CONTENTS

1. INTRODUCTION
- 1.1 Project Overview
- 1.2 Purpose

2. LITERATURE SURVEY
- 2.1 Existing problem
- 2.2 References
- 2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION
- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming
- 3.3 Proposed Solution
- 3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS
- 4.1 Functional requirement
- 4.2 Non-Functional requirements

5. PROJECT DESIGN
- 5.1 Data Flow Diagrams
- 5.2 Solution & Technical Architecture
- 5.3 User Stories

6. PROJECT PLANNING & SCHEDULING
- 6.1 Sprint Planning & Estimation
- 6.2 Sprint Delivery Schedule
- 6.3 Reports from JIRA

7. CODING & SOLUTIONING (Explain the features added in the project along with code)
- 7.1 Feature 1
- 7.2 Feature 2
- 7.3 Database Schema (if Applicable)

8. TESTING
- 8.1 Test Cases
- 8.2 User Acceptance Testing

9. RESULTS
- 9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

INTRODUCTION

1.1 Project Overview

User is buy the product online by chatbot instead of keyboard search. Keyboard Search Is not all time recommends correct product. Chatbot is normally recommends the product by user interest. The keyboard may not recommend the product user interest. The chat also manage the order details in the project. It is very easy the user is to order without any worry about. The user is only focus on the product not all other things in the website. The user is login the webpage. After the dashboard page is shows the dress. In the side the chatbot is here. The chatbot is use the user order the product. The is user selected. The chatbot is sent the mail to user email. Chatbot is send the notification when the product is arrived in the user location. The admin is login the website then the admin dashboard is open. The admin dashboard is gives the user product. The admin can view the user details. The admin dashboard have the update stock. The admin can update the stock using to update the stock. The website use the external chatbot. the chatbot are IBM Watson Assistance. The Website store data at the cloud database. The database are IBM DB2. It is SQL based database. The Website is upload the project in the cloud. It the project is accessed using the IBM Object Storage. The Object storage is use bucket to store the project. The website use the container. The container is Docker. It is used to upload the project to the cloud. The user is click the website to manage the massive amount of user.

1.2 Purpose:

Users to buy product to chatbot. It is very easy the user is use the website.

User can manage the order by chatbot.

User can display the product by the user interest. User can find the product with less time.

2. LITERATURE SURVEY

2.1 Existing Problem

Title	Year	Technology	Problem
Outfit Recommender System	2018	E-Commerce, Collaborative filtering	This problem refers to users with specific preferences that make the developer to hurdle to create such interface
Image base fashion recommender system	2021	Cross domain recommendation system, Flask, DevOps, HTML, CSS	Some don't support rich content items
Modern Fashion recommender system	2022	AWS, Docker, Artificial Intelligence, Python, Google Cloud Computing Engine.	Inaccurately estimate consumer's true preference stand to pull down willingness to pay for some items and increase of the

• **References**

Mohamed Elechi, Anis Mezhgan, Mariam Khem Khem, Monji Kherallah "Clothing Classification using Deep CNN Architecture based on Transfer Learning" Bussard, Lukas, Matthias Danton, Christian Leisner, Christian Wingert, Till Quack and Luc Van Gool. "Apparel Classification with Style." Congying Guan, Sheng Feng Qin, Yang Long, (2019) \"Apparel-based deep learning system design for apparel style recommendation\", International Journal of Clothing Science and Technology

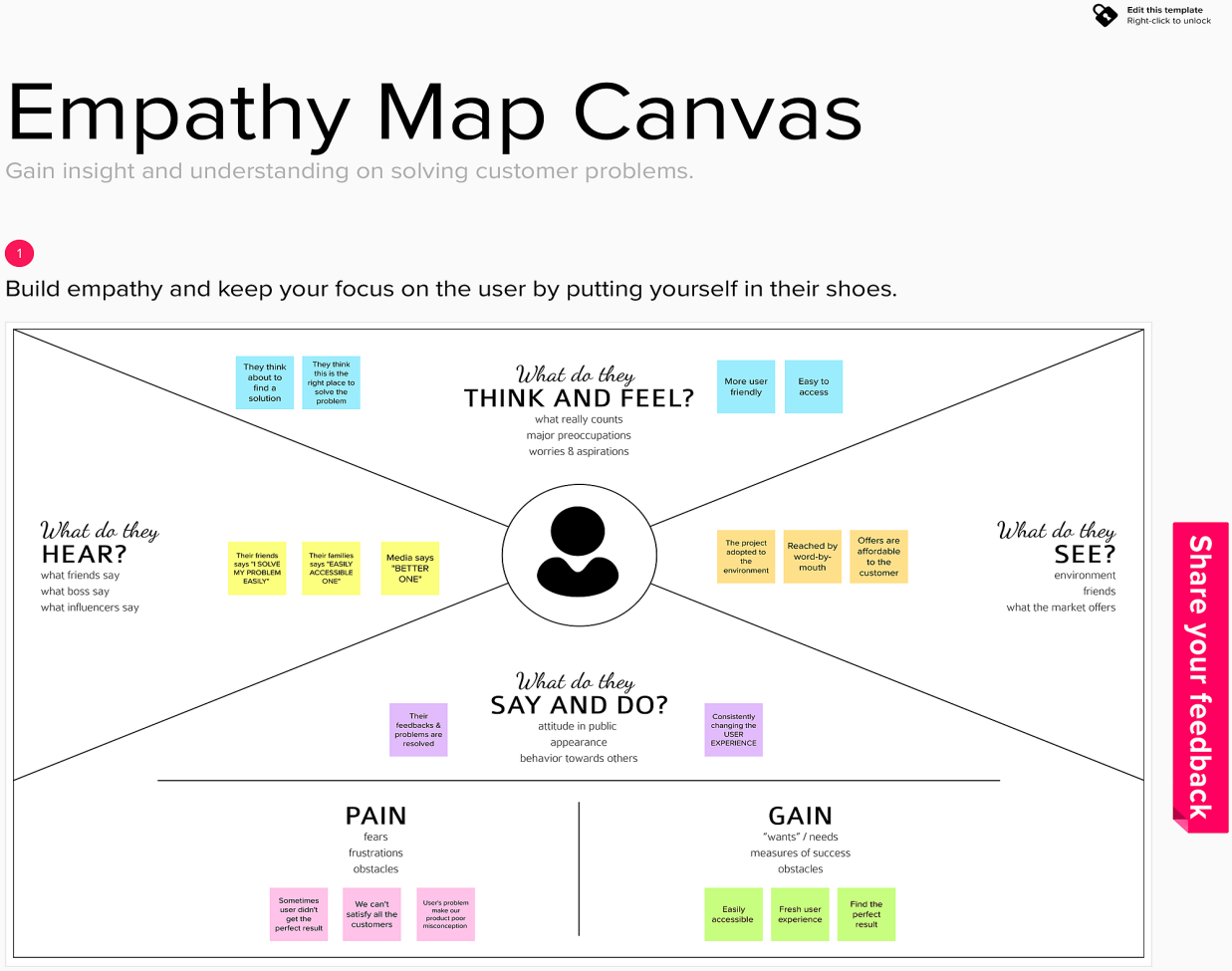
2.3 Problem Definition Statement

User entering the wrong input to get better results, Users is giving the opportunity to the chatbot

3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy map & Canvas Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges



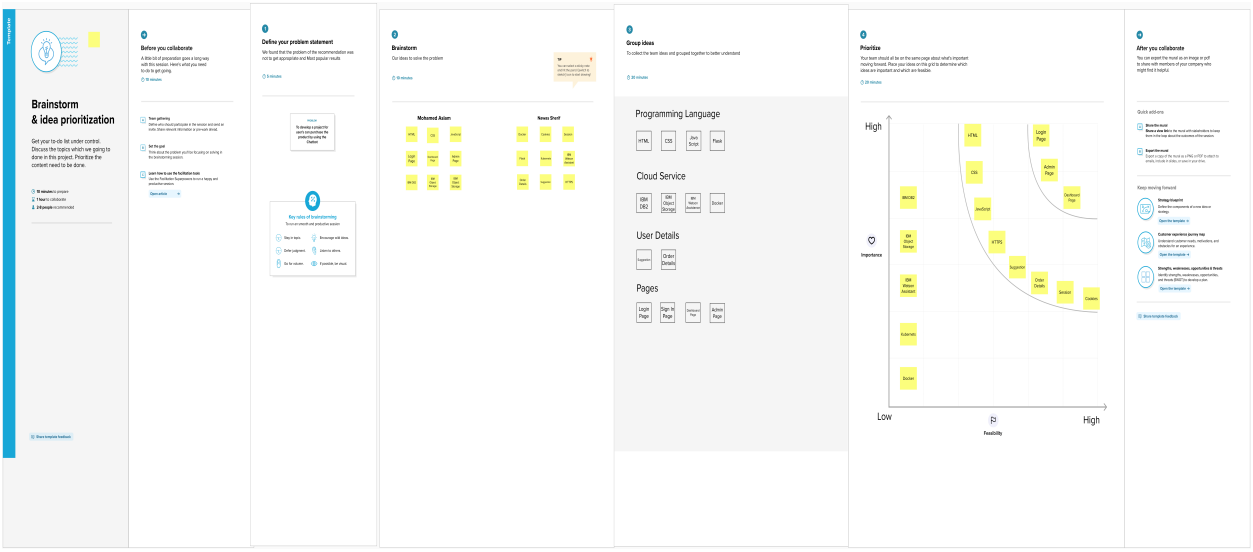
Reference:

https://app.mural.co/invitation/mural/ibmcloud7264/1663662849800?sender=u7b19bb3c874c312b4c5b0645 HYPERLINK

3.2 Ideation & Brainstorm

Thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions. Use this template in your own Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative brainstorming sessionssoyour team can unleash their imagination and start shaping concepts even if you're not sitting in the sameroom

Team Gathering, Collaboration and Select the Problem Statement



Reference:

https://app.mural.co/invitation/mural/ibmcloud7264/1668444580827?sender=u7b19bb3c874c312b4c5b0645

Proposed Solution Template:
Project team shall fill the following information in proposed solution template.

S No:	**Parameter**	**Description**
1	Problem Statement (Problem to be solved)	To solve the problem of finding the exact result they call for
2	Idea / Solution description	To propose a well defined Chatbot to identifying their perfect product with more relevant
3	Novelty / Uniqueness	To giving a fresh User Interface and great User Experience to use conveniently
4	Social Impact / Customer Satisfaction	Everyone is trying to create solution for the problem they need, If the Chatbot make a better result so everyone will trying to adopting their self's to the Chatbot feature
5	Business Model (Revenue Model)	By selling the Chatbot to Retailer to opt their product into it and make them more profitable
6	Scalability of the Solution	Adding extra features to the Chatbot for make better result

3.4 Proposed Solution Fit

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why.

Purpose:

• Solve complex problems in a way that fits the state of your customers.

• Succeed faster and increase your solution adoption by tapping into existing mediums and channels of behavior.

• Sharpen your communication and marketing strategy with the right triggers and messaging.

• Increase touch-points with your company by finding the right problembehavior fit and building trust by solving frequent annoyances, or urgent or costly problems.

• Understand the existing situation in order to improve it for your target group

Project Title: Smart Fashion Recommender Application

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMID45324

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) — CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

1) To solve their problem quickly
2) Customer understand the problem but can't solve it
3) Customer doesn't completely understand the interface

6. CUSTOMER CONSTRAINTS — CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

1) Lack of knowledge, awareness regarding the features available and offered
2) Second guess regarding its purpose

5. AVAILABLE SOLUTIONS — AS

Which solutions are available to the customers when they face the problem
or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

1) Customers could always go to some website community
2) Reference from social media platforms
3) Depending on Influencers to try and replicate

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS — J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

1) To make simple effective outfit combinations
2) To regulate customer's request
3) Get them offer deals for bargain

9. PROBLEM ROOT CAUSE — RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

1) Not recommending the perfect one
2) Not user friendly interface

7. BEHAVIOUR — BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

1) Dilemma regarding the right solution
2) Biased opinion, not open to options
3) Expecting better experience

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

Identify strong TR & EM

3. TRIGGERS — TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

1) Seeing their friends and colleagues better dressed then them
2) Bored with their general attire and everyday look
3) Insecure about their outfit choices

4. EMOTIONS: BEFORE / AFTER — EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

1) Feeling quite below average
2) Uncomfortable around well dressed people

10. YOUR SOLUTION — SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

1) To solve their problem by support them
2) Get their feedback and make convenient to use

8. CHANNELS of BEHAVIOUR — CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

8.1 Online:
1) Customer can receive the data in the app

8.2 Offline:
1) Customer can store the data in the device

Identify strong TR & EM

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Profile	Update in Profile Page
FR-4	Searching Product	Search Bar

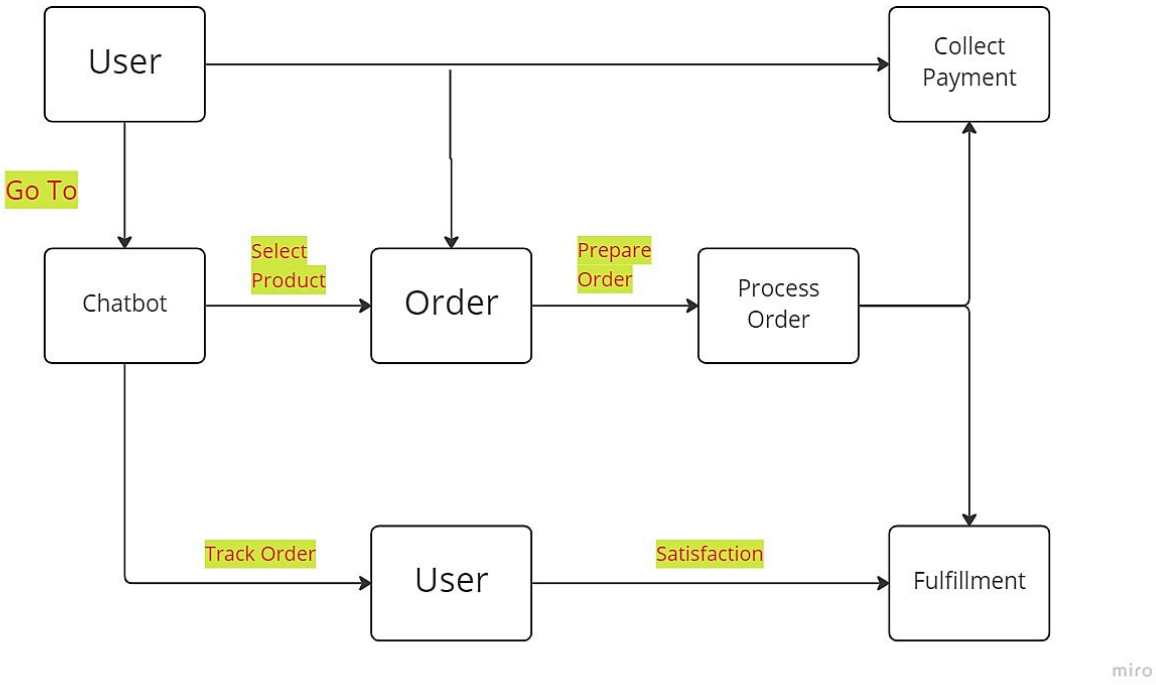
4.2 Non Functional Requirements

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Simple and Intuitive interface which provide rich experience
NFR-2	**Security**	Personal details are kept secure from third parties
NFR-3	**Reliability**	It is quite effortless to understand the features and implement them accordingly
NFR-4	**Performance**	Make the code simple as much as possible
NFR-5	**Availability**	Quite handy and accessible, performs required function as stated
NFR-6	**Scalability**	It can also be integrated as a software application for more attainability

5. PROJECT DESIGN

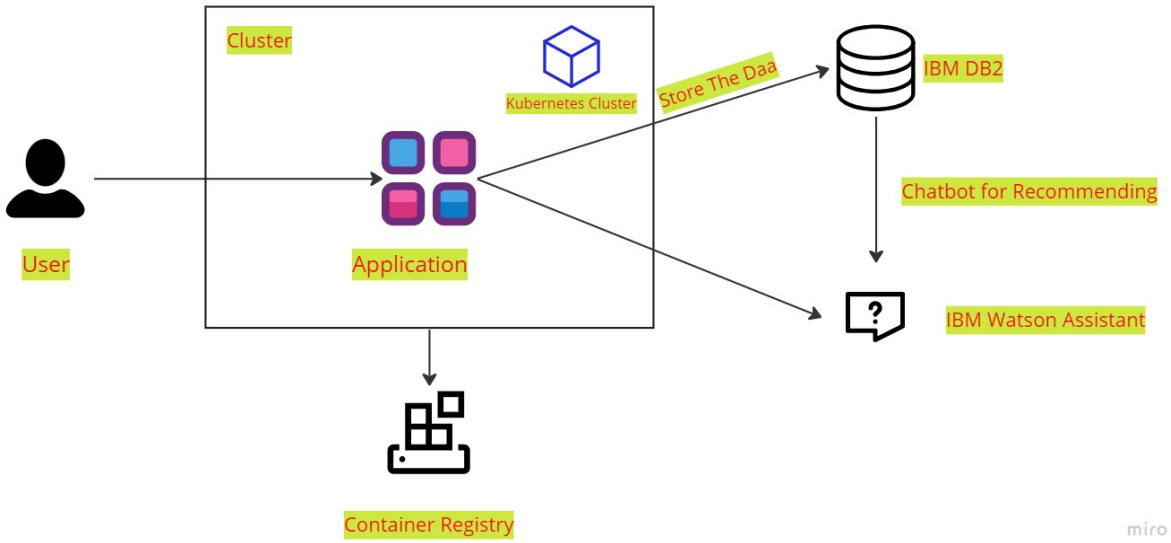
5.1 Data Flow Diagrams

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



5.2 Solution & Technical Architecture

Solution Architecture:



5.3 User Stories

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Panel	USN-1	The user will login into the website and gothrough the products available on the website	20	High	Mohamed Aslam, Mohamed Saki, Mohamed Rafee Nawas Sherif
Sprint-2	Admin panel	USN-2	The role of the admin is to check out the database about the stock and have a trackof all the things that the users are	20	High	Mohamed Aslam, Mohamed Saki, Mohamed

			purchasing.			Rafee Nawas Sherif
Sprint-3	Chat Bot	USN-3	The user can directly talk to Chatbot regarding the products. Get the recommendations based on information provided by the user.	20	High	Mohamed Aslam, Mohamed Saki, Mohamed Rafee Nawas Sherif
Sprint-4	Final delivery	USN-4	Container of applications using docker Kubernetes and deployment the application. Create the documentation and final submit the application	20	High	Mohamed Aslam, Mohamed Saki, Mohamed Rafee Nawas Sherif

6. PROJECT PLANNING AND SCHEDULING

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

6.1 Sprint Planning & Estimation

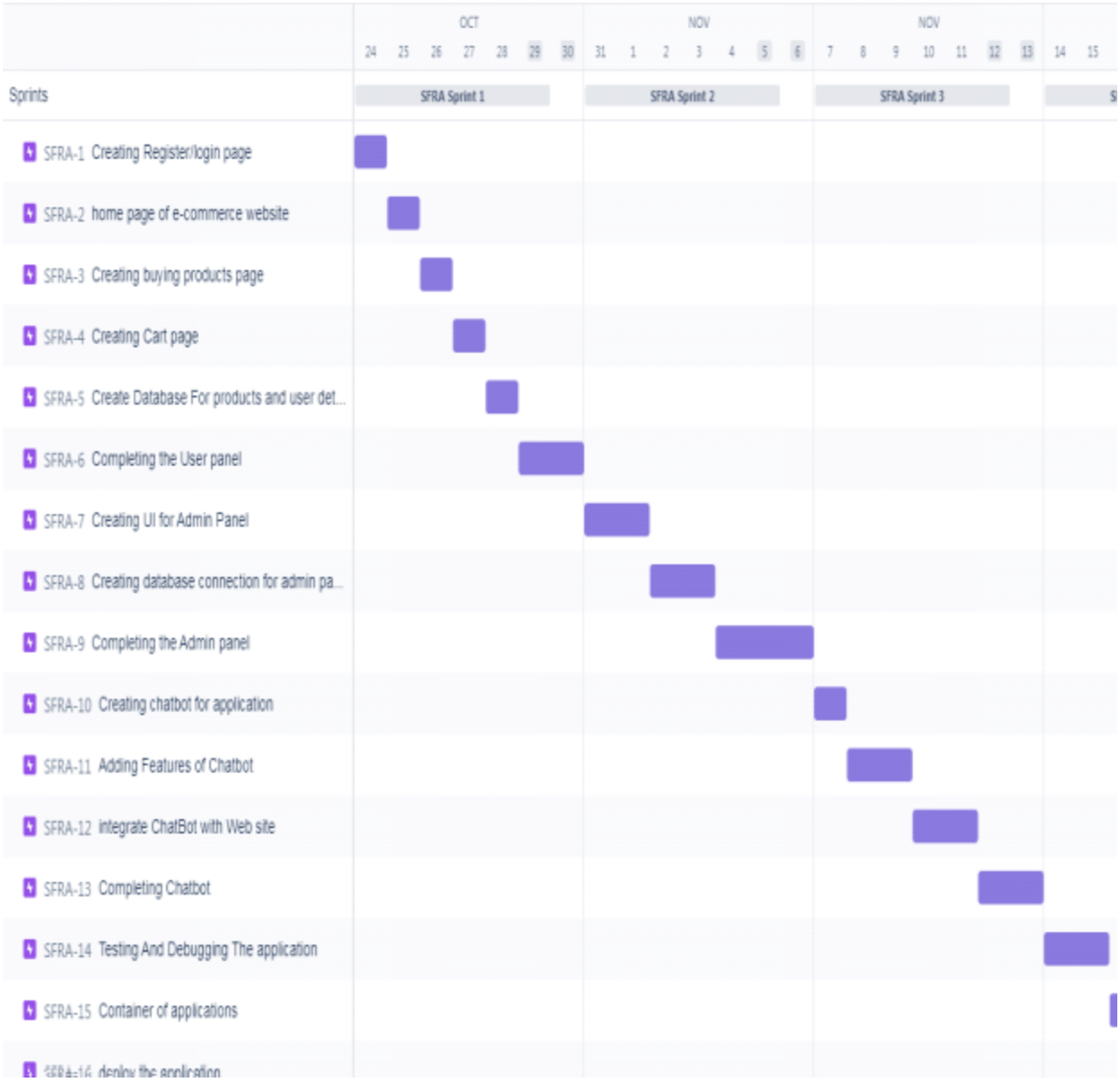
6.2 Sprint Delivery & Schedule

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

6.3 Report Jira Files

Burndown Chart:



7. CODING AND SOLUTION

admin.html

```
<html>
<head>
<title>Admin Dashboard</title>
<style>
 button{
       width:30%
       font-size:20px
       border-radius: 20px;
       background: transparent;
       text-shadow: 1px 1px 2px rgb(0,0,0,0.7);
   }
</style>
</head>
<body style="background-color:gray">
<h1 style="text-align:center">Admin Dashboard</h1>
<h3>Admin Page</h3>
<script>
function website()
{
i=1;
window.open("/admin",target="_self");
}
</script>
<button type="button" onClick="website()">Open</button> <br><br>
{orders}
</body>
</html>
```

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-to-fit=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>
   <style>
     .container-color{
      background-color: lightseagreen;
     }
     </style>
   <body>
  <!-- Modal -->
  <div class="modal fade" id="modalCenter" tabindex="-1" role="dialog" aria-
labelledby="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">Dashboard</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" data-
dismiss="modal">Keep Shopping</button>
                      <a href="/checkout/"><button type="button" class="btn btn-success
checkout">Fast Checkout</button></a>
                    </div>
                  </td>
                 </tr>
                </tfoot>
                {% else %}
                 <tr>
                  <td colSpan="7"><h3>Nothing In Your Cart :\</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" data-dismiss="modal">Continue
Shopping</button>
                    </div>
                  </td>
                 </tr>
                </tfoot>
                {% endif %}
               </table>
              </div>
             </div>
            </div>
           </div>
          </div>
         </div>
        </div>
       </div>
      </div>
     </div>
    </div>
   </div>
  </div>
  <header>
   <nav class="navbar fixed-top navbar-dark container-color navbar-expand-sm box-shadow">
    <style>
     a:hover{
      color:tomato;
```

```
  }
  </style>
<a href="/" class="navbar-brand d-flex align-items-center">
   <strong><i class="fa  fa-cart-plus">Store</i></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">Your Items</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" data-toggle="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" data-toggle="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 is removed from 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 Store</a></p>
        </div>
      </div>
    </div>
  </footer>
    <!-- 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 %}
Trendy Clothing Store - 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">Keep
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>Nothing in your cart :\</h3></td>
               </tr>
              </tbody>
              <tfoot>
               <tr>
                <td colSpan="7">Get some shirts now!<br />
                 <div>
```

```
                    <a href="/"><button type="button" class="btn btn-secondary" data-
dismiss="modal">Keep Shopping</button></a>
                  </div>
                 </td>
                </tr>
               </tfoot>
               {% endif %}
              </table>
             </div>
            </div>
           </div>
          </div>
         </div>
        </div>
       </main>

{% endblock %}
```

history.html

```
{% extends "base.html" %}
{% block title %}
Trendy Clothing Store - Home
{% endblock %}
{% block body %}
<!-- Main Store Body -->
    <div class="row">
      <div class="col-sm">
        <h2>Your Shopping History</h2>
        <p>Recently Purchased</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>
{% endblock %}
```

index.html

```
{% extends "base.html" %}
{% block title %}
Legendry Fashion- 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> Have A Good Day
      </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" style="background-color: lightseagreen;">
              <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>
<script>
 window.watsonAssistantChatOptions = {
  integrationID: "906c8aaf-7516-4f2a-a325-0906d32ec22d", // The ID of this integration.
  region: "us-south", // The region your integration is hosted in.
  serviceInstanceID: "f816c4ca-8e2f-4877-9aa9-47c9f8d6660e", // 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>

{% endblock %}
```

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-to-fit=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 - Log In</title>
<style>

body {
    display:flex;
    justify-content: center;
    align-items: center;
    color: brown;
}
    h1 {
      font-size: 30px;
      text-align: center;
      color: white;
      text-shadow: 5px 5px 4px rgb(0,0,0,.2);
      letter-spacing: 3px;
      margin-bottom: 30px;
      opacity: .8;
 }
 button {
```

```
    font-size: 30px;
    width: 30%;
    margin: 5%;
    margin-left: 35%;
    color: white;
    background: transparent;
    border: none;
    outline: none;
    box-shadow: 2px 2px 8px rgb(0,0,0,.5);
    text-shadow: 2px 2px 8px rgb(0,0,0,.5);
    border-radius: 8px;
    border-left: 1px solid rgb(255,255,255,0.3);
    border-top: 1px solid rgb(255,255,255,0.3);

}
 </style>
  </head>
  <style>
     .continer-color{
        background-color:greenyellow;
     }
  </style>
  <body>
  <header>
     <nav class="navbar fixed-top navbar-dark container-color 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 Store</strong>
      </a>
     </nav>
  </header><br />
  <main>
     <div class="container">
        <div class="row">
           <div class="col-sm">
              <h2>Log In</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> -->

                    <label for="username">Username:</label>
                    <input type="text" name=username id=username required><br><br>

                    <label for="password">Password:</label>
                    <input type="password" name=password id=password required>
                    <br>

                    <a href="#">Forget Password</a>
                    <br>
                    <button type="submit" class="btn btn-primary">Login</button>

                </form>
            </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-to-fit=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 Store - Register</title>
      <style>
body {
    display:flex;
    justify-content: center;
    align-items: center;
    color: brown;
  }
h1 {
    font-size: 30px;
    text-align: center;
    color:crimson;
    text-shadow: 2px 2px 4px rgb(0,0,0,.2);
    letter-spacing: 3px;
    margin-bottom: 30px;
    opacity: .8;
}

input {
    width: 80%;
```

```
    margin: 5%;
    background: transparent;
    font-size:15px;
    padding: 5px 5px;
    border-bottom:1px solid white;
    opacity:.8;
}

button {
    font-size: 15px;
    width: 30%;
    margin: 5%;
    margin-left: 35%;
    color: white;
    background: transparent;
    border: none;
    outline: none;
    box-shadow: 2px 2px 8px rgb(0,0,0,.5);
    text-shadow: 2px 2px 8px rgb(0,0,0,.5);
    border-radius: 8px;
    border-left: 1px solid rgb(255,255,255,0.3);
    border-top: 1px solid rgb(255,255,255,0.3);

}

   </style>
  </head>
  <style>
  .continer-color{
     background-color:greenyellow;
  }
</style>
  <body>
     <header>
        <nav class="navbar fixed-top navbar-dark container-color navbar-expand-sm box-
shadow">
          <a href="/" class="navbar-brand d-flex align-items-center">
             <strong style="text-color: tomato"><i class="fa fa-shopping-bag"></i>Smart Fashion
Store</strong>
          </a>
        </nav>
```

```
    </header><br />
    <main>
      <div class="container">
        <div class="row">
          <div class="col-sm">
            <h1>Register</h1>
            <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 alert-danger"></span><br /><br />
              <input type="password" name="password" id="password"
placeholder="Password" required > <span id="password-msg" class="alert alert-
danger"></span><br /><br />
              <input type="password" name="confirm" id="confirm" placeholder="Confirm
Password" required> <span id="confirm-msg" class="alert alert-danger"></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 btn-primary">Register</button>
            </form>
          </div>
        </div>
      </div>
    </main>
  <!-- Custom JS Scripts -->
    <script src="{{ url_for('static',filename='js/validate.js') }}"></script>
  </body>
</html>
```

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
if user=="admin" and pwd=="management":
        return render_template("admin.html",order="table")
    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)
@app.route("/admin")
def page():
    # 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=uid[0])
    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)
```

myscripts.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

```
// The submit button
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_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_MSG.show();
      valid = false;
   }

   if ( !PASSWORD.val() || PASSWORD.val().length < 8 )
   {
      //PASSWORD_MSG.html("");
      PASSWORD_MSG.show();
      valid = false;
   }

   if ( !CONFIRM.val() || PASSWORD.val() != CONFIRM.val() )
   {
      //CONFIRM_MSG.html("");
      CONFIRM_MSG.show();
```

```
        valid = false;
    }

    if ( !FNAME.val() )
    {
        //FNAME_MSG.html("");
        FNAME_MSG.show();
        valid = false;
    }

    if ( !LNAME.val() )
    {
        //LNAME_MSG.html("");
        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("");
        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 );
```

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-34937-1660279916.git

Project Demo Link: