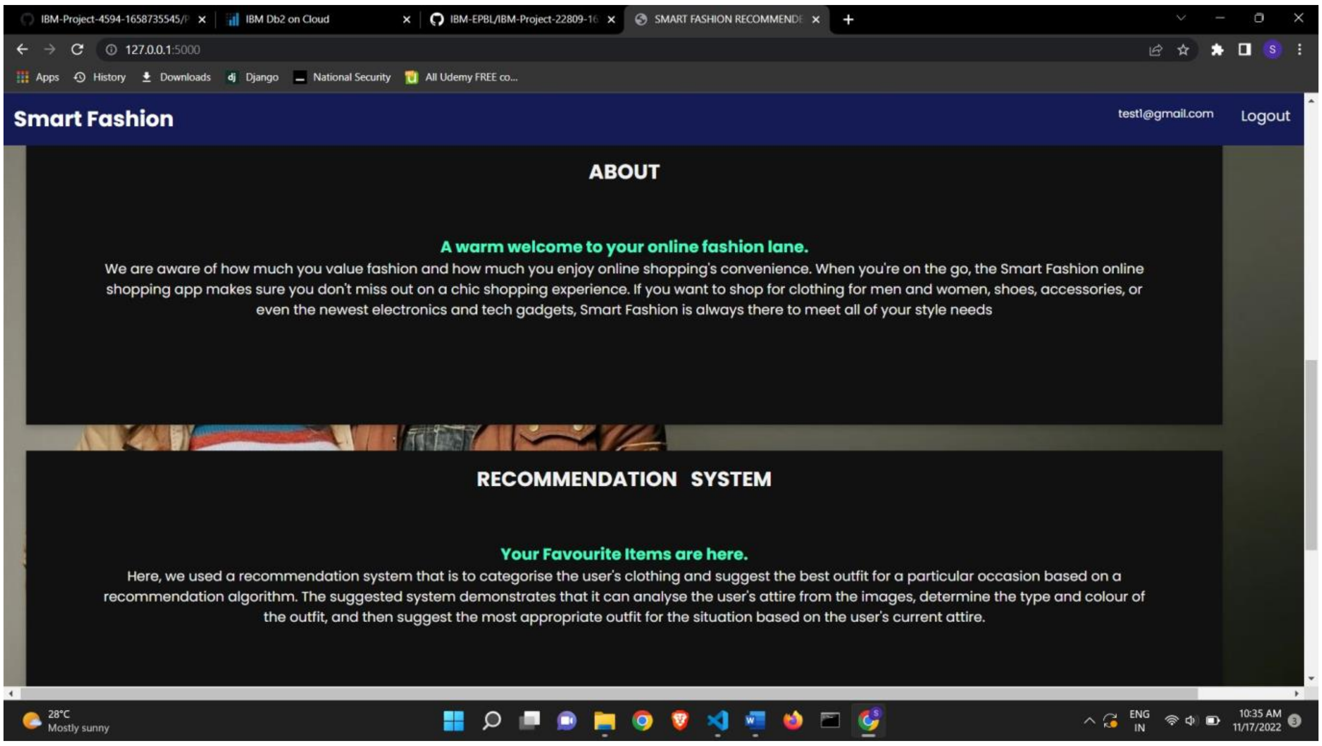
Name

ImageURL

Rate

CATEGORY: Shirt

ADD



ABOUT

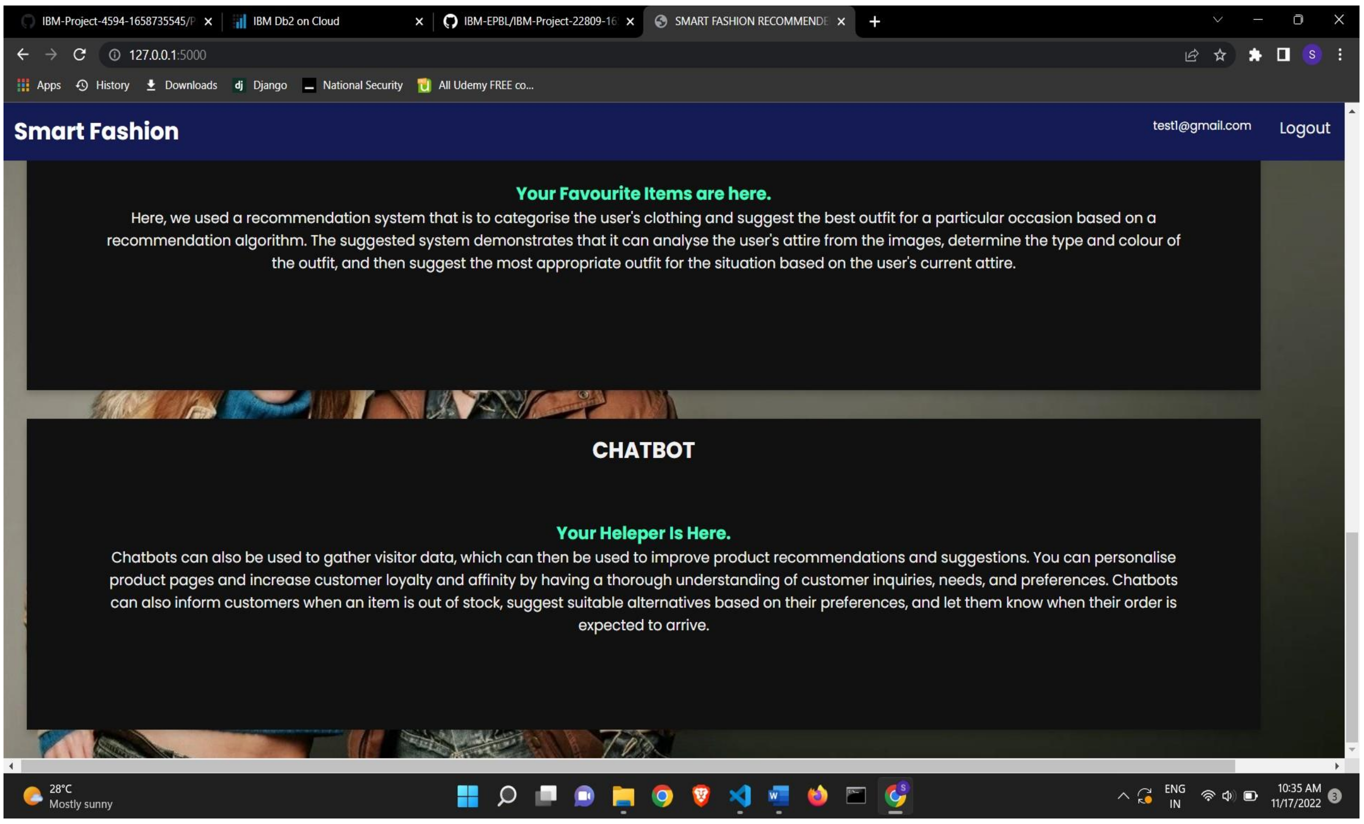
A warm welcome to your online fashion lane.

We are aware of how much you value fashion and how much you enjoy online shopping's convenience. When you're on the go, the Smart Fashion online shopping app makes sure you don't miss out on a chic shopping experience. If you want to shop for clothing for men and women, shoes, accessories, or even the newest electronics and tech gadgets, Smart Fashion is always there to meet all of your style needs

RECOMMENDATION SYSTEM

Your Favourite Items are here.

Here, we used a recommendation system that is to categorise the user's clothing and suggest the best outfit for a particular occasion based on a recommendation algorithm. The suggested system demonstrates that it can analyse the user's attire from the images, determine the type and colour of the outfit, and then suggest the most appropriate outfit for the situation based on the user's current attire.



Your Favourite Items are here.

Here, we used a recommendation system that is to categorise the user's clothing and suggest the best outfit for a particular occasion based on a recommendation algorithm. The suggested system demonstrates that it can analyse the user's attire from the images, determine the type and colour of the outfit, and then suggest the most appropriate outfit for the situation based on the user's current attire.

CHATBOT

Your Heleper Is Here.

Chatbots can also be used to gather visitor data, which can then be used to improve product recommendations and suggestions. You can personalise product pages and increase customer loyalty and affinity by having a thorough understanding of customer inquiries, needs, and preferences. Chatbots can also inform customers when an item is out of stock, suggest suitable alternatives based on their preferences, and let them know when their order is expected to arrive.

9 ADVANTAGES & DISADVANTAGES ADVANTAGES:

1. CUSTOMER SATISFACTION:

2. Customers frequently look at the products that were recommended to them during their previous browsing. mostly because they believe bigger prospects for quality products will present themselves. It would be beneficial if their browsing history from the prior session was available when they left the site and returned. This might aid and direct their e-Commerce endeavours in a similar manner as knowledgeable assistants at brick and mortar establishments. Client retention is a result of this kind of customer pleasure.

3. REVENUE:

There is less of a learning curve for online shoppers today because to years of research, experimentation, and implementation, mostly led by Amazon. Additionally, a wide range of algorithms have been investigated, put into practise, and shown to yield higher conversion rates than non-personalized product recommendations.

DISADVANTAGES:

1. LACK OF DATA

10 The fact that recommender systems require a lot of data in order to create recommendations is perhaps their largest problem. It's no accident that businesses like Google, Amazon, Netflix, and Last.fm, which have access to vast amounts of customer data, are those most associated with providing outstanding suggestions. A decent recommender system requires item data (from a catalogue or other form), user data (behavioural events), and then the magic algorithm does its thing, as shown in the slide below from Strands' talk at Recked. The likelihood of receiving useful recommendations increases with the amount of item and user data a recommender system has at its disposal.

11 CONCLUSION

The major method of communication, interest collection, and product recommendation used by the smart fashion recommender system is a chat bot. A chat bot's interaction with users is intended to enhance the user experience. Users do not have to go through different pages to find the right product. The system is set up to reduce the amount of time users spend looking for the right goods. Future improvements to the chatbot will allow for the addition of items to the shopping cart, the display of the contents of the cart, order history, and payment via the chatbot..

12 APPENDIX

Source 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">

  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
  <link href="//db.onlinewebfonts.com/c/d8a3c95906aec0c2483082a82e72cb40?family=WanderlustShine-Regular"
    rel="stylesheet" type="text/css" />



  <title>SMART FASHION RECOMMENDED APPLICATION</title>
  <style>
    @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap');
    @import url('https://fonts.googleapis.com/css2?family=Lobster&display=swap');

    @font-face {
      font-family: Wanderlust;
      src: url('/Wanderlust Letters-Font/OTF/WanderlustLetters-Regular.otf');
      font-weight: bold;
    }

    * {
      margin: 0;
      padding: 0;

      font-family: 'Poppins', sans-serif;
    }

    body {
      text-align: center;
      background-color: #23242a;
      color: #f2f2f2;
      background-image: url("https://wallpapercave.com/wp/wp6326202.jpg");
      background-attachment: fixed;
      background-size: cover;
    }



    .get {
```

```
    margin-bottom: 40%;
    margin-left: 38%;
    margin-top: 15%;
    width: 70%;
}

.about,
.cart,
.chatbot {
    margin: 2%;
    width: 90%;
    height: 300px;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
    background-color: #111111;
    transition: 0.3s;
    padding: 1%;
}

.txt {

    text-align: center;
    margin: 4% 5%;

}

.navbar {
    overflow: hidden;
    background-color: #151b54;
    position: fixed;
    top: 0;
    width: 100%;
    font-size: 3vh;
}

.navbar>a {
    float: right;
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 17px;
}

.navbar>a:hover {
    background: #ddd;
```

```
            color: black;
        }

        .title {
            float: left;
            padding: 12px 12px;
            color: #f2f2f2;
        }

        .con a {
            color: #f2f2f2;
            font-family: 'Poppins', sans-serif;
            font-weight: bolder;
            text-decoration: none;
            background: #111111;
            padding: 5px;
            border-radius: 10px;

        }

        .con a:hover {
            background: #f2f2f2;
            color: #111111;
            cursor: pointer;
            font-family: 'Poppins', sans-serif;
            font-weight: bolder;
        }

.container-fluid
{
width: 100%;
height: 100vh;
background: linear-gradient(to left, rgba(0,0,0,0.5)50%, rgba(0,0,0,0.5)50%), url(bg.jpg);
background-size: cover;
background-repeat: no-repeat;
}
.content
{
width: 1200px;
height: 100vh;
text-align: center;
margin: auto;
position: relative;
left: 350px;

}
```

```
.content h2
{
color: #fff;
padding-top: 20px;
font-size: 35px;
font-weight: 400;
position: relative;
top: 200px;
}
.content .btn
{
width: 170px;
height: 45px;
border-radius: 25px;
background-color: #e33;
color: #fff;
border: 2px solid #e33;
cursor: pointer;
font-weight: bold;
transition: 0.5s;
margin-top: 30px;
position: relative;
top: 200px;
}
.content .btn:hover
{
background-color: transparent;
color: #e33;
}

  </style>

</head>

<body>
  <div class="navbar">
    <a href="/logout">Logout </a>
    <a style="color: white; font-size: small;">{{email}}</a>
    <div class="title">
      <h3><a style="color: white; text-decoration: none;" href="/">Smart Fashion</a></h3>
    </div>
  </div>
  <div class="container-fluid">
    <div class="content">
      <h2>SMART FASHION RECOMMENDER APPLICATION</h2>
      <a href="/data"><button class="btn">VIEW MORE</button></a>
    </div>
</div>
<script type="module" src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
<script nomodule src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
<br>
<div class="about">
    <center>
        <h2>ABOUT</h2>
    </center>

    <div class="txt">
        <p>
        <h3 style="color:#45ffc1;">A warm welcome to your online fashion lane.</h3>

        We are aware of how much you value fashion and how much you enjoy online shopping's convenience. When you're
        on the go, the Smart Fashion online shopping app makes sure you don't miss out on a chic shopping
        experience. If you want to shop for clothing for men and women, shoes, accessories, or even the newest
        electronics and tech gadgets, Smart Fashion is always there to meet all of your style needs
        </p>
    </div>

</div>
<div class="cart">
    <center>
        <h2>RECOMMENDATION&numsp;SYSTEM</h2>
    </center>

    <div class="txt">
        <p>
        <h3 style="color:#45ffc1;">Your Favourite Items are here.</h3>

        Here, we used a recommendation system that is to categorise the user's clothing and suggest the best outfit
        for a particular occasion based on a recommendation algorithm. The suggested system demonstrates that it can
        analyse the user's attire from the images, determine the type and colour of the outfit, and then suggest the
        most appropriate outfit for the situation based on the user's current attire.
        </p>
    </div>

</div>
<div class="chatbot">
    <center>
        <h2>CHATBOT</h2>
    </center>

    <div class="txt">
        <p>
```

```
      <h3 style="color:#45ffc1;">Your Heleper Is Here.</h3>

      Chatbots can also be used to gather visitor data, which can then be used to improve product recommendations
      and suggestions. You can personalise product pages and increase customer loyalty and affinity by having a
      thorough understanding of customer inquiries, needs, and preferences. Chatbots can also inform customers
      when an item is out of stock, suggest suitable alternatives based on their preferences, and let them know
      when their order is expected to arrive.
      </p>
    </div>

  </div>
  <div class="devlopers"></div>
  <script>
    window.watsonAssistantChatOptions = {
      integrationID: "71a730ca-58bd-40be-8f3e-208a5db2545d", // The ID of this integration.
      region: "au-syd", // The region your integration is hosted in.
      serviceInstanceID: "9126d030-4dc5-4d4c-8c31-9c4a445bc485", // 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>

</html>

<!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">
  <title>Log in</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
  <style>
    @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap');
    @import url('https://fonts.googleapis.com/css2?family=Lobster&display=swap');

    * {
      margin: 0;
      padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}

.form {
    margin-top: 20px;
    height: 100vh;
    background: #28292d;
    z-index: 10;
    padding: 40px 40px;
    display: flex;
    flex-direction: column;
}

.form h2 {
    color: #ffffff;
    font-weight: 500;
    text-align: center;
    letter-spacing: 0.1em;
}

.inputBox {
    position: relative;
    width: 100%;
    margin-top: 35px;
}

.inputBox input {
    position: relative;
    width: 100%;
    padding: 10px 10px 10px;
    background: transparent;
    border: none;
    outline: none;
    color: white;
    font-size: 1em;
    letter-spacing: 0.05em;
    z-index: 10;
}

.inputBox span {
    position: absolute;
    left: 0;
    padding: 20px 0px 10px;
    font-size: 1em;
    color: white;
    pointer-events: none;
    letter-spacing: 0.05em;
    transition: 0.5s;
}

.inputBox i {
    position: absolute;
    left: 0;
    bottom: 0;
    width: 100%;
    height: 2px;
    background: #45ffc1;
    transition: 0.5s;
    pointer-events: none;
    z-index: 0;
}

.links {
    display: flex;
    justify-content: space-between;
}

.links>a {
    margin: 10px 0;
    font-size: 0.75em;
    color: #8f8f8f;
    text-decoration: none;
}

.links>a:hover,
.links>p a {
    color: #45ffc1;
    text-decoration: none;
}

input[type="submit"] {
    border: none;
    outline: none;
    background: #45ffc1;
    padding: 11px 25px;
    width: 100px;
    margin-top: 10px;
    border-radius: 4px;
    font-weight: 600;
    cursor: pointer;
}
```

```
input[type="submit"]:active {
    opacity: 0.8;
}



.font {
    font-family: 'Poppins', sans-serif;
    top: 0;
    margin-left: 120%;
    text-align: left;
}

.fa {
    color: aliceblue;



}

p {
    color: white;
}

.navbar {
    overflow: hidden;
    background-color: #151b54;
    position: fixed;
    top: 0;
    width: 100%;
    font-size: 3vh;
}

.navbar>a {
    float: right;
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 17px;
}

.navbar>a:hover {
    background: #ddd;
    color: black;
}
```

```
        .title {
            padding: 12px 12px;
            color: #f2f2f2;
        }
    </style>
</head>

<body>
    <div class="navbar">
        <a href="/login">SIGN IN </a>
        <a href="/register">SIGN UP</a>
        <a href="/adminlogin">ADMIN</a>
        <div class="title">
            <h3><a style="color: white; text-decoration: none;" href="/">Smart Fashion</a></h3>
        </div>
    </div>



    <form method="post">
        <div class="box">
            <div class="form">
                <h2>SIGN IN</h2>

                <div class="inputBox">


                    <input type="text" name="email" placeholder="Email" required>
                    <i></i>
                </div>

                <div class="inputBox">
                    <input type="password" name="password" placeholder="Password" required>
                    <i></i>
                </div>
                <div class="links">
                    <a href="#">Forget password</a>
                    <p><a href="/register">SIGN UP</a> or <a href="/adminlogin">Admin</a></a></p>
                </div>
                <input type="submit" value="Login">
            </div>
        </div>
    </form>

    <script>
        window.watsonAssistantChatOptions = {
            integrationID: "71a730ca-58bd-40be-8f3e-208a5db2545d", // The ID of this integration.
            region: "au-syd", // The region your integration is hosted in.
            serviceInstanceID: "9126d030-4dc5-4d4c-8c31-9c4a445bc485", // 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>

</html>

<!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">
    <title>Log in</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
</head>
<style>
    @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap');
    @import url('https://fonts.googleapis.com/css2?family=Lobster&display=swap');

    * {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
        font-family: 'Poppins', sans-serif;
    }

    .form {
        margin-top: 20px;
        height: 100vh;
        background: #28292d;
        z-index: 10;
        padding: 40px 40px;
        display: flex;
        flex-direction: column;
    }
```

```
.form h2 {
   color: #ffffff;
   font-weight: 500;
   text-align: center;
   letter-spacing: 0.1em;
}

.inputBox {
   position: relative;
   width: 100%;
   margin-top: 35px;
}

.inputBox input {
   position: relative;
   width: 100%;
   padding: 10px 10px 10px;
   background: transparent;
   border: none;
   outline: none;
   color: white;
   font-size: 1em;
   letter-spacing: 0.05em;
   z-index: 10;
}

.inputBox span {
   position: absolute;
   left: 0;
   padding: 20px 0px 10px;
   font-size: 1em;
   color: white;
   pointer-events: none;
   letter-spacing: 0.05em;
   transition: 0.5s;
}

.inputBox i {
   position: absolute;
   left: 0;
   bottom: 0;
   width: 100%;
   height: 2px;
   background: #45ffc1;
   transition: 0.5s;
   pointer-events: none;
```

```css
    z-index: 0;
}

.links {
    display: flex;
    justify-content: space-between;

}



.links>a {
    margin: 10px 0;
    font-size: 0.75em;
    color: #8f8f8f;
    text-decoration: none;
}

.links>a:hover,
.links>p a {
    color: #45ffc1;
    text-decoration: none;
}

input[type="submit"] {
    border: none;
    outline: none;
    background: #45ffc1;
    padding: 11px 25px;
    width: 100px;
    margin-top: 10px;
    border-radius: 4px;
    font-weight: 600;
    cursor: pointer;
}

input[type="submit"]:active {
    opacity: 0.8;
}



.font {
    font-family: 'Poppins', sans-serif;
    top: 0;
    margin-left: 120%;
    text-align: left;
}
```

```
    .fa {
        color: aliceblue;



    }

    .navbar {
        overflow: hidden;
        background-color: #151b54;
        position: fixed;
        top: 0;
        width: 100%;
        font-size: 3vh;
    }

    .navbar>a {
        float: right;
        display: block;
        color: #f2f2f2;
        text-align: center;
        padding: 14px 16px;
        text-decoration: none;
        font-size: 17px;
    }

    .navbar>a:hover {
        background: #ddd;
        color: black;
    }

    .title {
        padding: 12px 12px;
        color: #f2f2f2;
    }
</style>
</head>

<body>
    <div class="navbar">
        <a href="/login">SIGN IN </a>
        <a href="/register">SIGN UP</a>
        <a href="/adminlogin">ADMIN</a>
        <div class="title">
            <h3><a style="color: white; text-decoration: none;" href="/">Smart Fashion</a></h3>
        </div>
    </div>



    <form method="post">
        <div class="box">
            <div class="form">
                <h2>SIGN UP</h2>

                <div class="inputBox">


                    <input type="text" name="username" placeholder="Username" required>
                    <i></i>
                </div>
                <div class="inputBox">


                    <input type="email" name="email" placeholder="Email" required>
                    <i></i>
                </div>
                <div class="inputBox">


                    <input type="text" name="phoneno" placeholder="Phone Number" required>
                    <i></i>
                </div>
                <div class="inputBox">
                    <input type="password" name="password" placeholder="Password" required>
                    <i></i>
                </div>
                <div class="links">
                    <a href="#">Forget password</a>
                    <p style="color: white;"><a href="/login">Login</a> or <a href="/adminregister">Admin</a></p>
                </div>
                <input type="submit" value="Sign Up">
            </div>
        </div>
    </form>

</body>

</html>

<!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">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
<link href="//db.onlinewebfonts.com/c/d8a3c95906aec0c2483082a82e72cb40?family=WanderlustShine-Regular"
  rel="stylesheet" type="text/css" />



<title>SMART FASHION RECOMMENDED APPLICATION</title>
<style>
  @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap');
  @import url('https://fonts.googleapis.com/css2?family=Lobster&display=swap');

  @font-face {
    font-family: Wanderlust;
    src: url('/Wanderlust Letters-Font/OTF/WanderlustLetters-Regular.otf');
    font-weight: bold;
  }

  * {
    margin: 0;
    padding: 0;

    font-family: 'Poppins', sans-serif;
  }

  body {
    text-align: center;
    background-color: #23242a;
    color: #f2f2f2;
  }



  .get {
    margin-bottom: 40%;
    margin-left: 38%;
    margin-top: 15%;
    width: 70%;
  }

  .about,
  .cart,
  .chatbot {
    margin: 2%;
    width: 90%;
    height: 300px;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
    background-color: #111111;
    transition: 0.3s;
    padding: 1%;
}

.txt {


    text-align: center;
    margin: 4% 5%;


}

.navbar {
    overflow: hidden;
    background-color: #151b54;
    position: fixed;
    top: 0;
    width: 100%;
    font-size: 3vh;
}

.navbar>a {
    float: right;
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 17px;
}

.navbar>a:hover {
    background: #ddd;
    color: black;
}

.title {
    float: left;
    padding: 12px 12px;
    color: #f2f2f2;
}

.con a {
    color: #f2f2f2;
    font-family: 'Poppins', sans-serif;
    font-weight: bolder;
```

```
            text-decoration: none;
            background: #111111;
            padding: 5px;
            border-radius: 10px;

        }

        .con a:hover {
            background: #f2f2f2;
            color: #111111;
            cursor: pointer;
            font-family: 'Poppins', sans-serif;
            font-weight: bolder;
        }
        .card {
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
  max-width: 300px;
  margin: auto;
  text-align: center;
  font-family: arial;
}

.price {
  color: grey;
  font-size: 22px;
}

.card button {
  border: none;
  outline: 0;
  padding: 12px;
  color: white;
  background-color: #151b54;
  text-align: center;
  cursor: pointer;
  width: 100%;
  font-size: 18px;
}

.card button:hover {
  opacity: 0.7;
}
    </style>

</head>
```

```
<body>
    <div class="navbar">
        <a href="/logout">Logout </a>
        <div class="title">
            <h3><a style="color: white; text-decoration: none;" href="/">Smart Fashion</a></h3>
        </div>
    </div>
    <div style="display: flex; margin-top: 10%; flex-wrap: wrap; justify-content: center; align-items: center;">
    {% for item in shirts %}
    <div class="card" style="margin: 20px 50px;">
        <img src="{{item.IMAGE}}" style="width:250px; height: 350px;">
        <h1>{{item.NAME}}</h1>
        <p class="price">${{item.RATE}}</p>
        <p>CATEGORY: {{item.CATEGORIE}}</p>
        <p><button>Add to Cart</button></p>
    </div>
    {% endfor %}
    </div>
    <div style="display: flex; margin-top: 10%; flex-wrap: wrap; justify-content: center; align-items: center;">
    {% for item in pants %}
    <div class="card"  style="margin: 20px 50px;">
        <img src="{{item.IMAGE}}" style="width:250px; height: 350px;">
        <h1>{{item.NAME}}</h1>
        <p class="price">${{item.RATE}}</p>
        <p>CATEGORY: {{item.CATEGORIE}}</p>
        <p><button>Add to Cart</button></p>
    </div>
    <div style="display: flex; margin-top: 10%; flex-wrap: wrap; justify-content: center; align-items: center;">
        {% for item in shoes %}
        <div class="card" style="margin: 20px 50px;">
            <img src="{{item.IMAGE}}" style="width:250px; height: 350px;">
            <h1>{{item.NAME}}</h1>
            <p class="price">${{item.RATE}}</p>
            <p>CATEGORY: {{item.CATEGORIE}}</p>
            <p><button>Add to Cart</button></p>
        </div>
        {% endfor %}
    </div>
    {% endfor %}
    </div>
    <div style="display: flex; margin-top: 10%; flex-wrap: wrap; justify-content: center; align-items: center;">
    {% for item in watchs %}
    <div class="card"  style="margin: 20px 50px;">
        <img src="{{item.IMAGE}}" style="width:250px; height: 350px;">
        <h1>{{item.NAME}}</h1>
        <p class="price">${{item.RATE}}</p>
        <p>CATEGORY: {{item.CATEGORIE}}</p>
        <p><button>Add to Cart</button></p>
    </div>
    {% endfor %}
    </div>
    <script>
        window.watsonAssistantChatOptions = {
            integrationID: "71a730ca-58bd-40be-8f3e-208a5db2545d", // The ID of this integration.
            region: "au-syd", // The region your integration is hosted in.
            serviceInstanceID: "9126d030-4dc5-4d4c-8c31-9c4a445bc485", // 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>

</html>

<!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">
    <title>Log in</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
    <style>
        @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap');
        @import url('https://fonts.googleapis.com/css2?family=Lobster&display=swap');

        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: 'Poppins', sans-serif;
        }

        .form {
            margin-top: 20px;
            height: 100vh;
    background: #28292d;
    z-index: 10;
    padding: 40px 40px;
    display: flex;
    flex-direction: column;
}

.form h2 {
    color: #ffffff;
    font-weight: 500;
    text-align: center;
    letter-spacing: 0.1em;
}

.inputBox {
    position: relative;
    width: 100%;
    margin-top: 35px;
}

.inputBox input {
    position: relative;
    width: 100%;
    padding: 10px 10px 10px;
    background: transparent;
    border: none;
    outline: none;
    color: white;
    font-size: 1em;
    letter-spacing: 0.05em;
    z-index: 10;
}

.inputBox span {
    position: absolute;
    left: 0;
    padding: 20px 0px 10px;
    font-size: 1em;
    color: white;
    pointer-events: none;
    letter-spacing: 0.05em;
    transition: 0.5s;
}

.inputBox i {
    position: absolute;
    left: 0;
    bottom: 0;
    width: 100%;
    height: 2px;
    background: #45ffc1;
    transition: 0.5s;
    pointer-events: none;
    z-index: 0;
}

.links {
    display: flex;
    justify-content: space-between;

}

.links a {
    margin: 10px 0;
    font-size: 0.75em;
    color: #8f8f8f;
    text-decoration: none;
}

.links a:hover,
.links a:nth-child(2) {
    color: #45ffc1;
}

input[type="submit"] {
    border: none;
    outline: none;
    background: #45ffc1;
    padding: 11px 25px;
    width: 100px;
    margin-top: 10px;
    border-radius: 4px;
    font-weight: 600;
    cursor: pointer;
}

input[type="submit"]:active {
    opacity: 0.8;
}


.font {
```

```
            font-family: 'Poppins', sans-serif;
            top: 0;
            margin-left: 120%;
            text-align: left;
        }

        .fa {
            color: aliceblue;



        }

        .navbar {
            overflow: hidden;
            background-color: #151b54;
            position: fixed;
            top: 0;
            width: 100%;
            font-size: 3vh;
        }

        .navbar a {
            float: right;
            display: block;
            color: #f2f2f2;
            text-align: center;
            padding: 14px 16px;
            text-decoration: none;
            font-size: 17px;
        }

        .navbar a:hover {
            background: #ddd;
            color: black;
        }

        .title {
            padding: 12px 12px;
            color: #f2f2f2;
        }
    </style>
</head>

<body>
    <div class="navbar">
        <a href="/login">SIGN IN </a>
        <a href="/register">SIGN UP</a>
        <a href="/adminlogin">ADMIN</a>
        <div class="title">
            <h3>Smart Fashion</h3>
        </div>
    </div>



    <form method="post">
        <div class="box">
            <div class="form">
                <h2>ADMIN SIGN IN</h2>

                <div class="inputBox">


                    <input type="text" name="email" placeholder="Email" required>
                    <i></i>
                </div>

                <div class="inputBox">
                    <input type="password" name="password" placeholder="Password" required>
                    <i></i>
                </div>
                <div class="links">
                    <a href="#">Forget password</a>
                    <a href="/admin">SIGN UP</a>
                </div>
                <input type="submit" value="Login">
            </div>
        </div>
    </form>
    <script>
        window.watsonAssistantChatOptions = {
            integrationID: "71a730ca-58bd-40be-8f3e-208a5db2545d", // The ID of this integration.
            region: "au-syd", // The region your integration is hosted in.
            serviceInstanceID: "9126d030-4dc5-4d4c-8c31-9c4a445bc485", // 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>
```

```
</html>

<!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">
    <title>Log in</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
</head>
<style>
    @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap');
    @import url('https://fonts.googleapis.com/css2?family=Lobster&display=swap');

    * {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
        font-family: 'Poppins', sans-serif;
    }

    .form {
        margin-top: 20px;
        height: 100vh;
        background: #28292d;
        z-index: 10;
        padding: 40px 40px;
        display: flex;
        flex-direction: column;
    }

    .form h2 {
        color: #ffffff;
        font-weight: 500;
        text-align: center;
        letter-spacing: 0.1em;
    }

    .inputBox {
        position: relative;
        width: 100%;
        margin-top: 35px;
    }
```

```
.inputBox input {
    position: relative;
    width: 100%;
    padding: 10px 10px 10px;
    background: transparent;
    border: none;
    outline: none;
    color: white;
    font-size: 1em;
    letter-spacing: 0.05em;
    z-index: 10;
}

.inputBox span {
    position: absolute;
    left: 0;
    padding: 20px 0px 10px;
    font-size: 1em;
    color: white;
    pointer-events: none;
    letter-spacing: 0.05em;
    transition: 0.5s;
}

.inputBox i {
    position: absolute;
    left: 0;
    bottom: 0;
    width: 100%;
    height: 2px;
    background: #45ffc1;
    transition: 0.5s;
    pointer-events: none;
    z-index: 0;
}

.links {
    display: flex;
    justify-content: space-between;

}

.links a {
    margin: 10px 0;
    font-size: 0.75em;
```

```css
    color: #8f8f8f;
    text-decoration: none;
}

.links a:hover,
.links a:nth-child(2) {
    color: #45ffc1;
}

input[type="submit"] {
    border: none;
    outline: none;
    background: #45ffc1;
    padding: 11px 25px;
    width: 100px;
    margin-top: 10px;
    border-radius: 4px;
    font-weight: 600;
    cursor: pointer;
}

input[type="submit"]:active {
    opacity: 0.8;
}



.font {
    font-family: 'Poppins', sans-serif;
    top: 0;
    margin-left: 120%;
    text-align: left;
}

.fa {
    color: aliceblue;



}

.navbar {
    overflow: hidden;
    background-color: #151b54;
    position: fixed;
    top: 0;
    width: 100%;
    font-size: 3vh;
    }

    .navbar a {
        float: right;
        display: block;
        color: #f2f2f2;
        text-align: center;
        padding: 14px 16px;
        text-decoration: none;
        font-size: 17px;
    }

    .navbar a:hover {
        background: #ddd;
        color: black;
    }

    .title {
        padding: 12px 12px;
        color: #f2f2f2;
    }
</style>
</head>

<body>
    <div class="navbar">
        <a href="/login">SIGN IN </a>
        <a href="/register">SIGN UP</a>
        <a href="/adminlogin">ADMIN</a>
        <div class="title">
            <h3>Smart Fashion</h3>
        </div>
    </div>



    <form method="post">
        <div class="box">
            <div class="form">
                <h2> ADMIN SIGN UP</h2>

                <div class="inputBox">


                    <input type="text" name="username" placeholder="Username" required>
                    <i></i>
                </div>
                <div class="inputBox">
                    <input type="email" name="email" placeholder="Email" required>
                    <i></i>
                </div>
                <div class="inputBox">

                    <input type="text" name="phoneno" placeholder="Phone Number" required>
                    <i></i>
                </div>
                <div class="inputBox">
                    <input type="password" name="password" placeholder="Password" required>
                    <i></i>
                </div>
                <div class="links">
                    <a href="#">Forget password</a>
                    <a href="/login">Login</a>
                </div>
                <input type="submit" value="Sign Up">
            </div>
        </div>
    </form>
    <script>
        window.watsonAssistantChatOptions = {
            integrationID: "71a730ca-58bd-40be-8f3e-208a5db2545d", // The ID of this integration.
            region: "au-syd", // The region your integration is hosted in.
            serviceInstanceID: "9126d030-4dc5-4d4c-8c31-9c4a445bc485", // 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>

</html>

 <!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">
```

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
<link href="//db.onlinewebfonts.com/c/d8a3c95906aec0c2483082a82e72cb40?family=WanderlustShine-Regular"
  rel="stylesheet" type="text/css" />


<title>SMART FASHION RECOMMENDED APPLICATION</title>
<style>
  @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap');
  @import url('https://fonts.googleapis.com/css2?family=Lobster&display=swap');

  @font-face {
    font-family: Wanderlust;
    src: url('/Wanderlust Letters-Font/OTF/WanderlustLetters-Regular.otf');
    font-weight: bold;
  }

  * {
    margin: 0;
    padding: 0;

    font-family: 'Poppins', sans-serif;
  }

  body {
    text-align: center;
    background-color: #23242a;
    color: #f2f2f2;
  }



  .get {
    margin-bottom: 40%;
    margin-left: 38%;
    margin-top: 15%;
    width: 70%;
  }

  .about,
  .cart,
  .chatbot {
    margin: 2%;
    width: 90%;
    height: 300px;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
    background-color: #111111;
    transition: 0.3s;
    padding: 1%;
}

.txt {


    text-align: center;
    margin: 4% 5%;

}

.navbar {
    overflow: hidden;
    background-color: #151b54;
    position: fixed;
    top: 0;
    width: 100%;
    font-size: 3vh;
}

.navbar>a {
    float: right;
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 17px;
}

.navbar>a:hover {
    background: #ddd;
    color: black;
}

.title {
    float: left;
    padding: 12px 12px;
    color: #f2f2f2;
}

.con a {
    color: #f2f2f2;
    font-family: 'Poppins', sans-serif;
    font-weight: bolder;
    text-decoration: none;
    background: #111111;
    padding: 5px;
    border-radius: 10px;

}

.con a:hover {
    background: #f2f2f2;
    color: #111111;
    cursor: pointer;
    font-family: 'Poppins', sans-serif;
    font-weight: bolder;
}

.card {
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
    max-width: 300px;
    margin: auto;
    text-align: center;
    font-family: arial;
}

.price {
    color: grey;
    font-size: 22px;
}

.card button {
    border: none;
    outline: 0;
    padding: 12px;
    color: white;
    background-color: #000;
    text-align: center;
    cursor: pointer;
    width: 100%;
    font-size: 18px;
}

.card button:hover {
    opacity: 0.7;
}
input {
    position: relative;
    width: 100%;
    padding: 10px 10px 10px;
    background: transparent;
```

```
            border: none;
            outline: none;
            color: black;
            background-color: rgb(204, 204, 204);
            font-size: 1em;
            letter-spacing: 0.05em;
            z-index: 10;
            margin-bottom: 10px;
        }

        input[type="submit"] {
            border: none;
            outline: none;
            background: #45ffc1;
            padding: 11px 25px;
            width: 100px;
            margin-top: 10px;
            border-radius: 4px;
            font-weight: 600;
            cursor: pointer;
        }

        input[type="submit"]:active {
            opacity: 0.8;
        }
    </style>

</head>

<body>
    <div class="navbar">
        <a href="/logout">Logout </a>
        <div class="title">
            <h3><a style="color: white; text-decoration: none;" href="/">Smart Fashion</a></h3>
        </div>
    </div>
    <div style="display: flex; margin-top: 10%;justify-content: center; align-items: center;">
        <form method="post">
            <h2>ADD PRODUCT</h2>

            <input type="text" name="name" placeholder="Name" required>

            <input type="text" name="image" placeholder="ImageURL" required>

            <input type="text" name="rate" placeholder="Rate" required>
            <label for="CATEGORY">CATEGORY:</label>
            <select name="categorie" id="CATEGORY">
                <option value="shirt">Shirt</option>
                <option value="pant">Pant</option>
                <option value="watch">Watch</option>
                <option value="shoe">Shoe</option>
            </select>
            <br>
            <input type="submit" value="ADD">
        </form>
    </div>
    <script>
        window.watsonAssistantChatOptions = {
            integrationID: "71a730ca-58bd-40be-8f3e-208a5db2545d", // The ID of this integration.
            region: "au-syd", // The region your integration is hosted in.
            serviceInstanceID: "9126d030-4dc5-4d4c-8c31-9c4a445bc485", // 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>

</html>

import configparser
import ssl
from sendgrid.helpers.mail import Mail
from sendgrid import SendGridAPIClient
import secrets
from turtle import title
from unicodedata import category
from flask import Flask, render_template, request, redirect, url_for, session
import ibm_db
import bcrypt
import base64
import os

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=824dfd4d-99de-440d-9991-
629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30119;SECURITY=SSL;
SSLServerCertificateDigiCertGlobalRootCA.crt;PROTOCOL=TCPIP;UID=yxn13720;PWD=46gVfLcYJP6WedPZ;", "", "")
```

```
app = Flask(__name__)
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'

ssl._create_default_https_context = ssl._create_unverified_context

config = configparser.ConfigParser()
config.read("config.ini")

try:
    settings = config["SETTINGS"]
except:
    settings = {}

API = settings.get("APIKEY", None)
from_email = settings.get("FROM", None)
to_email = settings.get("TO", None)
subject = "Smart Fashion"
html_content = 'Fashion Prod'
print(API)



def sendMail(API, from_email, to_email, subject, html_content):
    if API != None and from_email != None and len(to_email) > 0:
        message = Mail(from_email, to_email, subject, html_content)
    try:
        sg = SendGridAPIClient(API)
        response = sg.send(message)
        print(response.status_code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e)



sendMail(API=API, from_email=from_email, to_email=to_email,
         subject=subject, html_content=html_content)



@app.route("/", methods=['GET'])
def home():
    if 'email' not in session:
        return redirect(url_for('login'))
    else:
        email = session.get('email')
    return render_template('home.html', email=email)
```

```
@app.route("/register", methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        phoneno = request.form['phoneno']
        password = request.form['password']

        if not username or not email or not phoneno or not password:
            return render_template('register.html', error='Please fill all fields')
        hash = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())
        query = "SELECT * FROM user_detail WHERE email=? OR phoneno=?"
        stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, phoneno)
        ibm_db.execute(stmt)
        isUser = ibm_db.fetch_assoc(stmt)
        if not isUser:
            insert_sql = "INSERT INTO user_detail(username, email, phoneno, password) 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, phoneno)
            ibm_db.bind_param(prep_stmt, 4, hash)
            ibm_db.execute(prep_stmt)
            return render_template('register.html', success="You can login")
        else:
            return render_template('register.html', error='Invalid Credentials')

    return render_template('register.html', name='Home')



@app.route("/login", methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

        if not email or not password:
            return render_template('login.html', error='Please fill all fields')
        query = "SELECT * FROM user_detail WHERE email=?"
        stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        isUser = ibm_db.fetch_assoc(stmt)
        print(isUser, password)

        if not isUser:
            return render_template('login.html', error='Invalid Credentials')

        isPasswordMatch = bcrypt.checkpw(password.encode(
            'utf-8'), isUser['PASSWORD'].encode('utf-8'))

        if not isPasswordMatch:
            return render_template('login.html', error='Invalid Credentials')

        session['email'] = isUser['EMAIL']
        return redirect(url_for('home'))

    return render_template('login.html', name='Home')


@app.route("/admin", methods=['GET', 'POST'])
def adregister():
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        phoneno = request.form['phoneno']
        password = request.form['password']

        if not username or not email or not phoneno or not password:
            return render_template('adminregister.html', error='Please fill all fields')
        hash = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())
        query = "SELECT * FROM admin_detail WHERE email=? OR phoneno=?"
        stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, phoneno)
        ibm_db.execute(stmt)
        isUser = ibm_db.fetch_assoc(stmt)
        if not isUser:
            insert_sql = "INSERT INTO admin_detail(username, email, phoneno, password) 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, phoneno)
            ibm_db.bind_param(prep_stmt, 4, hash)
            ibm_db.execute(prep_stmt)
            return render_template('adminregister.html', success="You can login")
        else:
            return render_template('adminregister.html', error='Invalid Credentials')
    return render_template('adminregister.html', name='Home')



@app.route("/adminlogin", methods=['GET', 'POST'])
def adlogin():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

        if not email or not password:
            return render_template('adminlogin.html', error='Please fill all fields')
        query = "SELECT * FROM admin_detail WHERE email=?"
        stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        isUser = ibm_db.fetch_assoc(stmt)
        print(isUser, password)

        if not isUser:
            return render_template('adminlogin.html', error='Invalid Credentials')

        isPasswordMatch = bcrypt.checkpw(password.encode(
            'utf-8'), isUser['PASSWORD'].encode('utf-8'))

        if not isPasswordMatch:
            return render_template('adminlogin.html', error='Invalid Credentials')

        session['email'] = isUser['EMAIL']
        return redirect(url_for('addproduct'))

    return render_template('adminlogin.html', name='Home')



@app.route("/addproduct", methods=['get', 'post'])
def addproduct():
    if request.method == 'POST':
        name = request.form['name']
        image = request.form['image']
        rate = request.form['rate']
        categorie = request.form['categorie']
        if categorie == 'shirt':
            insert_sql = "INSERT INTO SHIRT (name, image, categorie,rate) VALUES (?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, categorie)
            ibm_db.bind_param(prep_stmt, 4, rate)
            ibm_db.execute(prep_stmt)
        if categorie == 'pant':
            insert_sql = "INSERT INTO PANT(name, image, categorie,rate) VALUES (?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, categorie)
            ibm_db.bind_param(prep_stmt, 4, rate)
            ibm_db.execute(prep_stmt)
        if categorie == 'watch':
            insert_sql = "INSERT INTO WATCH(name, image, categorie, rate) VALUES (?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, categorie)
            ibm_db.bind_param(prep_stmt, 4, rate)
            ibm_db.execute(prep_stmt)
        if categorie == 'shoe':
            insert_sql = "INSERT INTO SHOE(name, image, categorie, rate) VALUES (?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, categorie)
            ibm_db.bind_param(prep_stmt, 4, rate)
            ibm_db.execute(prep_stmt)

    return render_template('addproducts.html', success="You can login")


@app.route("/data")
def display():
    shirt_list = []
    pant_list = []
    watch_list = []
    shoes_list = []

    # selecting_shirt
    sql = "SELECT * FROM SHIRT"
    stmt = ibm_db.exec_immediate(conn, sql)
    shirt = ibm_db.fetch_both(stmt)
    while shirt != False:
        shirt_list.append(shirt)
        shirt = ibm_db.fetch_both(stmt)

   # selecting_pant
```

```
    sql1 = "SELECT * FROM PANT"
    stmt1 = ibm_db.exec_immediate(conn, sql1)
    pant = ibm_db.fetch_both(stmt1)
    while pant != False:
        pant_list.append(pant)
        pant = ibm_db.fetch_both(stmt1)

# selecting_watch
    sql2 = "SELECT * FROM WATCH"
    stmt2 = ibm_db.exec_immediate(conn, sql2)
    watch = ibm_db.fetch_both(stmt2)
    while watch != False:
        watch_list.append(watch)
        watch = ibm_db.fetch_both(stmt2)

    # selecting_shoes
    sql3 = "SELECT * FROM SHOE"
    stmt3 = ibm_db.exec_immediate(conn, sql3)
    shoes = ibm_db.fetch_both(stmt3)
    while shoes != False:
        shoes_list.append(shoes)
        shoes = ibm_db.fetch_both(stmt3)

    # returning to HTML
    return render_template('data.html', shirts=shirt_list, pants=pant_list, watchs=watch_list, shoes=shoes_list)



@app.route('/logout')
def logout():
    session.pop('email', None)
    return redirect(url_for('login'))



if __name__ == '__main__':
    port = int(os.environ.get('PORT', 5000))
    app.run(port=port, host='0.0.0.0')
```

GitHub & Project Demo Link

https://drive.google.com/file/d/1z5u5O36DVAhc3YKCCDI2XM3BIzPZQwdi/view