HX8001-PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPERENEURSHIP

Submitted by

ANNAMALAI C 512219104002

EBINEZER PAUL LAZER J 512219104007

SANTHOSH K 512219104015

YOGESHWARAN P 512219104303

Pursuing final year of the degree

BACHELOR OF ENGINEERING

In

INFORMATION TECHNOLOGY & COMPUTER SCIENCE AND ENGINEERING
SKP ENGINEERING COLLEGE,

THIRUVANNAMALAI – 606611

NOV 2022

ANNA UNIVERSITY: CHENNAI 600 025

ACKNOWLEDGEMENT

SMART FASHION RECOMMENDER APPLICATION

IN CLOUD COMPUTING

I am very thankful to the management of SKP Engineering College, Tiruvannamalai for giving us this wonderful opportunity to study in this college and utilize all the facilities to the fullest.

My sincere thanks to lord blessing on us for the successful completion of my mini project. I thank proudly **Mr. K. KARUNANITHI, B.E., MBA**, Chairman of my college for providing the facility to do the mini project. I am grateful for my Principal **Dr.S.BASKARAN, M.E, Ph.D.**, for his constant support and encouragement to our project.

I express my sincere thanks to **Dr.N.PURUSHOTHAMAN,M.E, Ph.D.,** Head of the Department of Computer science and Supervisor **L.SANKARAN, M.E** for us for guidance,timely suggestions and word motivation.

It is a Great pleasure to express my gratitude and thanks towards my project Guide **L.SANKARAN, M.E** for his uninterruptable suggestions and words of improvements regarding this mini project, which played a major role in guiding me in my track.

We also extend our thanks to our Class Advisor and every member of the faculty who provided valuable academic guidance, and their co-operation to do this successful project. Finally, we would like to thank our parents, relatives and friends for their encouragement and enthusiastic co-operation.

Project Report Format

1. **INTRODUCTION**
    - Project Overview
    - Purpose
2. **LITERATURE SURVEY**
    - Existing problem
    - References
    - Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
    - Empathy Map Canvas
    - Ideation & Brainstorming
    - Proposed Solution
    - Problem Solution fit
4. **REQUIREMENT ANALYSIS**
    - Functional requirement
    - Non-Functional requirements
5. **PROJECT DESIGN**
    - Data Flow Diagrams
    - Solution & Technical Architecture
    - User Stories
6. **PROJECT PLANNING & SCHEDULING**
    - Sprint Planning & Estimation
    - Sprint Delivery Schedule
    - Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
    - Feature 1
    - Feature 2

Smart Fashion Recommender Application - Report

Introduection

- Clothing is a kind of symbol that represents people's internal perceptions through their outer appearance.
- It conveys information about their choices, faith, personality, profession, social status, and attitude towards life.
- Therefore, clothing is believed to be a nonverbal way of communicating and a major part of people's outer appearance [1].
- Recent technological advancements have enabled consumers to track current fashion trendsaround the globe, which influence their choices [2,3].
- The fashion choices of consumers depend on many factors, such as demographics, geographic location, individual preferences, interpersonal influences, age, gender, season, and culture [4–8].
- Moreover, previous fashion recommendation research shows that fashion preferences vary not only from country to country but also from city to city [9].
- The combination of fashion preferences and the above mentioned factors associated with clothing choices could transmit the image features for a better understanding of consumers' preferences

Project Overview

- Recommendation system (RS) is referred to as a decision-making approach for usersunder a multidimensional information environment .
- RS has also been defined as ane-commerce tool, which helps consumers search based on knowledge that is related to aconsumer's choices and preferences .
- RS also assists in augmenting social processesby using the recommendations of other users when there is no abundant personal information or knowledge of the alternatives .
- RS handles the complication of informationoverload that consumers usually encounter by offering customized service, exclusive content, and personalized recommendations .
- There are multiple phases involved in the recommendation system that develop thefoundation of any state-of-the-art recommendation system.
- These are defined as the information collection phase, the learning phase, and the recommendation phase.
- The interrelationship of these phases involved in the recommendation process.
- It shows that information collection is the initial stage of RS, which is followed by the learning phase and the recommendation phase.
- The recommendation provided in the last phase can be generated based on information gat

LITERATURE SURVEY

Existing problem

References

- Fashion Recommendation Systems, Models and Methods: A Review ,Informatics /26 July 2021
- A Review of Modern Fashion Recommend-er Systems,ACM Comput /December 2021
- Smart Fashion: A Review of AI Applications in the Fashion & Apparel Industry

Problem Statement Definition

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love. A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

PUBLICATION /YEAR	TITLE	OVERVIEW	POSITIVE ASPECTS	LIMITATIONS
Informatics / 26 July 2021	Fashion Recommendation Systems, Models and Methods: A Review	To the best of the authors' knowledge, this is the first scholarly article to review the state-of-the-art fashion recommendation systems and the corresponding filtering techniques. In addition, this review also explores various potential models that could be implemented to develop fashion recommendation systems in the future.	The conclusion of this review paper is this helps in learning about the users' interest in fashion through social media and other medias and recommends the choices after deep understanding on the users' interest.	The focus of this comprehensive review paper was to explore fashion recommendation-based articles published in last decade that explicitly described their frameworks, algorithms, and filtering techniques. To achieve this goal, the articles were searched using keywords relevant to the topic title instead of using the PRISMA technique. However, it did not affect the article extraction methodology, because the authors included and studied all the research papers relevant to the research focus.

ACM Comput / December 2021	A Review of Modern Fashion Recommend-er Systems	The goal of this survey is to provide a review of recommender systems that operate in the specific vertical domain of garment and fashion products. We have identified the most pressing challenges in fashion RS research and created a taxonomy that categorizes the literature according to the objective they are trying to accomplish and type of side-information.We have also identified the most important evaluation goals an perspectives and the most commonly used datasets and evaluation metrics.	In this survey, we have analyzed and classified the RS that function in a specific vertical market: clothes and fashion goods. In particular, we have introduced a taxonomy of fashion recommender systems, which categorizes them according to the task and type of side information .We have also identified the most important evaluation goals and perspectives exploited by the community, together with the most common datasets and evaluation metrics.	It limits the recommendation by only the task done by the users'. So it's unable to learn about users interests in other fields unlike the one defined above.
	Smart Fashion: A Review of AI Applications in the Fashion & Apparel Industry			

IDEATION & PROPOSED SOLUTION

Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours 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 hisor her goals and challenges.

Empathy map for Smart Fashion Recommender Application

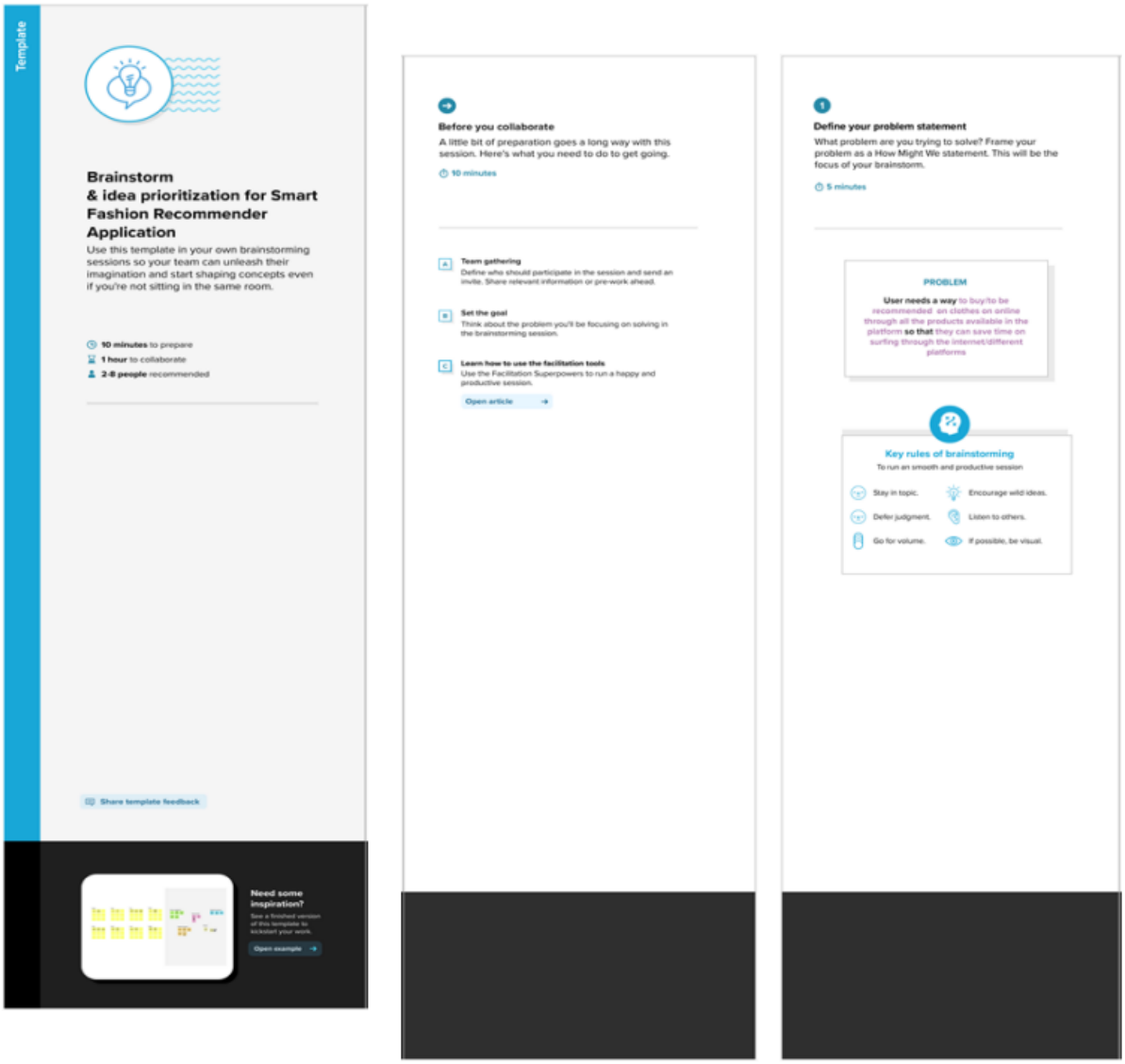
Ideation & Brainstorming

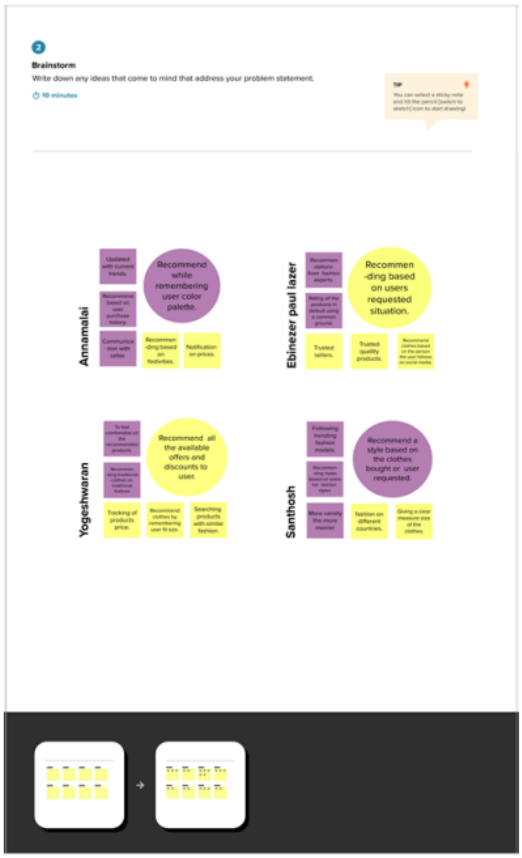
Brainstorm & Idea Prioritization Template:

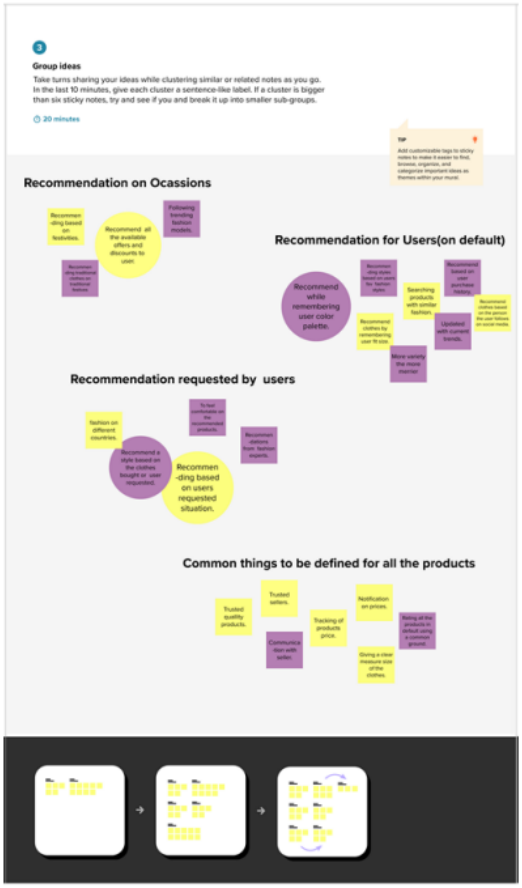
Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich number of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Brainstorm & Idea Prioritization Template for Smart Fashion Recommender Application




2
Brainstorm
Write down any ideas that come to mind that address your problem statement.
10 minutes
TIP
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!
Annamalai
Updated with current trends.
Recommend based on user purchase history.
Communication with seller.
Recommen -ding based on festivities.
Notification on prices.
Recommend while remembering user color palette.
Ebinezer paul lazer
Recommen -dations from fashion experts.
Rating all the products in default using a common ground.
Trusted sellers.
Trusted quality products.
Recommend clothes based on the person the user follows on social media.
Recommen -ding based on users requested situation.
Yogeshwaran
To feel comfortable on the recommended products.
Recommen -ding traditional clothes on traditional festivals.
Tracking of products price.
Recommend clothes by remembering user fit size.
Searching products with similar fashion.
Recommend all the available offers and discounts to user.
Santhosh
Following trending fashion models.
Recommen -ding styles based on users for fashion styles.
More variety the more merrier.
fashion on different countries.
Giving a clear measure size of the clothes.
Recommend a style based on the clothes bought or user requested.


3
Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.
20 minutes
TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.
Recommendation on Ocassions
Recommend all the available offers and discounts to user.
Following trending fashion models.
Recommen -ding based on festivities.
Recommend while remembering user color palette.
Recommendation for Users(on default)
Recommen -ding styles based on users for fashion styles.
Searching products with similar fashion.
Recommend clothes by remembering user fit size.
Updated with current trends.
More variety the more merrier
Recommendation requested by users
fashion on different countries.
To feel comfortable on the recommended products.
Recommen -dations from fashion experts.
Recommend a style based on the clothes bought or user requested.
Recommen -ding based on users requested situation.
Common things to be defined for all the products
Trusted sellers.
Notification on prices.
Trusted quality products.
Tracking of products price.
Rating all the products in default using a common ground.
Communication with seller.
Giving a clear measure size of the clothes.

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using

After you collaborate

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

Quick add-ons

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

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

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template →

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template →

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template →

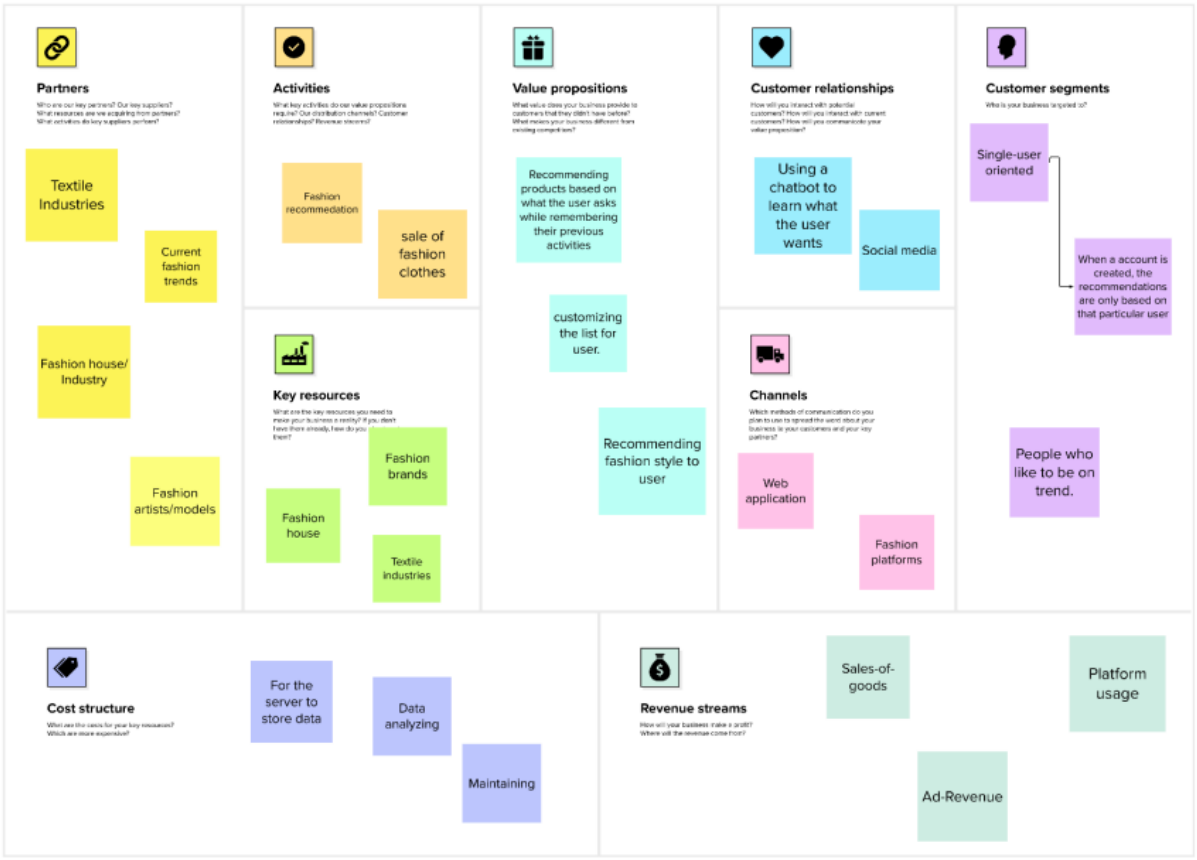
Share template feedback

Proposed Solution

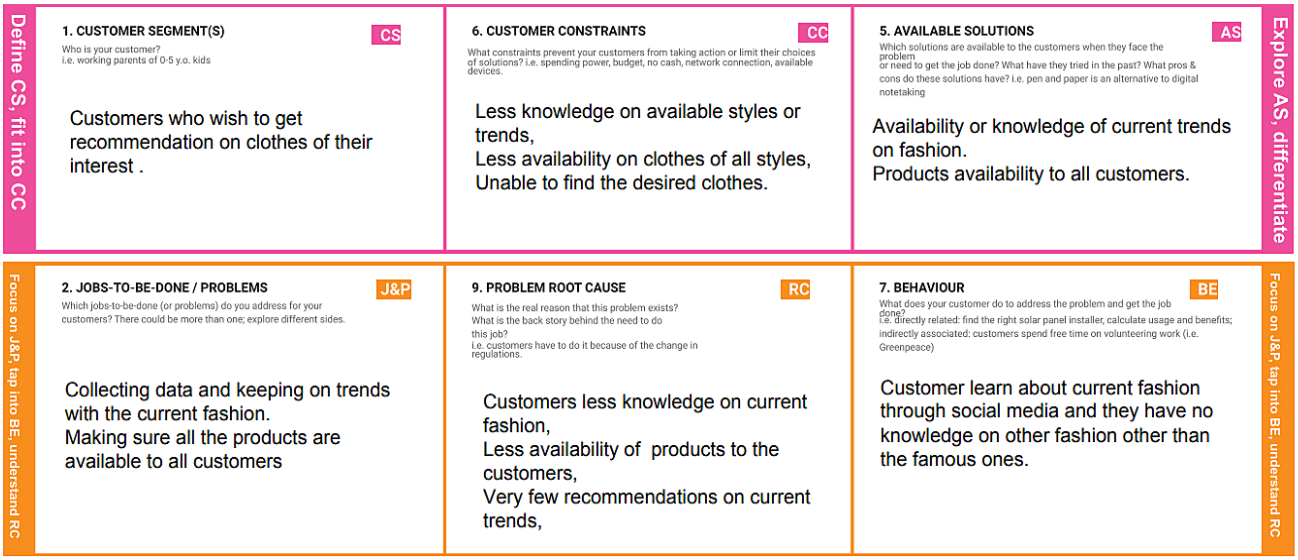
Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	User needs a way to buy/to be recommended on clothes on online through all the products available in the platform so that they can save time on surfing through the internet/different platforms.
2.	Idea / Solution description	The idea is to learn all the current trends of the fashion and reaching a conclusion based on the users' responses.
3.	Novelty / Uniqueness	The uniqueness of this project is the system recommends based on what user wants and what user likes/prefers based on previous activities and when a customized closet is created, the system may provide clothing style based on the request given by the user.
4.	Social Impact / Customer Satisfaction	When a user gets recommendation that satisfies them without even putting a effort on their side, then there is no need for them search on other platforms, at the same time when the user increases, other platforms may sign up with our platform.
5.	Business Model (Revenue Model)	Business model is attached below.
6.	Scalability of the Solution	The first thing will be where the user gets based on what user requests, after a considerable time, where the system can predict what user wants and sorting down the list to more personalised. The next will be where the user gets a fashion/clothing style recommendation from his bought history and creating a personalized closet. The final step will be, creating a common platform for fashion designers, fashion artists and other fashion professionals to interact or showcase their work to our registered users.

Business Model for Smart fashion Recommender Application:



Problem Solution fit

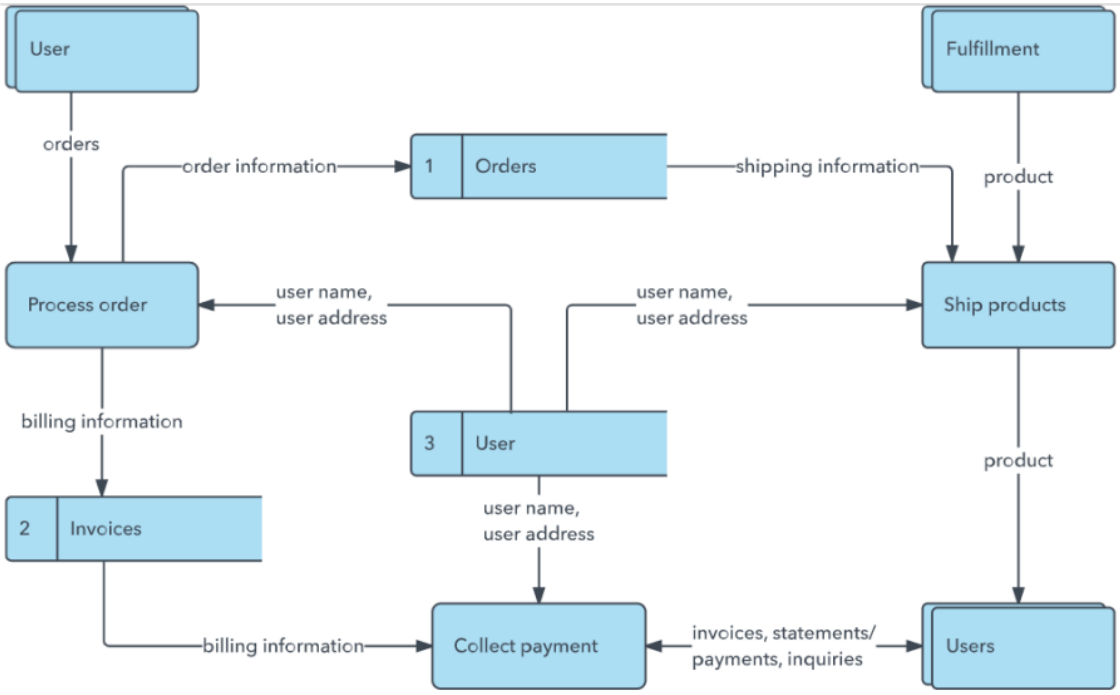


REQUIREMENT ANALYSIS

5.PROJECT DESIGN

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.



User Stories

list of the user stories for the product.

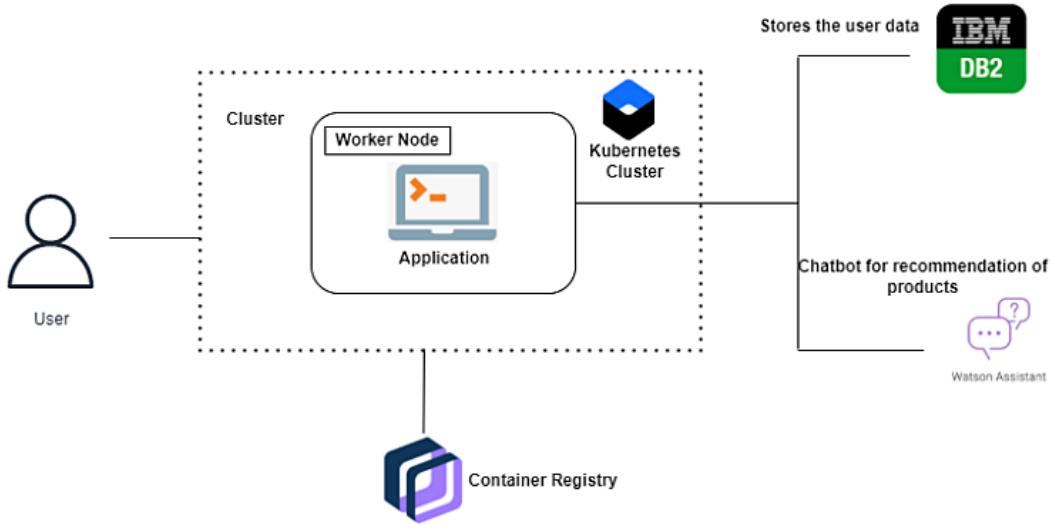
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access my account/ login	High	Sprint-1
	Dashboard	USN-6	As a user, I can view the dashboard and by products		High	Sprint-2
Customer (Web user)	Registration / Customers	USN-7	As a user,I can register for the application by entering my email,password, and confirming my password.	I can access my account / dashboard		Sprint-2
Customer Care Executive	Contact with customers	USN-8	As a Customer , customer care executive can solve the customer requirements and feedback	I can receive calls from customers	High	Sprint-1
Administrator	Check stock and price, orders	USN-9	As a administrative , I can check the database and stock details and alter the prices	I am the admin of the company	High	Sprint-2

Solution & Technical Architecture

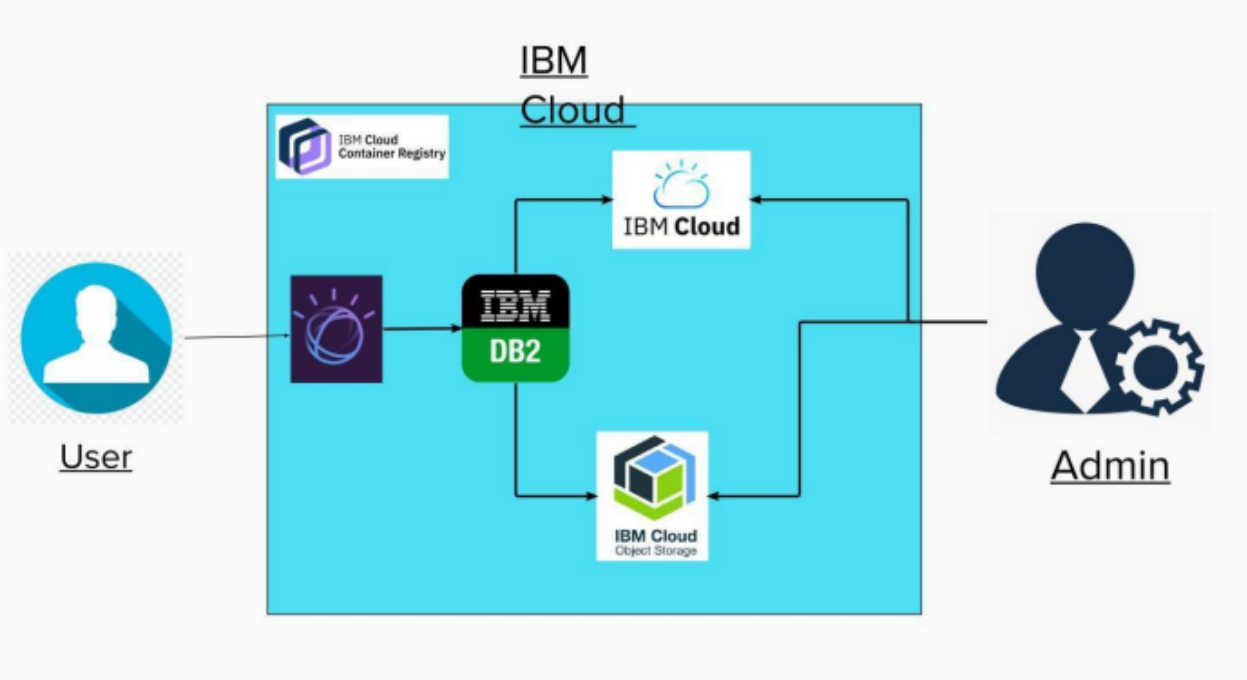
Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

Solution Architecture:



Technical Architecture:



User Stories

- The user will login into the website and go through the products available on the website
- The role of the admin is to check out the database about the stock and have a track of all the things that the users are purchasing.
- The user can directly talk to Chatbot regarding the products. Get the recommendations based on information provided by the user
- Container of applications using docker Kubernetes and deployment the application.Create the documentation and final submit the application

6. PROJECT PLANNING & SCHEDULING

Sprint Planning & Estimation

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	EBINEZER PAUL LAZER.J ANNAMALAI.C SANTHOSH.K YOGESHWARAN.P
Sprint-2	Admin panel	USN-2	The role of the admin is to check out the database about the stock and have a track of all the things that the users are purchasing.	20	High	EBINEZER PAUL LAZER.J ANNAMALAI.C SANTHOSH.K YOGESHWARA N.P
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	EBINEZER PAUL LAZER.J ANNAMALAI.C SANTHOSH.K YOGESHWARAN.P
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	EBINEZER PAUL LAZER.J ANNAMALAI.C SANTHOSH.K YOGESHWARAN.P

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

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022		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

6.3 Reports from JIRA

7.CODING & SOLUTIONING(Explain the features added in the project along with code)

7.1 Feature 1

The first feature in this application is adding a database registering Users and admins to the database and verify the user.

Registration - User

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Uregister - aesy</title>
    <meta property="og:title" content="Uregister - aesy" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta charset="utf-8" />
    <meta property="twitter:card" content="summary_large_image" />

    <style data-tag="reset-style-sheet">
        html {  line-height: 1.15;}body {  margin: 0;}* {  box-sizing: border-box;  border-width: 0;  border-style:  solid;}p,li,ul,pre,div,h1,h2,h3,h4,h5,h6,figure,blockquote,figcaption {  margin: 0;  padding: 0;}button {  background-color: transparent;}button,input,optgroup,select,textarea {  font-family: inherit;  font-size: 100%;  line-height: 1.15;  margin: 0;}button,select {  text-transform: none;}button,[type="button"],[type="reset"],[type="submit"] {  -webkit-appearance: button;}button::-moz-focus-inner,[type="button"]::-moz-focus-inner,[type="reset"]::-moz-focus-inner,[type="submit"]::-moz-focus-inner {  border-style: none;  padding: 0;}button:-moz-focus,[type="button"]:-moz-focus,[type="reset"]:-moz-focus,[type="submit"]:-moz-focus {  outline: 1px dotted ButtonText;}a {  color: inherit;  text-decoration: inherit;}input {  padding: 2px 4px;}img { display: block;}html { scroll-behavior: smooth  }
    </style>
    <style data-tag="default-style-sheet">
      html {
        font-family: Inter;
```

```
      font-size: 16px;
    }

    body {
      font-weight: 400;
      font-style:normal;
      text-decoration: none;
      text-transform: none;
      letter-spacing: normal;
      line-height: 1.15;
      color: var(--dl-color-gray-black);
      background-color: var(--dl-color-gray-white);

    }
  </style>
  <link
    rel="stylesheet"

href="https://fonts.googleapis.com/css2?family=Raleway:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&amp;display=swap"
    data-tag="font"
  />
  <link
    rel="stylesheet"

href="https://fonts.googleapis.com/css2?family=Inter:wght@100;200;300;400;500;600;700;800;900&amp;display=swap"
    data-tag="font"
  />
  <!--This is the head section-->
  <!-- <style> ... </style> -->
  <link rel="stylesheet" href="./style.css" />
```

```
</head>
<body>
  <div>
    <link href="./uregister.css" rel="stylesheet" />

    <div class="uregister-container">
      <div class="uregister-container1">
        <h1 class="uregister-heading">User Registration</h1>
        <form class="uregister-form">
          <div class="uregister-remember">
            <input
              type="checkbox"
              checked=""
              disabled=""
              required=""
              autocomplete="on"
              class="uregister-checkbox"
            />
            <span class="uregister-text">Remember me</span>
          </div>
        </form>
        <div class="uregister-container2">
          <span class="uregister-text01">
            <br />
            <br />
            <br />
            <span>E-mail :</span>
            <br />
            <br />
            <span>Username :</span>
            <br />
            <br />
            <span>Password :</span>
```

```
      <br />
    </span>
    <button class="uregister-button button">Register me</button>
  </div>
  <div class="uregister-container3">
    <input
      type="email"
      target="email"
      required=""
      placeholder="placeholder"
      class="uregister-textinput input"
    />
    <input
      type="text"
      required=""
      placeholder="placeholder"
      class="uregister-textinput1 input"
    />
    <input
      type="password"
      required=""
      placeholder="placeholder"
      class="uregister-textinput2 input"
    />
  </div>
</div>
<header data-role="Accordion" class="uregister-header">
  <div class="uregister-container4">
    <span class="uregister-text13">AESY</span>
  </div>
  <div class="uregister-separator"></div>
  <nav class="uregister-nav">
    <nav
```

```
          class="navigation-links2-nav navigation-links2-root-class-name19"
        >
          <span class="navigation-links2-text"><span>About</span></span>
          <span class="navigation-links2-text1"><span>Home</span></span>
          <span class="navigation-links2-text2">
            <span>Contact Us</span>
          </span>
        </nav>
      </nav>
      <div data-role="AccordionHeader" class="uregister-accordion-header">
        <svg viewBox="0 0 1024 1024" class="uregister-icon">
          <path
            d="M128 554.667h768c23.552 0 42.667-19.115 42.667-42.667s-19.115-42.667-42.667-42.667h-768c-23.552 0-42.667 19.115-42.667 42.667s19.115 42.667 42.667 42.667zM128 298.667h768c23.552 0 42.667-19.115 42.667-42.667s-19.115-42.667-42.667-42.667h-768c-23.552 0-42.667 19.115-42.667 42.667s19.115 42.667 42.667 42.667zM128 810.667h768c23.552 0 42.667-19.115 42.667-42.667s-19.115-42.667-42.667-42.667h-768c-23.552 0-42.667 19.115-42.667 42.667s19.115 42.667 42.667 42.667z"
          ></path>
        </svg>
      </div>
      <div data-role="AccordionContent" class="uregister-accordion-content">
        <div class="uregister-nav1">
          <nav
            class="navigation-links2-nav navigation-links2-root-class-name20"
          >
            <span class="navigation-links2-text"><span>About</span></span>
            <span class="navigation-links2-text1"><span>Home</span></span>
            <span class="navigation-links2-text2">
              <span>Contact Us</span>
            </span>
          </nav>
        </div>
```

```
        </div>
       </header>
     </div>
   </div>
   <script
     data-section-id="header"
     src="https://unpkg.com/@teleporthq/teleport-custom-scripts"
   ></script>
  </body>
</html>
```

Registration - Admin

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Aregister1 - aesy</title>
    <meta property="og:title" content="Aregister1 - aesy" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta charset="utf-8" />
    <meta property="twitter:card" content="summary_large_image" />

    <style data-tag="reset-style-sheet">
        html { line-height: 1.15;}body { margin: 0;}* { box-sizing: border-box; border-width: 0; border-style: solid;}p,li,ul,pre,div,h1,h2,h3,h4,h5,h6,figure,blockquote,figcaption { margin: 0; padding: 0;}button { background-color: transparent;}button,input,optgroup,select,textarea { font-family: inherit; font-size: 100%; line-height: 1.15; margin: 0;}button,select { text-transform: none;}button,[type="button"],[type="reset"],[type="submit"] { -webkit-appearance: button;}button::-moz-focus-inner,[type="button"]::-moz-focus-inner,[type="reset"]::-moz-focus-inner,[type="submit"]::-moz-focus-inner { border-style: none; padding: 0;}button:-moz-focus,[type="button"]:-moz-focus,[type="reset"]:-moz-focus,[type="submit"]:-moz-focus { outline: 1px dotted ButtonText;}a { color: inherit; text-decoration: inherit;}input { padding: 2px 4px;}img { display: block;}html { scroll-behavior: smooth }
    </style>
```

```
<style data-tag="default-style-sheet">
 html {
  font-family: Inter;
  font-size: 16px;
 }

 body {
  font-weight: 400;
  font-style:normal;
  text-decoration: none;
  text-transform: none;
  letter-spacing: normal;
  line-height: 1.15;
  color: var(--dl-color-gray-black);
  background-color: var(--dl-color-gray-white);

 }
</style>
<link
 rel="stylesheet"

href="https://fonts.googleapis.com/css2?family=Raleway:ital,wght@0,100;0,200;0,300;0,400;0,5
00;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&amp;display=
swap"
 data-tag="font"
/>
<link
 rel="stylesheet"

href="https://fonts.googleapis.com/css2?family=Inter:wght@100;200;300;400;500;600;700;800;
900&amp;display=swap"
 data-tag="font"
/>
```

```
  <!--This is the head section-->
  <!-- <style> ... </style> -->
  <link rel="stylesheet" href="./style.css" />
</head>
<body>
  <div>
    <link href="./aregister1.css" rel="stylesheet" />

    <div class="aregister1-container">
      <div class="aregister1-container1">
        <h1 class="aregister1-heading">Admin Registration</h1>
        <form class="aregister1-form">
          <div class="aregister1-remember">
            <input
              type="checkbox"
              checked=""
              disabled=""
              required=""
              autocomplete="on"
              class="aregister1-checkbox"
            />
            <span class="aregister1-text">Remember me</span>
          </div>
        </form>
        <div class="aregister1-container2">
          <span class="aregister1-text01">
            <br />
            <br />
            <br />
            <span>Employee Id :</span>
            <br />
            <br />
            <span>Username :</span>
```

```
      <br />
      <br />
      <span>Password :</span>
      <br />
    </span>
    <button class="aregister1-button button">Register me</button>
  </div>
  <div class="aregister1-container3">
    <input
      type="text"
      target="email"
      required=""
      placeholder="placeholder"
      class="aregister1-textinput input"
    />
    <input
      type="text"
      required=""
      placeholder="placeholder"
      class="aregister1-textinput1 input"
    />
    <input
      type="password"
      required=""
      placeholder="placeholder"
      class="aregister1-textinput2 input"
    />
  </div>
</div>
<header data-role="Accordion" class="aregister1-header">
  <div class="aregister1-container4">
    <span class="aregister1-text13">AESY - Admin pannel</span>
  </div>
```

```
<div class="aregister1-separator"></div>
<nav class="aregister1-nav">
 <nav
  class="navigation-links2-nav navigation-links2-root-class-name19"
 >
  <span class="navigation-links2-text"><span>About</span></span>
  <span class="navigation-links2-text1"><span>Home</span></span>
  <span class="navigation-links2-text2">
   <span>Contact Us</span>
  </span>
 </nav>
</nav>
<div data-role="AccordionHeader" class="aregister1-accordion-header">
 <svg viewBox="0 0 1024 1024" class="aregister1-icon">
  <path
   d="M128 554.667h768c23.552 0 42.667-19.115 42.667-42.667s-19.115-42.667-42.667-42.667h-768c-23.552 0-42.667 19.115-42.667 42.667s19.115 42.667 42.667 42.667zM128 298.667h768c23.552 0 42.667-19.115 42.667-42.667s-19.115-42.667-42.667-42.667h-768c-23.552 0-42.667 19.115-42.667 42.667s19.115 42.667 42.667 42.667zM128 810.667h768c23.552 0 42.667-19.115 42.667-42.667s-19.115-42.667-42.667-42.667h-768c-23.552 0-42.667 19.115-42.667 42.667s19.115 42.667 42.667 42.667z"
  ></path>
 </svg>
</div>
<div
 data-role="AccordionContent"
 class="aregister1-accordion-content"
>
 <div class="aregister1-nav1">
  <nav
   class="navigation-links2-nav navigation-links2-root-class-name20"
  >
   <span class="navigation-links2-text"><span>About</span></span>
```

```
            <span class="navigation-links2-text1"><span>Home</span></span>
            <span class="navigation-links2-text2">
              <span>Contact Us</span>
            </span>
          </nav>
        </div>
      </div>
    </header>
  </div>
 </div>
 <script
  data-section-id="header"
  src="https://unpkg.com/@teleporthq/teleport-custom-scripts"
 ></script>
</body>
</html>
```

Login - User

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <title>Ulogin - aesy</title>
  <meta property="og:title" content="Ulogin - aesy" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta charset="utf-8" />
  <meta property="twitter:card" content="summary_large_image" />

  <style data-tag="reset-style-sheet">
      html { line-height: 1.15;}body { margin: 0;}* { box-sizing: border-box; border-width: 0; border-style: solid;}p,li,ul,pre,div,h1,h2,h3,h4,h5,h6,figure,blockquote,figcaption { margin: 0; padding: 0;}button { background-color: transparent;}button,input,optgroup,select,textarea { font-family: inherit; font-size: 100%; line-height: 1.15; margin: 0;}button,select { text-transform: none;}button,[type="button"],[type="reset"],[type="submit"] { -webkit-appearance: button;}button::-
```

moz-focus-inner,[type="button"]::-moz-focus-inner,[type="reset"]::-moz-focus-inner,[type="submit"]::-moz-focus-inner { border-style: none; padding: 0;}button:-moz-focus,[type="button"]:-moz-focus,[type="reset"]:-moz-focus,[type="submit"]:-moz-focus { outline: 1px dotted ButtonText;}a { color: inherit; text-decoration: inherit;}input { padding: 2px 4px;}img { display: block;}html { scroll-behavior: smooth }

```
  </style>
  <style data-tag="default-style-sheet">
    html {
      font-family: Inter;
      font-size: 16px;
    }

    body {
      font-weight: 400;
      font-style:normal;
      text-decoration: none;
      text-transform: none;
      letter-spacing: normal;
      line-height: 1.15;
      color: var(--dl-color-gray-black);
      background-color: var(--dl-color-gray-white);

    }
  </style>
  <link
    rel="stylesheet"

href="https://fonts.googleapis.com/css2?family=Raleway:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&amp;display=swap"
    data-tag="font"
  />
  <link
```

```
      rel="stylesheet"

href="https://fonts.googleapis.com/css2?family=Inter:wght@100;200;300;400;500;600;700;800;
900&amp;display=swap"
      data-tag="font"
    />
    <!--This is the head section-->
    <!-- <style> ... </style> -->
    <link rel="stylesheet" href="./style.css" />
  </head>
  <body>
    <div>
      <link href="./ulogin.css" rel="stylesheet" />

      <div class="ulogin-container">
        <div class="ulogin-container1">
          <h1 class="ulogin-heading">User Login</h1>
          <form class="ulogin-form">
            <div class="ulogin-remember">
              <input
                type="checkbox"
                checked=""
                disabled=""
                required=""
                autocomplete="on"
                class="ulogin-checkbox"
              />
              <span class="ulogin-text">Remember me</span>
            </div>
          </form>
          <div class="ulogin-container2">
            <span class="ulogin-text01">
              <br />
```

```
      <br />
      <br />
      <span>E-mail :</span>
      <br />
      <br />
      <span>Password :</span>
      <br />
    </span>
    <button class="ulogin-button button">Log in</button>
  </div>
  <div class="ulogin-container3">
    <input
      type="email"
      target="email"
      required=""
      placeholder="placeholder"
      class="ulogin-textinput input"
    />
    <input
      type="password"
      required=""
      placeholder="placeholder"
      class="ulogin-textinput1 input"
    />
  </div>
</div>
<header data-role="Accordion" class="ulogin-header">
  <div class="ulogin-container4">
    <span class="ulogin-text10">AESY</span>
  </div>
  <div class="ulogin-separator"></div>
  <nav class="ulogin-nav">
    <nav
```

```
          class="navigation-links2-nav navigation-links2-root-class-name19"
        >
          <span class="navigation-links2-text"><span>About</span></span>
          <span class="navigation-links2-text1"><span>Home</span></span>
          <span class="navigation-links2-text2">
            <span>Contact Us</span>
          </span>
        </nav>
      </nav>
      <div data-role="AccordionHeader" class="ulogin-accordion-header">
        <svg viewBox="0 0 1024 1024" class="ulogin-icon">
          <path
            d="M128 554.667h768c23.552 0 42.667-19.115 42.667-42.667s-19.115-42.667-42.667-42.667h-768c-23.552 0-42.667 19.115-42.667 42.667s19.115 42.667 42.667 42.667zM128 298.667h768c23.552 0 42.667-19.115 42.667-42.667s-19.115-42.667-42.667-42.667h-768c-23.552 0-42.667 19.115-42.667 42.667s19.115 42.667 42.667 42.667zM128 810.667h768c23.552 0 42.667-19.115 42.667-42.667s-19.115-42.667-42.667-42.667h-768c-23.552 0-42.667 19.115-42.667 42.667s19.115 42.667 42.667 42.667z"
          ></path>
        </svg>
      </div>
      <div data-role="AccordionContent" class="ulogin-accordion-content">
        <div class="ulogin-nav1">
          <nav
            class="navigation-links2-nav navigation-links2-root-class-name20"
          >
            <span class="navigation-links2-text"><span>About</span></span>
            <span class="navigation-links2-text1"><span>Home</span></span>
            <span class="navigation-links2-text2">
              <span>Contact Us</span>
            </span>
          </nav>
        </div>
```

```
        </div>
       </header>
     </div>
   </div>
   <script
     data-section-id="header"
     src="https://unpkg.com/@teleporthq/teleport-custom-scripts"
   ></script>
  </body>
</html>
```

Login - Admin

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Alogin1 - aesy</title>
    <meta property="og:title" content="Alogin1 - aesy" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta charset="utf-8" />
    <meta property="twitter:card" content="summary_large_image" />

    <style data-tag="reset-style-sheet">
        html { line-height: 1.15;}body { margin: 0;}* { box-sizing: border-box; border-width: 0; border-style: solid;}p,li,ul,pre,div,h1,h2,h3,h4,h5,h6,figure,blockquote,figcaption { margin: 0; padding: 0;}button { background-color: transparent;}button,input,optgroup,select,textarea { font-family: inherit; font-size: 100%; line-height: 1.15; margin: 0;}button,select { text-transform: none;}button,[type="button"],[type="reset"],[type="submit"] { -webkit-appearance: button;}button::-moz-focus-inner,[type="button"]::-moz-focus-inner,[type="reset"]::-moz-focus-inner,[type="submit"]::-moz-focus-inner { border-style: none; padding: 0;}button:-moz-focus,[type="button"]:-moz-focus,[type="reset"]:-moz-focus,[type="submit"]:-moz-focus { outline: 1px dotted ButtonText;}a { color: inherit; text-decoration: inherit;}input { padding: 2px 4px;}img { display: block;}html { scroll-behavior: smooth }
```

```
  </style>
  <style data-tag="default-style-sheet">
   html {
    font-family: Inter;
    font-size: 16px;
   }

   body {
    font-weight: 400;
    font-style:normal;
    text-decoration: none;
    text-transform: none;
    letter-spacing: normal;
    line-height: 1.15;
    color: var(--dl-color-gray-black);
    background-color: var(--dl-color-gray-white);

   }
  </style>
  <link
   rel="stylesheet"

href="https://fonts.googleapis.com/css2?family=Raleway:ital,wght@0,100;0,200;0,300;0,400;0,5
00;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&amp;amp;display=
swap"
   data-tag="font"
  />
  <link
   rel="stylesheet"

href="https://fonts.googleapis.com/css2?family=Inter:wght@100;200;300;400;500;600;700;800;
900&amp;amp;display=swap"
   data-tag="font"
```

```
  />
  <!--This is the head section-->
  <!-- <style> ... </style> -->
  <link rel="stylesheet" href="./style.css" />
</head>
<body>
  <div>
    <link href="./alogin1.css" rel="stylesheet" />
    <div class="alogin1-container">
      <div class="alogin1-container1">
        <h1 class="alogin1-heading">Admin Log-in</h1>
        <form class="alogin1-form">
          <div class="alogin1-remember">
            <input
              type="checkbox"
              checked=""
              disabled=""
              required=""
              autocomplete="on"
              class="alogin1-checkbox"
            />
            <span class="alogin1-text">Remember me</span>
          </div>
        </form>
        <div class="alogin1-container2">
          <span class="alogin1-text01">
            <br />
            <br />
            <br />
            <span>Employee id :</span>
            <br />
            <br />
            <span>Password :</span>
```

```
      <br />
    </span>
    <button class="alogin1-button button">Log in</button>
  </div>
  <div class="alogin1-container3">
    <input
      type="text"
      target="email"
      required=""
      placeholder="placeholder"
      class="alogin1-textinput input"
    />
    <input
      type="password"
      required=""
      placeholder="placeholder"
      class="alogin1-textinput1 input"
    />
  </div>
</div>
<header data-role="Accordion" class="alogin1-header">
  <div class="alogin1-container4">
    <span class="alogin1-text10">AESY</span>
  </div>
  <div class="alogin1-separator"></div>
  <nav class="alogin1-nav">
    <nav
      class="navigation-links2-nav navigation-links2-root-class-name19"
    >
      <span class="navigation-links2-text"><span>About</span></span>
      <span class="navigation-links2-text1"><span>Home</span></span>
      <span class="navigation-links2-text2">
        <span>Contact Us</span>
```

```
          </span>
         </nav>
        </nav>
        <div data-role="AccordionHeader" class="alogin1-accordion-header">
         <svg viewBox="0 0 1024 1024" class="alogin1-icon">
          <path
                  d="M128 554.667h768c23.552 0 42.667-19.115 42.667-42.667s-19.115-42.667-
42.667-42.667h-768c-23.552 0-42.667 19.115-42.667 42.667s19.115 42.667 42.667
42.667zM128 298.667h768c23.552 0 42.667-19.115 42.667-42.667s-19.115-42.667-42.667-
42.667h-768c-23.552 0-42.667 19.115-42.667 42.667s19.115 42.667 42.667 42.667zM128
810.667h768c23.552 0 42.667-19.115 42.667-42.667s-19.115-42.667-42.667-42.667h-768c-
23.552 0-42.667 19.115-42.667 42.667s19.115 42.667 42.667 42.667z"
          ></path>
         </svg>
        </div>
        <div data-role="AccordionContent" class="alogin1-accordion-content">
         <div class="alogin1-nav1">
          <nav
           class="navigation-links2-nav navigation-links2-root-class-name20"
          >
           <span class="navigation-links2-text"><span>About</span></span>
           <span class="navigation-links2-text1"><span>Home</span></span>
           <span class="navigation-links2-text2">
            <span>Contact Us</span>
           </span>
          </nav>
         </div>
        </div>
       </header>
      </div>
     </div>
     <script
      data-section-id="header"
```

```
    src="https://unpkg.com/@teleporthq/teleport-custom-scripts"
  ></script>
 </body>
</html>
```

7.2. Feature 2

The second feature added to this application is embeding IBM watson chatbot intothe application to provide services about the application.

Chatbot:

```
<script>
        window.watsonAssistantChatOptions = {
          integrationID: "fea421dd-6f49-43dd-8c1e-016331e4ccc1", // The ID of this integration.
          region: "au-syd", // The region your integration is hosted in.
          serviceInstanceID: "a567db96-97a3-4d9d-acb4-9dbd4fc0c3d3", // 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>
```

7.3. Database schema:

```
import sqlite3

#Open database
conn = sqlite3.connect('database.db')

#Create table
conn.execute('''CREATE TABLE users
            (userId INTEGER PRIMARY KEY,
```

```
                password TEXT,
                email TEXT,
                firstName TEXT,
                lastName TEXT,
                address1 TEXT,
                address2 TEXT,
                zipcode TEXT,
                city TEXT,
                state TEXT,
                country TEXT,
                phone TEXT
                )'''')

conn.execute('''CREATE TABLE products
                (productId INTEGER PRIMARY KEY,
                name TEXT,
                price REAL,
                description TEXT,
                image TEXT,
                stock INTEGER,
                categoryId INTEGER,
                FOREIGN KEY(categoryId) REFERENCES categories(categoryId)
                )'''')

conn.execute('''CREATE TABLE kart
                (userId INTEGER,
                productId INTEGER,
                FOREIGN KEY(userId) REFERENCES users(userId),
                FOREIGN KEY(productId) REFERENCES products(productId)
                )'''')

conn.execute('''CREATE TABLE categories
                (categoryId INTEGER PRIMARY KEY,
```

```
    name TEXT
    )'")


conn.close()
```

8. TESTING

8.1 Test cases

Test Case ID	BU_001	**Test Case Description**	Test the Login Functionality of application		
Created By	Mark	**Reviewed By**	Bill	**Version**	2.1

QA Tester's Log	Review comments from Bill incorporate in version 2.1

Tester's Name	Ebinezer	**Date Tested**	17 nov 2022	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**
1	Access to Chrome Browser
2	Access to Internet explore
3	Access to opera
4	

S #	**Test Data**
1	Userid = mg12345
2	Pass = df12@434c
3	
4	

Test Scenario	Verify on entering valid userid and password, the customer can login

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**

1	Navigate to http://demo.guru99.com	Site should open	As Expected	Pass
2	Enter Userid & Password	Credential can be entered	As Expected	Pass
3	Click Submit	Customer is logged in	As Expected	Pass
4	Chatbot	Should have conversation	As Expected	pass
5	Database	Should be editable	As Expected	pass
6	Cart database	Shoud be working	As Expected	pass

9.Advantages & Disadvantages

Advantages

- Drive Traffic
- Deliver Relevant Content
- Engage Shoppers
- Convert Shoppers to Customers
- Increase Average Order Value
- Increase Number of Items per Order
- Control Merchandising and Inventory Rules
- Reduce Workload and Overhead

Disadvantages

- Significant investments required
- Too many choices
- The complex onboarding process
- Lack of data analytics capability
- The 'cold start' problem
- Inability to capture changes in user behavior
- Privacy concerns

10.Conclusion

The Fashion Recommendation System is mainly used to recommend the best possible outfit combinations to a user who has no fashion sense based on their wardrobe . It may not always provide the best possible outfit to wear for an occasion as the system is dependent completely on the clothes present in the user's wardrobe. Also another reason is that fashion is highly dependent on the time period. However the system does a great job in inculcating a fashion sense among the users and can provide the best recommendations based on the user's wardrobe. Since the system is implemented as a website, it is very easy for the end users to access as well as use. The scope of this system can be expanded by including the ability to detect the various design and patterns on clothing, and to increase the number of occasions

11. FUTURE SCOPE

The objective of recommender systems is to **provide recommendations based on recorded information on the users' preferences**. These systems use information filtering techniques to process information and provide the user with potentially more relevant items.

12. APPENDIX

Main.py

```
from flask import *
import sqlite3, hashlib, os
from werkzeug.utils import secure_filename

app = Flask(__name__)
app.secret_key = 'random string'
UPLOAD_FOLDER = 'static/uploads'
ALLOWED_EXTENSIONS = set(['jpeg', 'jpg', 'png', 'gif'])
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def getLoginDetails():
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
```

```
        if 'email' not in session:
            loggedIn = False
            firstName = ''
            noOfItems = 0
        else:
            loggedIn = True
            cur.execute("SELECT userId, firstName FROM users WHERE email = ?",
(session['email'], ))
            userId, firstName = cur.fetchone()
            cur.execute("SELECT count(productId) FROM kart WHERE userId = ?", (userId, ))
            noOfItems = cur.fetchone()[0]
    conn.close()
    return (loggedIn, firstName, noOfItems)

@app.route("/")
def root():
    loggedIn, firstName, noOfItems = getLoginDetails()
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
        cur.execute('SELECT productId, name, price, description, image, stock FROM
products')
        itemData = cur.fetchall()
        cur.execute('SELECT categoryId, name FROM categories')
        categoryData = cur.fetchall()
    itemData = parse(itemData)
    return render_template('home.html', itemData=itemData, loggedIn=loggedIn,
firstName=firstName, noOfItems=noOfItems, categoryData=categoryData)

@app.route("/add")
def admin():
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
        cur.execute("SELECT categoryId, name FROM categories")
        categories = cur.fetchall()
    conn.close()
    return render_template('add.html', categories=categories)
```

```
@app.route("/addItem", methods=["GET", "POST"])
def addItem():
    if request.method == "POST":
        name = request.form['name']
        price = float(request.form['price'])
        description = request.form['description']
        stock = int(request.form['stock'])
        categoryId = int(request.form['category'])

        #Uploading image procedure
        image = request.files['image']
        if image and allowed_file(image.filename):
            filename = secure_filename(image.filename)
            image.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        imagename = filename
        with sqlite3.connect('database.db') as conn:
            try:
                cur = conn.cursor()
                cur.execute('''INSERT INTO products (name, price, description, image, stock,
categoryId) VALUES (?, ?, ?, ?, ?, ?)''', (name, price, description, imagename, stock,
categoryId))
                conn.commit()
                msg="added successfully"
            except:
                msg="error occured"
                conn.rollback()
        conn.close()
        print(msg)
        return redirect(url_for('root'))

@app.route("/remove")
def remove():
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
        cur.execute('SELECT productId, name, price, description, image, stock FROM
products')
        data = cur.fetchall()
```

```
    conn.close()
    return render_template('remove.html', data=data)

@app.route("/removeItem")
def removeItem():
    productId = request.args.get('productId')
    with sqlite3.connect('database.db') as conn:
        try:
            cur = conn.cursor()
            cur.execute('DELETE FROM products WHERE productID = ?', (productId, ))
            conn.commit()
            msg = "Deleted successsfully"
        except:
            conn.rollback()
            msg = "Error occured"
    conn.close()
    print(msg)
    return redirect(url_for('root'))

@app.route("/displayCategory")
def displayCategory():
        loggedIn, firstName, noOfItems = getLoginDetails()
        categoryId = request.args.get("categoryId")
        with sqlite3.connect('database.db') as conn:
            cur = conn.cursor()
            cur.execute("SELECT products.productId, products.name, products.price,
products.image, categories.name FROM products, categories WHERE
products.categoryId = categories.categoryId AND categories.categoryId = ?",
(categoryId, ))
            data = cur.fetchall()
        conn.close()
        categoryName = data[0][4]
        data = parse(data)
        return render_template('displayCategory.html', data=data, loggedIn=loggedIn,
firstName=firstName, noOfItems=noOfItems, categoryName=categoryName)

@app.route("/account/profile")
```

```
def profileHome():
    if 'email' not in session:
        return redirect(url_for('root'))
    loggedIn, firstName, noOfItems = getLoginDetails()
    return render_template("profileHome.html", loggedIn=loggedIn, firstName=firstName,
noOfItems=noOfItems)

@app.route("/account/profile/edit")
def editProfile():
    if 'email' not in session:
        return redirect(url_for('root'))
    loggedIn, firstName, noOfItems = getLoginDetails()
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
        cur.execute("SELECT userId, email, firstName, lastName, address1, address2,
zipcode, city, state, country, phone FROM users WHERE email = ?", (session['email'], ))
        profileData = cur.fetchone()
    conn.close()
    return render_template("editProfile.html", profileData=profileData, loggedIn=loggedIn,
firstName=firstName, noOfItems=noOfItems)

@app.route("/account/profile/changePassword", methods=["GET", "POST"])
def changePassword():
    if 'email' not in session:
        return redirect(url_for('loginForm'))
    if request.method == "POST":
        oldPassword = request.form['oldpassword']
        oldPassword = hashlib.md5(oldPassword.encode()).hexdigest()
        newPassword = request.form['newpassword']
        newPassword = hashlib.md5(newPassword.encode()).hexdigest()
        with sqlite3.connect('database.db') as conn:
            cur = conn.cursor()
            cur.execute("SELECT userId, password FROM users WHERE email = ?",
(session['email'], ))
            userId, password = cur.fetchone()
            if (password == oldPassword):
                try:
```

```
                cur.execute("UPDATE users SET password = ? WHERE userId = ?",
(newPassword, userId))
                conn.commit()
                msg="Changed successfully"
            except:
                conn.rollback()
                msg = "Failed"
            return render_template("changePassword.html", msg=msg)
        else:
            msg = "Wrong password"
        conn.close()
        return render_template("changePassword.html", msg=msg)
    else:
        return render_template("changePassword.html")

@app.route("/updateProfile", methods=["GET", "POST"])
def updateProfile():
    if request.method == 'POST':
        email = request.form['email']
        firstName = request.form['firstName']
        lastName = request.form['lastName']
        address1 = request.form['address1']
        address2 = request.form['address2']
        zipcode = request.form['zipcode']
        city = request.form['city']
        state = request.form['state']
        country = request.form['country']
        phone = request.form['phone']
        with sqlite3.connect('database.db') as con:
            try:
                cur = con.cursor()
                cur.execute('UPDATE users SET firstName = ?, lastName = ?, address1 = ?,
address2 = ?, zipcode = ?, city = ?, state = ?, country = ?, phone = ? WHERE email = ?',
(firstName, lastName, address1, address2, zipcode, city, state, country, phone, email))

                con.commit()
                msg = "Saved Successfully"
```

```
            except:
                con.rollback()
                msg = "Error occured"
        con.close()
        return redirect(url_for('editProfile'))

@app.route("/loginForm")
def loginForm():
    if 'email' in session:
        return redirect(url_for('root'))
    else:
        return render_template('login.html', error='')

@app.route("/login", methods = ['POST', 'GET'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        if is_valid(email, password):
            session['email'] = email
            return redirect(url_for('root'))
        else:
            error = 'Invalid UserId / Password'
            return render_template('login.html', error=error)

@app.route("/productDescription")
def productDescription():
    loggedIn, firstName, noOfItems = getLoginDetails()
    productId = request.args.get('productId')
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
        cur.execute('SELECT productId, name, price, description, image, stock FROM
products WHERE productId = ?', (productId, ))
        productData = cur.fetchone()
    conn.close()
    return render_template("productDescription.html", data=productData, loggedIn =
loggedIn, firstName = firstName, noOfItems = noOfItems)
```

```
@app.route("/addToCart")
def addToCart():
    if 'email' not in session:
        return redirect(url_for('loginForm'))
    else:
        productId = int(request.args.get('productId'))
        with sqlite3.connect('database.db') as conn:
            cur = conn.cursor()
            cur.execute("SELECT userId FROM users WHERE email = ?", (session['email'], ))
            userId = cur.fetchone()[0]
            try:
                cur.execute("INSERT INTO kart (userId, productId) VALUES (?, ?)", (userId,
productId))
                conn.commit()
                msg = "Added successfully"
            except:
                conn.rollback()
                msg = "Error occured"
        conn.close()
        return redirect(url_for('root'))

@app.route("/cart")
def cart():
    if 'email' not in session:
        return redirect(url_for('loginForm'))
    loggedIn, firstName, noOfItems = getLoginDetails()
    email = session['email']
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
        cur.execute("SELECT userId FROM users WHERE email = ?", (email, ))
        userId = cur.fetchone()[0]
        cur.execute("SELECT products.productId, products.name, products.price,
products.image FROM products, kart WHERE products.productId = kart.productId AND
kart.userId = ?", (userId, ))
        products = cur.fetchall()
    totalPrice = 0
```

```
    for row in products:
        totalPrice += row[2]
    return render_template("cart.html", products = products, totalPrice=totalPrice,
loggedIn=loggedIn, firstName=firstName, noOfItems=noOfItems)

@app.route("/removeFromCart")
def removeFromCart():
    if 'email' not in session:
        return redirect(url_for('loginForm'))
    email = session['email']
    productId = int(request.args.get('productId'))
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
        cur.execute("SELECT userId FROM users WHERE email = ?", (email, ))
        userId = cur.fetchone()[0]
        try:
            cur.execute("DELETE FROM kart WHERE userId = ? AND productId = ?", (userId,
productId))
            conn.commit()
            msg = "removed successfully"
        except:
            conn.rollback()
            msg = "error occured"
    conn.close()
    return redirect(url_for('root'))

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

def is_valid(email, password):
    con = sqlite3.connect('database.db')
    cur = con.cursor()
    cur.execute('SELECT email, password FROM users')
    data = cur.fetchall()
    for row in data:
```

```
        if row[0] == email and row[1] == hashlib.md5(password.encode()).hexdigest():
            return True
    return False

@app.route("/register", methods = ['GET', 'POST'])
def register():
    if request.method == 'POST':
        #Parse form data
        password = request.form['password']
        email = request.form['email']
        firstName = request.form['firstName']
        lastName = request.form['lastName']
        address1 = request.form['address1']
        address2 = request.form['address2']
        zipcode = request.form['zipcode']
        city = request.form['city']
        state = request.form['state']
        country = request.form['country']
        phone = request.form['phone']

        with sqlite3.connect('database.db') as con:
            try:
                cur = con.cursor()
                cur.execute('INSERT INTO users (password, email, firstName, lastName,
address1, address2, zipcode, city, state, country, phone) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?)', (hashlib.md5(password.encode()).hexdigest(), email, firstName, lastName, address1,
address2, zipcode, city, state, country, phone))

                con.commit()

                msg = "Registered Successfully"
            except:
                con.rollback()
                msg = "Error occured"
        con.close()
        return render_template("login.html", error=msg)
```

```
@app.route("/registerationForm")
def registrationForm():
   return render_template("register.html")

def allowed_file(filename):
   return '.' in filename and \
         filename.rsplit('.', 1)[1] in ALLOWED_EXTENSIONS

def parse(data):
   ans = []
   i = 0
   while i < len(data):
      curr = []
      for j in range(7):
         if i >= len(data):
            break
         curr.append(data[i])
         i += 1
      ans.append(curr)
   return ans

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

Home.html:

```
<!DOCTYPE HTML>
<html>
<head>
<title>Welcome</title>
<link rel="stylesheet" href={{ url_for('static', filename='css/home.css') }} />
<link rel="stylesheet" href={{ url_for('static', filename='css/topStyle.css') }} />
</head>
<body>
```

```
<div id="title">
	<a href="/">
		<img id="logo" src= {{ url_for('static', filename='images/logo.png') }} />
	</a>
	<form>
		<input id="searchBox" type="text" name="searchQuery">
		<input id="searchButton" type="submit" value="Search">
	</form>

	{% if not loggedIn %}
	<div id="signInButton">
		<a class="link" href="/loginForm">Sign In</a>
	</div>
	{% else %}
	<div class="dropdown">
		<button class="dropbtn">Hello, <br>{{firstName}}</button>
		<div class="dropdown-content">
			<a href="/account/orders">Your orders</a>
			<a href="/account/profile">Your profile</a>
			<hr>
			<a href="/logout">Sign Out</a>
		</div>
	</div>
	{% endif %}
	<div id="kart">
		<a class="link" href="/cart">
			<img src={{url_for('static', filename='images/shoppingCart.png')}}
id="cartIcon" />
			CART {{noOfItems}}
```

```
                </a>
        </div>
</div>
<div class="display">
        <div class="displayCategory">
                <h2>Shop by Category: </h2>
                <ul>
                        {% for row in categoryData %}
                        <li><a
href="/displayCategory?categoryId={{row[0]}}">{{row[1]}}</a></li>
                        {% endfor %}
                </ul>
        </div>
        <div>
                <h2>Items</h2>
                {% for data in itemData %}
                <table>
                        <tr id="productName">
                                {% for row in data %}
                                <td>
                                        {{row[1]}}
                                </td>
                                {% endfor %}
                        </tr>
                        <tr id="productImage">
                                {% for row in data %}
                                <td>
                                        <a href="/productDescription?productId={{row[0]}}">
                                                <img  src={{ url_for('static', filename='uploads/'
```

```
+ row[4]) }} id="itemImage" />
                                        </a>
                                    </td>
                                    {% endfor %}
                                </tr>
                                <tr id="productPrice">
                                    {% for row in data %}
                                    <td>
                                        ${{row[2]}}
                                    </td>
                                    {% endfor %}
                                </tr>
                            </table>
                            {% endfor %}
                    </div>
            </div>
            <script>
                    window.watsonAssistantChatOptions = {
                      integrationID: "fea421dd-6f49-43dd-8c1e-016331e4ccc1", // The ID of this integration.
                      region: "au-syd", // The region your integration is hosted in.
                      serviceInstanceID: "a567db96-97a3-4d9d-acb4-9dbd4fc0c3d3", // The ID of your service instance.
                      onLoad: function(instance) { instance.render(); }
                    };
                    setTimeout(function(){
                      const t=document.createElement('script');
                      t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" + (window.watsonAssistantChatOptions.clientVersion || 'latest') +
```

```
"/WatsonAssistantChatEntry.js";
            document.head.appendChild(t);
          });
  </script>
</body>
</html>
```

Database.py:

```
import sqlite3

#Open database
conn = sqlite3.connect('database.db')

#Create table
conn.execute('''CREATE TABLE users
                    (userId INTEGER PRIMARY KEY,
                    password TEXT,
                    email TEXT,
                    firstName TEXT,
                    lastName TEXT,
                    address1 TEXT,
                    address2 TEXT,
                    zipcode TEXT,
                    city TEXT,
                    state TEXT,
                    country TEXT,
                    phone TEXT
                    )''')
```

```
conn.execute('''CREATE TABLE products
                (productId INTEGER PRIMARY KEY,
                name TEXT,
                price REAL,
                description TEXT,
                image TEXT,
                stock INTEGER,
                categoryId INTEGER,
                FOREIGN KEY(categoryId) REFERENCES categories(categoryId)
                )''')

conn.execute('''CREATE TABLE kart
                (userId INTEGER,
                productId INTEGER,
                FOREIGN KEY(userId) REFERENCES users(userId),
                FOREIGN KEY(productId) REFERENCES products(productId)
                )''')

conn.execute('''CREATE TABLE categories
                (categoryId INTEGER PRIMARY KEY,
                name TEXT
                )''')



conn.close()
```

GitHub link:

https://github.com/IBM-EPBL/IBM-Project-28054-1660106140

Demo link: