RMK COLLEGE OF ENGINEERING AND TECHNOLOGY

SMART FASHION RECOMMENDER APPLICATION

TEAM ID: PNT2022TMID14283

Team Members:

- JEEVANANDHAM B.M (111619104047)
- DEEPAK KUMAR .A (111619104018)
- JUROO J.S (111619104049)
- KISHORE .R (111619104060)
- KARTHIK .M (111619104053)

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.

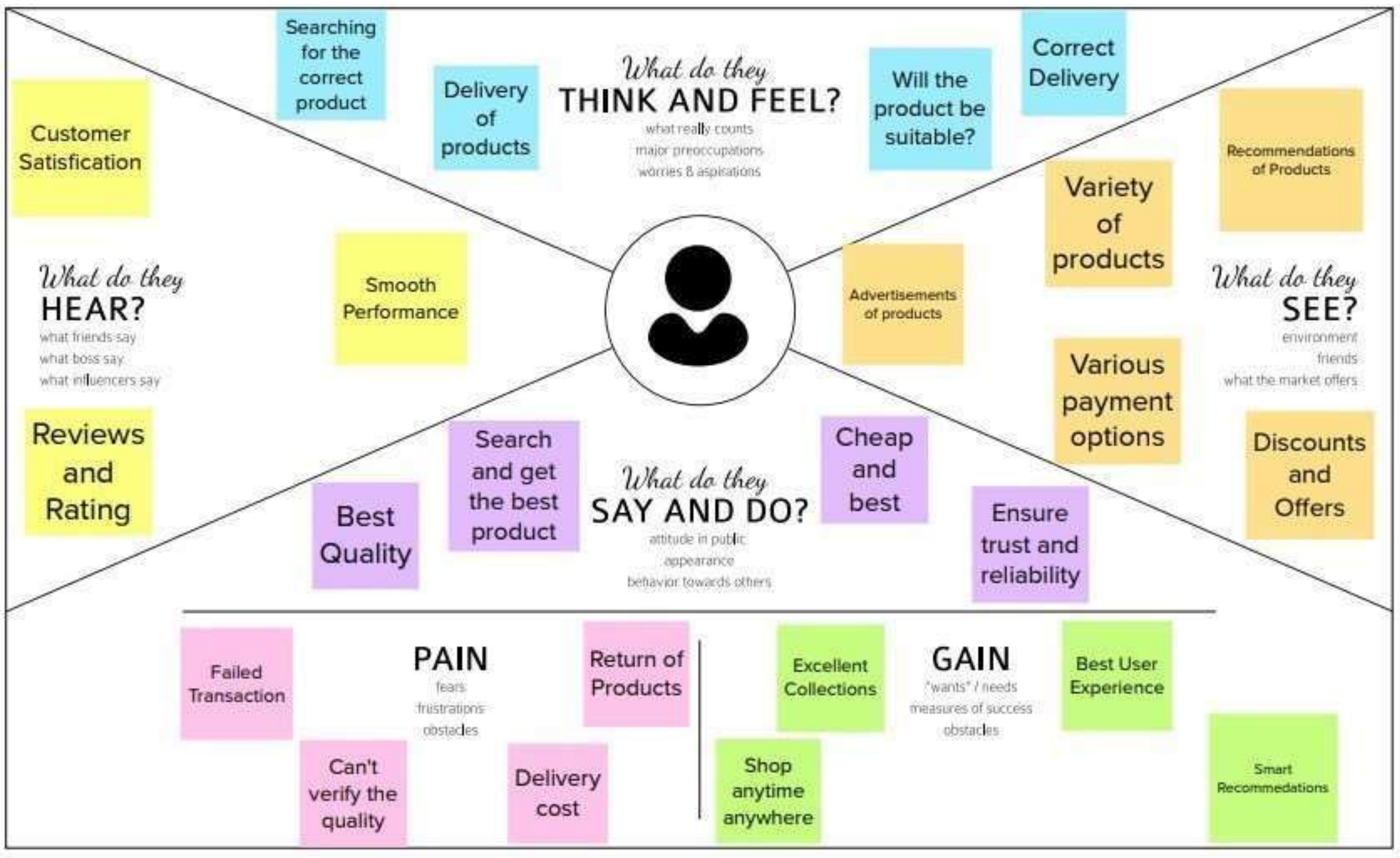
2.2 Problem Statement Definition:

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

3.3 Proposed Solution

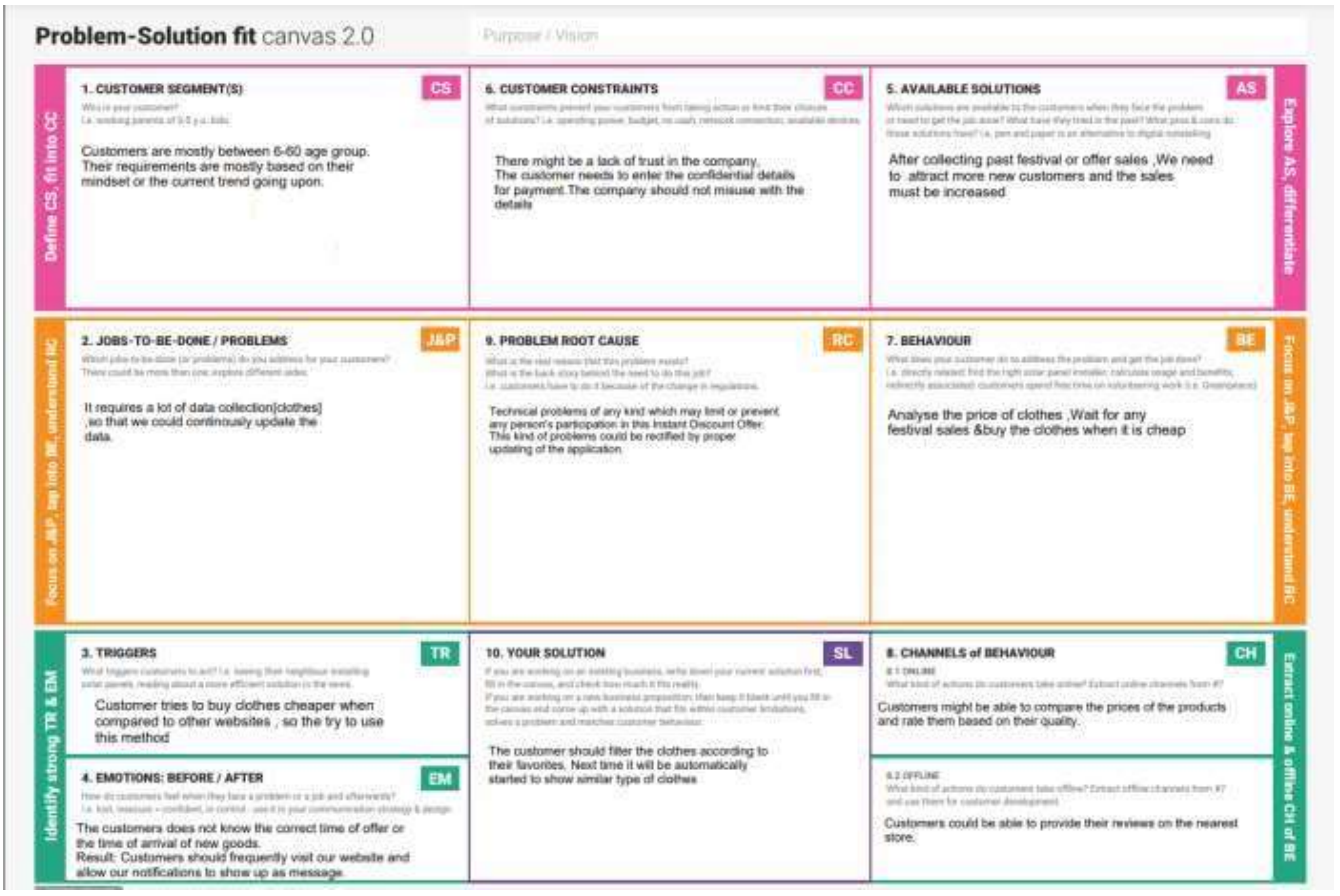
S.No.		Parameter	Description

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 Mod el (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



Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Customers are mostly between 6-60 age group. Their requirements are mostly based on their mindset or the current trend going upon.

6. CUSTOMER CONSTRAINTS CC

There might be a lack of trust in the company. The customer needs to enter the confidential details for payment.The company should not misuse with the details

5. AVAILABLE SOLUTIONS AS

After collecting past festival or offer sales ,We need to attract more new customers and the sales must be increased

Explore AS, differentiate

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

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

It requires a lot of data collection[clothes] ,so that we could continously update the data.

9. PROBLEM ROOT CAUSE RC

Technical problems of any kind which may limit or prevent any person's participation in this Instant Discount Offer. This kind of problems could be rectified by proper updating of the application.

7. BEHAVIOUR BE

Analyse the price of clothes ,Wait for any festival sales &buy the clothes when it is cheap

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

Identify strong TR & EM

3. TRIGGERS TR

Customer tries to buy clothes cheaper when compared to other websites , so the try to use this method

4. EMOTIONS: BEFORE / AFTER EM

The customers does not know the correct time of offer or the time of arrival of new goods.
Result: Customers should frequently visit our website and allow our notifications to show up as message.

10. YOUR SOLUTION SL

The customer should filter the clothes according to their favorites. Next time it will be automatically started to show similar type of clothes

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

Customers might be able to compare the prices of the products and rate them based on their quality.

8.2 OFFLINE

Customers could be able to provide their reviews on the nearest store.

Extract online & offline CH of BE

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	**Functional requirement**	**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

4.2 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

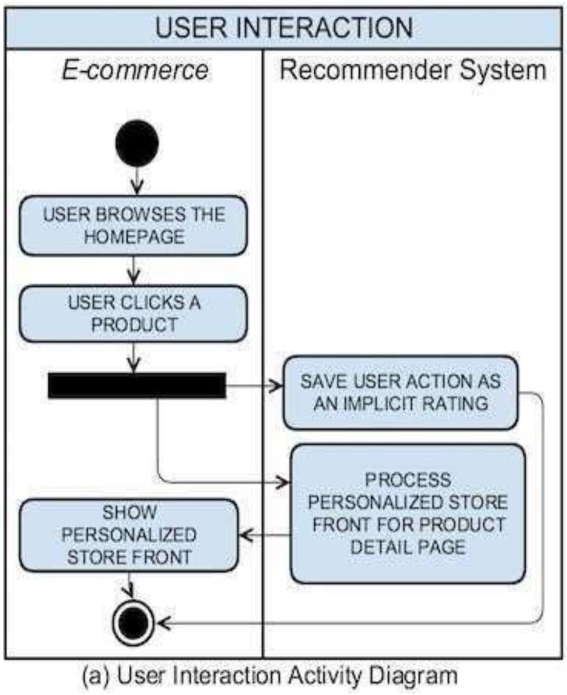
5.1 Data Flow Diagrams:

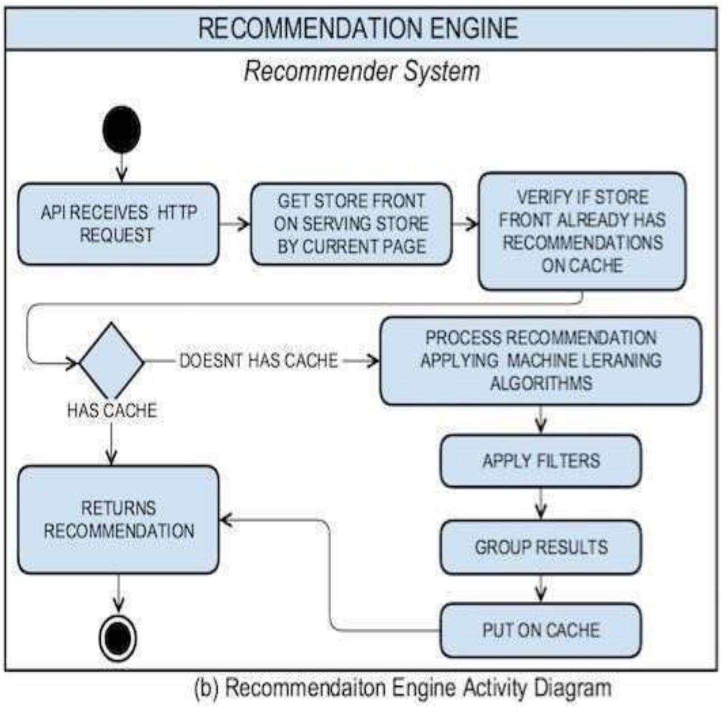
FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	a chatbot and user-friendly UI for effective usability
NFR-2	**Security**	A protected connection Requests and answers should be transmitted via HTTPS.
NFR-3	**Reliability**	To prevent programme termination, the system should manage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

5.2 Solution & Technical Architecture:

ARCHITECTURE DIAGRAM

User Interaction Activity Diagram Recommendation Engine Activity Diagram


USER INTERACTION
E-commerce
Recommender System
USER BROWSES THE HOMEPAGE
USER CLICKS A PRODUCT
SAVE USER ACTION AS AN IMPLICIT RATING
PROCESS PERSONALIZED STORE FRONT FOR PRODUCT DETAIL PAGE
SHOW PERSONALIZED STORE FRONT
(a) User Interaction Activity Diagram


RECOMMENDATION ENGINE
Recommender System
API RECEIVES HTTP REQUEST
GET STORE FRONT ON SERVING STORE BY CURRENT PAGE
VERIFY IF STORE FRONT ALREADY HAS RECOMMENDATIONS ON CACHE
DOESNT HAS CACHE
HAS CACHE
PROCESS RECOMMENDATION APPLYING MACHINE LERANING ALGORITHMS
APPLY FILTERS
GROUP RESULTS
PUT ON CACHE
RETURNS RECOMMENDATION
(b) Recommendaiton Engine Activity Diagram

The proposed system is divided into Four parts:

- Image pre-processing
- Recommendation Engine
- Web Scraping
- Web App

Image pre-processing: Image processing is the process of using computer algorithms to perform manipulations on digital images. The main goal of image processing is to refine the image data by removing the distorted noise and by enhancing image pixels. An image is nothing more than a two-dimensional array of number between 0 and 255. It is defined by the mathematical function f(x,y) where x and y are the two co- ordinates horizontally and vertically. The value of f(x,y) at any point is giving the pixel value at that point of an image. The steps to pre-process the image are as follows

Recommendation Engine:

A recommendation engine filters the information using different algorithms and recommends the relevant items to users. It first captures thepast behaviour of a customer and recommends products which the users might be likely to buy. The working of recommendation engine is as follows:

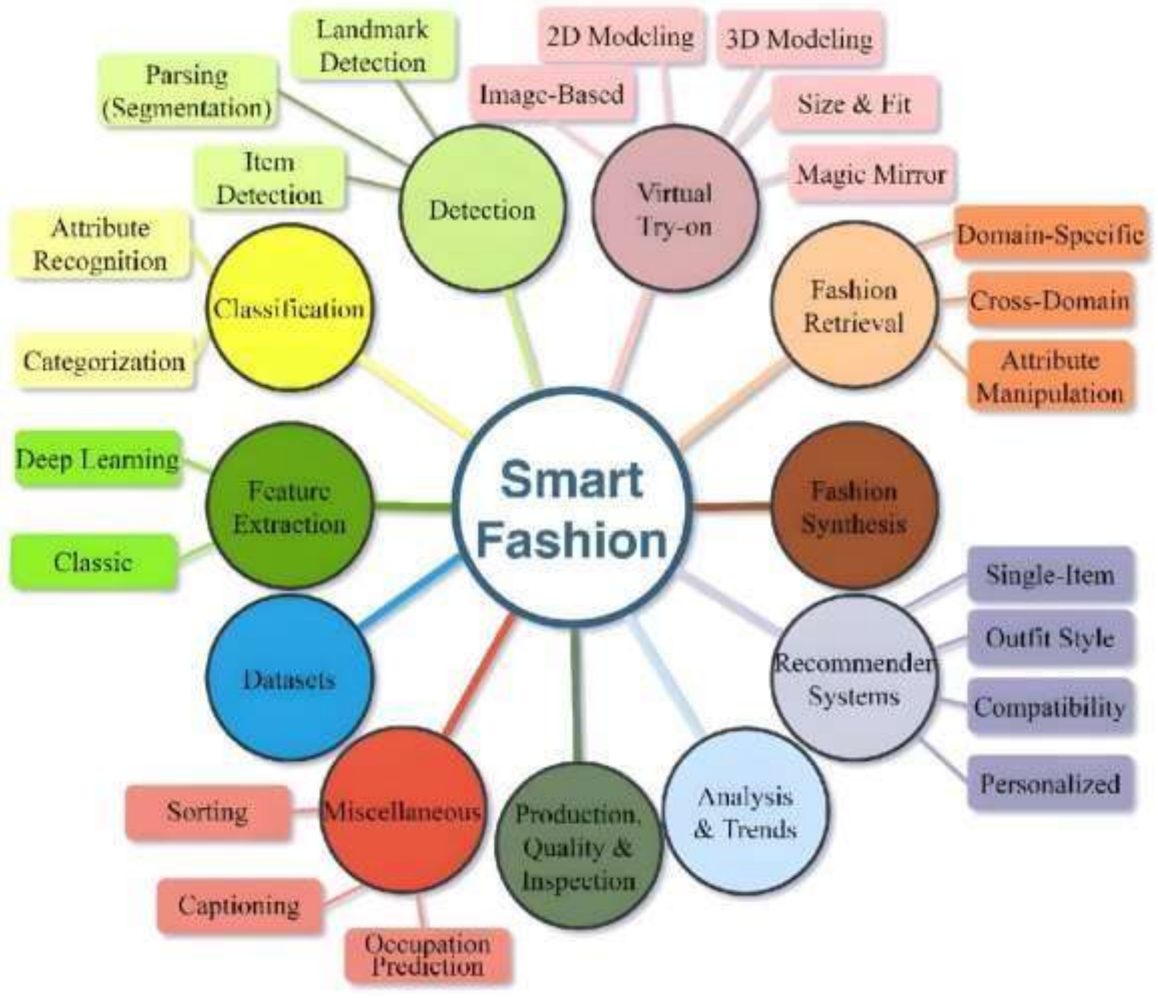
- **Collection of Data**
- **Analzying of Data**

Web Scraping:

Web scraping is the process of automating the process of data extraction in a fast and efficient manner.It
implements the use of crawlers or robots that automaticallyscan specific pages on a website and extract the required information. For the extraction of product data on a large scale, we implement a piece of code (called a 'web scraper') that requests a particular product page on an e-commerce website. In return, the website replies with the requested web page. Once the page is received, the scraper will parse its HTML code and extract relevant data from it. When the data extraction process is completed, the tool finally converts the data into the desired format. Now, since the web scraper is anautomated program, it can repeat this process thousands of times on a large number of product pages, and across several e- commerce websites.

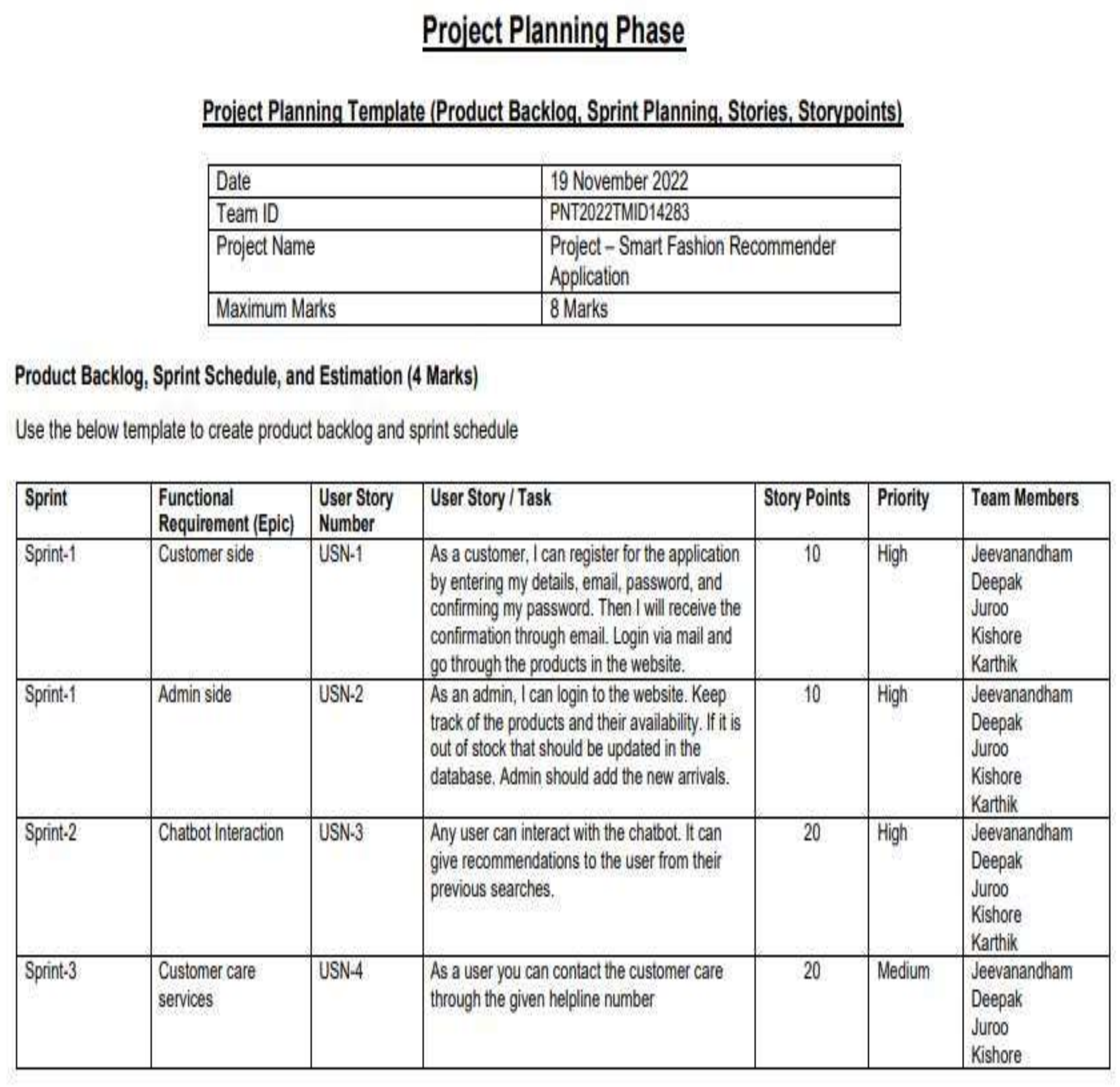
Web App:

Front End Design: Vue.js framework is used to create ainteractive interface for the web app. Back End Design: Flask framework is used to create aRESTful API.



6 PROJECT PLANNING & SCHEDULING

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



Project Planning Phase

Project Planning Template (Product Backlog, Sprint Planning, Stories, Storypoints)

Date	19 November 2022
Team ID	PNT2022TMID14283
Project Name	Project – Smart Fashion Recommender Application
Maximum Marks	8 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Customer side	USN-1	As a customer, I can register for the application by entering my details, email, password, and confirming my password. Then I will receive the confirmation through email. Login via mail and go through the products in the website.	10	High	Jeevanandham Deepak Juroo Kishore Karthik
Sprint-1	Admin side	USN-2	As an admin, I can login to the website. Keep track of the products and their availability. If it is out of stock that should be updated in the database. Admin should add the new arrivals.	10	High	Jeevanandham Deepak Juroo Kishore Karthik
Sprint-2	Chatbot Interaction	USN-3	Any user can interact with the chatbot. It can give recommendations to the user from their previous searches.	20	High	Jeevanandham Deepak Juroo Kishore Karthik
Sprint-3	Customer care services	USN-4	As a user you can contact the customer care through the given helpline number	20	Medium	Jeevanandham Deepak Juroo Kishore

Sprint-4	Feedback, Ratings.	USN-5	User can give their purchase experience through providing feedback, ratings etc.	20	High	Jeevanandham Deepak Juroo Kishore Karthik

Project Tracker, Velocity & Burndown Chart: (4 Marks)

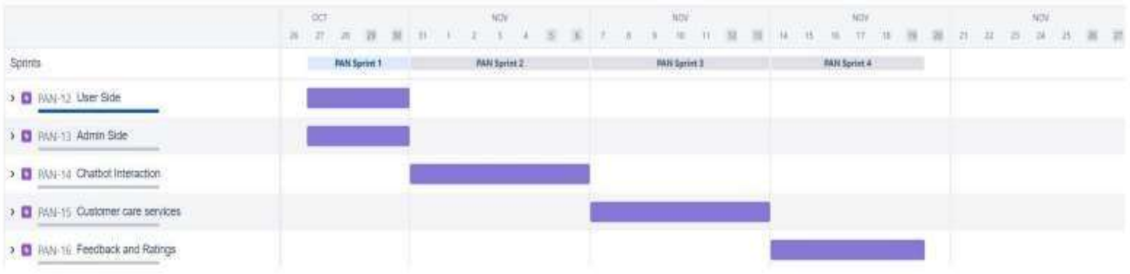
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	15 Nov 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	17 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	19 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Burndown Chart:



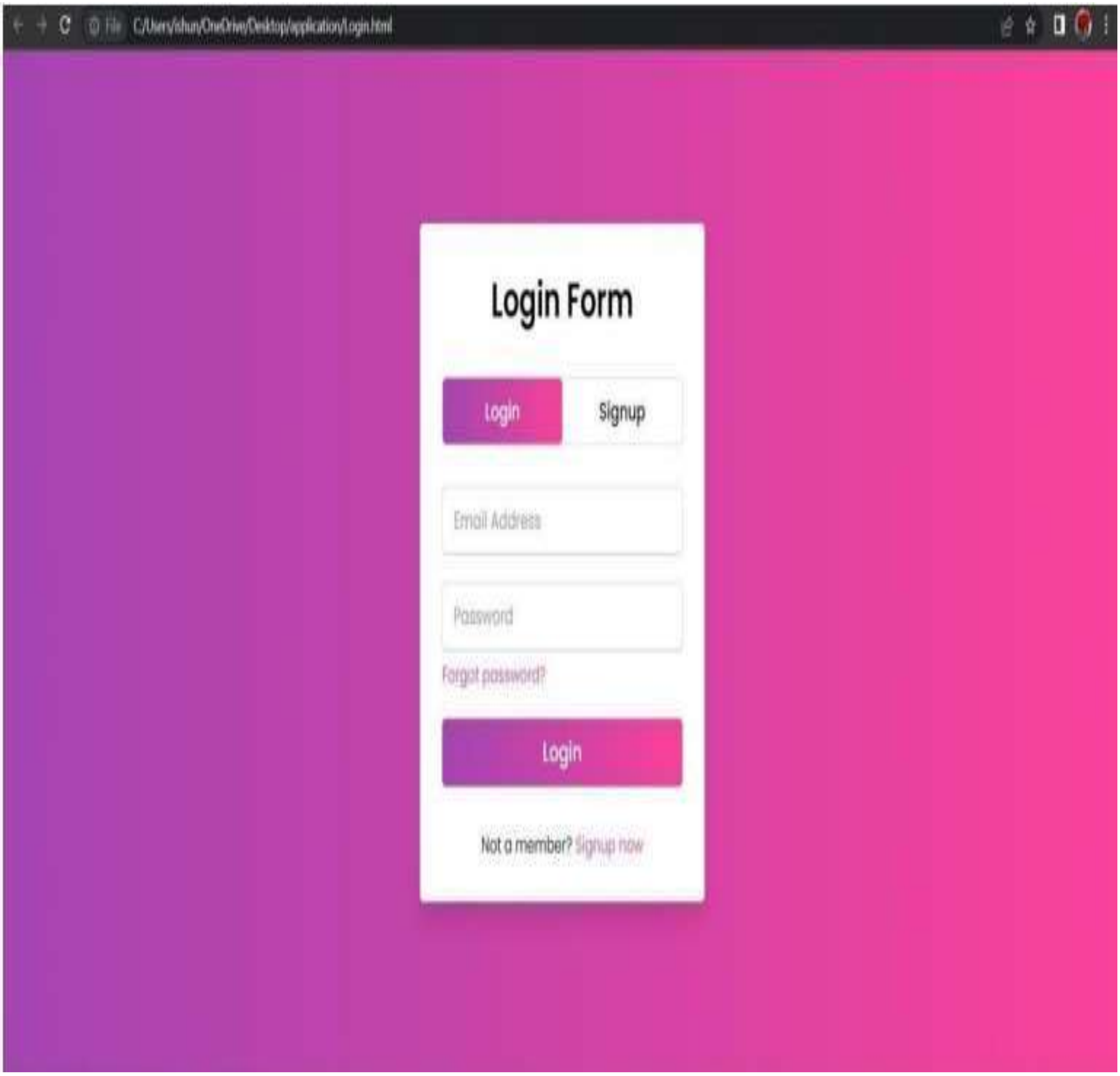
7 TESTING

7.1 Test Cases

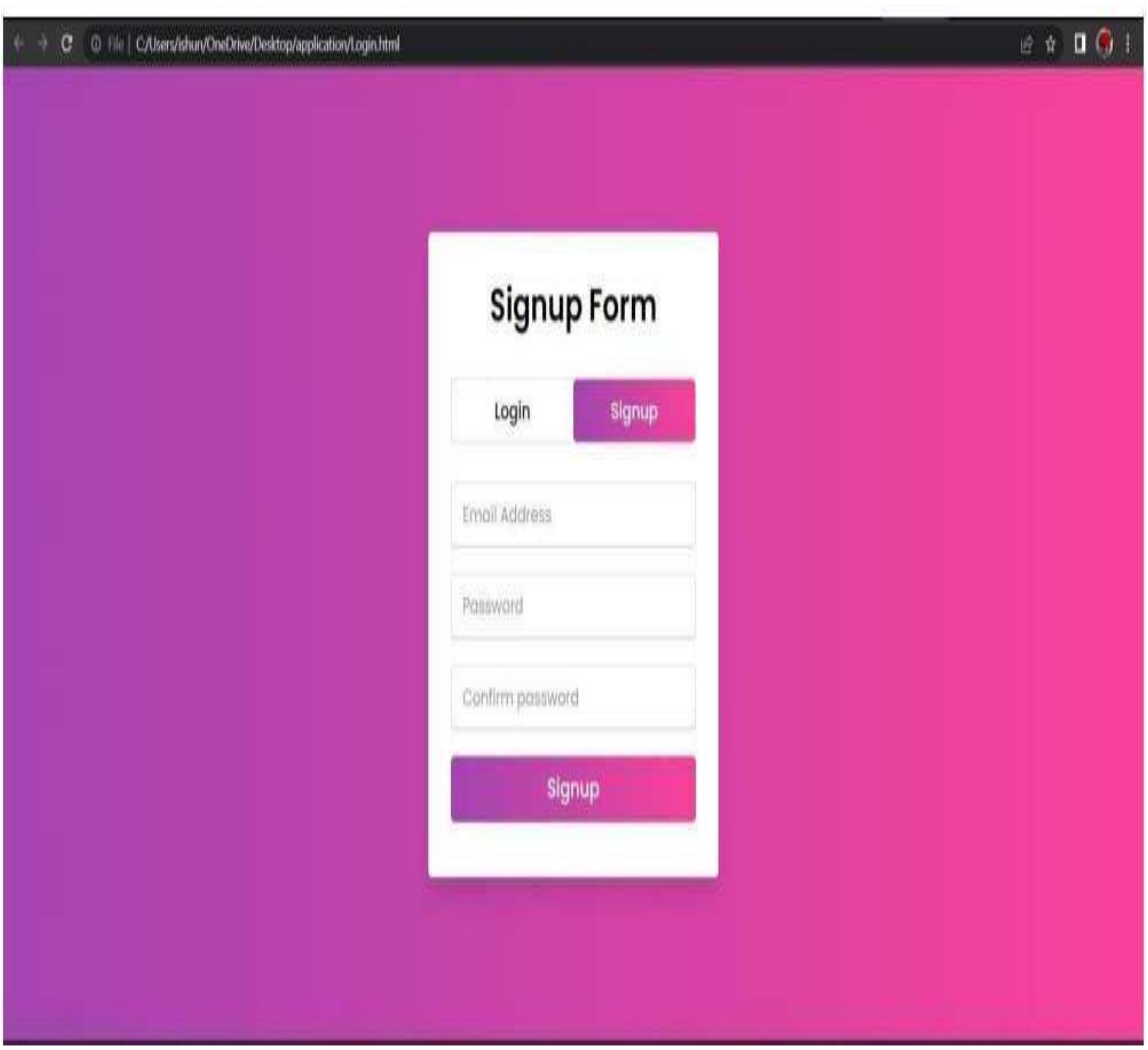
TEST CASE ID	Test Description	Input	Excepted result	Remarks
Test case – 1	Enter a field	User detail	Successfully registered	Pass
Test case – 2	Enter username and password	User details	Redirect to main page	Pass
Test case – 3	Username and password are empty	User details	Username and password are still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

8 RESULTS

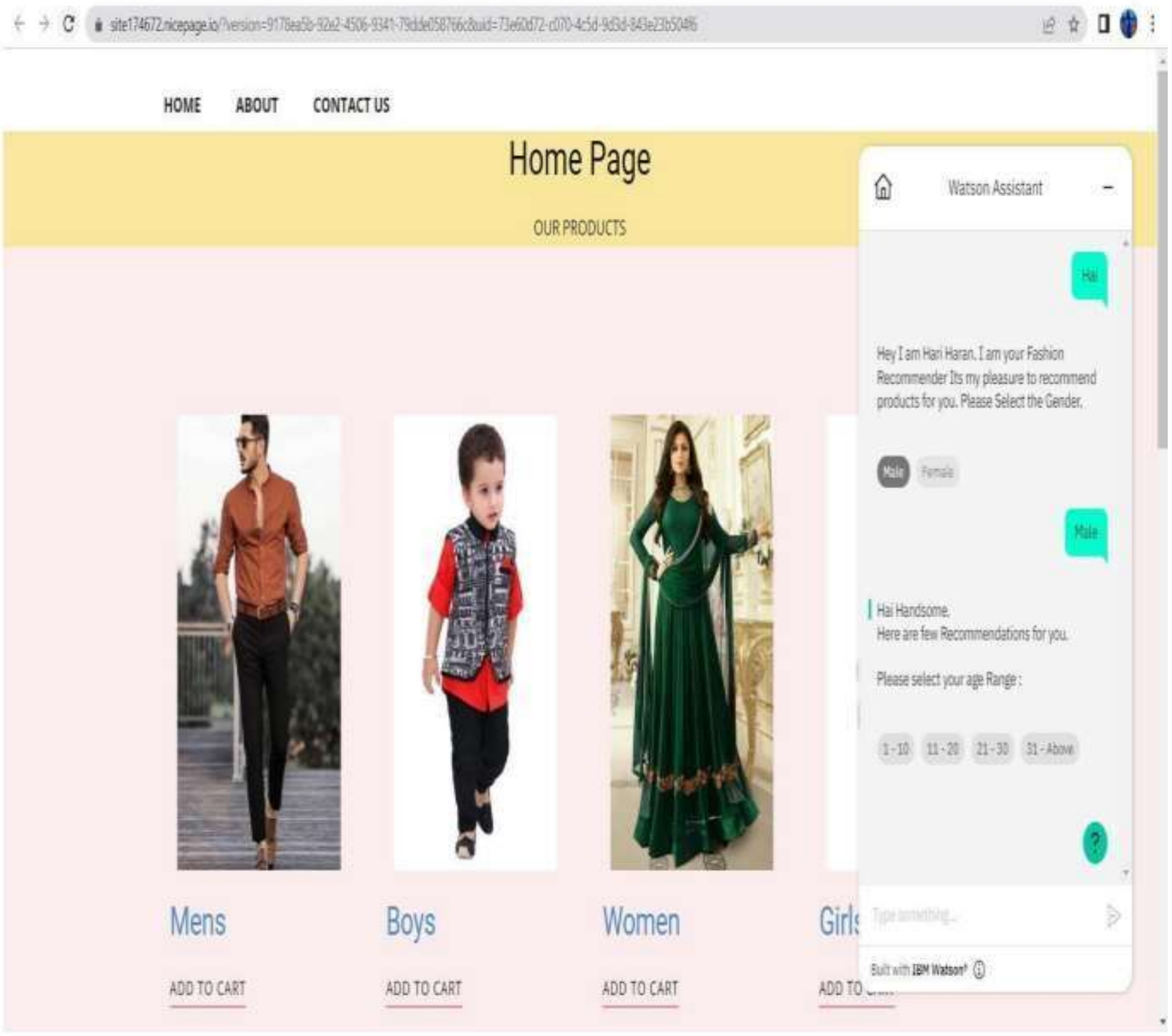
Login page :



Sign in Page:


C:/Users/ishun/OneDrive/Desktop/application/Login.html
Signup Form
Login
Signup
Email Address
Password
Confirm password
Signup

Home Page:



Hey I am Hari Haran. I am your Fashion Recommender Its my pleasure to recommend products for you. Please Select the Gender.

Hai Handsome,
Here are few Recommendations for you.

Please select your age Range :

9. ADVANTAGES & DISADVANTAGEs:

1. CUSTOMER SATISFACTION:

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.

2. 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.

10. DISADVANTAGES:

1. LACK OF DATA

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.

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

BASE .HTML:

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

    <head>
        <meta content="text/html;charset=utf-8" http-equiv="Content-Type">
        <meta content="utf-8" http-equiv="encoding">
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-tofit=no">
        <meta name="theme-color" content="#000000">
        <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico">
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
        <link href="{{ url_for('static',filename='css/custom.css') }}" rel="stylesheet" type="text/css" />
        <script defer src="https://use.fontawesome.com/releases/v5.0.6/js/all.js"></script>         <script
src="https://code.jquery.com/jquery-1.11.0.min.js"></script>

        <title>{% block title %}{% endblock %}</title>
    </head>
    <body>
  <!-- Modal -->
  <div class="modal fade" id="modalCenter" tabindex="-1" role="dialog"
arialabelledby="exampleModalCenterTitle" aria-hidden="true">
    <div class="modal-dialog modal-dialog-centered modal-lg" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="exampleModalLongTitle">Shopping Cart</h5>
          <button type="button" class="close" data-dismiss="modal" aria-label="Close">             <span aria-
hidden="true">&times;</span>
          </button>
        </div>
        <div class="modal-body">
          <div id="shoppingCart">
            <div class="container">
              <div class="row">
                <div class="col-sm">
                  <table class="table table-sm">
                    <thead>
                      <tr>
                        <th scope="col">#</th>
                        <th scope="col">Item</th>
                        <th scope="col">Name</th>
                        <th scope="col">Quantity</th>
                        <th scope="col">Unit Price</th>
                        <th scope="col">Sub-Total</th>
                        <th scope="col"></th>
                      </tr>
                    </thead>
                    <tbody>
                    <!-- For Each shirt -->
                    {% if shopLen != 0 %}
                    {% for i in range(shopLen) %}
                      <tr>
                        <th scope="row">{{ i + 1 }}</th>
                        <td><img src="/static/img/{{ shoppingCart[i]["image"] }}" width="30px" alt="{{
shoppingCart[i]["samplename"] }}" /></td>
                        <td>{{ shoppingCart[i]["samplename"] }}</td>
                        <td>{{ shoppingCart[i]['SUM(qty)'] }}</td>
                        <td>{{ '${:,.2f}'.format(shoppingCart[i]["price"]) }}</td>
                        <td>{{ '${:,.2f}'.format(shoppingCart[i]['SUM(subTotal)']) }}</td><!--
                        <td>
                          <form action="/remove/" methods="GET">
                            <input type="hidden" name="id" value="{{ shoppingCart[i]["id"] }}" />
                            <button type="submit" class="btn btn-secondary btn-sm"
id="removeFromCart">Remove</button>
                          </form>
                        </td>-->
                      </tr>
                    </tbody>
                    {% endfor %}
                    <tfoot>
                      <tr>
                        <td colSpan="7">Total: {{ '${:,.2f}'.format(total) }}<br /><br />
<div class="modal-footer">
                            <a href="/cart/"><button type="button" class="btn btn-primary checkout">Make
Changes</button></a>
                            <button type="button" class="btn btn-primary checkout" datadismiss="modal">Continue
Shopping</button>
                            <a href="/checkout/"><button type="button" class="btn btn-success checkout">Quick
Checkout</button></a>
                          </div>
                        </td>
                      </tr>
                    </tfoot>
                    {% else %}
                      <tr>
                        <td colSpan="7"><h3>Your cart is empty :\</h3></td>
                      </tr>
                    </tbody>
                    <tfoot>
                      <tr>
                        <td colSpan="7">Get some shirts now!<br />
                          <div class="modal-footer">
                            <button type="button" class="btn btn-primary" datadismiss="modal">Continue
Shopping</button>
                          </div>
                        </td>
                      </tr>
                    </tfoot>
                    {% endif %}
                  </table>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
  <header>
    <nav class="navbar fixed-top navbar-dark bg-dark navbar-expand-sm box-shadow">
      <a href="/" class="navbar-brand d-flex align-items-center">
          <strong><i class="fa  fa-cart-plus"></i> Smart Fashion Recommender Application</strong>
      </a>
      {% if session %}
      <ul class="navbar-nav mr-auto">
        <li class="nav-item"><a href="/logout/" class="nav-link">Logout</a></li>
        <li class="nav-item"><a href="/history/" class="nav-link">You Bought</a></li>       {% else %}
      <ul class="navbar-nav mr-auto">
        <li class="nav-item"><a href="/new/" class="nav-link">Register</a></li>         <li
class="nav-item"><a href="/login/" class="nav-link">Login</a></li>

      {% endif %}
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbardrop" datatoggle="dropdown">
            Filter By
          </a>
          <div class="dropdown-menu">
              <a class="dropdown-item" href="/">All</a>
              <a class="dropdown-item" href="/filter/?typeClothes=shirt">Shirts</a>
              <a class="dropdown-item" href="/filter/?typeClothes=pant">Trousers</a>
              <a class="dropdown-item" href="/filter/?typeClothes=shoe">Shoes</a>
              <a class="dropdown-item" href="/filter/?kind=casual">Casual Clothing</a>
              <a class="dropdown-item" href="/filter/?kind=formal">Formal Clothing</a>
              <a class="dropdown-item" href="/filter/?sale=1">On Sale</a>
              <a class="dropdown-item" href="/filter/?price=1">Price $0-$000</a>           </div>
        </li>
      </ul>
      <div>
        <button class="navbar-toggler" style="display:inline" type="button" datatoggle="modal" data-
target="#modalCenter">
          <span class="glyphicon glyphicon-shopping-cart" data-toggle="modal" data-target="">             <i
class="fas fa-shopping-cart"></i>
            <span class="counter">No. of Items: {{ totItems }}</span>
            <span class="counter">Total: ${{ '{:,.2f}'.format(total) }}</span>           </span>
        </button>
      </div>
    </nav>
  </header><br />
  <main>
    <div class="container">
```

```
      {% if display == 1 %}
      <div class="alert alert-success flashMessage" style="text-align:center">
        <strong>Your item was successfully removed from shopping cart!</strong>       </div>
      {% endif %}
    {% block body %}{% endblock %}
    <footer>
        <div class="container">
            <div class="row">
                <div class="col-md">
                    <hr />
                    <p>&#169; <a href="/">Smart Fashion Recommender Application</a></p>
                </div>
            </div>
        </div>
    </footer>     <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>

        <!-- jQuery first, then Popper.js, then Bootstrap JS -->
        <script src="https://code.jquery.com/jquery-1.11.0.min.js"></script>
        <!-- <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>-->
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>
        <!-- Custom JS Scripts -->
        <script src="{{ url_for('static',filename='js/myscripts.js') }}"></script>         <script
 src="{{ url_for('static',filename='js/validate.js') }}"></script>     </body>
</html>
```

Cart.html:

```
{% extends "base.html" %}

{% block title %}
Smart Fashion Recommender Application - Home
{% endblock %}

{% block body %}
<!-- Main Store Body -->
  <div aria-hidden="true">
    <div>
```

```
      <div>
        <div>
          <h5 class="modal-title" id="exampleModalLongTitle">Shopping Cart</h5>
          <button type="button" class="close" data-dismiss="modal" aria-label="Close">           </button>
        </div>
        <div>
          <div id="shoppingCart">
            <div class="container">
              <div class="row">
                <div class="col-sm">
                  <table class="table table-sm">
                    <thead>
                      <tr>
                        <th scope="col">#</th>
                        <th scope="col">Item</th>
                        <th scope="col">samplename</th>
                        <th scope="col">Quantity</th>
                        <th scope="col">Unit Price</th>
                        <th scope="col">Sub-Total</th>
                        <th scope="col"></th>
                      </tr>
                    </thead>
                    <tbody>
                    <!-- For Each shirt -->
                    {% if shopLen != 0 %}
                    {% for i in range(shopLen) %}
                      <tr>
                        <th scope="row">{{ i + 1 }}</th>
                        <td><img src="/static/img/{{ shoppingCart[i]["image"] }}" width="30px" alt="{{
shoppingCart[i]["samplename"] }}" /></td>
                        <td>{{ shoppingCart[i]["samplename"] }}</td>
                        <td><form action="/update/">
                            <input type="hidden" name="id" value="{{shoppingCart[i]["id"]}}" />
<input type="number" name="quantity" min="1" max="10" size="5" value="{{ shoppingCart[i]['SUM(qty)']
}}">
                            <button type="submit" class="btn btn-warning checkout">Update</button>
</form></td>
                        <td>{{ '${:,.2f}'.format(shoppingCart[i]["price"]) }}</td>
                        <td>{{ '${:,.2f}'.format(shoppingCart[i]['SUM(subTotal)']) }}</td>
                        <td>
                          <form action="/remove/" methods="GET">
                            <input type="hidden" name="id" value="{{ shoppingCart[i]["id"] }}" />
<button type="submit" class="btn btn-secondary btn-sm" id="removeFromCart">Remove</button>
                          </form>
                        </td>
                      </tr>
                    </tbody>
                    {% endfor %}
                    <tfoot>
                      <tr>
                        <td colSpan="7">Total: {{ '${:,.2f}'.format(total) }}<br /><br />
                          <div class="modal-footer">
                            <a href="/"><button type="button" class="btn btn-primary
```

```
checkout">Continue Shopping</button></a>
                          <a href="/checkout/"><button type="button" class="btn btn-success checkout">Proceed to
Checkout</button></a>
                        </div>
                      </td>
                    </tr>
                  </tfoot>
                  {% else %}
                    <tr>
                      <td colSpan="7"><h3>Your cart is empty :\</h3></td>
                    </tr>
                  </tbody>
                  <tfoot>
                    <tr>
                      <td colSpan="7">Get some shirts now!<br />
                        <div>
                          <a href="/"><button type="button" class="btn btn-secondary"
datadismiss="modal">Continue Shopping</button></a>
                        </div>
                      </td>
                    </tr>
                  </tfoot>
                  {% endif %}
                </table>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
  </div>
</main>
```

Index.html:

```
{% extends "base.html" %}

{% block title %}
Smart Fashion Recommender Application - Home
{% endblock %}

{% block body %}
<!-- Main Store Body -->
    {% if session['user'] %}
        <div class="alert alert-warning alert-dismissible fade show" role="alert">
          <button type="button" class="close" data-dismiss="alert" aria-label="Close">             <span aria-
hidden="true">&times;</span>
          </button>
            <strong>Welcome, {{ session['user'] }}</strong> Hope you have a pleasant experience shopping with
us.
        </div>
    {% endif %}
      <div class="row" id="shirtCard">
      {% for i in range(shirtsLen) %}
          <div class="col-sm">
              <div class="card text-center">
                  <div class="card-body">
                    <form action="/buy/" methods="POST">
                        <h5 class="card-title">{{shirts[i]["typeClothes"].capitalize()}}</h5>
                      <img src="/static/img/{{shirts[i]["image"]}}" class="shirt" alt="" />
                      <h5 class="card-text">{{shirts[i]["samplename"]}}</h5>
                      {% if shirts[i]["onSale"] %}
                        <img src="/static/img/sale-icon.png" width="26px" />
                        <h4 class="card-text price" style="color:red; display:inline">{{
'{:,.2f}'.format(shirts[i]["onSalePrice"]) }}</h4>
                      {% else %}
                        <h4 class="card-text price">{{ '{:,.2f}'.format(shirts[i]["price"]) }}</h4>                        {%
endif %}
                      <div class="stepper-input">
                          <span class="decrement target">-</span>
                          <input class="quantity" name="quantity" value='0' />
                          <span class="increment target">+</span>
</div>
                      <input type="hidden" name="id" value="{{shirts[i]["id"]}}" />
{% if not session %}
                      <input type="hidden" name="loggedin" value="0" />
                      {% else %}
                      <input type="hidden" name="loggedin" value="1" />

                      {% endif %}
                      <input type="submit" class="btn btn-primary addToCart" value="Add To Cart" /><br /><br />
                      <div class="alert alert-danger flashMessage" style="text-align: center; display:none; font-
size:0.9em;"></div>
                    </form>
                  </div>
              </div>
          </div>
      {% endfor %}
      </div>
    </div>
  </main>
```

History.html:

```
{% extends "base.html" %}

{% block title %}
Smart Fashion Recommender Application - Home
{% endblock %}

{% block body %}
<!-- Main Store Body -->
      <div class="row">
        <div class="col-sm">
```

```
          <h2>Your Shopping History</h2>
          <p>Items you've bought in the past.</p>
          <table class="table table-sm">
            <thead>
              <tr>
                <th scope="col">#</th>
                <th scope="col">Item</th>
                <th scope="col">Name</th>
                <th scope="col">Quantity</th>
                <th scope="col">Date</th>
                <th scope="col"></th>
              </tr>
            </thead>
            <tbody>
            <!-- For Each shirt -->
            {% for i in range(myShirtsLen) %}
              <tr>
                <th scope="row">{{ i + 1 }}</th>
                <td><img src="/static/img/{{ myShirts[i]["image"] }}" width="30px" alt="{{
myShirts[i]["samplename"] }}" /></td>
                <td>{{ myShirts[i]["samplename"] }}</td>
                <td>{{ myShirts[i]["quantity"] }}</td>
                <td>{{ myShirts[i]["date"] }}</td>
                <td><a href="/filter/?id={{ myShirts[i]["id"] }}"><button type="button" class="btn btn-
warning">Buy Again</button></a></td>
              </tr>
            </tbody>
            {% endfor %}
            <tfoot>
            </tfoot>
          </table>
        </div>
      </div>
    </div>
  </main>
```

Login.html:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta content="text/html;charset=utf-8" http-equiv="Content-Type">
        <meta content="utf-8" http-equiv="encoding">
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-tofit=no">
        <meta name="theme-color" content="#000000">
        <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico">
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
        <link href="{{ url_for('static',filename='css/custom.css') }}" rel="stylesheet" type="text/css" />
        <script defer src="https://use.fontawesome.com/releases/v5.0.6/js/all.js"></script>         <script
src="https://code.jquery.com/jquery-1.11.0.min.js"></script>
```

```
        <title>Smart Fashion Recommender Application - Log In</title>     </head>
    <body>
    <header>
        <nav class="navbar fixed-top navbar-dark bg-dark navbar-expand-sm box-shadow">
          <a href="/" class="navbar-brand d-flex align-items-center">
              <strong><i class="fa  fa-cart-plus"></i> Smart Fashion Recommender Application</strong>
          </a>
        </nav>
    </header><br />
    <main>
        <div class="container">
            <div class="row">
                <div class="col-sm">
                    <h2>Log In to Buy</h2>
                    <p>{{ msg }}</p>
                    <div>
                        <form action="/logged/" class="form" method="post">
                            <div>
                                <input type="text" name="username" autofocus placeholder="Username">
                                <input type="password" name="password" placeholder="Password">
                                <button type="submit" class="btn btn-primary">Login</button>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </main>
    </body>
</html>
```

New.html:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta content="text/html;charset=utf-8" http-equiv="Content-Type">
        <meta content="utf-8" http-equiv="encoding">
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-tofit=no">
        <meta name="theme-color" content="#000000">
        <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico">
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
        <link href="{{ url_for('static',filename='css/custom.css') }}" rel="stylesheet" type="text/css" />
        <script defer src="https://use.fontawesome.com/releases/v5.0.6/js/all.js"></script>        <script
src="https://code.jquery.com/jquery-1.11.0.min.js"></script>

        <title>Smart Fashion Recommender Application - Register</title>     </head>
    <body>
```

```
        <header>
            <nav class="navbar fixed-top navbar-dark bg-dark navbar-expand-sm box-shadow">
              <a href="/" class="navbar-brand d-flex align-items-center">
                  <strong><i class="fa fa-shopping-bag"></i> Smart Fashion Recommender Application</strong>
              </a>
            </nav>
        </header><br />
        <main>
            <div class="container">
                <div class="row">
                    <div class="col-sm">
                        <h2>Register</h2>
                        <p>{{msg}}</p>
                        <form action="/register/" class="form" method="post">
<input type="text" name="username" id="username"
placeholder="Username" autofocus required > <span id="user-msg" class="alert alertdanger"></span><br
/><br />
                            <input type="password" name="password" id="password" placeholder="Password" required
> <span id="password-msg" class="alert alertdanger"></span><br /><br />
                            <input type="password" name="confirm" id="confirm"
placeholder="Confirm Password" required> <span id="confirm-msg" class="alert alertdanger"></span><br
/><br />
                            <input type="text" name="fname" id="fname" placeholder="First Name" required> <span
id="fname-msg" class="alert alert-danger"></span><br /><br />
                            <input type="text" name="lname" id="lname" placeholder="Last Name" required>
<span id="lname-msg" class="alert alert-danger"></span><br /><br />                            <input
type="email" name="email" id="email" placeholder="Email" required> <span id="email-msg"
class="alert alert-danger"></span><br /><br /><br />
                            <button type="reset" class="btn btn-secondary">Clear</button>
                            <button type="submit" id="submit" class="btn btnprimary">Register</button>
                        </form>
                    </div>
                </div>
            </div>
        </main>
    <!-- Custom JS Scripts -->
        <script src="{{ url_for('static',filename='js/validate.js') }}"></script>
    </body>
</html>
```

Custom.css:

```
.card:hover {     border-color:red;
box-shadow: 1px 2px red;
}

.card {     margin-bottom: 1em;
background-color: pink;
}
```

```
.price {     color:
seagreen;     font-
weight: bold;
}

.price:before {     content:
'$';
}

.shirt {
    margin-bottom:       10px;
width:  200px;           height:
200px;
}

.stepper-input{     display:
flex;     display: -webkit-
flex;     color: #222;     max-
width: 120px;     margin:
10px auto;     text-align:
center;
}

header {     margin-bottom:
50px;
}

.shirtCart {     width:
25px;
}

.add {     text-transform:
uppercase;     font-size: 0.8em;
font-weight: bold;     color:
white;
}

.checkout {     text-transform:
uppercase;     font-size: 0.8em;
font-weight: bold;
}

.add:hover {     background-color:
sandybrown;     border-color:
sandybrown;
}

tr {     text-align:
center;
}

.modal-header {     border-bottom:
0px;
```

```
}

.counter {     font-size:
0.6em;     margin-left:
1em;     font-weight:
bold;
}


.increment, .decrement{
        height: 24px;        width:
24px;           border: 1px solid
#222;          text-align: center;
box-sizing:           border-box;
border-radius: 50%;        text-
decoration: none;          color:
#222;            font-size: 24px;
line-height:                    22px;
display:              inline-block;
cursor: pointer;
    }

.decrement:hover,
.increment:hover {     color:
green;
}

.decrement:active,  .increment:active {
background-color: green;
    color: white;
}


.quantity{         height:
24px;         width: 48px;
text-align: center;
margin: 0 12px;
border-radius: 2px;
        border: 1px solid #222;

}

body {     margin:
0;
    font-family: -apple-system,BlinkMacSystemFont,"Segoe UI",Roboto,"Helvetica
Neue",Arial,sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI Symbol";     font-size:
1rem;     font-weight: 400;     line-height: 1.5;     color: #212529;
    text-align: left;
    background-color: beige;
}

.bg-dark {
    background-color: grey!important;
```

```
}
```

Myscript.js:

```
$(".target").on("click", function() {
   let $button = $(this);
   let oldVal = parseInt($button.parent().find("input").val());     let
newVal = 0;

   if ($button.text() == '+') {
      newVal = oldVal + 1;
   }

   else {
      if (oldVal > 0) {
         newVal = oldVal - 1;
      }
else {
         newVal = 0;
      }
   }

   $button.parent().find("input").val(newVal);
});




$('.addToCart').on("click", function(event) {
   console.log('hello');
   if($(this).prev().prev().prev().find("input").val() == '0') {         event.preventDefault();
      $(this).next().next().next().html("You need to select at least one clothing.");
      $(this).next().next().next().css("display", "block");
      $(this).next().next().next().delay(3000).slideUp();
   }

   if ($(this).prev().val() == "0") {            event.preventDefault();
         $(this).next().next().next().html("You need to log in to buy.");
         $(this).next().next().next().css("display", "block");
$(this).next().next().next().delay(3000).slideUp();
      }
});


$(".flashMessage").delay(3000).slideUp();
```

Validate.js:

```
const SUBMIT = $( "#submit" );

// Each of the fields and error message divs const
USERNAME = $( "#username" );
```

```
const USERNAME_MSG = $( "#user-msg" );

const PASSWORD = $( "#password" );
const PASSWORD_MSG = $( "#password-msg" );

const CONFIRM = $( "#confirm" );
const CONFIRM_MSG = $( "#confirm-msg" );

const FNAME = $( "#fname" );
const FNAME_MSG = $( "#fname-msg" );

const LNAME = $( "#lname" );
const LNAME_MSG = $( "#lname-msg" );

const EMAIL = $( "#email" );
const EMAIL_MSG = $( "#email-msg" );

/**
 * Resets the error message fields and makes the submit  * button
visible.
 */
function reset_form ( )
{
   USERNAME_MSG.html( "" );
   USERNAME_MSG.hide();
   PASSWORD_MSG.html( "" );
   PASSWORD_MSG.hide();
   CONFIRM_MSG.html( "" );
   CONFIRM_MSG.hide();
   LNAME_MSG.html( "" );
   LNAME_MSG.hide();
   FNAME_MSG.html( "" );
   FNAME_MSG.hide();
   EMAIL_MSG.html( "" );
   EMAIL_MSG.hide();
   SUBMIT.show();
}

/**
 * Validates the information in the register form so that  * the
server is not required to check this information.
 */
function validate ( )
{     let valid = true;
   reset_form ( );


   // This currently checks to see if the username is
   // present and if it is at least 5 characters in length.
   if ( !USERNAME.val() || USERNAME.val().length < 5  )
   {
      // Show an invalid input message
      USERNAME_MSG.html( "Username must be 5 characters or more" );
```

```
        USERNAME_MSG.show();
        // Indicate the type of bad input in the console.
console.log( "Bad username" );         // Indicate that the
form is invalid.
        valid = false;
    }
    // TODO: Add your additional checks here.


    if ( USERNAME.val() != USERNAME.val().toLowerCase())
    {
        USERNAME_MSG.html("Username must be all lowercase");
        USERNAME_MSG.show();
        valid = false;
    }

    if ( !PASSWORD.val() || PASSWORD.val().length < 8 )
    {
        PASSWORD_MSG.html("Password needs to be at least 8 characters long");
        PASSWORD_MSG.show();
        valid = false;
    }

    if ( !CONFIRM.val() || PASSWORD.val() != CONFIRM.val() )
    {
        CONFIRM_MSG.html("Passwords don't match");
        CONFIRM_MSG.show();
        valid = false;
    }

    if ( !FNAME.val() )
    {
        FNAME_MSG.html("First name must not be empty");
        FNAME_MSG.show();
        valid = false;
    }

    if ( !LNAME.val() )
    {
        LNAME_MSG.html("Last name must not be empty");
        LNAME_MSG.show();
        valid = false;
    }

    var x = EMAIL.val().trim();     var atpos = x.indexOf("@");     var
dotpos = x.lastIndexOf(".");     if ( atpos < 1 || dotpos < atpos + 2 || dotpos
+ 2 >= x.length ) {
        EMAIL_MSG.html("You need to enter a valid email address");
        EMAIL_MSG.show();         valid
= false;
    }
```

```
   // If the form is valid, reset error messages     if (
valid )
   {
      reset_form ( );
   }
}

// Bind the validate function to the required events.
$(document).ready ( validate );
USERNAME.change ( validate );
PASSWORD.change ( validate );
CONFIRM.change ( validate );
LNAME.change ( validate );
FNAME.change ( validate );
EMAIL.change ( validate );
```

Application.py:

```
from cs50 import SQL from
flask_session import Session
from flask import Flask, render_template, redirect, request, session, jsonify from datetime
import datetime

# # Instantiate Flask object named app
app = Flask(_name_)

# # Configure sessions
app.config["SESSION_PERMANENT"] = False app.config["SESSION_TYPE"] =
"filesystem"
Session(app)

# Creates a connection to the database db =
SQL ( "sqlite:///data.db" )


@app.route("/") def
index():
   shirts = db.execute("SELECT * FROM shirts ORDER BY onSalePrice")     shirtsLen
= len(shirts)     # Initialize variables     shoppingCart = []     shopLen =
len(shoppingCart)     totItems, total, display = 0, 0, 0     if 'user' in session:
      shoppingCart = db.execute("SELECT samplename, image, SUM(qty), SUM(subTotal), price, id FROM
cart GROUP BY samplename")
      shopLen = len(shoppingCart)        for i
in range(shopLen):
         total += shoppingCart[i]["SUM(subTotal)"]            totItems
+= shoppingCart[i]["SUM(qty)"]
      shirts = db.execute("SELECT * FROM shirts ORDER BY onSalePrice ASC")        shirtsLen =
len(shirts)        return render_template ("index.html", shoppingCart=shoppingCart, shirts=shirts,
shopLen=shopLen, shirtsLen=shirtsLen, total=total, totItems=totItems, display=display,
session=session )
   return render_template ( "index.html", shirts=shirts, shoppingCart=shoppingCart, shirtsLen=shirtsLen,
shopLen=shopLen, total=total, totItems=totItems, display=display) @app.route("/buy/") def buy():
   # Initialize shopping cart variables
shoppingCart = []     shopLen =
len(shoppingCart)     totItems, total, display
= 0, 0, 0     qty =
int(request.args.get('quantity'))     if session:
       # Store id of the selected shirt         id
= int(request.args.get('id'))
       # Select info of selected shirt from database
       goods = db.execute("SELECT * FROM shirts WHERE id = :id", id=id)
       # Extract values from selected shirt record         #
Check if shirt is on sale to determine price
if(goods[0]["onSale"] == 1):             price =
goods[0]["onSalePrice"]         else:
           price = goods[0]["price"]
       samplename = goods[0]["samplename"]
       image = goods[0]["image"]         subTotal =
qty * price
       # Insert selected shirt into shopping cart
       db.execute("INSERT INTO cart (id, qty, samplename, image, price, subTotal) VALUES (:id, :qty,
:samplename, :image, :price, :subTotal)", id=id, qty=qty, samplename=samplename, image=image,
price=price, subTotal=subTotal)
       shoppingCart = db.execute("SELECT samplename, image, SUM(qty), SUM(subTotal), price, id FROM
cart GROUP BY samplename")
       shopLen = len(shoppingCart)         # Rebuild shopping
cart         for i in range(shopLen):             total +=
shoppingCart[i]["SUM(subTotal)"]             totItems +=
shoppingCart[i]["SUM(qty)"]         # Select all shirts for
home page view
       shirts = db.execute("SELECT * FROM shirts ORDER BY samplename ASC")         shirtsLen =
len(shirts)         # Go back to home page         return render_template ("index.html",
shoppingCart=shoppingCart, shirts=shirts, shopLen=shopLen, shirtsLen=shirtsLen, total=total,
totItems=totItems, display=display, session=session )


@app.route("/update/") def
update():
   # Initialize shopping cart variables
shoppingCart = []     shopLen =
len(shoppingCart)     totItems, total, display
= 0, 0, 0     qty =
int(request.args.get('quantity'))     if session:
       # Store id of the selected shirt
       id = int(request.args.get('id'))
       db.execute("DELETE FROM cart WHERE id = :id", id=id)
       # Select info of selected shirt from database
       goods = db.execute("SELECT * FROM shirts WHERE id = :id", id=id)
       # Extract values from selected shirt record         #
Check if shirt is on sale to determine price
if(goods[0]["onSale"] == 1):             price =
goods[0]["onSalePrice"]         else:
           price = goods[0]["price"]
       samplename = goods[0]["samplename"]
```

```
        image = goods[0]["image"]         subTotal =
qty * price
        # Insert selected shirt into shopping cart
        db.execute("INSERT INTO cart (id, qty, samplename, image, price, subTotal) VALUES (:id, :qty,
:samplename, :image, :price, :subTotal)", id=id, qty=qty, samplename=samplename, image=image,
price=price, subTotal=subTotal)
        shoppingCart = db.execute("SELECT samplename, image, SUM(qty), SUM(subTotal), price, id FROM
cart GROUP BY samplename")
        shopLen = len(shoppingCart)
# Rebuild shopping cart         for i in
range(shopLen):
            total += shoppingCart[i]["SUM(subTotal)"]             totItems
+= shoppingCart[i]["SUM(qty)"]
        # Go back to cart page
        return render_template ("cart.html", shoppingCart=shoppingCart, shopLen=shopLen, total=total,
totItems=totItems, display=display, session=session )


@app.route("/filter/") def
filter():
    if request.args.get('typeClothes'):
        query = request.args.get('typeClothes')
        shirts = db.execute("SELECT * FROM shirts WHERE typeClothes = :query ORDER BY samplename
ASC", query=query )     if request.args.get('sale'):
        query = request.args.get('sale')
        shirts = db.execute("SELECT * FROM shirts WHERE onSale = :query ORDER BY samplename
ASC", query=query)     if request.args.get('id'):         query = int(request.args.get('id'))
        shirts = db.execute("SELECT * FROM shirts WHERE id = :query ORDER BY samplename
ASC", query=query)     if request.args.get('kind'):         query = request.args.get('kind')
        shirts = db.execute("SELECT * FROM shirts WHERE kind = :query ORDER BY samplename
ASC", query=query)     if request.args.get('price'):         query = request.args.get('price')
        shirts = db.execute("SELECT * FROM shirts ORDER BY onSalePrice ASC")     shirtsLen = len(shirts)
    # Initialize shopping cart variables
shoppingCart = []     shopLen =
len(shoppingCart)     totItems, total,
display = 0, 0, 0     if 'user' in session:
        # Rebuild shopping cart
        shoppingCart = db.execute("SELECT samplename, image, SUM(qty), SUM(subTotal), price, id FROM
cart GROUP BY samplename")
        shopLen = len(shoppingCart)         for i in
range(shopLen):             total +=
shoppingCart[i]["SUM(subTotal)"]             totItems +=
shoppingCart[i]["SUM(qty)"]
        # Render filtered view         return render_template ("index.html", shoppingCart=shoppingCart,
shirts=shirts, shopLen=shopLen, shirtsLen=shirtsLen, total=total, totItems=totItems, display=display,
session=session )     # Render filtered view     return render_template ( "index.html", shirts=shirts,
shoppingCart=shoppingCart, shirtsLen=shirtsLen, shopLen=shopLen, total=total, totItems=totItems,
display=display)


@app.route("/checkout/") def
checkout():
   order = db.execute("SELECT * from cart")     #
Update purchase history of current customer     for item
in order:
       db.execute("INSERT INTO purchases (uid, id, samplename, image, quantity) VALUES(:uid, :id,
:samplename, :image, :quantity)", uid=session["uid"], id=item["id"], samplename=item["samplename"],
image=item["image"], quantity=item["qty"] )
   # Clear shopping cart     db.execute("DELETE
from cart")     shoppingCart = []     shopLen =
len(shoppingCart)     totItems, total, display = 0, 0, 0
# Redirect to home page     return redirect('/')


@app.route("/remove/", methods=["GET"]) def
remove():
   # Get the id of shirt selected to be removed     out
= int(request.args.get("id"))     # Remove shirt from
shopping cart
   db.execute("DELETE from cart WHERE id=:id", id=out)
   # Initialize shopping cart variables
totItems, total, display = 0, 0, 0     #
Rebuild shopping cart
   shoppingCart = db.execute("SELECT samplename, image, SUM(qty), SUM(subTotal), price, id FROM
cart GROUP BY samplename")
   shopLen = len(shoppingCart)     for i in
range(shopLen):         total +=
shoppingCart[i]["SUM(subTotal)"]         totItems +=
shoppingCart[i]["SUM(qty)"]
   # Turn on "remove success" flag
   display = 1
   # Render shopping cart
   return render_template ("cart.html", shoppingCart=shoppingCart, shopLen=shopLen, total=total,
totItems=totItems, display=display, session=session )


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


@app.route("/new/", methods=["GET"]) def new():
   # Render log in page
   return render_template("new.html")


@app.route("/logged/", methods=["POST"] ) def
logged():
   # Get log in info from log in form     user =
request.form["username"].lower()     pwd =
request.form["password"]
   #pwd = str(sha1(request.form["password"].encode('utf-8')).hexdigest())     # Make
sure form input is not blank and re-render log in page if blank     if user == "" or pwd
== "":         return render_template ( "login.html" )
   # Find out if info in form matches a record in user database
```

```
    query = "SELECT * FROM users WHERE username = :user AND password = :pwd"     rows =
db.execute ( query, user=user, pwd=pwd )

    # If username and password match a record in database, set session variables     if
len(rows) == 1:         session['user'] = user         session['time'] = datetime.now( )
session['uid'] = rows[0]["id"]     # Redirect to Home Page     if 'user' in session:
        return redirect ( "/" )
    # If username is not in the database return the log in page
    return render_template ( "login.html", msg="Wrong username or password." )


@app.route("/history/") def
history():
    # Initialize shopping cart variables
shoppingCart = []     shopLen =
len(shoppingCart)     totItems, total,
display = 0, 0, 0
    # Retrieve all shirts ever bought by current user
    myShirts = db.execute("SELECT * FROM purchases WHERE uid=:uid", uid=session["uid"])
    myShirtsLen = len(myShirts)
    # Render table with shopping history of current user     return render_template("history.html",
shoppingCart=shoppingCart, shopLen=shopLen, total=total, totItems=totItems, display=display,
session=session, myShirts=myShirts, myShirtsLen=myShirtsLen)


@app.route("/logout/") def
logout():
    # clear shopping cart
    db.execute("DELETE from cart")
    # Forget any user_id     session.clear()
    # Redirect user to login form
    return redirect("/")


@app.route("/register/", methods=["POST"] ) def
registration():     # Get info from form     username =
request.form["username"]     password =
request.form["password"]     confirm =
request.form["confirm"]     fname =
request.form["fname"]     lname =
request.form["lname"]     email =
request.form["email"]
    # See if username already in the database
    rows = db.execute( "SELECT * FROM users WHERE username = :username ", username = username )
    # If username already exists, alert user     if len(
rows ) > 0:
        return render_template ( "new.html", msg="Username already exists!" )
    # If new user, upload his/her info into the users database
    new = db.execute ( "INSERT INTO users (username, password, fname, lname, email) VALUES
(:username, :password, :fname, :lname, :email)",
                    username=username, password=password, fname=fname, lname=lname, email=email )
    # Render login template
    return render_template ( "login.html" )
```

```
@app.route("/cart/") def
cart():
    if 'user' in session:
        # Clear shopping cart variables
totItems, total, display = 0, 0, 0         # Grab
info currently in database
        shoppingCart = db.execute("SELECT samplename, image, SUM(qty), SUM(subTotal), price, id FROM
cart GROUP BY samplename")
        # Get variable values         shopLen =
len(shoppingCart)         for i in range(shopLen):
total += shoppingCart[i]["SUM(subTotal)"]
totItems += shoppingCart[i]["SUM(qty)"]
    # Render shopping cart
    return render_template("cart.html", shoppingCart=shoppingCart, shopLen=shopLen, total=total,
totItems=totItems, display=display, session=session)
```

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-46727-1660755075/tree/main