SMART FASHION RECOMMENDER APPLICATION

HX 8001- Professional Readiness For Innovation, Employability and Entrepreneurship

IBM-Project-44922-1660727459

TEAM ID : PNT2022TMID31802

Submitted by

ELAVARASAN E (721219104016)

JAKITH AHAMED A (721219104019)

LOGHA PRIYA A (721219104028)

SAMRAJ S (721219104041)

SANMATHI M (721219104042)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

ABSTRACT

Fashion is perceived as a meaningful way of self-expressing that people use for different purposes. It seems to be an integral part of every person in modern societies, from everyday life to exceptional events and occasions. Fashionable products are highly demanded, and consequently, fashion is perceived as a desirable and profitable industry. Although this massive demand for fashion products provides an excellent opportunity for companies to invest in fashion-related sectors, it also faces different challenges in answering their customer needs.

In recent years, the textile and fashion industries have witnessed an enormous amount of growth in fast fashion. On e-commerce platforms, where numerous choices are available, an efficient recommendation system is required to sort, order, and efficiently convey relevant product content or information to users. Smart Fashion Recommender Application have attracted a huge amount of attention from fast fashion retailers as they provide a personalized shopping experience to consumers. Smart Fashion Recommender Application have been introduced to address these needs.

1 INTRODUCTION

1.1 OVERVIEW:

The project, Smart Fashion Recommender Application is an idea based on purchase. The main motive of this project is to share fashion products with multiple users.

The Smart Fashion Recommender Application for online purchase. Fashion is a kind of symbol that represents people's internal perception through their outer appearance, it conveys information about their choice's faith personality profession social status and attitude towards life. Users use this application for purchasing their own needs. The platform allows the users to searching the products and purchase in online mode and Also user add their favorite product in cart. The recent technological advancements have enabled consumers to track current fashion trends around the globe which influence their choice. The fashion choices of consumers depend on many factors such as demographics, geographic location, individual preferences, interpersonal influences, age, gender, season and culture. Moreover, previous fashion recommendation research shows that fashion preferences vary not only from country to country but also from city to city.

The User can track the status order by logging into their accounts where the current order is present along with all his previous orders so that they can check their previous orders also. User can get real time updates of his orders so he/she can be readily available to collecting his/her delivery.

1.2 PURPOSE:

Fashion products may be available in some of the metro or urban areas but the desired fashion products the people want will not be available in all kind of rural areas. A recommendation system is a system that is programmed to predict future preferable items from a large set of collections. This system works either by using user preferences or by using the items most preferred by all users. The main challenge in building a fashion recommendation system is that it is a very dynamic industry. It changes very often when it comes to seasons, festivals, pandemic conditions like corona virus and many more. To overcome this problem, sharing of fashion products via internet by developing an application.

2 LITERATURE SURVEY

2.1 Existing problem:

This application is intended to provide information about fashion industries have witnessed an enormous amount of growth in fast fashion, where numerous choices are available, an efficient recommendation system is required to sort, order, and efficiently convey relevant product content or information to users.

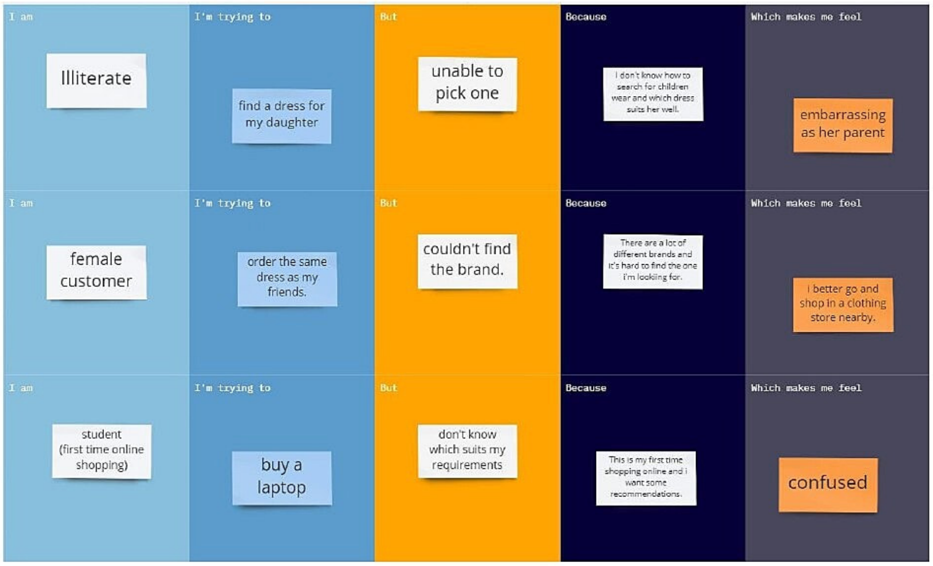
Smart fashion recommendation application has attracted a huge amount of attention from fast fashion retailers as they provide a personalized shopping experience to consumers. With the technological advancements, this branch of artificial intelligence exhibits a tremendous amount of potential in image processing, parsing, classification, and segmentation.

2.2 References:

https://play.google.com/store/apps/details?id=com.yourclosetapp.app.freecloset

2.3 Problem Statement Definition:

Smart fashion recommendation application has attracted a huge amount of attention from fast fashion retailers as they provide a personalized shopping experience to consumers. With the technological advancements, this branch of artificial intelligence exhibits a tremendous amount of potential in image processing, parsing, classification, and segmentation.

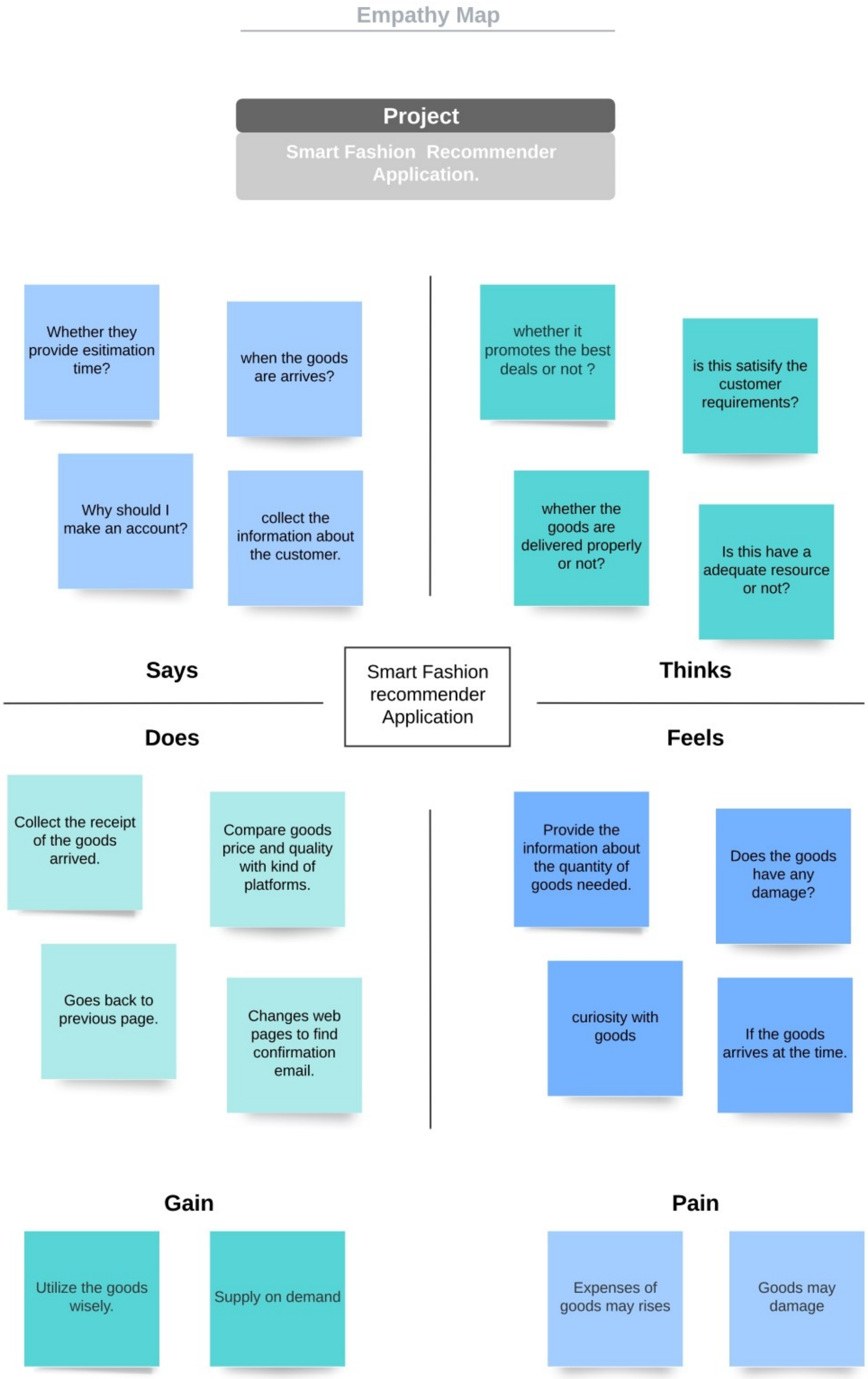


I am	I'm trying to	But	Because	Which makes me feel
Illiterate	find a dress for my daughter	unable to pick one	I don't know how to search for children wear and which dress suits her well.	embarrassing as her parent
female customer	order the same dress as my friends.	couldn't find the brand.	There are a lot of different brands and it's hard to find the one i'm lookiing for.	i better go and shop in a clothing store nearby.
student (first time online shopping)	buy a laptop	don't know which suits my requirements	This is my first time shopping online and i want some recommendations.	confused

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Illiterate	find a dress for my daughter	unable to pick one	I don't know how to search for children wear and which dress suits her well.	embarrassing as her parent
PS-2	female customer	order the same dress as my friends.	couldn't find the brand.	There are a lot of different brands and it's hard to find the one i'm lookiing for.	i better go and shop in a clothing store nearby.
PS-3	student	buy a laptop	don't know which suits my requirements	This is my first time shopping online and i want some recommendations.	confused

3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas:



Empathy Map

Project

Smart Fashion Recommender Application.

Smart Fashion recommender Application

Says

Whether they provide esitimation time?

when the goods are arrives?

Why should I make an account?

collect the information about the customer.

Thinks

whether it promotes the best deals or not ?

is this satisify the customer requirements?

whether the goods are delivered properly or not?

Is this have a adequate resource or not?

Does

Collect the receipt of the goods arrived.

Compare goods price and quality with kind of platforms.

Goes back to previous page.

Changes web pages to find confirmation email.

Feels

Provide the information about the quantity of goods needed.

Does the goods have any damage?

curiosity with goods

If the goods arrives at the time.

Gain

Utilize the goods wisely.

Supply on demand

Pain

Expenses of goods may rises

Goods may damage

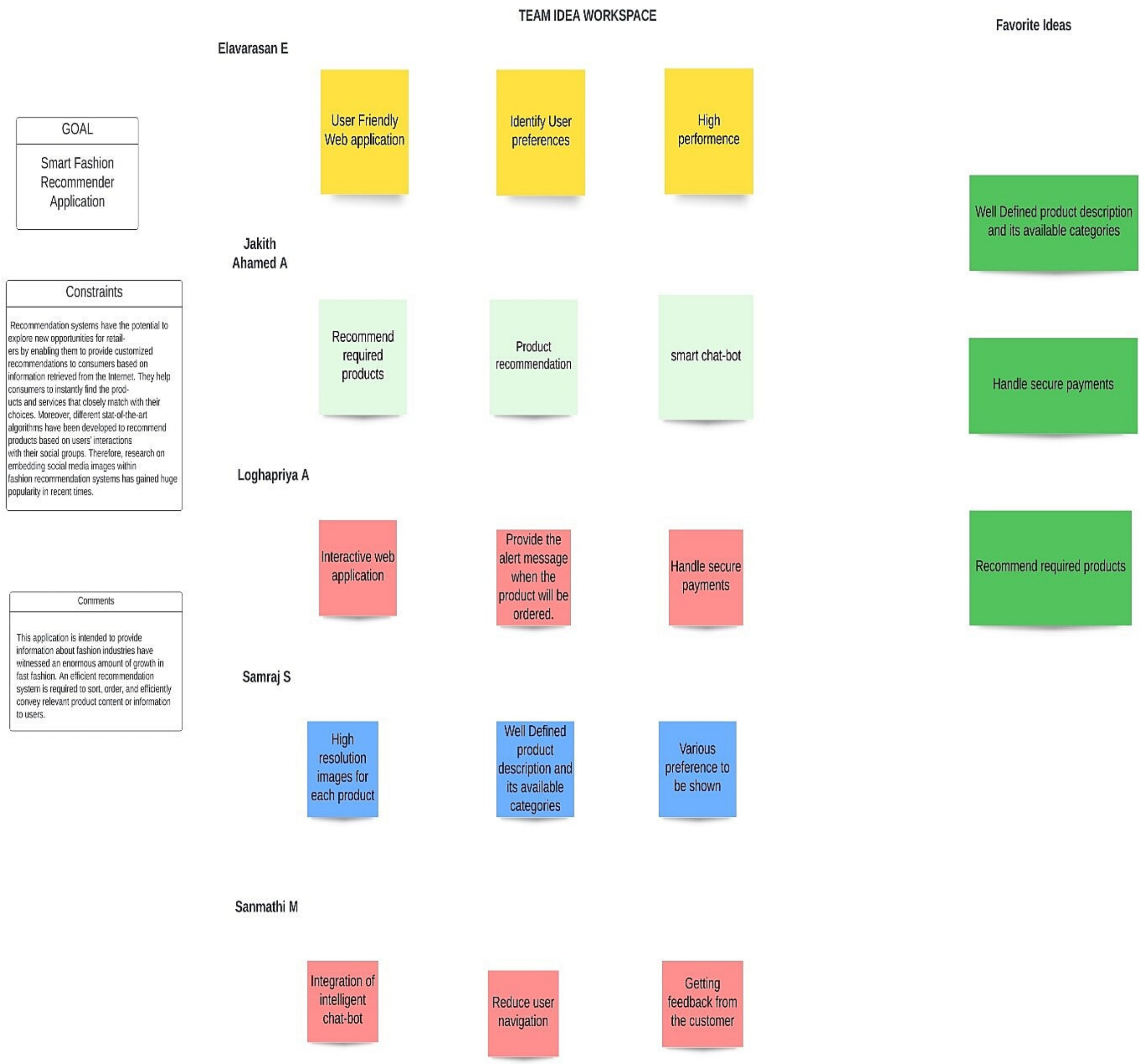
An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. This tool helps build empathy towards users and helps design teams shift focus from the product to the users who are going to use the product. Empathy map visualize customer needs, condense customer data into a brief chart.

Empathy maps can be used whenever you find a need to immerse yourself in a user's environment. Empathy maps can also be read and understood quite easily, making them a great tool for communicating information about the user to other members of the design team. It's critical to help others on your team and in your company to cultivate a deep understanding of the user's behaviours and empathy for their needs. This helps ensure that users' needs will take CenterStage in design decision making, since everyone contributing to the product's development can work to serve the same set of personas that reflect the same set of needs and goals.

3.2 Ideation & Brainstorming:

Brainstorming, a specific technique that is utilized to generate new ideas. It ideation is commonly more thought as being an individual pursuit, while brainstorming is almost always a group activity.

Brainstorming is a great way to generate many ideas by leveraging the collective thinking of the group, engaging with each other, listening, and building an others ideas. It allows people to think more freely without fear of judgment, it encourages open and ongoing collaboration to solve problems and generate innovative ideas, it helps teams to generate a large number of ideas quickly which can be refined and merged to create the ideal solution.



GOAL

Smart Fashion Recommender Application

Constraints

Recommendation systems have the potential to explore new opportunities for retailers by enabling them to provide customized recommendations to consumers based on information retrieved from the Internet. They help consumers to instantly find the products and services that closely match with their choices. Moreover, different stat-of-the-art algorithms have been developed to recommend products based on users' interactions with their social groups. Therefore, research on embedding social media images within fashion recommendation systems has gained huge popularity in recent times.

Comments

This application is intended to provide information about fashion industries have witnessed an enormous amount of growth in fast fashion. An efficient recommendation system is required to sort, order, and efficiently convey relevant product content or information to users.

TEAM IDEA WORKSPACE

Elavarasan E

- User Friendly Web application
- Identify User preferences
- High performence

Jakith Ahamed A

- Recommend required products
- Product recommendation
- smart chat-bot

Loghapriya A

- Interactive web application
- Provide the alert message when the product will be ordered.
- Handle secure payments

Samraj S

- High resolution images for each product
- Well Defined product description and its available categories
- Various preference to be shown

Sanmathi M

- Integration of intelligent chat-bot
- Reduce user navigation
- Getting feedback from the customer

Favorite Ideas

- Well Defined product description and its available categories
- Handle secure payments
- Recommend required products

3.3 Proposed Solution:

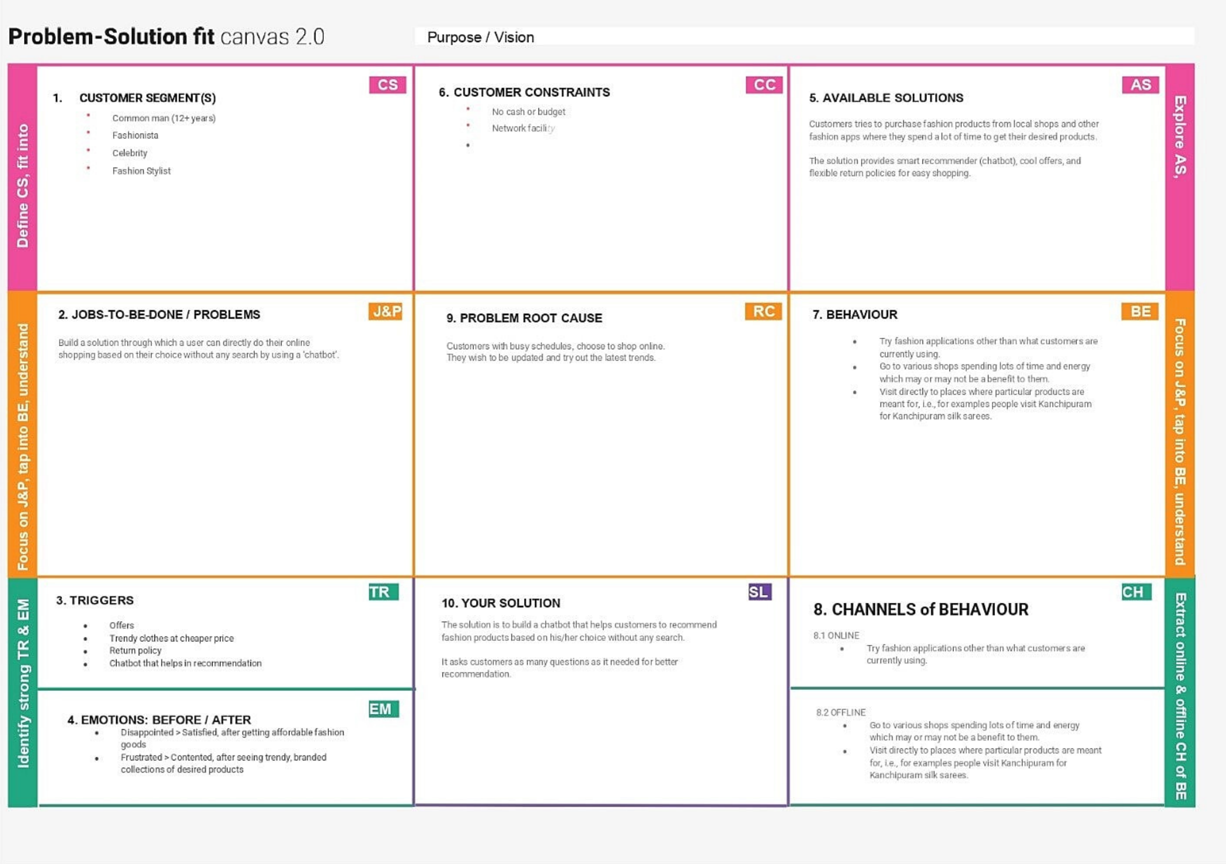
Proposed solution should related the current situation to a desired result and describe the benefits that will accrue when the desired is achived.

S. NO	**Parameter**	**Description**
1.	Problem Statement (Problem to be solved)	Fashion Recommender system with an increase in the standard of living, peoples' attention gradually moved towards fashion that is concerned to be a popular aesthetic expression. However, given too many options of garments on that has presented new challenges to the customers in identifying their correct outfit.
2.	Idea / Solution description	The goal of a recommender system is to provide personalized suggestions to users, based on large volumes of historical Feedback, by uncovering hidden dimensions that describe the Preferences of users and the properties of the items they consume.
3.	Novelty/ Uniqueness	The novelty of the project is for collections generated by the model to contain some aspects of the input but with serendipity to pleasantly surprise users.

4.	Social Impact / Customer Satisfaction	Users need to manually select their preferences like size, cost etc...
5.	Business Model (Revenue Model)	Fashion recommendation systems and methods to personalize clothing are also included in this subcategory, in fact, these systems are meant to develop collaborative filtering to predict user preferences online, when data from purchase history are lacking, as well as content-based filtering, to support consumers’ decision-making process, improve the customer experience and increase sales.
6.	Scalability of the Solution	The extracted results can then be evaluated by the designer and the preferred products can be saved on their dashboard over time and for each product search. If the user is not satisfied by the recommendations, they have the ability either to renew their preferences or ask for new recommendations

3.4 Problem Solution fit:

Problem solution fit that you have found a problem with customer and that the solution you have realized for its actually solves the customer problem.



Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into

1. CUSTOMER SEGMENT(S) CS

- Common man (12+ years)
- Fashionista
- Celebrity
- Fashion Stylist

6. CUSTOMER CONSTRAINTS CC

- No cash or budget
- Network facility
-

5. AVAILABLE SOLUTIONS AS

Customers tries to purchase fashion products from local shops and other fashion apps where they spend a lot of time to get their desired products.

The solution provides smart recommender (chatbot), cool offers, and flexible return policies for easy shopping.

Explore AS,

Focus on J&P, tap into BE, understand

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

Build a solution through which a user can directly do their online shopping based on their choice without any search by using a 'chatbot'.

9. PROBLEM ROOT CAUSE RC

Customers with busy schedules, choose to shop online. They wish to be updated and try out the latest trends.

7. BEHAVIOUR BE

- Try fashion applications other than what customers are currently using.
- Go to various shops spending lots of time and energy which may or may not be a benefit to them.
- Visit directly to places where particular products are meant for, i.e., for examples people visit Kanchipuram for Kanchipuram silk sarees.

Focus on J&P, tap into BE, understand

Identify strong TR & EM

3. TRIGGERS TR

- Offers
- Trendy clothes at cheaper price
- Return policy
- Chatbot that helps in recommendation

4. EMOTIONS: BEFORE / AFTER EM

- Disappointed > Satisfied, after getting affordable fashion goods
- Frustrated > Contented, after seeing trendy, branded collections of desired products

10. YOUR SOLUTION SL

The solution is to build a chatbot that helps customers to recommend fashion products based on his/her choice without any search.

It asks customers as many questions as it needed for better recommendation.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

- Try fashion applications other than what customers are currently using.

8.2 OFFLINE

- Go to various shops spending lots of time and energy which may or may not be a benefit to them.
- Visit directly to places where particular products are meant for, i.e., for examples people visit Kanchipuram for Kanchipuram silk sarees.

Extract online & offline CH of BE

4.REQUIREMENT ANALYSIS

4.1 Functional requirement:

Following are the functional requirements of the proposed solution:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through registration Form, Gmail, mobile number.
FR-2	User Confirmation	User confirmation via email and email – OTP
FR-3	Live chat - Chatbot	User recommendations can be made by the chatbot depending on their interests. It may advertise the day's top specials and promotions. It will keep a database of the customer's information and orders. If the order is accepted, the chatbot will notify the customers. Additionally, chatbots can be used to gather customer feedback.
FR-4	Checking item availability	Item availability in specific locations
FR-5	Shopping cart	My cart button, Add-to-cart button, Remove-from-cart button.
FR-6	Super-fast checkout	Online transfer, Credit card payment, Paying with mobile wallets

FR-7	Checking the shipping status	Option to easily check the shipping status of items ordered in the store.

4.2 Non-Functional requirements:

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

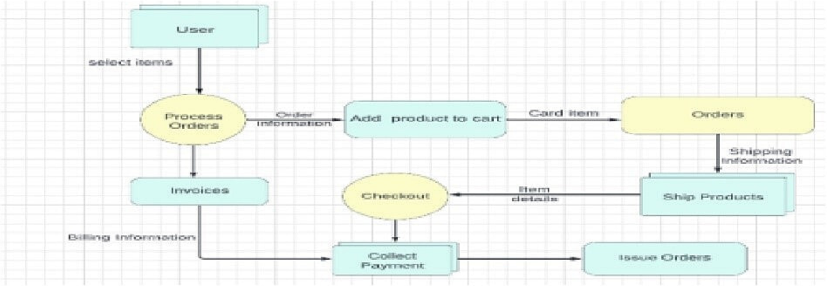
FR. NO	**Non-Functional Requirements**	**Description**
NFR-1	Usability	If people search on google for a product you offer it should be on the first page of result and good quality images that will attract buyers.
NFR-2	Security	This Application will collect a lot of users’ private information to complete a purchase (banking, shipping/home address, email, etc.) Data protection is the priority.
NFR-3	Reliability	Ability of the software to perform critical tasks like collecting and securing customer data, providing payment gateway to function correctly in a given environment, for a particular amount of time.
NFR-4	Performance	Speed up the webpage and Site optimization based on the data analysis.

		Good use of the product description.
NFR-5	Availability	The administrator needs to look up the stock availability in the database.
NFR-6	Scalability	Having a plan to handle demand peaks. Avoid downtime, preserve the customer experience, and ensure deliveries go out on time at all costs. Chatbots to provide scalable customer Support.

5.PROJECT DESIGN

5.1 Data Flow Diagrams:

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement.

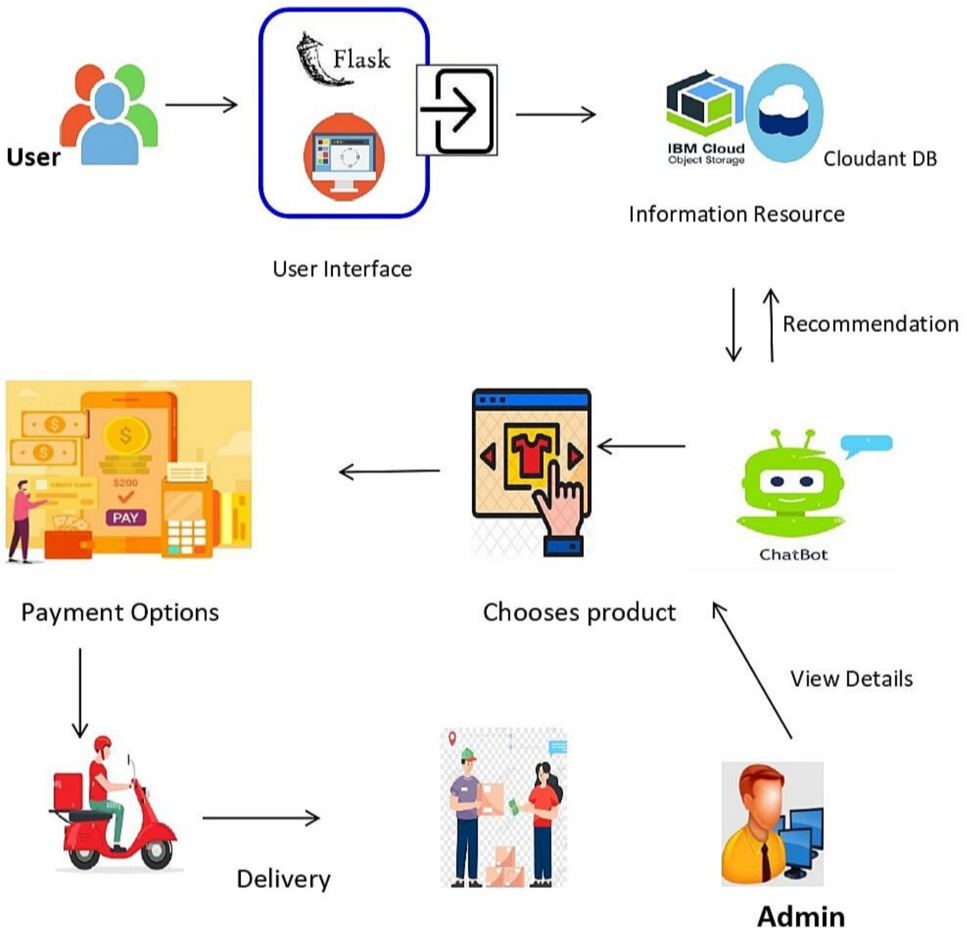


A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various subprocesses the data moves through. DFDs are built using standardized symbols and notation to describe various entities and their relationships.

5.2 Solution & Technical Architecture:

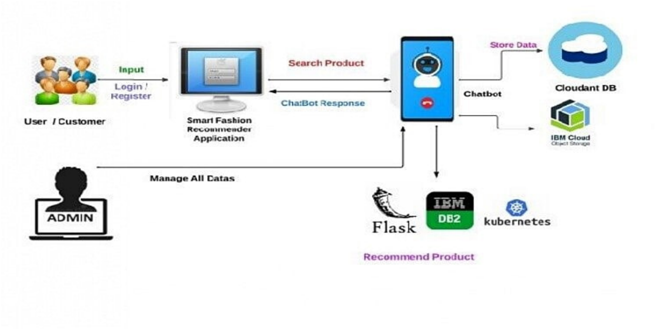
Solution Architecture:

Solution architecture in the context of software development, you first need to think about what a solution is. Even though this might seem quite basic, it illustrates why solution architecture is one of the most important processes when re-designing your IT landscape.



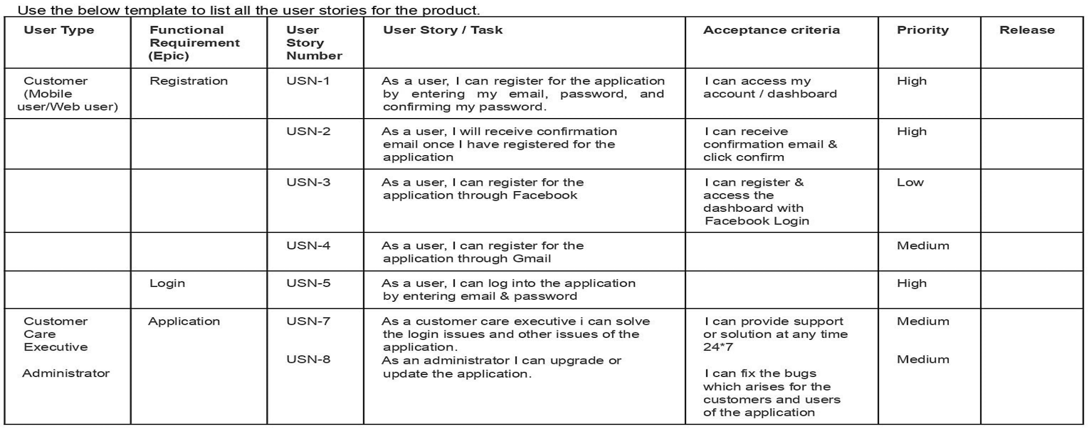
Technical Architecture:

Technical Architecture (TA) is a form of IT architecture that is used to design computer systems. It involves the development of a technical blueprint with regard to the arrangement, interaction, and interdependence of all elements so that system-relevant requirements are met.



5.3 User Stories:

A user story is an informal, general explanation of a software feature written from the perspective of the end user or customer. The purpose of a user story is to articulate how a piece of work will deliver a particular value back to the customer.



Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user/Web 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	
		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	
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	
		USN-4	As a user, I can register for the application through Gmail		Medium	
	Login	USN-5	As a user, I can log into the application by entering email & password		High	
Customer Care Executive Administrator	Application	USN-7 USN-8	As a customer care executive i can solve the login issues and other issues of the application. As an administrator I can upgrade or update the application.	I can provide support or solution at any time 24*7 I can fix the bugs which arises for the customers and users of the application	Medium Medium	

6.PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation:

Title	Description	Date
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information by referring the technical papers, research publications, journals etc.	29 AUGUST 2022
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem Statements that are to be solved by this project.	5 SEPTEMBER 2022
Ideation	List the ideas by organizing a brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	12 SEPTEMBER 2022
Proposed Solution	Prepare the proposed solution document, which includes novelty, feasibility of idea, revenue model, social impact, scalability of solution, etc.	19 SEPTEMBER 2022
Problem Solution Fit	Prepare problem - solution fit document.	26 SEPTEMBER 2022
Solution Architecture	Prepare solution architecture document.	1 OCTOBER 2022

Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	3 OCTOBER 2022
Functional Requirement	Prepare the functional requirement document.	10 OCTOBER 2022
Data Flow Diagrams	Draw the data flow diagrams and submit for review.	15 OCTOBER 2022
Technology Architecture	Prepare the technology architecture diagram.	15 OCTOBER 2022
Prepare Milestone & Activity List	Prepare the milestones & activity list of the project.	17 OCTOBER 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	17 NOVEMBER 2022

6.2 Sprint Delivery Schedule:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Admin Panel	USN-1	The user will login into the website and go through the products available on the website	20	High	E Elavarasan A Jakith Ahamed A Logha Priya S Samraj M Sanmathi
Sprint-2	User 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	E Elavarasan A Jakith Ahamed A Logha Priya S Samraj M Sanmathi
Sprint-3	Chatbot	USN-3	The user can directly talk to Chatbot regarding the products. Get the recommendations based on information provided by the user	20	High	E Elavarasan A Jakith Ahamed A Logha Priya S Samraj M Sanmathi

Sprint-4	Testing & deploy	USN-4	Container of applications using docker Kubernetes and deployment the application.	20	High	E Elavarasan A Jakith Ahamed A Logha Priya S Samraj M Sanmathi

AV= Sprint Duration/Velocity= 20/10=2

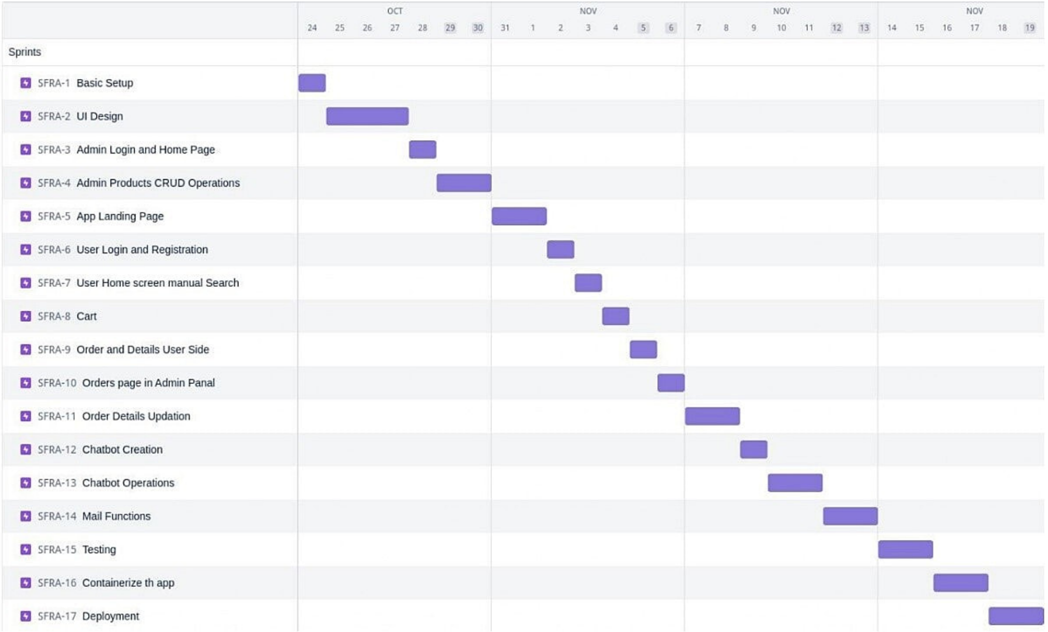
Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Story Points Completed (As on Planned End Date)**	**Sprint Release Date (Actual)**
Sprint-1	20	6 days	24 Oct 2022		29 Oct 2022
Sprint-2	20	6 days	05 Nov 2022		05 Nov 2022
Sprint-3	20	6 days	12 Nov 2022		12 Nov 2022
Sprint-4	20	6 days	19 Nov 2022		19 Nov 2022

Velocity:

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

per iteration unit (story points per day)

6.3 Reports from JIRA:



Sprints

SFRA-1 Basic Setup

SFRA-2 UI Design

SFRA-3 Admin Login and Home Page

SFRA-4 Admin Products CRUD Operations

SFRA-5 App Landing Page

SFRA-6 User Login and Registration

SFRA-7 User Home screen manual Search

SFRA-8 Cart

SFRA-9 Order and Details User Side

SFRA-10 Orders page in Admin Panal

SFRA-11 Order Details Updation

SFRA-12 Chatbot Creation

SFRA-13 Chatbot Operations

SFRA-14 Mail Functions

SFRA-15 Testing

SFRA-16 Containerize th app

SFRA-17 Deployment

7.CODING & SOLUTION

7.1 Feature:

HOME PAGE:

```
 <!DOCTYPE html>
<html lang="en">
 <head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <title>Document</title>
 <link
 rel="stylesheet"
 href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
 integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
 crossorigin="anonymous"
 />
 <link
 rel="stylesheet"
 href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
```

```
integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
crossorigin="anonymous"
/>
<link rel="stylesheet" href="Style.css" />
</head>
<body>
<!-- <div class="container"> -->
<div class="box">
<div class="row">
<div class="col-sm-5 col-xs-1 box1">
<div class="inline-text">
<h1>Login</h1>
<p>
If you can't stop thinking<br/>
About it...<br />
Just buy it.
</p>
<img src="C:\Users\sanmathi\Downloads\logu.jpg" width="60px"; height="60px">
</div>
```

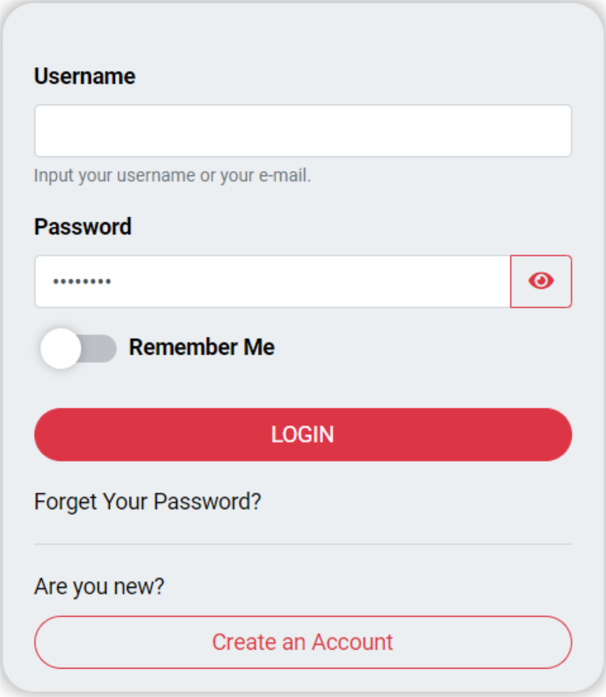
```html
</div>
<div class="col-sm-6 col-xs-1 box2">
<div class="user-id user-data">
<input type="email " name="" id="" required=""/>
<label>Enter Email</label>
</div>
<div class="user-id user-data">
<input type="password" name="" id="" required=""/>
<label>Enter Password</label>


</div>


<span><a href="#">Forgot?</span></a>
<div class="user-id button">
<input type="submit" name="" id="" value="Login" />
</div>
<div class="user-id">
<p class="footer"><a href="file:///C:/Users/sanmathi/Downloads/flipcart%20login%20form/flipcart/signup.html?txt=Logha+Priya&email=loghapriyaa%40gmail.com&pswd=Loghu%400709&pswd=Loghu%400709">New User? Create an account</a></p>
```

```
    </div>

    </div>

    </div>

    </div>

    </div>

    </body>

   </html>
```

Login.css:

```
body{

 margin:0;

 padding:0;

 background:#000;

}

.box{

 border-radius:3px;

 position: absolute;

 top:50%;

 left:50%;

 transform:translate(-50%,-50%);

 width:750px;
 height: 560px;

 background:white;

 overflow: hidden;

}

.inline-text{

 color:white;

 position: absolute;

 top: 32px;

 left: 55px;



}

.inline-text h1{

font-size:25px;

margin-bottom: 20px;

}

.inline-text p{

 color:rgb(228, 228, 228);

 opacity:0.9;

 font-size:13px;

 letter-spacing: 1px;
```

```
}

.box1{



 background:url(//img1a.flixcart.com/www/linchpin/fk-cp-zion/img/login_img_dec4bf.png);

 height: 100vh;

 background-position:center;

 background-repeat: no-repeat;

 background-color:#808080;

}

.box2 .user-id{

 position: relative;

 left:20px;

 top:50px;

 padding:10px 20px;



}

.box2 .user-id input{

 width:100%;

 height: 40px;

}
```



Signup Page:

```
<!DOCTYPE html>

<html>

<head>

<title>Smart Fashion Recommender Application</title>

<link rel="stylesheet" type="text/css" href="sign.css">

<link href="https://fonts.googleapis.com/css2?family=Jost:wght@500&display=swap" rel="stylesheet">

</head>

<body>

<div class="main">

<input type="checkbox" id="chk" aria-hidden="true">
```

```
<div class="signup">

<form>

<label for="chk" aria-hidden="true">Sign up</label>

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

<input type="email" name="email" placeholder="Email" required="">

<input type="password" name="pswd" placeholder="Create Password" required="">

 <input type="password" name="pswd" placeholder="Conform Password" required="">

<button>SUBMIT</button>

</form>

</div>

</div>

</body>

</html>
```

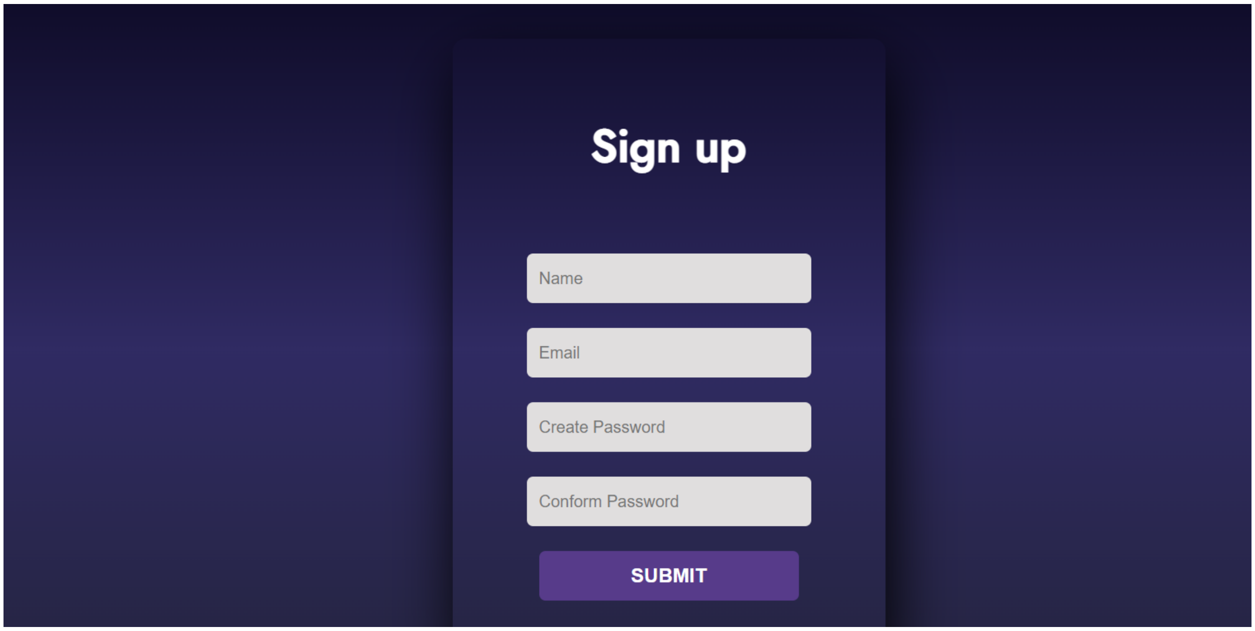
Signup.css:

```
body{

margin: 0;

padding: 0;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;
```

```
font-family: 'Jost', sans-serif;
background: linear-gradient(to bottom, #0f0c29, #302b63, #24243e);
}
.main{
width: 350px;
height: 500px;
background: red;
overflow: hidden;
background: url("https://doc-08-2c-docs.googleusercontent.com/docs/securesc/68c90smiglihng9534mvqmq1946dmis5/fo0picsp1nhiucmc0l25s29respgpr4j/1631524275000/03522360960922298374/03522360960922298374/1Sx0jhdpEpnNIydS4rnN4kHSJtU1EyWka?e=view&authuser=0&nonce=gcrocepgbb17m&user=03522360960922298374&hash=tfhgbs86ka6divo3llbvp93mg4csvb38") no-repeat center/ cover;
border-radius: 10px;
box-shadow: 5px 20px 50px #000;
}
#chk{
display: none;
}
.signup{
position: relative;
width:100%;
height: 100%;
}
label{
```

```
color: #fff;
font-size: 2.3em;
justify-content: center;
display: flex;
margin: 60px;
font-weight: bold;
cursor: pointer;
transition: .5s ease-in-out;
}
input{
width: 60%;
height: 20px;
background: #e0dede;
justify-content: center;
display: flex;
margin: 20px auto;
padding: 10px;
border: none;
outline: none;
border-radius: 5px;
}
button{
width: 60%;
```

```
height: 40px;
margin: 10px auto;
justify-content: center;
display: block;
color: #fff;
background: #573b8a;
font-size: 1em;
font-weight: bold;
margin-top: 20px;
outline: none;
border: none;
border-radius: 5px;
transition: .2s ease-in;
cursor: pointer;
}
button:hover{
background: #6d44b8;
}
#chk:checked ~ .signup label{
transform: scale(.6);
}
```



Sign up

Name

Email

Create Password

Conform Password

7.2 Feature 2:

Admin Page:

index.html

```
<!DOCTYPE html>

<html lang="en" dir="ltr">

 <head>

 <meta charset="utf-8">

 <title>Login & Signup Form | CodingNepal</title>

 <link rel="stylesheet" href="style.css">

 <meta name="viewport" content="width=device-width, initial-scale=1.0">

 </head>

 <body>
```

```
<div class="wrapper">
<div class="title-text">
<div class="title login">Admin Login</div>
<div class="title signup">Admin Signup</div>
</div>
<div class="form-container">
<div class="slide-controls">
<input type="radio" name="slide" id="login" checked>
<input type="radio" name="slide" id="signup">
<label for="login" class="slide login">Login</label>
<label for="signup" class="slide signup">Signup</label>
<div class="slider-tab"></div>
</div>
<div class="form-inner">
<form action="#" class="login">
<div class="field">
<input type="text" placeholder="Email Address" required>
</div>
<div class="field">
<input type="password" placeholder="Password" required>
```

```
</div>
<div class="pass-link"><a href="#">Forgot password?</a></div>
<div class="field btn">
<div class="btn-layer"></div>
<input type="submit" value="Login">
</div>

</form>
<form action="#" class="signup">
<div class="field">
<input type="text" placeholder="Email Address" required>
</div>
<div class="field">
<input type="password" placeholder="Password" required>
</div>
<div class="field">
<input type="password" placeholder="Confirm password" required>
</div>
<div class="field btn">
<div class="btn-layer"></div>
```

```
<input type="submit" value="Signup">

</div>

</form>

</div>

</div>

</div>


<script>

const loginText = document.querySelector(".title-text .login");

const loginForm = document.querySelector("form.login");

const loginBtn = document.querySelector("label.login");

const signupBtn = document.querySelector("label.signup");

const signupLink = document.querySelector("form .signup-link a");

signupBtn.onclick = (()=>{

loginForm.style.marginLeft = "-50%";

loginText.style.marginLeft = "-50%";

});

loginBtn.onclick = (()=>{

loginForm.style.marginLeft = "0%";

loginText.style.marginLeft = "0%";
```

```
 });
 signupLink.onclick = (()=>{
 signupBtn.click();
 return false;
 });
 </script>
</body>
</html>
```

Style.css:

```
@import url('https://fonts.googleapis.com/css?family=Poppins:400,500,600,700&display=swap');
*{
 margin: 0;
 padding: 0;
 box-sizing: border-box;
 font-family: 'Poppins', sans-serif;
}
html,body{
 display: grid;
 height: 100%;
 width: 100%;
 place-items: center;

 background: -webkit-linear-gradient(left, #a445b2, #fa4299);

}

::selection{

 background: #fa4299;

 color: #fff;

}

.wrapper{

 overflow: hidden;

 max-width: 390px;

 background: #fff;

 padding: 30px;

 border-radius: 5px;

 box-shadow: 0px 15px 20px rgba(0,0,0,0.1);

}

.wrapper .title-text{

 display: flex;

 width: 200%;

}

.wrapper .title{
 width: 50%;

 font-size: 35px;

 font-weight: 600;

 text-align: center;

 transition: all 0.6s cubic-bezier(0.68,-0.55,0.265,1.55);

}

.wrapper .slide-controls{

 position: relative;

 display: flex;

 height: 50px;

 width: 100%;

 overflow: hidden;

 margin: 30px 0 10px 0;

 justify-content: space-between;

 border: 1px solid lightgrey;

 border-radius: 5px;

}

.slide-controls .slide{

 height: 100%;

 width: 100%;
 color: #fff;

 font-size: 18px;

 font-weight: 500;

 text-align: center;

 line-height: 48px;

 cursor: pointer;

 z-index: 1;

 transition: all 0.6s ease;

}

.slide-controls label.signup{

 color: #000;

}

.slide-controls .slider-tab{

 position: absolute;

 height: 100%;

 width: 50%;

 left: 0;

 z-index: 0;

 border-radius: 5px;

 background: -webkit-linear-gradient(left, #a445b2, #fa4299);
 transition: all 0.6s cubic-bezier(0.68,-0.55,0.265,1.55);

}

input[type="radio"]{

 display: none;

}

#signup:checked ~ .slider-tab{

 left: 50%;

}

#signup:checked ~ label.signup{

 color: #fff;

 cursor: default;

 user-select: none;

}

#signup:checked ~ label.login{

 color: #000;

}

#login:checked ~ label.signup{

 color: #000;

}

#login:checked ~ label.login{
 cursor: default;
 user-select: none;
}
.wrapper .form-container{
 width: 100%;
 overflow: hidden;
}
.form-container .form-inner{
 display: flex;
 width: 200%;
}
.form-container .form-inner form{
 width: 50%;
 transition: all 0.6s cubic-bezier(0.68,-0.55,0.265,1.55);
}
.form-inner form .field{
 height: 50px;
 width: 100%;
 margin-top: 20px;
}
```

```
.form-inner form .field input{

 height: 100%;

 width: 100%;

 outline: none;

 padding-left: 15px;

 border-radius: 5px;

 border: 1px solid lightgrey;

 border-bottom-width: 2px;

 font-size: 17px;

 transition: all 0.3s ease;

}

.form-inner form .field input:focus{

 border-color: #fc83bb;

 /* box-shadow: inset 0 0 3px #fb6aae; */

}

.form-inner form .field input::placeholder{

 color: #999;

 transition: all 0.3s ease;

}

form .field input:focus::placeholder{
 color: #b3b3b3;

}

.form-inner form .pass-link{

 margin-top: 5px;

}

.form-inner form .signup-link{

 text-align: center;

 margin-top: 30px;

}

.form-inner form .pass-link a,

.form-inner form .signup-link a{

 color: #fa4299;

 text-decoration: none;

}

.form-inner form .pass-link a:hover,

.form-inner form .signup-link a:hover{

 text-decoration: underline;

}

form .btn{

 height: 50px;
 width: 100%;
 border-radius: 5px;
 position: relative;
 overflow: hidden;
}
form .btn .btn-layer{
 height: 100%;
 width: 300%;
 position: absolute;
 left: -100%;
 background: -webkit-linear-gradient(right, #a445b2, #fa4299, #a445b2, #fa4299);
 border-radius: 5px;
 transition: all 0.4s ease;;
}
form .btn:hover .btn-layer{
 left: 0;
}
form .btn input[type="submit"]{
 height: 100%;
 width: 100%;
 z-index: 1;

 position: relative;

 background: none;

 border: none;

 color: #fff;

 padding-left: 0;

 border-radius: 5px;

 font-size: 20px;

 font-weight: 500;

 cursor: pointer;

}
```

Feedback Form:

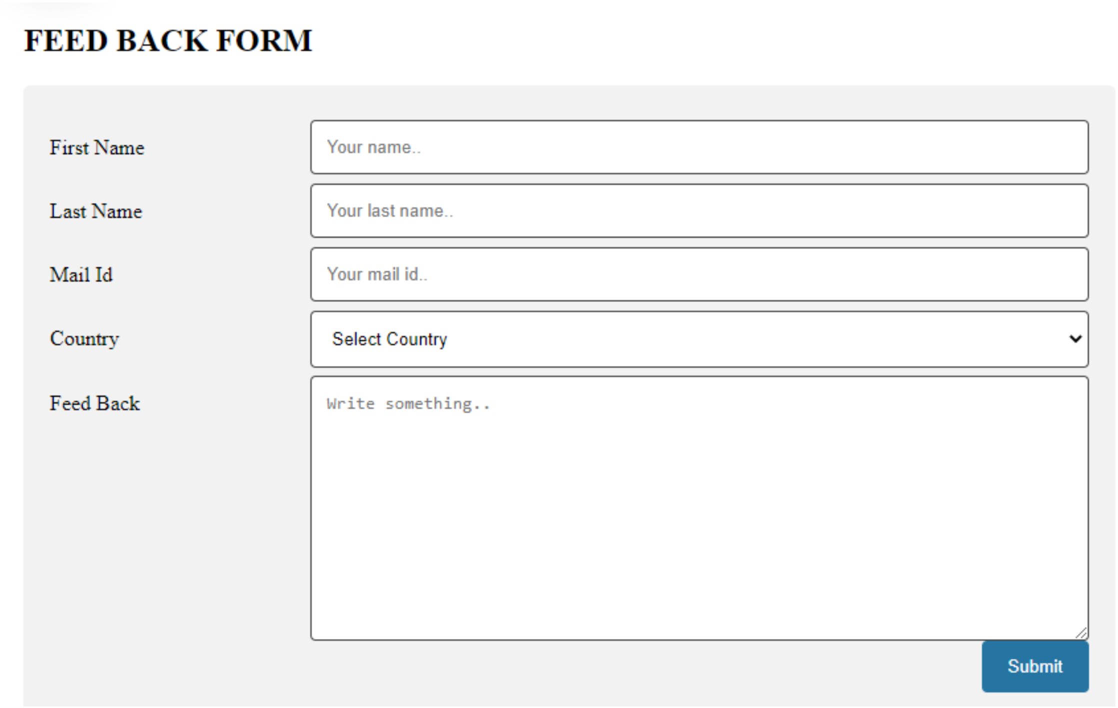
```
<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

* {

  box-sizing: border-box;
```

```
}


input[type=text], select, textarea {
  width: 100%;
  padding: 12px;
  border: 1px solid rgb(70, 68, 68);
  border-radius: 4px;
  resize: vertical;
}
input[type=email], select, textarea {
  width: 100%;
  padding: 12px;
  border: 1px solid rgb(70, 68, 68);
  border-radius: 4px;
  resize: vertical;
}


label {
  padding: 12px 12px 12px 0;
  display: inline-block;
```

```
}


input[type=submit] {
  background-color: rgb(37, 116, 161);
  color: white;
  padding: 12px 20px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  float: right;
}


input[type=submit]:hover {
  background-color: #45a049;
}


.container {
  border-radius: 5px;
  background-color: #f2f2f2;
  padding: 20px;
```

```
}


.col-25 {
  float: left;
  width: 25%;
  margin-top: 6px;
}


.col-75 {
  float: left;
  width: 75%;
  margin-top: 6px;
}


/* Clear floats after the columns */
.row:after {
  content: "";
  display: table;
  clear: both;
}
```

```
/* Responsive layout - when the screen is less than 600px wide, make the two columns stack on top of each other instead of next to each other */
</style>
</head>
<body>
<h2>FEED BACK FORM</h2>
<div class="container">
  <form>
    <div class="row">
      <div class="col-25">
        <label for="fname">First Name</label>
      </div>
      <div class="col-75">
        <input type="text" id="fname" name="firstname" placeholder="Your name..">
      </div>
    </div>
    <div class="row">
      <div class="col-25">
        <label for="lname">Last Name</label>
      </div>

      <div class="col-75">

        <input type="text" id="lname" name="lastname" placeholder="Your last name..">

      </div>

    </div>

    <div class="row">

        <div class="col-25">

          <label for="email">Mail Id</label>

        </div>

        <div class="col-75">

          <input type="email" id="email" name="mailid" placeholder="Your mail id..">

        </div>

      </div>
```



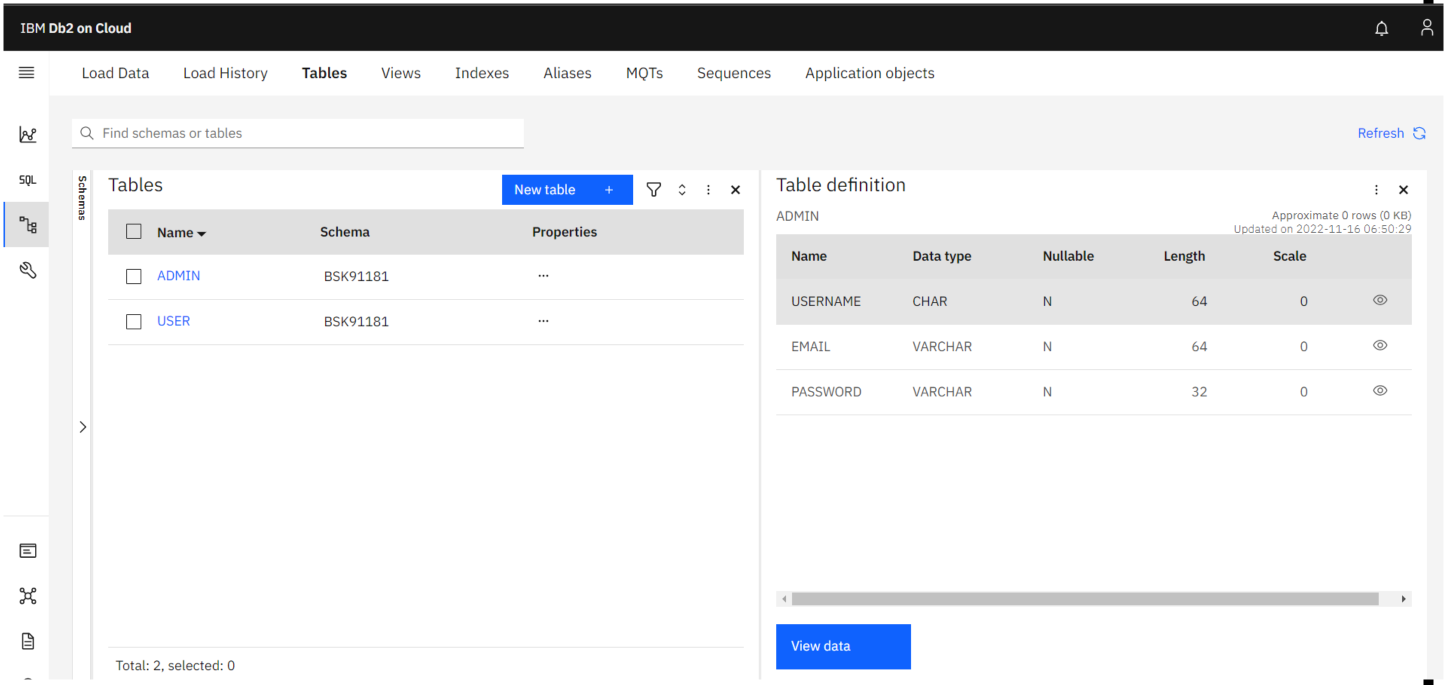
FEED BACK FORM

First Name	Your name..
Last Name	Your last name..
Mail Id	Your mail id..
Country	Select Country
Feed Back	Write something..

IBM Watson:

7.3 Database Schema (if Applicable):

```
from flask import Flask,request,render_template
import ibm_db
from db import *
app = Flask(__name__)
import ibm_db

conn = ibm_db.connect(dbconnect(), "", "")

@app.route('/')
def login():  # put application's code here
    return render_template("Login.html")
@app.route('/login',methods=['POST'])
def page():
    return render_template("registration.html")
@app.route('/register')
def register():
    print("checked")
    username1 = request.form['username']
    password1 = request.form['password']
    sql = "SELECT * FROM user WHERE username=?"
    statement = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(statement, 1, username1)
    ibm_db.execute(statement)
    acc = ibm_db.fetch_assoc(statement)
    if acc:
        print("username already exists")
        return render_template("registration.html")
    else:
        sql = "INSERT INTO user(username,password) values(?,?)"
        statement = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(statement, 1, username1)

        ibm_db.bind_param(statement, 2, password1)

        ibm_db.execute(statement)

        print("created")

        return render_template("Login.html")



if __name__ == '__main__':

    app.run()
```



Tables

☐	Name	Schema	Properties
☐	ADMIN	BSK91181	···
☐	USER	BSK91181	···

Total: 2, selected: 0

Table definition

ADMIN

Approximate 0 rows (0 KB)
Updated on 2022-11-16 06:50:29

Name	Data type	Nullable	Length	Scale
USERNAME	CHAR	N	64	0
EMAIL	VARCHAR	N	64	0
PASSWORD	VARCHAR	N	32	0

8.TESTING

8.1Test Cases:

A test case has components that describe input, action and an expected response, in order to determine if a feature of an application is working correctly. A test case is a set
of instructions on “HOW” to validate a particular test objective/target, which when followed
will tell us if the expected behavior of the system is satisfied or not.
Characteristics of a good test case:

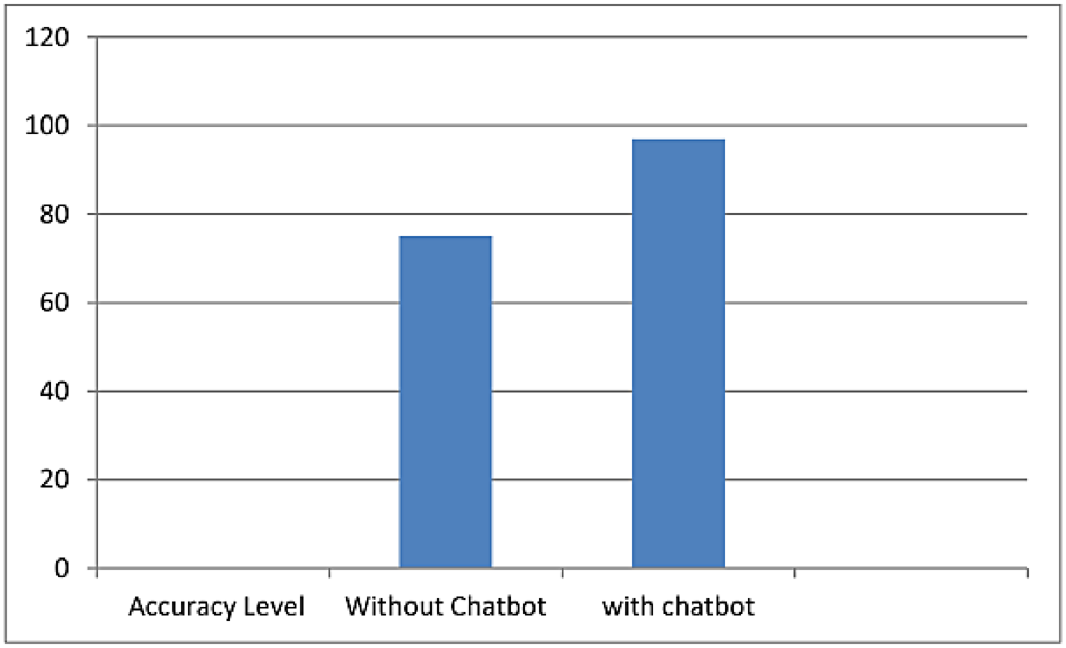
- Accurate: Exacts the purpose.
- Economical: No unnecessary steps or words.
- Traceable: Capable of being traced to requirements.
- Repeatable: Can be used to perform the test over and over.
- Reusable: Can be reused if necessary.

8.2User Acceptance Testing:

This sort of testing is carried out by users, clients, or other authorised bodies to identify the requirements and operational procedures of an application or piece of software. The most crucial stage of testing is acceptance testing since it determines whether or not the customer will accept the application or programme. It could entail the application's U.I., performance, usability, and usefulness. It is also referred to as end-user testing, operational acceptance testing, and user acceptance testing (UAT).

9. RESULTS:

9.1Performance Metrics:



10.ADVANTAGES & DISADVANTAGES

ADVANTAGE:

- Automation of existing manual information systems.
- Implement chatbot system makes this application user friendly
- Keep track of daily information exchange at the server by the administrator.
- Recommending products based on the user search
- Communicate with the bot makes customer satisfied for choose the product

DISADVANTAGE:

- Difficult to decision making for an administrator to improve
- Immediate response to the queries is difficult.
- More stationary use so they are expensive.
- Manual system is takes more time.
- Existing system is manually, so it increases the chances of errors.

11.CONCLUSION

In order to enhance user interaction with the e-commerce website, we have created a chatbot that is based on a website. The chatbot has a pre-stored list of responses, but it also considers dynamic user input, which makes it more likely to offer pertinent answers and product recommendations. Newer goods under the relevant category may be readily added and withdrawn without requiring any changes to the stored chatbot replies since the product database is independent of the responses that were previously saved.

12.FUTURE SCOPE

Future iterations of this project may add more features, such as a comprehensive chatbot application for the healthcare sector or another business. It is easy to make additional enhancements to this system because of the way it was designed. The modification of the project would increase the system's adaptability. Furthermore, the functionalities are provided in a way that will improve the system's performance.

13.APPENDIX

Source Code

```
from flask import Flask, render_template, flash, request,session
from flask import Flask, render_template, request, jsonify
import datetime
import re
import ibm_db
import pandas
import ibm_db_dbi
from sqlalchemy import create_engine

engine = create_engine('sqlite://',
echo = False)

dsn_hostname = "1bbf73c5-d84a-4bb0-85b9-
ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud"
dsn_uid = "ysc77612"
dsn_pwd = "oUWwH90LqzyyOSfH"

dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "BLUDB"
dsn_port = "32286"
dsn_protocol = "TCPIP"
dsn_security = "SSL"

dsn = (
"DRIVER={0};"
"DATABASE={1};"
"HOSTNAME={2};"
"PORT={3};"
"PROTOCOL={4};"
"UID={5};"

"PWD={6};"
"SECURITY={7};").format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol, dsn_uid, dsn_pwd,dsn_security)

try:
conn = ibm_db.connect(dsn, "", "")
print ("Connected to database: ", dsn_database, "as user: ", dsn_uid, "on host: ", dsn_hostname)

except:
print ("Unable to connect: ", ibm_db.conn_errormsg() )

app = Flask(__name__)
app.config.from_object(__name__)
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'

@app.route("/")
def homepage():

return render_template('index.html')
```

```
@app.route("/AdminLogin")
def AdminLogin():

return render_template('AdminLogin.html')


@app.route("/NewUser")
def NewUser():

return render_template('NewUser.html')
@app.route("/UserLogin")
def UserLogin():

return render_template('UserLogin.html')
@app.route("/AdminHome")
def AdminHome():
conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)

selectQuery = "SELECT * from regtb "
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('Employee_Data',
con=engine,
if_exists='append')
# run a sql query
data = engine.execute("SELECT * FROM Employee_Data").fetchall()
```

```
return render_template('AdminHome.html', data=data)
@app.route("/NewProduct")
def NewProduct():

return render_template('NewProduct.html')


@app.route("/ProductInfo")
def ProductInfo():
conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)

selectQuery = "SELECT * from protb "
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('Employee_Data',
con=engine,
if_exists='append')

# run a sql query
print(engine.execute("SELECT * FROM Employee_Data").fetchall())

return render_template('ProductInfo.html', data=engine.execute("SELECT *
FROM Employee_Data").fetchall())
@app.route("/SalesInfo")
def SalesInfo():
return render_template('SalesInfo.html')
@app.route("/Search")
```

```
def Search():

conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)

selectQuery = "SELECT * from protb "
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('Employee_Data',
con=engine,
if_exists='append')
# run a sql query
print(engine.execute("SELECT * FROM Employee_Data").fetchall())
return render_template('ViewProduct.html', data=engine.execute("SELECT *
FROM Employee_Data").fetchall())
@app.route("/viewproduct", methods=['GET', 'POST'])
def viewproduct():
searc = request.form['subcat']
conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)

selectQuery = "SELECT * from protb where SubCategory like '%" + searc + "%' "
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('Employee_Data',
con=engine,
if_exists='append')
# run a sql query
```

```
print(engine.execute("SELECT * FROM Employee_Data").fetchall())

return render_template('ViewProduct.html', data=engine.execute("SELECT *
FROM Employee_Data").fetchall())
@app.route("/RNewUser", methods=['GET', 'POST'])
def RNewUser():
if request.method == 'POST':

name1 = request.form['name']
gender1 = request.form['gender']
Age = request.form['age']
email = request.form['email']
address = request.form['address']
pnumber = request.form['phone']
uname = request.form['uname']
password = request.form['psw']
conn = ibm_db.connect(dsn, "", "")
insertQuery = "INSERT INTO regtb VALUES ('" + name1 + "','" + gender1 +
"','" + Age + "','" + email + "','" + pnumber + "','" + address + "','" + uname + "','" +
password + "')"
insert_table = ibm_db.exec_immediate (conn, insertQuery)
print(insert_table)
return render_template('userlogin.html')
@app.route("/RNewProduct", methods=['GET', 'POST'])
def RNewProduct():
if request.method == 'POST':
```

```
file = request.files['fileupload']
file.save("static/upload/" + file.filename)
ProductId =request.form['pid']
Gender =request.form['gender']
Category =request.form['cat']
SubCategory=request.form['subcat']
ProductType=request.form['ptype']
Colour=request.form['color']
Usage=request.form['usage']
ProductTitle=request.form['ptitle']
price = request.form['price']
Image= file.filename
ImageURL="static/upload/" + file.filename
conn = ibm_db.connect(dsn, "", "")

insertQuery = "INSERT INTO protb VALUES ('"+ ProductId +"','" + Gender +
"','" + Category + "','" + SubCategory + "','" + ProductType + "','" + Colour +
"','"+Usage +"','"+ProductTitle+"','"+ Image +"','"+ ImageURL +"','"+ price +"')"
insert_table = ibm_db.exec_immediate(conn, insertQuery)

data1 = 'Record Saved!'
return render_template('goback.html', data=data1)
@app.route("/userlogin", methods=['GET', 'POST'])
def userlogin():
error = None
if request.method == 'POST':
username = request.form['uname']
```

```
password = request.form['password']
session['uname'] = request.form['uname']

conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)

selectQuery = "SELECT * from regtb where UserName='" + username + "' and
password='" + password + "'"
dataframe = pandas.read_sql(selectQuery, pd_conn)

if dataframe.empty:
data1 = 'Username or Password is wrong'
return render_template('goback.html', data=data1)
else:
print("Login")

selectQuery = "SELECT * from regtb where UserName='" + username + "'
and password='" + password + "'"
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('Employee_Data',
con=engine,
if_exists='append')
# run a sql query
print(engine.execute("SELECT * FROM Employee_Data").fetchall())
return render_template('UserHome.html', data=engine.execute("SELECT *
FROM Employee_Data").fetchall())
@app.route("/adminlogin", methods=['GET', 'POST'])
```

```
def adminlogin():
error = None
if request.method == 'POST':
username = request.form['uname']
password = request.form['password']
conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)
selectQuery = "SELECT * from admintb where LASTNAME='" + username + "'
and FIRSTNAME='" + password + "'"
dataframe = pandas.read_sql(selectQuery, pd_conn)
if dataframe.empty:
data1 = 'Username or Password is wrong'
return render_template('goback.html', data=data1)
else:
print("Login")
selectQuery = "SELECT * from regtb "
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('Employee_Data', con=engine,if_exists='append')

# run a sql query
print(engine.execute("SELECT * FROM Employee_Data").fetchall())

return render_template('AdminHome.html', data=engine.execute("SELECT *
FROM Employee_Data").fetchall())
@app.route("/Remove", methods=['GET'])
def Remove():
```

```
pid = request.args.get('id')
conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)
insertQuery = "Delete from protb where id='"+ pid +"'""
insert_table = ibm_db.exec_immediate(conn, insertQuery)
selectQuery = "SELECT * from protb "
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('Employee_Data',
con=engine,
if_exists='append')
# run a sql query
print(engine.execute("SELECT * FROM Employee_Data").fetchall())
return render_template('ProductInfo.html', data=engine.execute("SELECT *
FROM Employee_Data").fetchall())
@app.route("/fullInfo")
def fullInfo():
pid = request.args.get('pid')
session['pid'] = pid
conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)
selectQuery = "SELECT * FROM protb where ProductId='" + pid + "' "
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('Employee_Data',
con=engine,
if_exists='append')
# run a sql query
print(engine.execute("SELECT * FROM Employee_Data").fetchall())
```

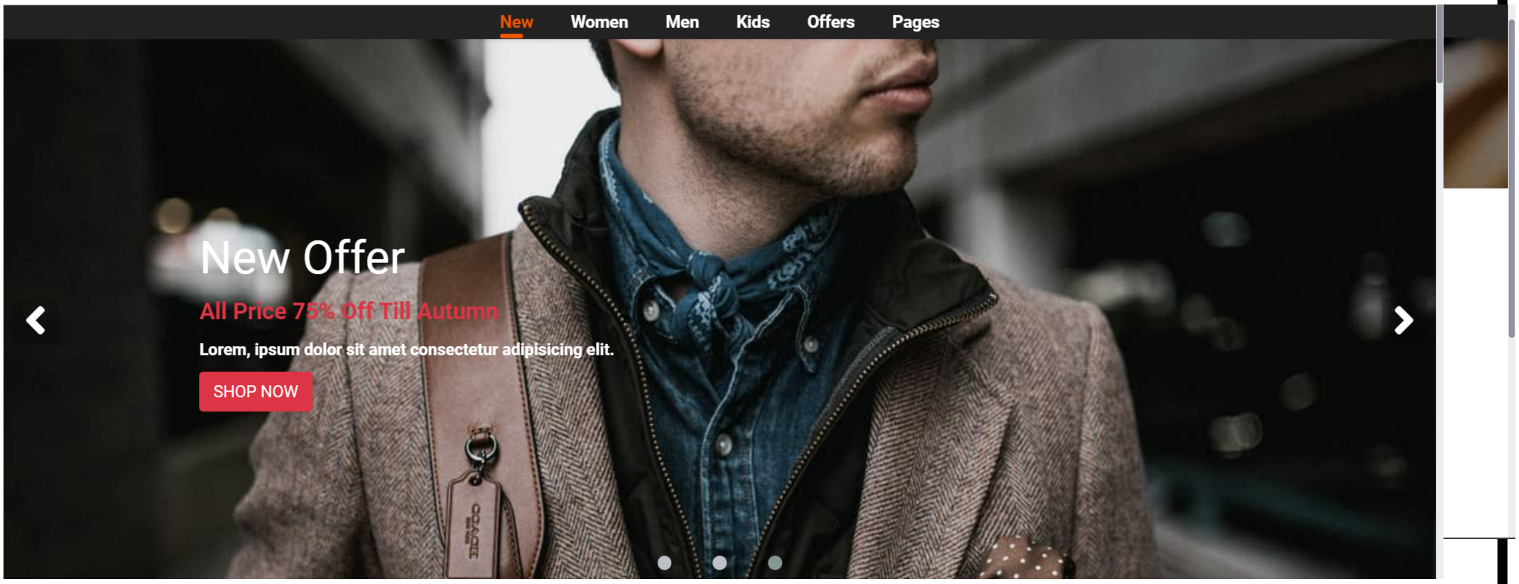
```
return render_template('ProductFullInfo.html', data=engine.execute("SELECT * FROM Employee_Data").fetchall())

@app.route("/Book", methods=['GET', 'POST'])
def Book():
if request.method == 'POST':
uname = session['uname']
pid = session['pid']
qty = request.form['qty']
ctype = request.form['ctype']
cardno = request.form['cardno']
cvno = request.form['cvno']
Bookingid = "
ProductName ="
UserName= uname
Mobile="
Email="
Qty = qty
Amount="
CardType = ctype
CardNo = cardno
CvNo = cvno
date = datetime.datetime.now().strftime('%d-%b-%Y')
conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)
selectQuery = "SELECT * FROM protb where ProductId='" + pid + "' "
```

```
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('Employee_Data',con=engine,if_exists='append')
data = engine.execute("SELECT * FROM Employee_Data").fetchall()
for item in data:
ProductName = item[8]
price = item[11]
print(price)
Amount = float(price) * float(Qty)
print(Amount)
selectQuery1 ="SELECT * FROM regtb where UserName='" + uname + "'"
dataframe = pandas.read_sql(selectQuery1, pd_conn)

dataframe.to_sql('regtb', con=engine, if_exists='append')
data1 = engine.execute("SELECT * FROM regtb").fetchall()
for item1 in data1:
Mobile = item1[5]
Email = item1[4]
selectQuery = "SELECT * FROM booktb"
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('booktb', con=engine, if_exists='append')
data2 = engine.execute("SELECT * FROM booktb").fetchall()
count = 0
for item in data2:
count+=1
Bookingid="BOOKID00" + str(count)

insertQuery = "INSERT INTO booktb VALUES ('" + Bookingid + "','"+
ProductName +"','" + price + "','" + uname + "','" + Mobile + "','" + Email + "','" +
str(Qty) + "','" + str(Amount) + "','"+ str(CardType) +"','"+ str(CardNo) +"','"+
str(CvNo) +"','"+ str(date) +"')"
insert_table = ibm_db.exec_immediate(conn, insertQuery)
sendmsg(Email,"order received delivery in one week ")
selectQuery = "SELECT * FROM booktb where UserName= '" + uname + "' "
dataframe = pandas.read_sql(selectQuery, pd_conn)

dataframe.to_sql('booktb1', con=engine, if_exists='append')
data = engine.execute("SELECT * FROM booktb1").fetchall()
return render_template('UOrderInfo.html', data=data)
def sendmsg(Mailid,message):
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders

fromaddr = "sampletest685@gmail.com"

toaddr = Mailid

# instance of MIMEMultipart
msg = MIMEMultipart()

# storing the senders email address
msg['From'] = fromaddr
```

```
# storing the receivers email address
msg['To'] = toaddr

# storing the subject
msg['Subject'] = "Alert"

# string to store the body of the mail
body = message

# attach the body with the msg instance
msg.attach(MIMEText(body, 'plain'))

# creates SMTP session
s = smtplib.SMTP('smtp.gmail.com', 587)

# start TLS for security
s.starttls()
# Authentication
s.login(fromaddr, "hneucvnontsuwgpj")
# Converts the Multipart msg into a string
text = msg.as_string()
# sending the mail
s.sendmail(fromaddr, toaddr, text)
# terminating the session
s.quit()
@app.route("/UOrderInfo")
```

```
def UOrderInfo():
uname = session['uname']
conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)
selectQuery = "SELECT * FROM booktb where UserName= '" + uname + "' "
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('booktb1', con=engine, if_exists='append')
data = engine.execute("SELECT * FROM booktb1").fetchall()
return render_template('UOrderInfo.html', data=data)
@app.route("/UserHome")
def UserHome():
uname = session['uname']
conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)
selectQuery = "SELECT * FROM regtb where UserName= '" + uname + "' "
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('booktb1', con=engine, if_exists='append')
data = engine.execute("SELECT * FROM booktb1").fetchall()
return render_template('UserHome.html', data=data)
@app.route("/ASalesInfo")
def ASalesInfo():
conn = ibm_db.connect(dsn, "", "")
pd_conn = ibm_db_dbi.Connection(conn)
selectQuery = "SELECT * FROM booktb "
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('booktb', con=engine, if_exists='append')
```

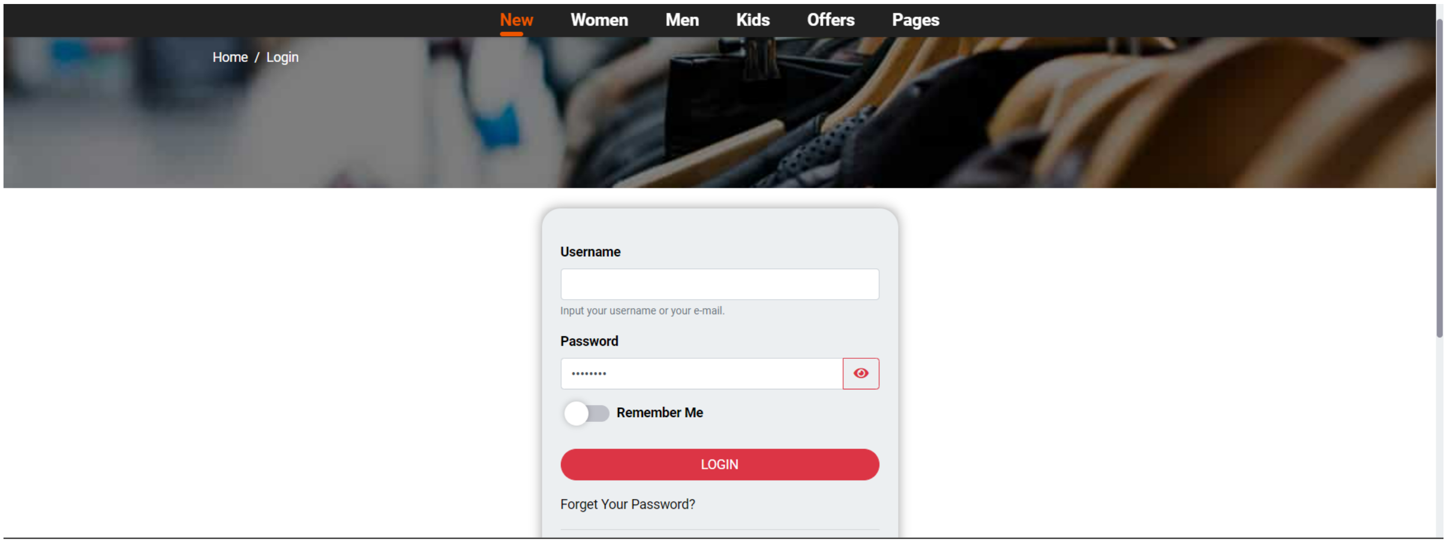
```
data = engine.execute("SELECT * FROM booktb").fetchall()
return render_template('ASalesInfo.html', data=data)
def main():
app.run(debug=True, use_reloader=True)
if __name__ == '__main__':
main()
```



New Offer

All Price 75% Off Till Autumn

Lorem, ipsum dolor sit amet consectetur adipisicing elit.



Input your username or your e-mail.

Forget Your Password?

GitHub & Project Demo Link:

Github Link:

https://github.com/IBM-EPBL/IBM-Project-44922-1660727459.

Project Link:

https://drive.google.com/drive/folders/1QTm9ZmwbASKkfX8GLmDBfRkXF90WZJEg