SMART FASHION RECOMMENDER APPLICATION

HX 8001-

Professional Readiness For Innovation, Employability and Entrepreneurship

IBM-Project- 15181-1659594640

TEAM ID : PNT2022TMID08329

Submitted by

NADIPINENI BHARATH KUMAR	**(810419104065)**
TIPPISETTI SRAVAN KUMAR	**(810419104117)**
TANNERU SURYA KALYAN CHAKRAVARTHI	**(810419104115)**
PALLA BHANU SIVA SANKAR	**(810419104075)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE

(AUTONOMOUS)

PERAMBALUR-621212

TABLE OF CONTENTS

1. INTRODUCTION.....1
   1.1 Project Overview
   1.2 Purpose
2. LITERATURESURVEY.....3
   2.1 Existing problem
   2.2 References
   2.3 Problem Statement Definition
3. IDEATION&PROPOSEDSOLUTION.....10
   3.1 Empathy Map Canvas
   3.2 Ideation & Brainstorming
   3.3 Proposed Solution
   3.4 Problem Solution fit
4. REQUIREMENTANALYSIS.....12
   4.1 Functional requirement
   4.2 Non-Functional requirements
5. PROJECTDESIGN.....14
   5.1 Data Flow Diagrams
   5.2 Solution & Technical Architecture
   5.3 User Stories
6. PROJECTPLANNING&SCHEDULING.....19
   6.1 Sprint Planning & Estimation
   6.2 Sprint Delivery Schedule
   6.3 Reports from JIRA
7. CODING&SOLUTIONING.....21
   7.1 Registration Page
   7.2 Dashboard Page
   7.3 Data base Schema (DB2 and SQL_LITE3)
8. TESTING.....30
   8.1 Test Cases
   8.2 User Acceptance Testing
9. RESULTS.....32
   9.1 Performance Metrics

10. ADVANTAGES &DISADVANTAGES……33
11. CONCLUSION……34
12. FUTURE SCOPE……35
13. APPENDIX……36
Source Code
GitHub & Project Demo Link

SMART FASHION RECOMMENDER APPLICATION

ABSTRACT

A Web based shopping application projects that act as a central database containing various products in stock along with their title, category and cost. . Online Shopping is fast gaining ground as an accepted and used business paradigm. More and more business houses are implementing web sites providing functionality for performing commercial transactions over the web. It is reasonable to say that the process of shopping on the web is becoming common place. Many internet markets are currently beginning to replace the traditional market. Due to the intense competition in the online market, shoppers expect sellers to deliver exceptional customer care; hence many online shops offer round-the-clock support. If performed manually, this job undoubtedly costs a lot of money. The proposed chatbot system is used as a tool to perform and serve general conversations about the product, stock, orders and payments questions. Capable of answering and solve the users query about the purchase process and many other details regarding to it. The bot must be able to respond promptly and accurately which is applied to the e-commerce application. It helps you to make a decision which product is suitable for you and especially helpful when you have not narrowed down the criteria for the product. Its functions basically like an online automated assistan

1. INTRODUCTION

1.1 PROJECT OVERVIEW

In many modern apps, especially those that provide the user intelligence help, the usage of chatbots is quite common. In reality, these systems frequently have chatbots that can read user inquiries and give the appropriate replies quickly and accurately in order to speed up the support. The use of chatbots as human-computer interactions has gained significant traction. The term "chatbot" refers to a computer software that tries to replicate a written discussion in an effort to briefly deceive a human user. In essence, a chatbot is a conversational agent that can communicate with users in natural language on a certain topic. On the internet, there are several chat bots that are being used for amusement, customer service, education, and other purposes. The goal of the e-commerce chatbot is to provide customers with quick answers to their questions. The user may even search for specific products on the website. It is the graphical user interface. Products are categorized in different Category and each Category has Some Sub Category. And also chatbot system is applied to this shopping website for recommending products and solves the users query regarding to buy the product. However, a chatbot simply symbolises the logical development of a Question Answering system utilising Natural Language Processing from a technical standpoint (NLP).

1.2 PURPOSE

This project is to develop a general purpose online store where any product can be bought from the comfort of home through the Internet. A user visiting the website can see a wide range of products arranged in respective categories. The user may select desired product and view its price. The user may even search for specific products on the website. It is the graphical user interface. Products are categorized in different Category and each Category has Some Sub Category. And also chatbot system is applied to this shopping website for recommending products and solves the users query regarding to buy the product.

2. LITERATURE REVIEW

2.1 EXISTING PROBLEM

2.1.1 TITLE: Smart Chatbot System for E-Commerce Assitance based on AIML, 2020 AUTHOR NAME: Arif Nursetyo

Because the competitive nature of the internet market necessitates outstanding service from sellers to customers, many online companies offer complete 24-hour service. If done manually, this job will undoubtedly cost a lot of money. A chatbot may be used to automatically purchase online. The bot must then be able to respond accurately and quickly. The purpose of this research is to propose an intelligent chatbot system based on Artificial Intelligence Mark-up Language (AIML) that may be utilised as an e-commerce assistant. This Chatbot is used with the Telegram app. AIML will be used to handle user input queries through three stages: parsing, pattern matching, and data crawling. User requests in the AIML process are divided into three categories: general queries, computations, and stock checks. Where the computation request immobilises the order and payment processes. Based on the results of 300 trials, the proposed method can satisfactorily respond to all user requests, with an average response time of 3.4 seconds.

2.1.2 TITLE: An E-Commerce Website based Chatbot, 2020
AUTHOR NAME: Siddharth Gupta

A chatbot, sometimes known as a chatbot, is a computer software that simulates an intelligent interaction with one or more human users using aural or written means. Chatbots can be programmed to make small conversation or to operate as a medium of engagement with users, providing them with solutions to common inquiries. The chatbot understands context and responds based on the message it receives. A chatbot is only one example of AI. Chatbots were initially created for entertainment purposes, and some of them have been designed to pass the Turing Test. This paper discusses a chatbot that runs on a webpage. This chatbot can make interacting with the website easier. The bot communicates with the user in Simple Language. This chatbot is tied to an online store. This website offers a selection of items with varying features. The chatbot assists you in determining which product is best for you. This is especially useful if you haven't narrowed down the product's requirements. It acts essentially as an online automated assistant.

2.1.3 TITLE: The Impact of Anthropomorphism on Consumers' Purchase Decision in Chatbot Commerce, 2022

AUTHOR NAME: Min Chung Han

Many organisations have used chatbots with human-like speaking abilities to help clients with online buying and customer support; however, it is unclear whether or not their efforts will be rewarded by consumers engaging with them. Will young, digital-native customers utilise Chatbots as much as we predicted? If this is the case, what would motivate them to adopt chatbots? There have been no academic studies investigating customers' attitudes and acceptance of chatbot commerce. This study attempts to answer this issue by investigating the effects of anthropomorphism on customers' perceptions of mobile messenger chatbots, as well as its impact on behavioural decision making. The data demonstrate that anthropomorphism influences consumers' inclinations to purchase through chatbot commerce. Because chatbots are becoming more widespread in online customer care and e-commerce, it is critical to understand young customers' psychological processes for human-like speaking chatbots, as well as their effect on consumers' behavioural intents to purchase things via chatbots. This study attempts to fill a gap in the literature by investigating the influence of anthropomorphism on customers' perceptions of mobile messenger chatbots and its impact on behavioural decision making.

2.1.4 TITLE: The role of the chatbot on customer purchase intention: towards digital relational sales, 2021

AUTHOR NAME: Giulio Maggiore

The goal of this article is to investigate the extent to which a discussion with a chatbot on an official website might alter product value perception and impact customer purchase intention. The study also looks into the function of chatbots in brand familiarity and the impact of prior interactions with chatbots on customer purchasing intentions. Two laboratory tests were carried out to assess the effect of chatbot use on consumer purchase intent. Chatbots on websites increase the value perception of hedonic and utilitarian commodities toward utilitarian and hedonic values, respectively. Chatbots increase brand recognition by minimising the influence of familiarity on client purchase intent. Retailers can use successful chatbots while managing and creating retail websites to increase consumer purchasing intent. Companies should carefully build and manage chatbots, monitoring user participation and customer experience quality to feed a calibrated continuous improvement process on company objectives.

2.1.5 TITLE: Research and Development of an E-commerce with Sales Chatbot, 2021
AUTHOR NAME: Mostaqim Hossain

E-commerce is the process of doing business through computer networks. A person sitting in front of a computer may access all of the internet's resources to buy or sell something. Unlike traditional commerce, which requires a person to travel and collect products, ecommerce has made it easier for people to remove physical labour and save time. This online company management system has also helped merchants save money on initial investments such as storefronts and employees. The goal is to create a sophisticated ecommerce system with a smart chatbot to deliver a user-friendly experience for customers. Customers will save time since the chatbot in the system are easily accessible. We created a real-time chatbot with intelligent features using natural language processing. We attempted to tackle some of the difficulties with our country's existing e-commerce platform on this smart e-commerce website. The registration system, login system, search system, order system, add product system, view product system, order received system, and an interface to connect with the bot are all part of the system.

2.2 REFERENCES

[1]. Arif Nursetyo, Smart Chatbot System for E-Commerce Assitance based on AIML, 2020

[2]. Siddharth Gupta, An E-Commerce Website based Chatbot, 2020

[3]. Min Chung Han, The Impact of Anthropomorphism on Consumers' Purchase Decision in Chatbot Commerce, 2022\

[4]. Giulio Maggiore, The role of the chatbot on customer purchase intention: towards digital relational sales, 2021

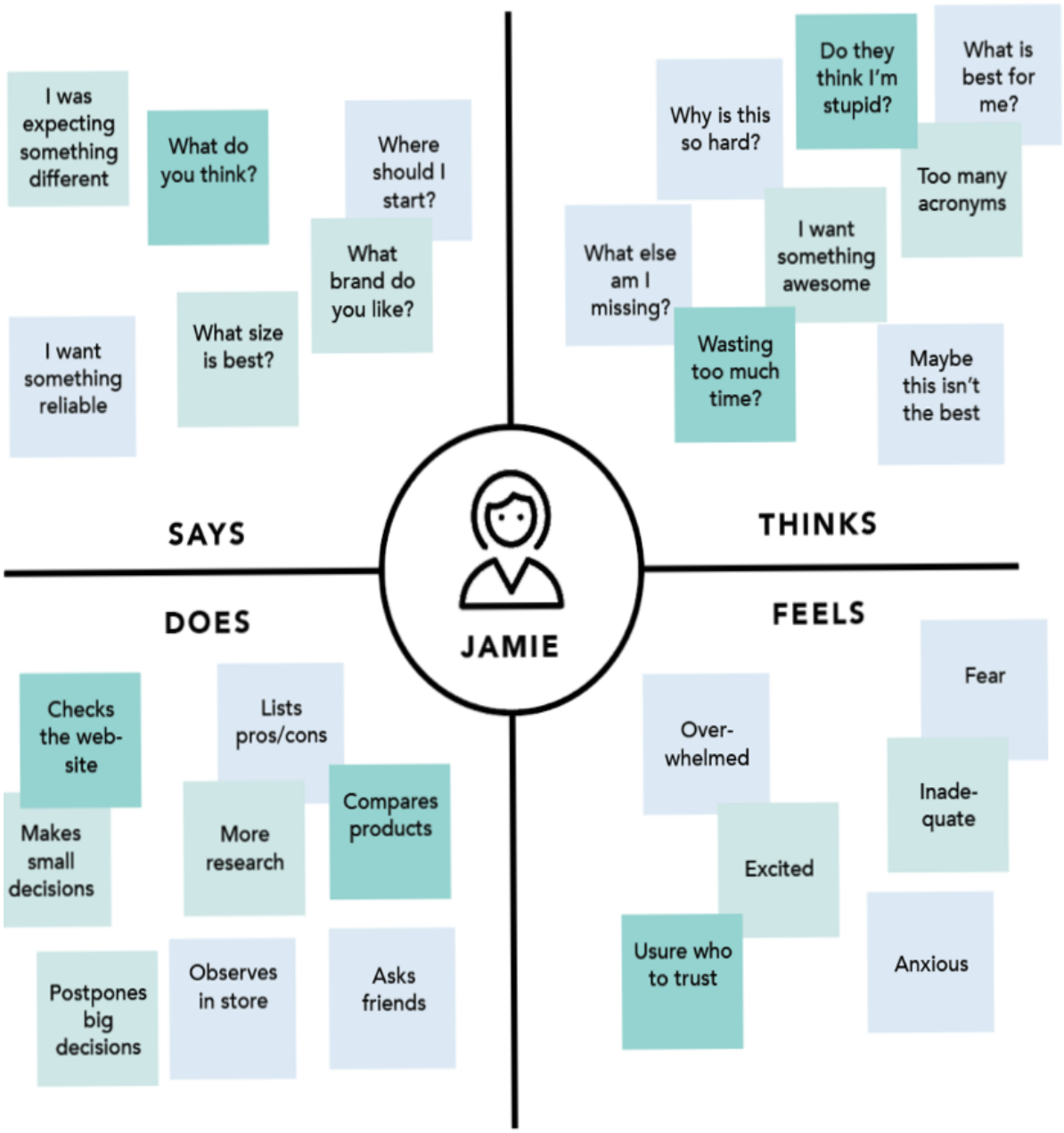
[5]. Mostaqim Hossain, Research and Development of an E-commerce with Sales Chatbot, 2021

2.3 PROBLEM STATEMENT DEFINITION

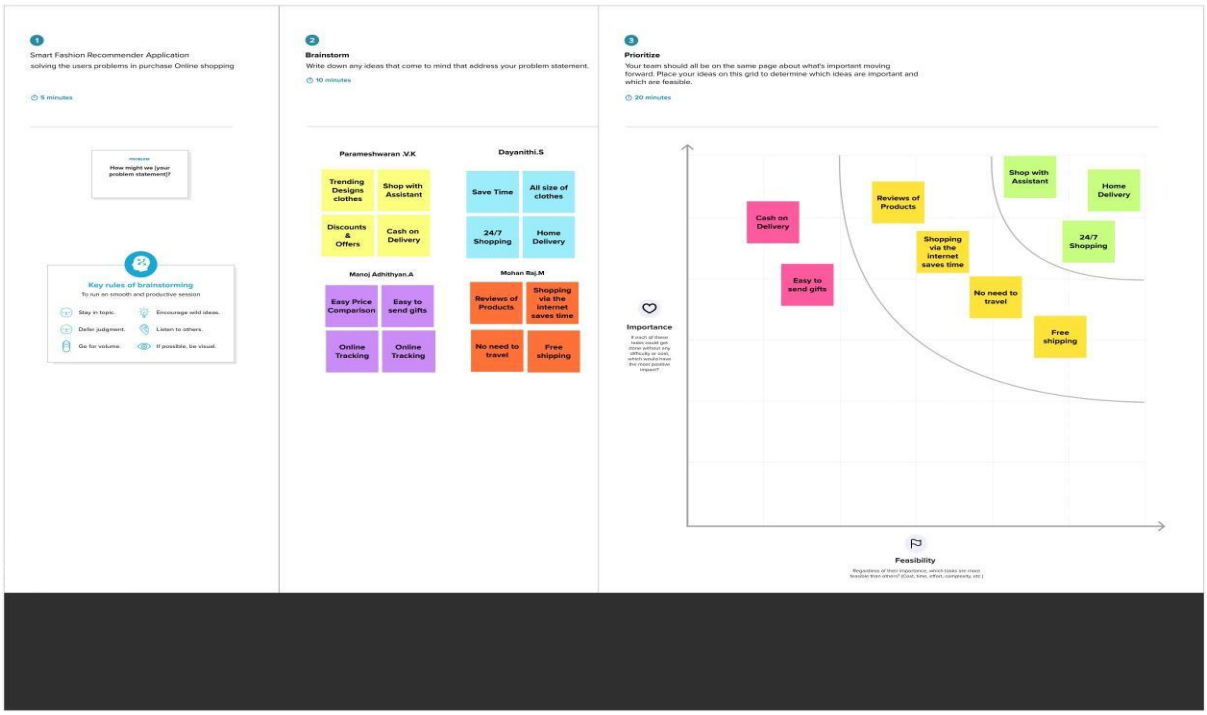
In existing, the system is operated manually. For sales of paper work, the system offers a prepared online store system. When products need to be purchased and a bill has to be made for the sale of items, the indent is prepared. The system requires a lot of paperwork to maintain accessory details, and it might be challenging for users to search for certain accessories in the database. The lack of a bot communication mechanism on the current website makes it difficult for users.

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION & BRAINSTORMING



3.3 PROPOSED SOLUTION

The proposed system is developed after a detailed study about the requirements requested by the user. Proposed system is a computerized one, where all the limitations of manual system are compensated. Product details of online shopping system have simplified the working information and make a user friendly environment, where the user is provided with much flexibility to manage effectively. It helps the retailer to generate desirable reports more quickly and also to produce better results. A chatbot system able to respond quickly and promptly and also aids to manage and resolve the customers need.

3.4 PROBLEM SOLUTION FIT

The system requires a lot of paperwork to maintain accessory details, and it might be challenging for users to search for certain accessories in the database. The lack of a bot communication mechanism on the current website makes it difficult for users. A chatbot system is proposed and able to respond quickly and promptly and also aids to manage and resolve the customers need. It helps the retailer to generate desirable reports more quickly and also to produce better results.

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

MODULES DESCRIPTION

Framework Creation

In this module, design the framework for e-commerce application to the customer, to get answers without any human assistance. Admin can train keywords with answers for future processing. Chatbots are such kind of computer programs that interact with users using natural languages. For all kind of chatbot the flow is same, though each chatbot is specific in its own area knowledge that is one input from human is matched against the knowledge base of chatbot.

Add Products

In this module, admin can add the product such as cosmetics things and clothes and monitor the details of customers

View Products

In this module, customer can view the products and buy whatever they need and these details are tracked by an admin and stored in the database

Post Questions

In this module, customer can ask query regarding to buy the product to the chatbot system

Recommend Results

In this module, the chatbot system responds and resolves the customer query and based on the customer search, it will recommend the products. This approach helps to customer as well as the admin to improve the efficiency of website and products.

4.2 NON FUNCTIONAL REQUIREMENTS

Usability

The system shall allow the users to access the system with pc using web application. The system uses a web application as an interface. The system is user friendly which makes the system easy

Availability

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

Scalability

Scalability is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands.

Security

A security requirement is a statement of needed security functionality that ensures one of many different security properties of software is being satisfied.

Performance

The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 seconds to appear on the screen.

Reliability

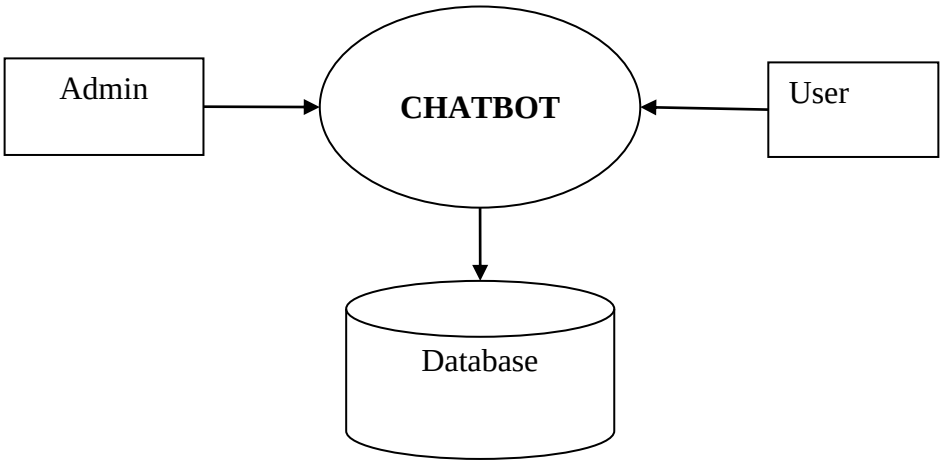
The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week. 24 hours a day.

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

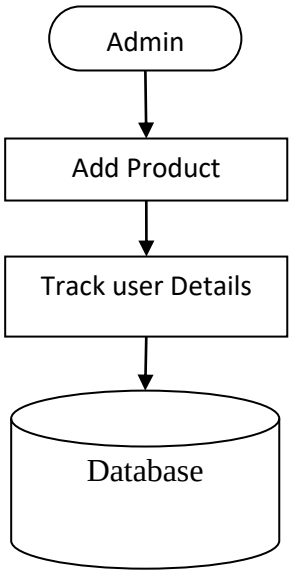
LEVEL 0

The Level 0 DFD shows how the system is divided into 'sub-systems' (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.



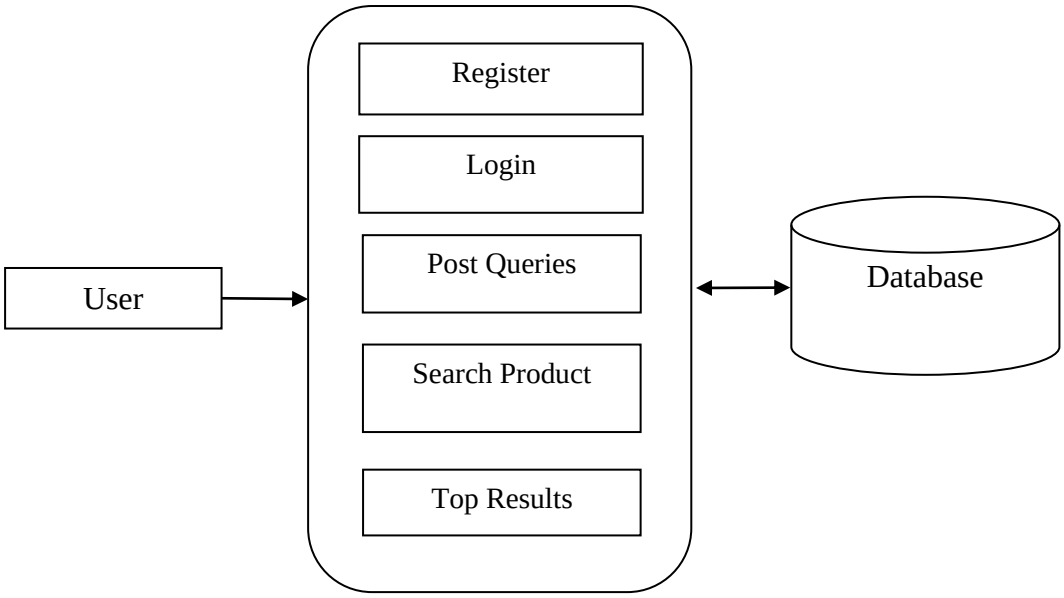
LEVEL 1

The next stage is to create the Level 1 Data Flow Diagram. This highlights the main functions carried out by the system. As a rule, to describe the system was using between two and seven functions - two being a simple system and seven being a complicated system. This enables us to keep the model manageable on screen or paper.



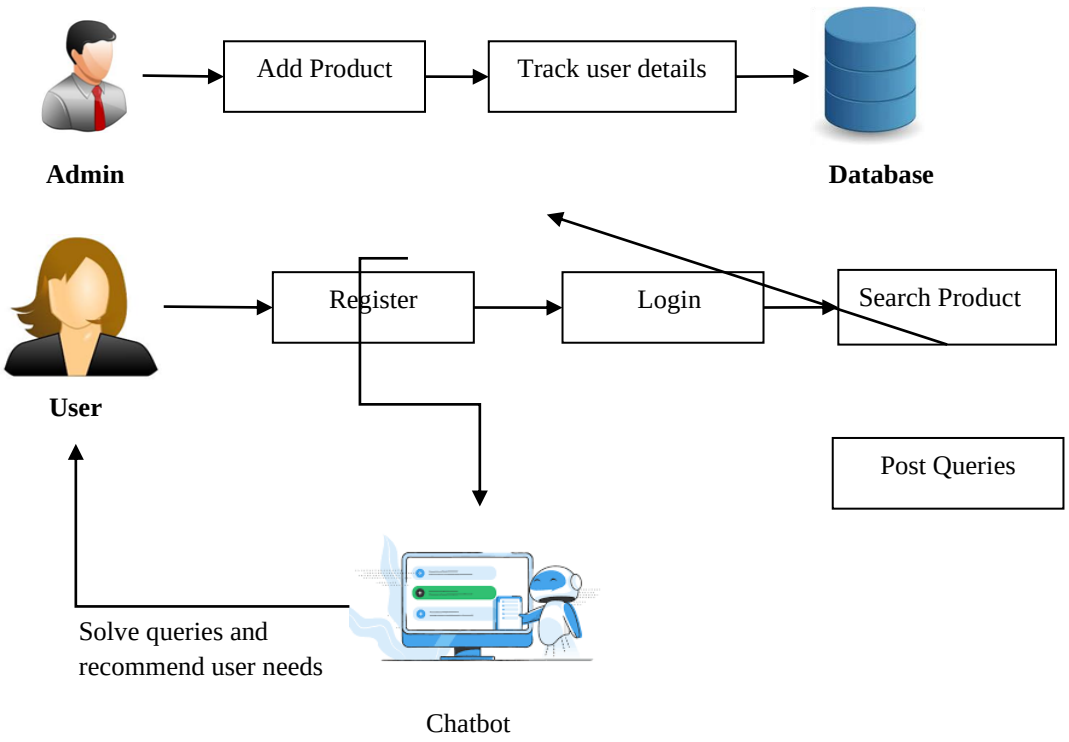
LEVEL 2

A Data Flow Diagram (DFD) tracks processes and their data paths within the business or system boundary under investigation. A DFD defines each domain boundary and illustrates the logical movement and transformation of data within the defined boundary. The diagram shows 'what' input data enters the domain, 'what' logical processes the domain applies to that data, and 'what' output data leaves the domain. Essentially, a DFD is a tool for process modelling and one of the oldest.



5.2 SOLUTION & TECHNICAL ARCHITECTURE

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture; collectively these are called architecture description languages (ADLs).



5.3 USER STORIES

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority
Sprint-1	User Panel	USN-1	The user will login into the website and go through the products available on the website	High
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.	High
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.	High
Sprint-4	final delivery	USN-4	Container of applications using docker kubernetes and deployment the application. Create the documentation and final submit the application	High

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

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

6.2 SPRINT DELIVERY SCHEDULE

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority
Sprint-1	User Panel	USN-1	The user will login into the website and go through the products available on the website	High
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.	High
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.	High
Sprint-4	final delivery	USN-4	Container of applications using docker kubernetes and deployment the application. Create the documentation and final submit the application	High

6.3 REPORTS FROM JIRA

7. CODING & SOLUTIONING

7.1 FEATURE 1

Products page (code):

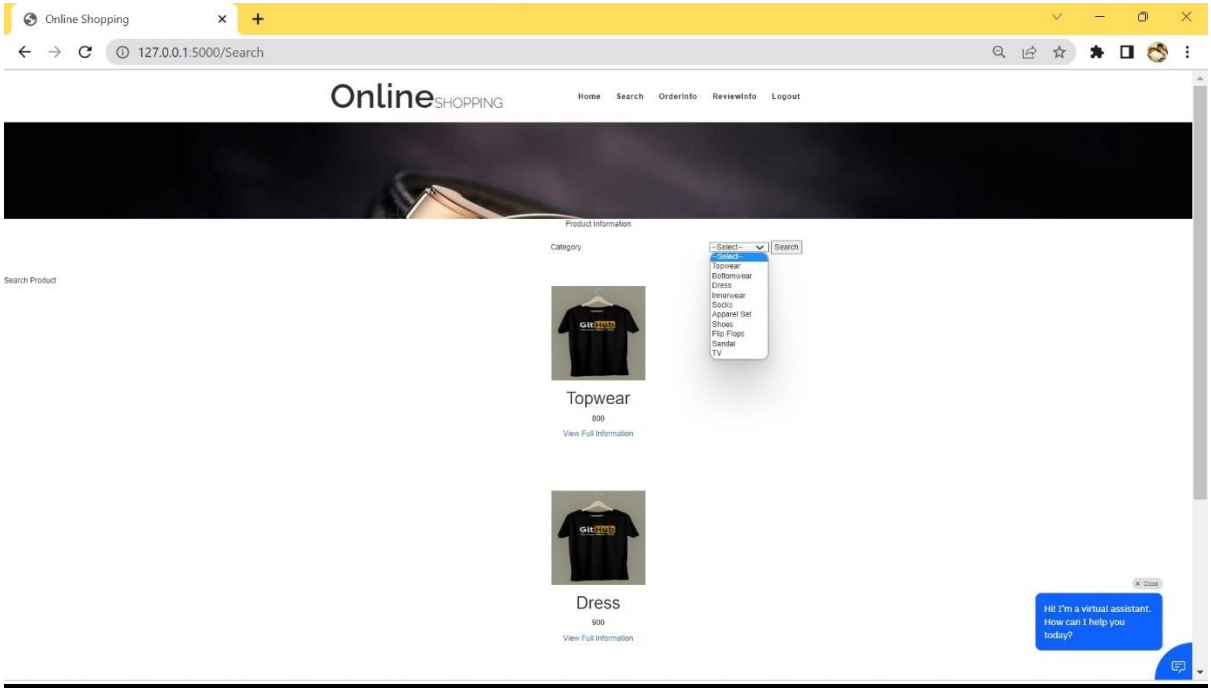
```
<html>
<body>
<div class="banner about-banner">
      <div class="header about-header">
         <div class="container">
            <div class="header-left">
               <div class="w3layouts-logo">
                  <h1>
                     <a href="#">Online<span>Shopping</span></a>
                  </h1>
               </div>
            </div>
            <div class="header-right">
               <div class="top-nav">
                  <nav class="navbar navbar-default">
                        <div class="navbar-header">
                           <button type="button" class="navbar-toggle collapsed"
data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">
                              <span class="sr-only">Toggle navigation</span>
                              <span class="icon-bar"></span>
                              <span class="icon-bar"></span>
                              <span class="icon-bar"></span>
                           </button>
                        </div>
                     <!-- Collect the nav links, forms, and other content for
toggling -->
                     <div class="collapse navbar-collapse" id="bs-example-navbar-
collapse-1">
                        <ul class="nav navbar-nav">

                           <li><a href="/UserHome">Home</a></li>
                           <li><a href="/Search">Search</a></li>
                           <li><a href="/UOrderInfo">OrderInfo</a></li>
                           <li><a href="/UReviewInfo">ReviewInfo</a></li>
                                    <li><a href="/">Logout</a></li>

                        </ul>
                        <div class="clearfix"> </div>
                     </div>
                  </nav>
               </div>
               <div class="agileinfo-social-grids">
                  <ul>
                     <li><a href="#"><i class="fa fa-facebook"></i></a></li>
                     <li><a href="#"><i class="fa fa-twitter"></i></a></li>
                     <li><a href="#"><i class="fa fa-rss"></i></a></li>
                     <li><a href="#"><i class="fa fa-vk"></i></a></li>
                  </ul>
               </div>
               <div class="clearfix"> </div>
            </div>
```

```
            <div class="clearfix"> </div>
        </div>
      </div>
      <div class="about-heading">
         <div class="container">

            <h2>  </h2>
                <h2>   </h2>
        </div>
      </div>
   </div>
<div align="center" class="style2"> Product Information </div>

    <form id="form1" name="form1" method="post" action="/viewproduct">
          <table width="100%" border="0">
               <tr>
                 <td width="46%"> </td>
                 <td width="54%" colspan="2"> </td>
               </tr>
               <tr>
                 <td rowspan="3"> </td>
                 <td>Category</td>
                 <td><select name="subcat" id="subcat">
                   <option value="--Select--">--Select--</option>
                   <option value="Topwear">Topwear</option>
                   <option value="Bottomwear">Bottomwear</option>
                   <option value="Dress">Dress</option>
                   <option value="Innerwear">Innerwear</option>
                   <option value="Socks">Socks</option>
                   <option value="Apparel Set">Apparel Set</option>
                   <option value="Shoes">Shoes</option>
                   <option value="Flip Flops">Flip Flops</option>
                   <option value="Sandal">Sandal</option>
                   <option value="TV">TV</option>
                 </select>
                 <input type="submit" name="Submit" value="Search" /></td>
               </tr>
               <tr>
                 <td colspan="2"> </td>
               </tr>
               <tr>
                 <td colspan="2"> </td>
               </tr>
               <tr>
                 <td>Search Product </td>
                 <td colspan="2"> </td>
               </tr>
               <tr>

                 <td colspan="3">    {% for item in data %}
   <div>


        <div class="card" align="center">
  <img src="{{item[10]}}" alt="{{item[4]}}"  width="200" height="200">
  <h1>{{item[4]}}</h1>
  <p class="price">{{item[11]}}</p>

         <p><div align="center"><a href="fullInfo?pid={{item[1]}}">View Full
```

```
Information</a></div> </p>

        <p>            </p>
        <p>  </p>
        <p>  </p>
</div>

     <div>     </div>
     <div>     </div>
     {% endfor %}    </div></td>
               </tr>
            </table>
</form>
     <!-- copyright -->
   <div class="copyright">
     <div class="container">
        <p>©  All rights reserved | Design by <a href="#">Online Shopping</a></p>
     </div>
   </div>

<script>
  window.watsonAssistantChatOptions = {
    integrationID: "8d5041f5-9017-49db-9aa8-d28dd7214b24", // The ID of this
integration.
    region: "us-south", // The region your integration is hosted in.
    serviceInstanceID: "38398766-818e-483c-a986-96a5c42e07a1", // 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>
   <!-- //copyright -->
   <script src="static/js/responsiveslides.min.js"></script>
   <script src="static/js/SmoothScroll.min.js"></script>
<script type="text/javascript" src="static/js/move-top.js"></script>
<script type="text/javascript" src="static/js/easing.js"></script>
   <!-- here stars scrolling icon -->
   <script type="text/javascript">
       $(document).ready(function () {
           /*
           var defaults = {
           containerID: 'toTop', // fading element id
           containerHoverID: 'toTopHover', // fading element hover id
           scrollSpeed: 1200,
           easingType: 'linear'
           };
           */

           $().UItoTop({ easingType: 'easeOutQuart' });

       });
   </script>
```

```
<!-- //here ends scrolling icon -->
</body>
</html>
```

Products page (output):



7.2 FEATURE 2

IBM Watson chat assistant(code)

```
<!DOCTYPE html>
<html>

<head lang="en">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Chatbot</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.6.3/css/font-
awesome.min.css" rel="stylesheet">
    <style type="text/css">
        .fixed-panel {
            min-height: 400px;
            max-height: 400px;
            background-color: #19313c;
            color: white;
```

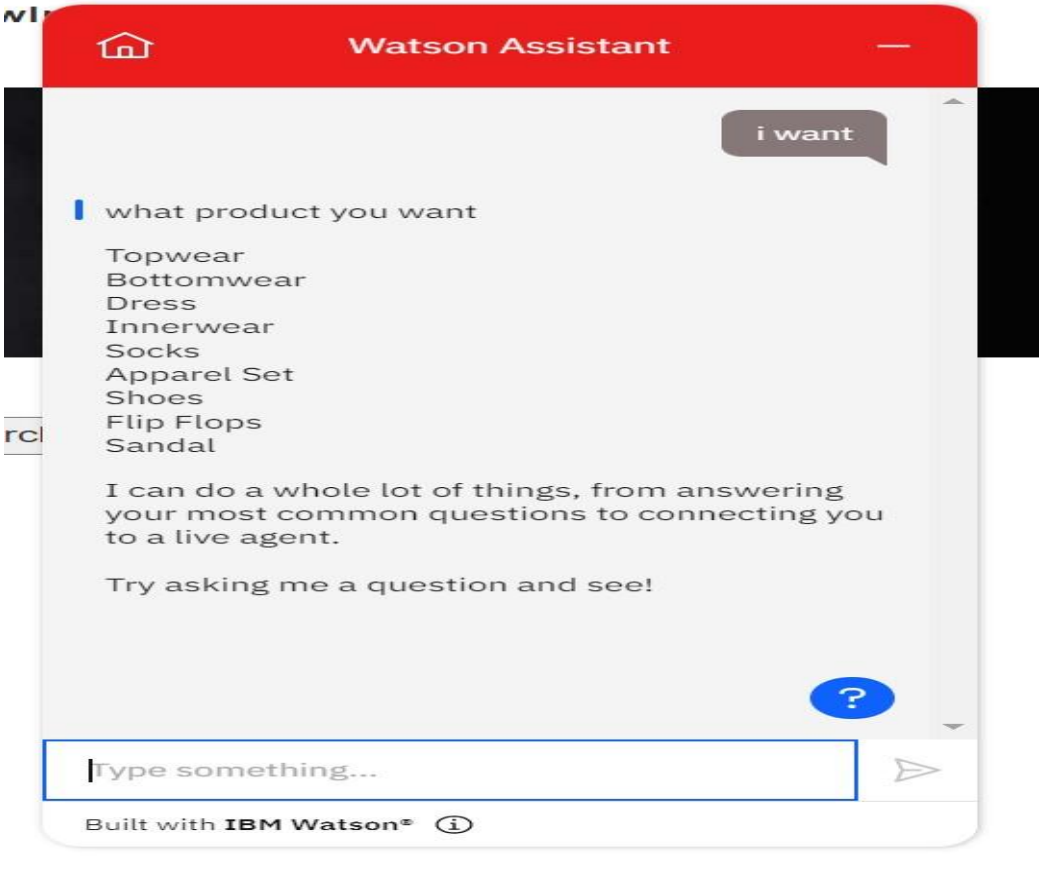
```
            overflow: auto;
        }

        .media-list {
            overflow: auto;
            clear: both;
            display: table;
            overflow-wrap: break-word;
            word-wrap: break-word;
            word-break: normal;
            line-break: strict;
        }

        .panel {
            margin-bottom: 20px;
            background-color: #fff;
            border: 6px solid transparent;
            border-radius: 25px;
            -webkit-box-shadow: 0 1px 1px rgba(0, 0, 0, .05);
            box-shadow: 0 1px 1px rgba(0, 0, 0, .05);
        }

        .panel-info {
            border-color: #0c2735;
        }

        .panel-info>.panel-heading {
            color: white;
            background-color: #0c2735;
            border-color: #0c2735;
        }

        .panel-footer {
            padding: 10px 15px;
            background-color: #0c2735;
            border-top: 1px solid #0c2735;
            border-bottom-right-radius: 3px;
            border-bottom-left-radius: 3px
        }

        body {
            background: rgb(96, 143, 149);
            /* Old browsers */
            background: -moz-linear-gradient(-45deg, rgba(96, 143, 149, 1) 0%,
rgba(0, 133, 136, 1) 9%, rgba(12, 36, 73, 1) 52%, rgba(26, 30, 59, 1) 100%);
            /* FF3.6-15 */
            background: -webkit-linear-gradient(-45deg, rgba(96, 143, 149, 1) 0%,
rgba(0, 133, 136, 1) 9%, rgba(12, 36, 73, 1) 52%, rgba(26, 30, 59, 1) 100%);
            /* Chrome10-25,Safari5.1-6 */
            background: linear-gradient(135deg, rgba(96, 143, 149, 1) 0%, rgba(0,
133, 136, 1) 9%, rgba(12, 36, 73, 1) 52%, rgba(26, 30, 59, 1) 100%);
            /* W3C, IE10+, FF16+, Chrome26+, Opera12+, Safari7+ */
            filter:
progid:DXImageTransform.Microsoft.gradient(startColorstr='#608f95',
endColorstr='#1a1e3b', GradientType=1);
            /* IE6-9 fallback on horizontal gradient */
        }
    </style>
</head>
```

```
<body>
    <div class="container background-color: rgb(255,0,255);">
        <div class="row">
            <h3 class="text-center"><small><strong>ChatBot</strong></small>
                <font color="white"> </font><small><strong></strong></small>
                <font color="white"></font>
            </h3>

            <div class="col-md-4 col-md-offset-4">
                <div id="chatPanel" class="panel panel-info">
                    <div class="panel-heading">
                        <strong><span class="glyphicon glyphicon-globe"></span>
Talk with Me !!! (You: Green / Bot:
                            White) </strong>
                    </div>
                    <div class="panel-body fixed-panel">
                        <ul class="media-list">
                        </ul>
                    </div>
                    <div class="panel-footer">
                        <form method="post" id="chatbot-form">
                            <div class="input-group">
                                <input type="text" class="form-control"
placeholder="Enter Message" name="messageText"
                                    id="messageText" autofocus />
                                <span class="input-group-btn">
                                    <button class="btn btn-info" type="button"
id="chatbot-form-btn">Send</button>
                                     <button class="btn btn-info" type="button"
                                        id="chatbot-form-btn-voice">Voice</button>
                                    <button class="btn btn-info" type="button"
                                        id="chatbot-form-btn-clear">Clear</button>

                                </span>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script
>
    <script>
        var exports = {};
    </script>
    <script src="https://unpkg.com/speech-to-text@0.7.4/lib/index.js"></script>
    <script>
        $(function () {
            var synth = window.speechSynthesis;

            var msg = new SpeechSynthesisUtterance();
            var voices = synth.getVoices();
            msg.voice = voices[0];
            msg.rate = 1;
            msg.pitch = 1;
```

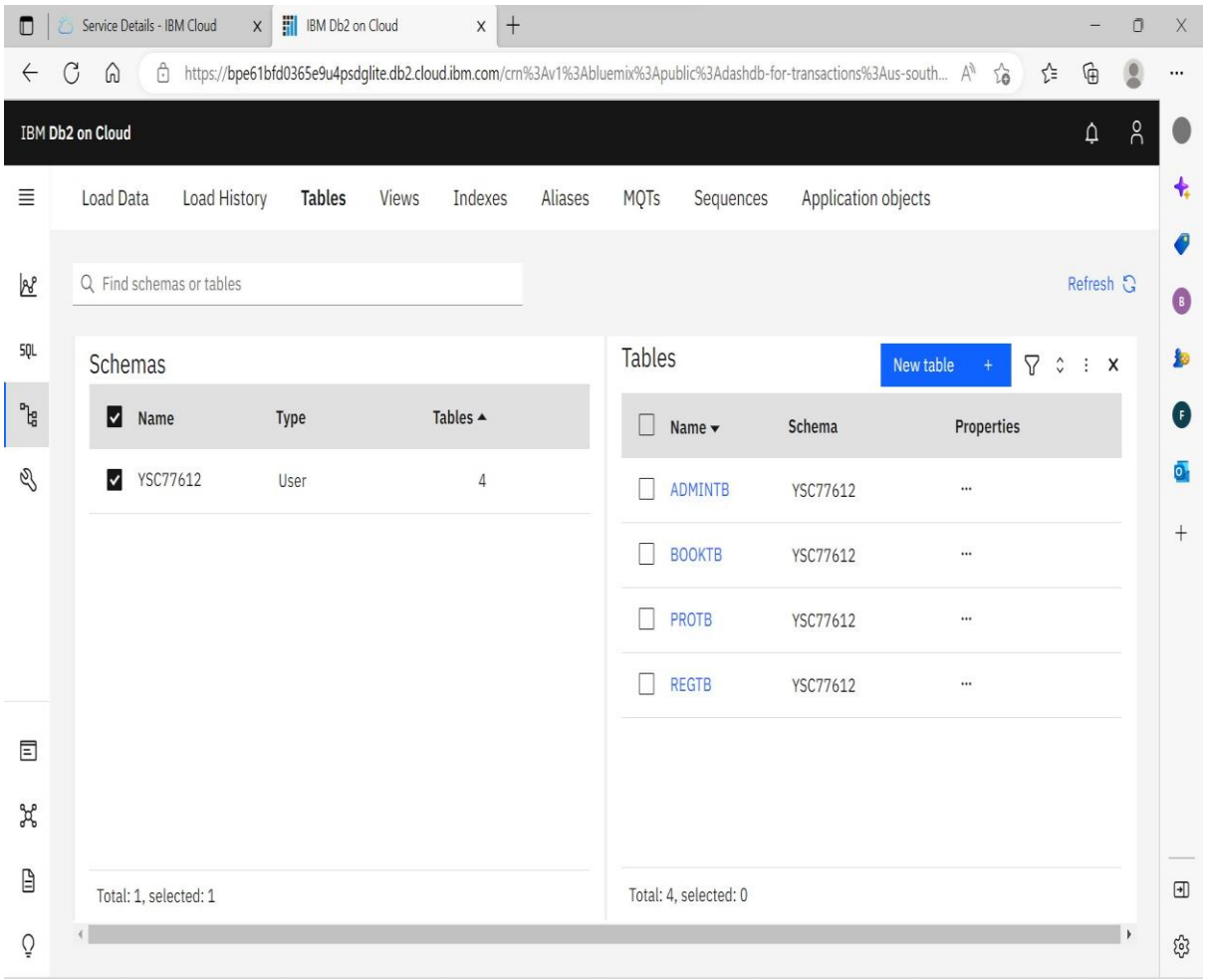
```
            $('#chatbot-form-btn').click(function (e) {
                e.preventDefault();
                $('#chatbot-form').submit();
            });
            $('#chatbot-form-btn-clear').click(function (e) {
                e.preventDefault();
                $('#chatPanel').find('.media-list').html('');
            });
            $('#chatbot-form-btn-voice').click(function (e) {
                e.preventDefault();

                var onAnythingSaid = function (text) {
                    console.log('Interim text: ', text);
                };
                var onFinalised = function (text) {
                    console.log('Finalised text: ', text);
                    $('#messageText').val(text);
                };
                var onFinishedListening = function () {
                    // $('#chatbot-form-btn').click();
                };

                try {
                    var listener = new SpeechToText(onAnythingSaid, onFinalised,
onFinishedListening);
                    listener.startListening();

                    setTimeout(function () {
                        listener.stopListening();
                        if ($('#messageText').val()) {
                            $('#chatbot-form-btn').click();
                        }
                    }, 5000);
                } catch (error) {
                    console.log(error);
                }
            });

            $('#chatbot-form').submit(function (e) {
                e.preventDefault();
                var message = $('#messageText').val();
                $(".media-list").append(
                    '<li class="media"><div class="media-body"><div
class="media"><div style = "text-align:right; color : #2EFE2E" class="media-
body">' +
                    message + '<hr/></div></div></div></li>');

                $.ajax({
                    type: "POST",
                    url: "/ask",
                    data: $(this).serialize(),
                    success: function (response) {
                        $('#messageText').val('');
                        var answer = response.answer;
                        const chatPanel = document.getElementById("chatPanel");
                        $(".media-list").append(
                            '<li class="media"><div class="media-body"><div
class="media"><div style = "color : white" class="media-body">' +
```

```
                        answer + '<hr/></div></div></div></li>');
                    $(".fixed-panel").stop().animate({
                        scrollTop: $(".fixed-panel")[0].scrollHeight
                    }, 1000);

                    msg.text = answer;
                    speechSynthesis.speak(msg);
                },
                error: function (error) {
                    console.log(error);
                }
            });
        });
    });
    </script>
</body>

</html>
```

IBM Watson chat assistant(output)



7.3 DATABASE SCHEMA



8. TESTING

8.1 TEST 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.

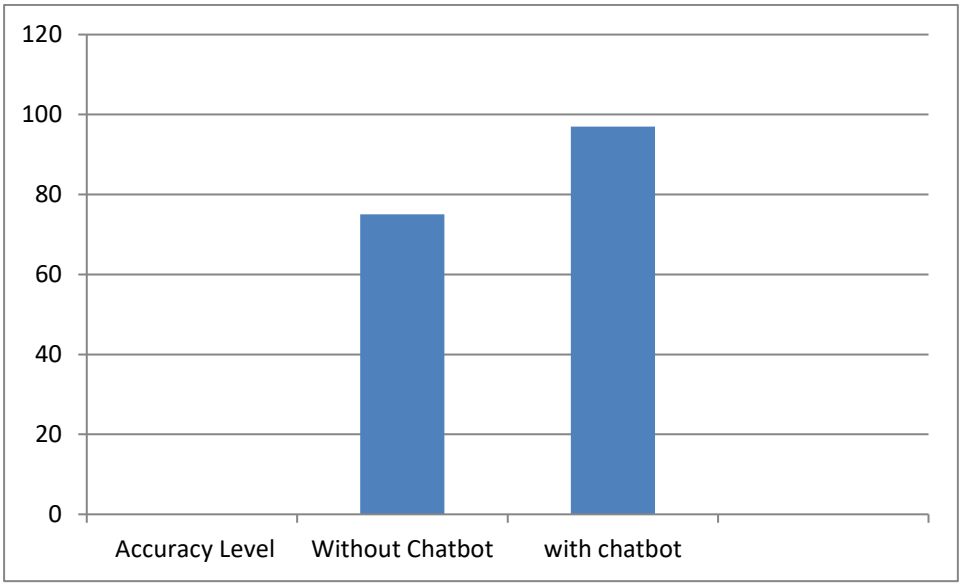
S.NO	**Scenario**	**Input**	**Excepted output**	**Actual output**
1	User login	User name and password	Login	Login success.
2	Search Product	Show product list	Purchase successfully	User details are stored in a database.
3	Post Queries	Ask questions to chatbot	Get result from chatbot	Details are stored in a database.

8.2 USER 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

9.1 PERFORMANCE 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())
```

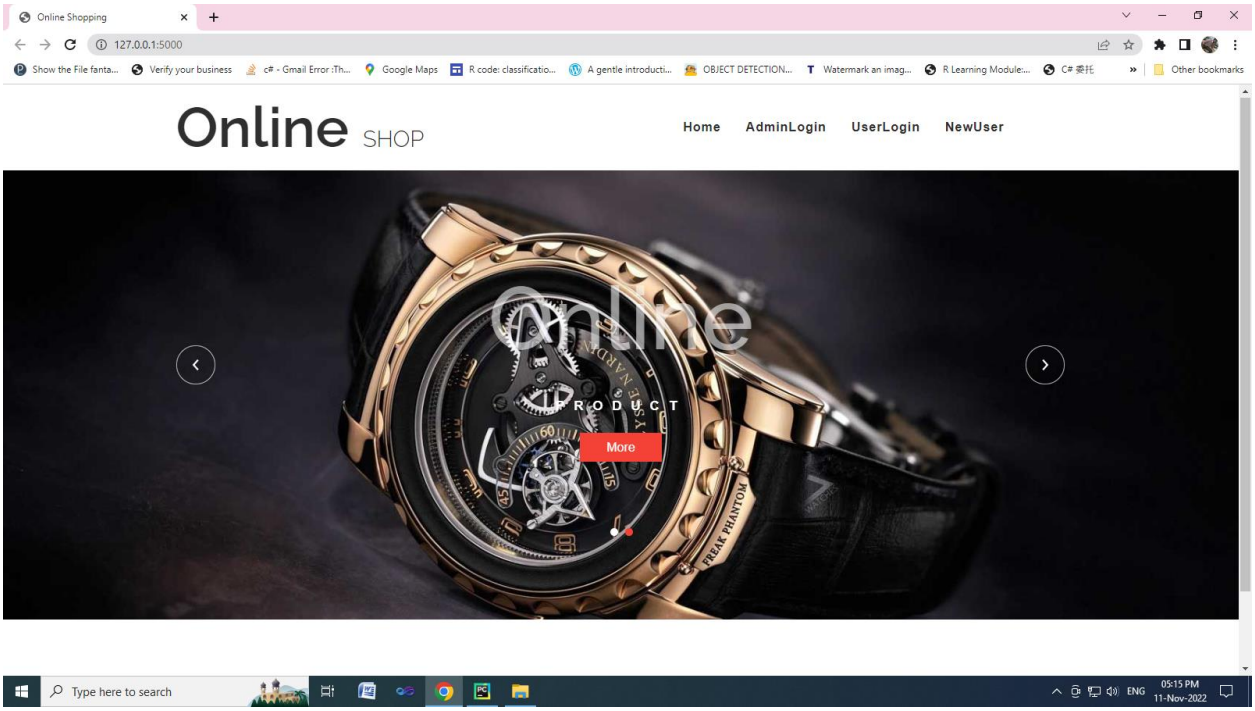
```
        return render_template('AdminHome.html', data=engine.execute("SELECT *
FROM Employee_Data").fetchall())
@app.route("/Remove", methods=['GET'])
def Remove():
    pid = request.args.get('id')
    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    insertQuery = "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())
```

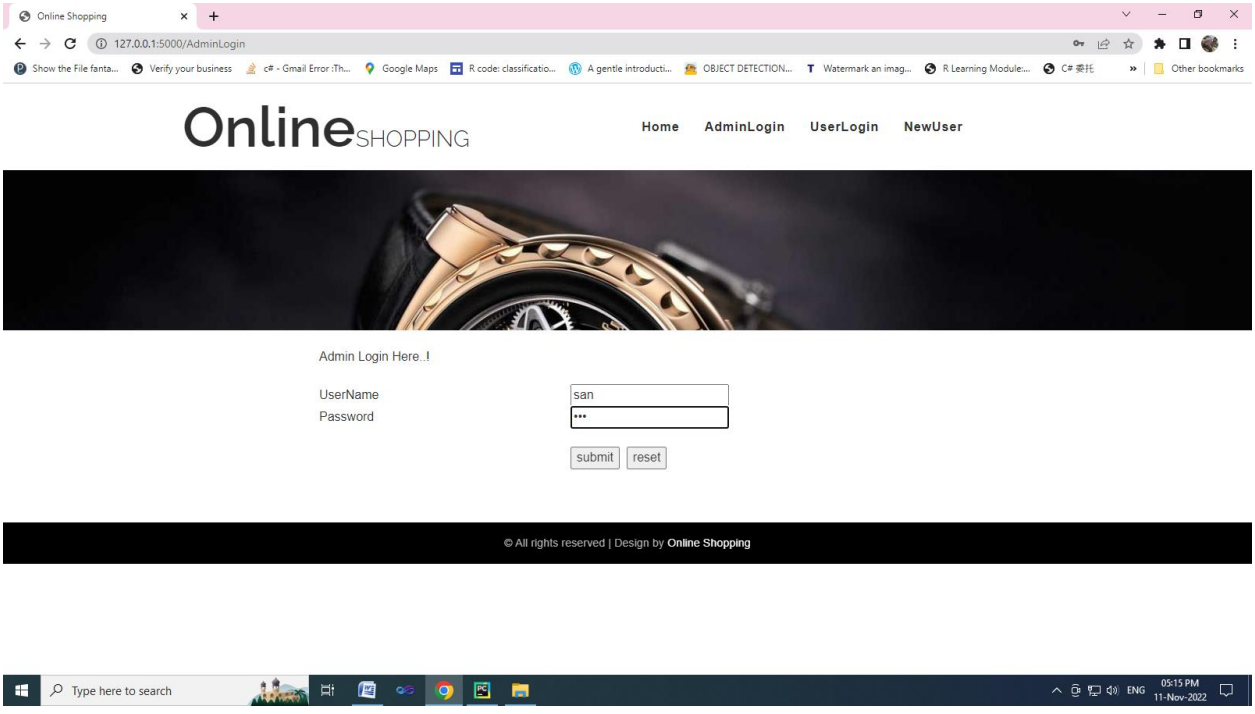
```
@app.route("/Book", methods=['GET', 'POST'])
def Book():
    if request.method == 'POST':
        uname = session['uname']
        pid = session['pid']
        qty = request.form['qty']
        ctype = request.form['ctype']
        cardno = request.form['cardno']
        cvno = request.form['cvno']
        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)
```

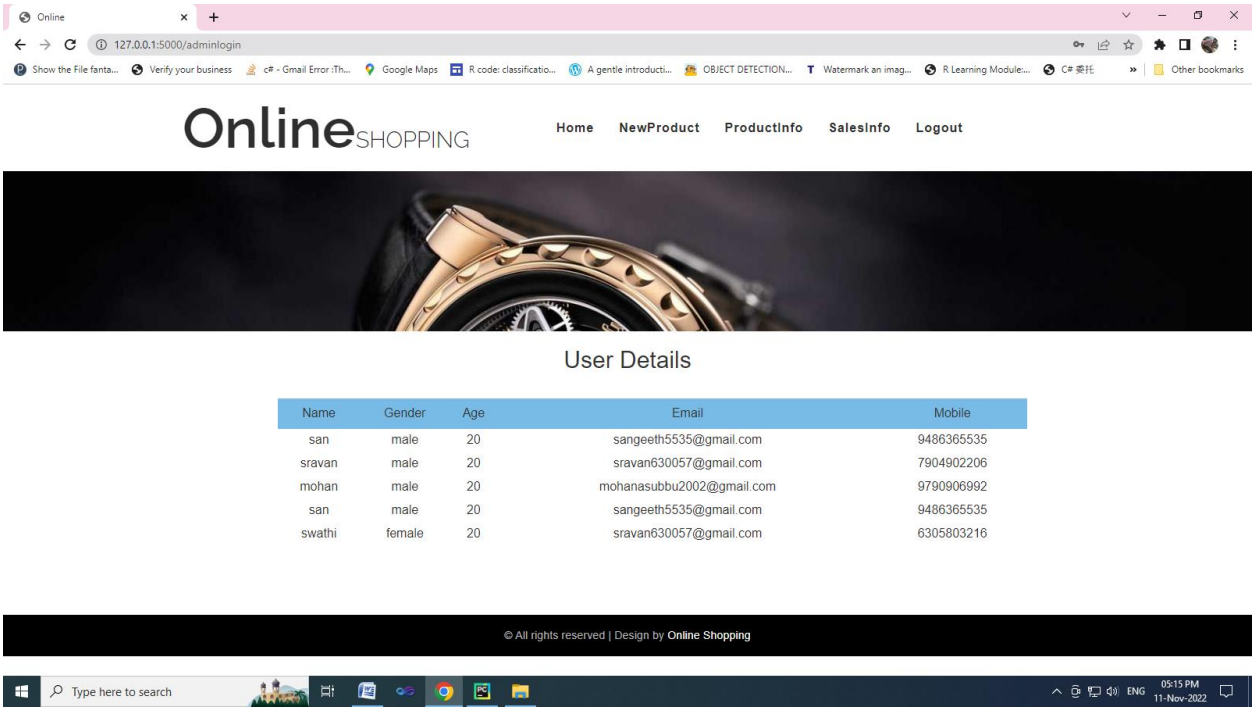
```
        dataframe.to_sql('regtb', con=engine, if_exists='append')
        data1 = engine.execute("SELECT * FROM regtb").fetchall()
        for item1 in data1:
            Mobile = item1[5]
            Email = item1[4]
        selectQuery = "SELECT  *  FROM  booktb"
        dataframe = pandas.read_sql(selectQuery, 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"
```

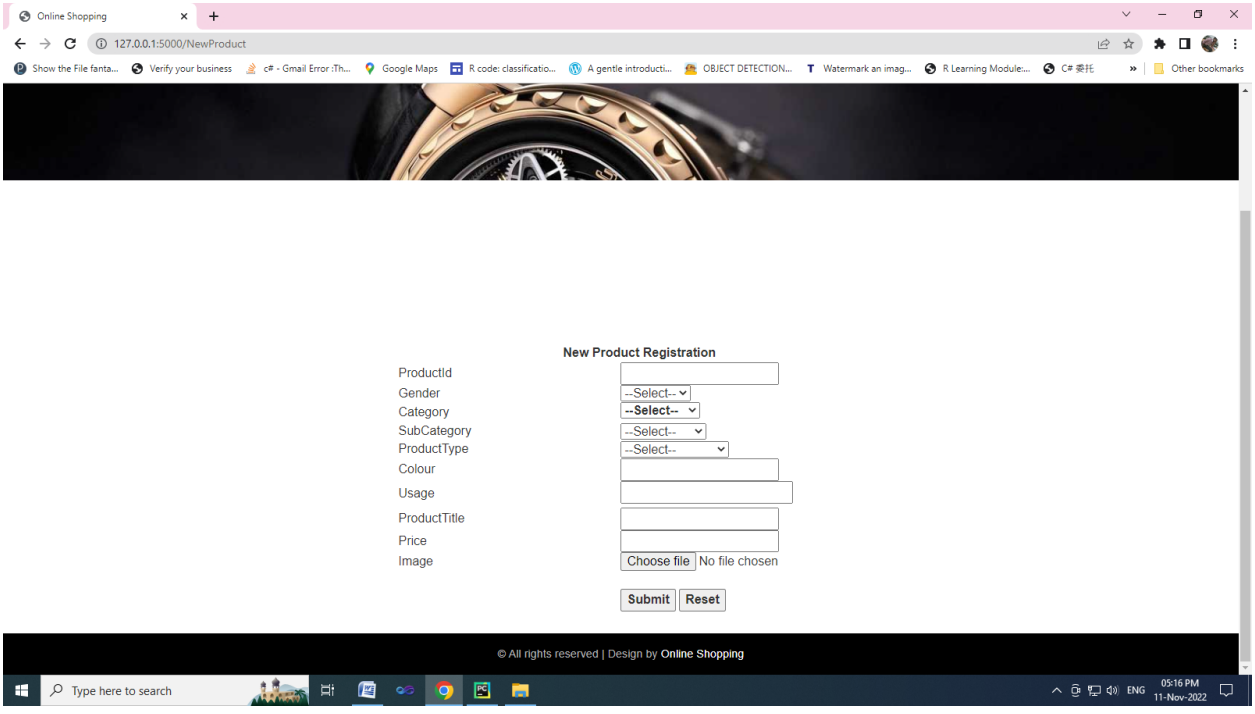
```
    toaddr = Mailid

    # 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")
```

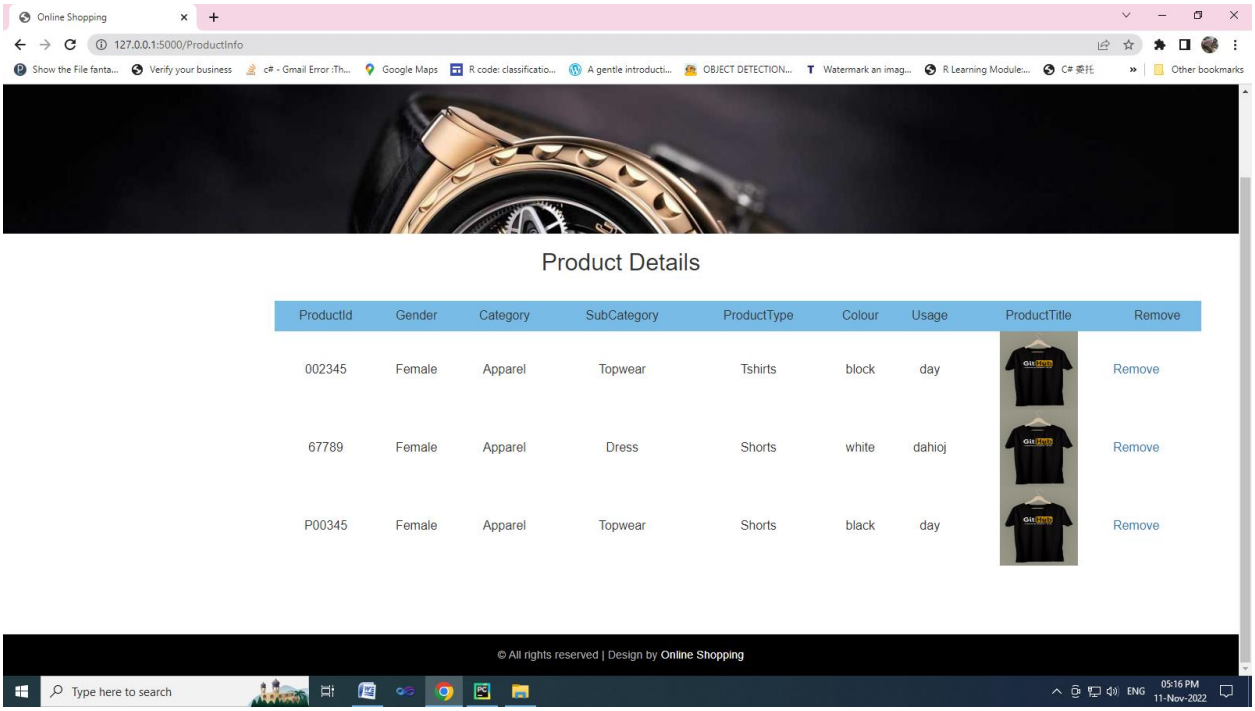
```
def UOrderInfo():
    uname = session['uname']
    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM booktb where  UserName= '" + uname + "' "
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('booktb1', con=engine, 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()
```

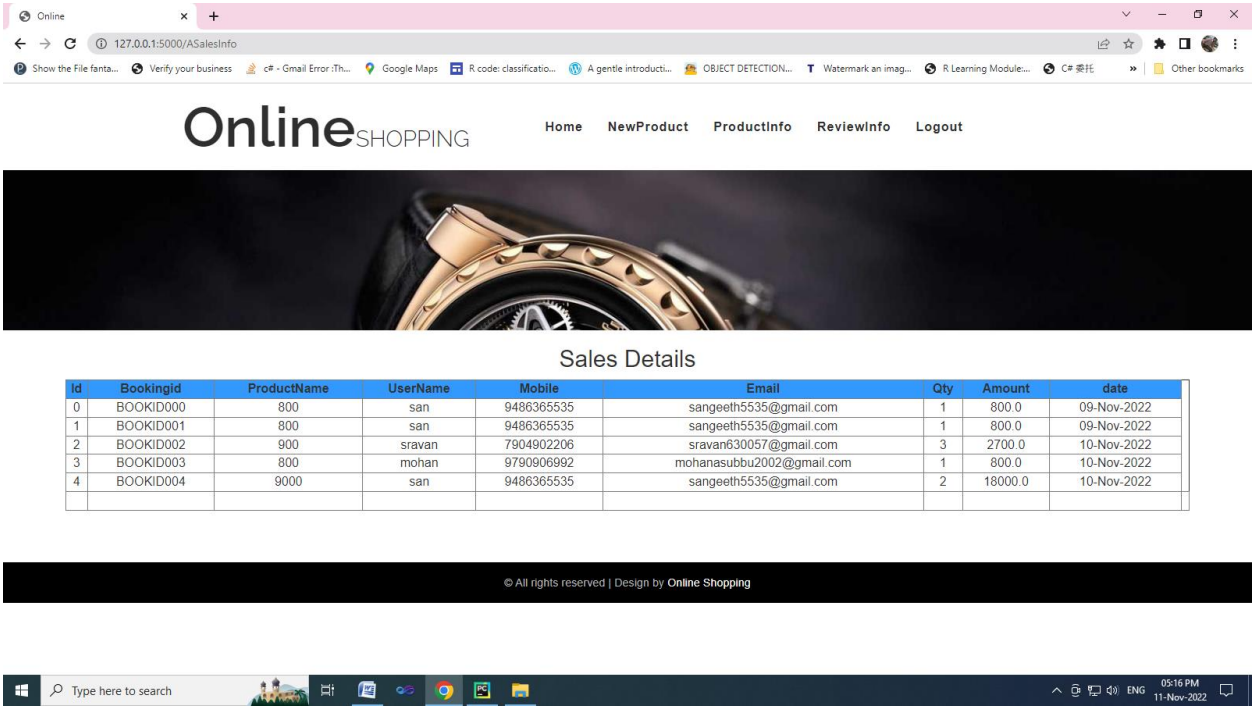

Online Shopping
127.0.0.1:5000
Online SHOP
Home
AdminLogin
UserLogin
NewUser
PRODUCT
More
Type here to search
ENG
05:15 PM
11-Nov-2022

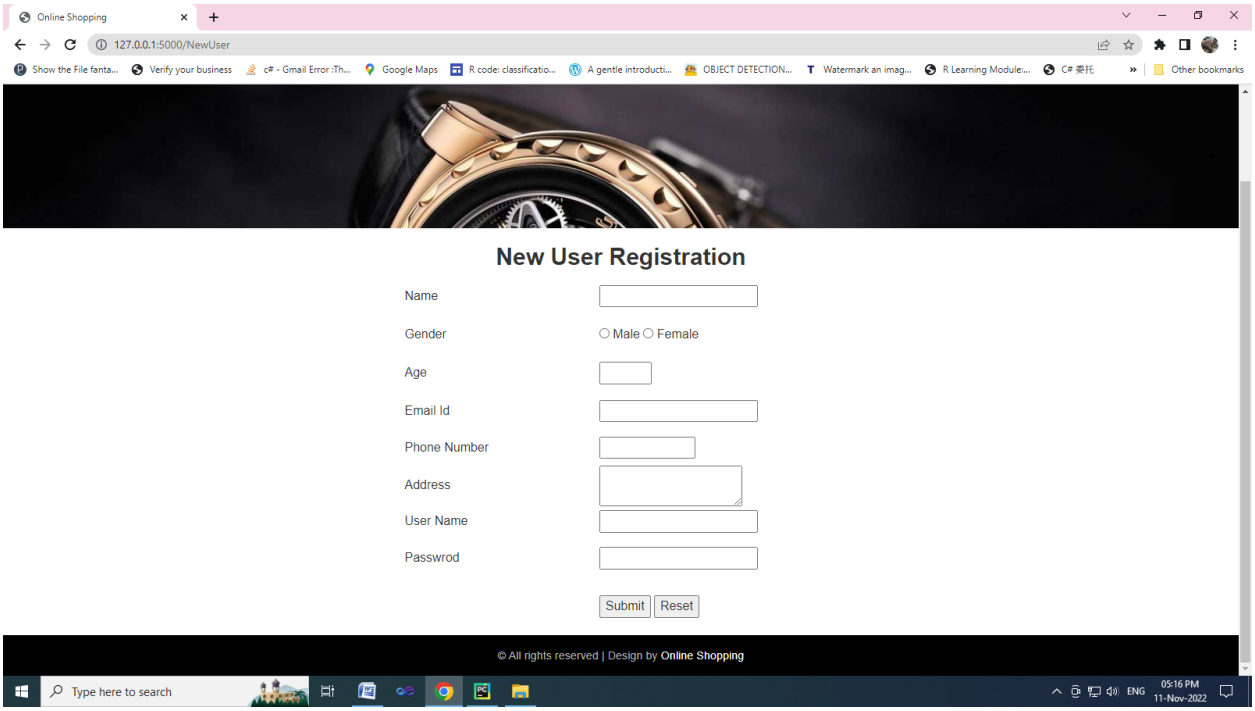

Online Shopping
127.0.0.1:5000/AdminLogin
Online SHOPPING
Home
AdminLogin
UserLogin
NewUser
Admin Login Here..!
UserName
san
Password
submit
reset
© All rights reserved | Design by Online Shopping
Type here to search
ENG
05:15 PM
11-Nov-2022

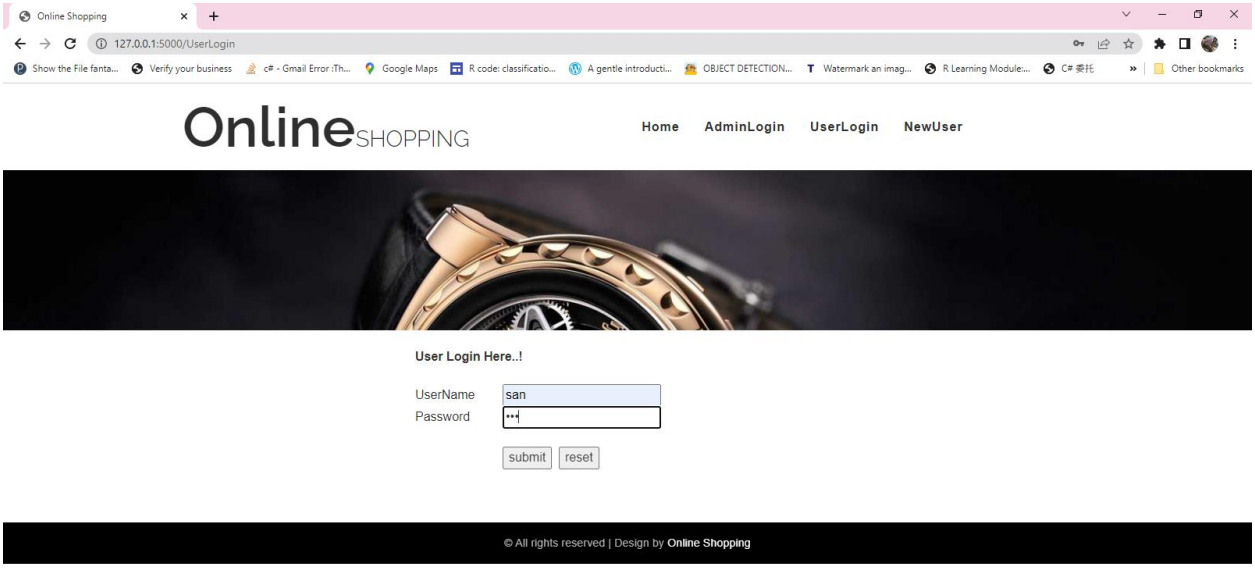

Online
127.0.0.1:5000/adminlogin
Online SHOPPING
Home
NewProduct
ProductInfo
SalesInfo
Logout
User Details
Name Gender Age Email Mobile
san male 20 sangeeth5535@gmail.com 9486365535
sravan male 20 sravan630057@gmail.com 7904902206
mohan male 20 mohanasubbu2002@gmail.com 9790906992
san male 20 sangeeth5535@gmail.com 9486365535
swathi female 20 sravan630057@gmail.com 6305803216
© All rights reserved | Design by Online Shopping
Type here to search
ENG
05:15 PM
11-Nov-2022


Online Shopping
127.0.0.1:5000/NewProduct
New Product Registration
ProductId
Gender
--Select--
Category
--Select--
SubCategory
--Select--
ProductType
--Select--
Colour
Usage
ProductTitle
Price
Image
Choose file
No file chosen
Submit
Reset
© All rights reserved | Design by Online Shopping
Type here to search
ENG
05:16 PM
11-Nov-2022

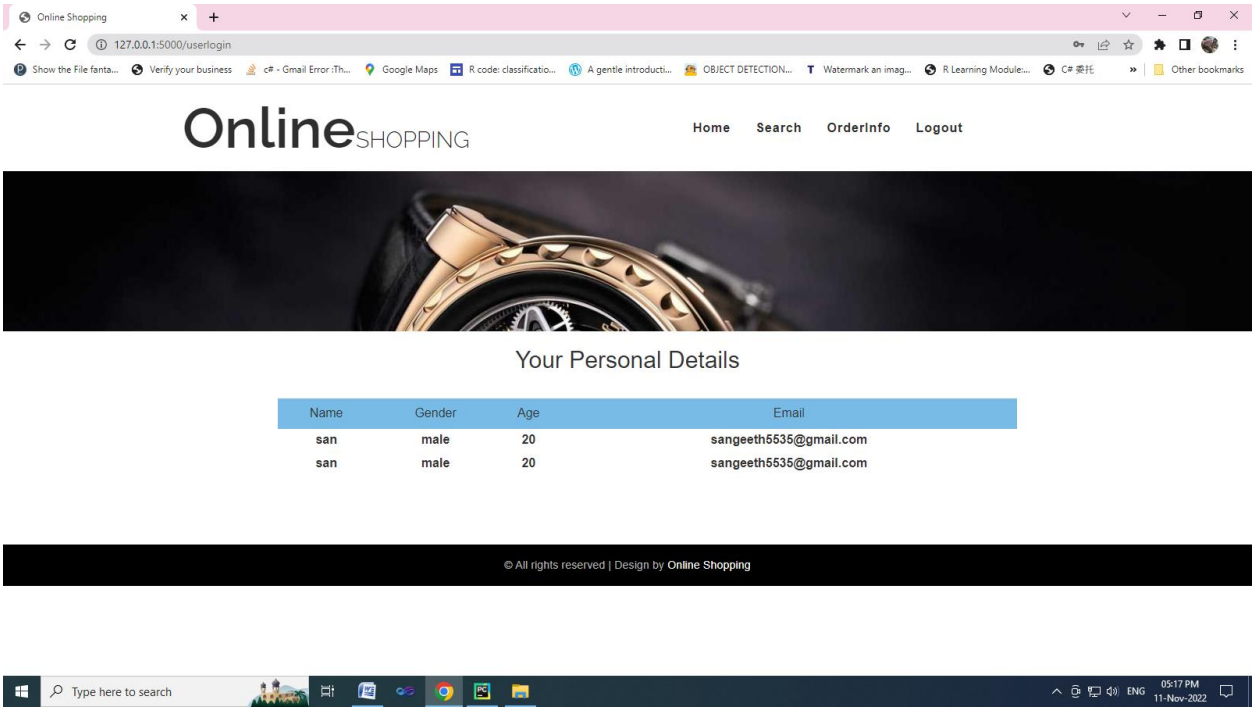

Online Shopping
127.0.0.1:5000/ProductInfo
Product Details
ProductId Gender Category SubCategory ProductType Colour Usage ProductTitle Remove
002345 Female Apparel Topwear Tshirts block day Remove
67789 Female Apparel Dress Shorts white dahioj Remove
P00345 Female Apparel Topwear Shorts black day Remove
© All rights reserved | Design by Online Shopping
Type here to search
ENG 05:16 PM 11-Nov-2022


Online
127.0.0.1:5000/ASalesInfo
Online SHOPPING
Home NewProduct ProductInfo ReviewInfo Logout
Sales Details
Id Bookingid ProductName UserName Mobile Email Qty Amount date
0 BOOKID000 800 san 9486365535 sangeeth5535@gmail.com 1 800.0 09-Nov-2022
1 BOOKID001 800 san 9486365535 sangeeth5535@gmail.com 1 800.0 09-Nov-2022
2 BOOKID002 900 sravan 7904902206 sravan630057@gmail.com 3 2700.0 10-Nov-2022
3 BOOKID003 800 mohan 9790906992 mohanasubbu2002@gmail.com 1 800.0 10-Nov-2022
4 BOOKID004 9000 san 9486365535 sangeeth5535@gmail.com 2 18000.0 10-Nov-2022
© All rights reserved | Design by Online Shopping
Type here to search
ENG 05:16 PM 11-Nov-2022

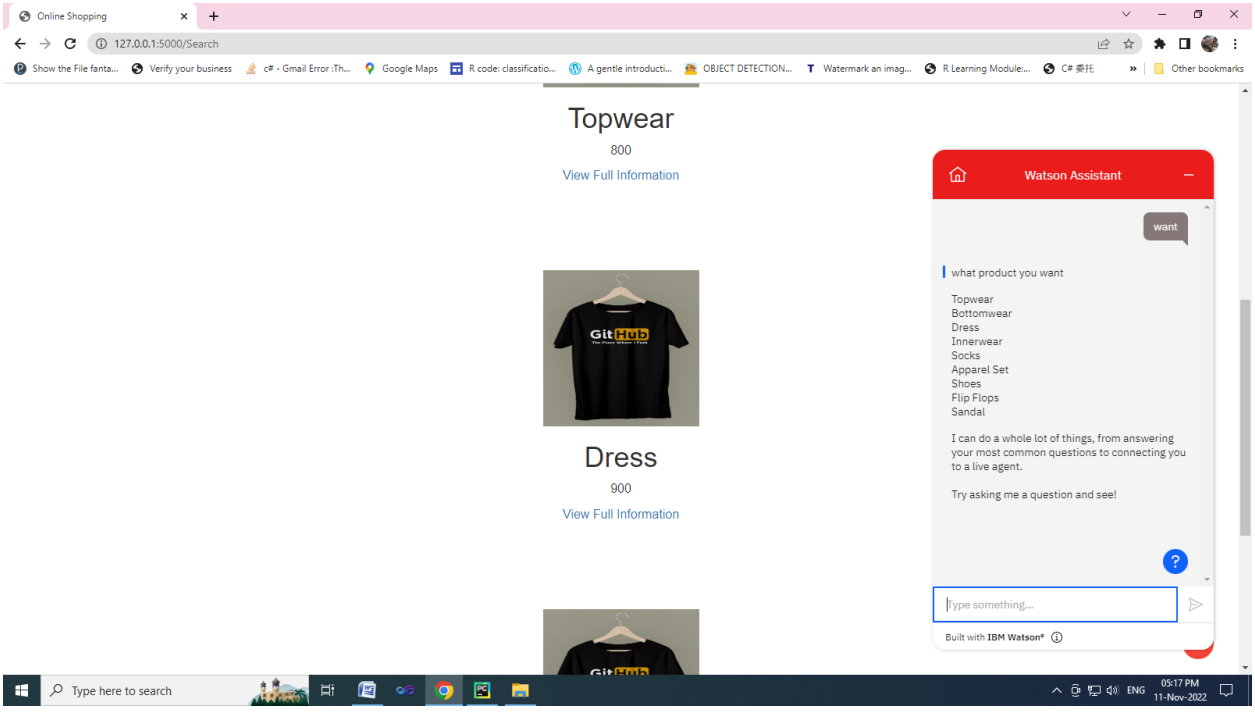

Online Shopping
127.0.0.1:5000/NewUser
New User Registration
Name
Gender
Male Female
Age
Email Id
Phone Number
Address
User Name
Passwrod
Submit
Reset
© All rights reserved | Design by Online Shopping
Type here to search
ENG
05:16 PM
11-Nov-2022

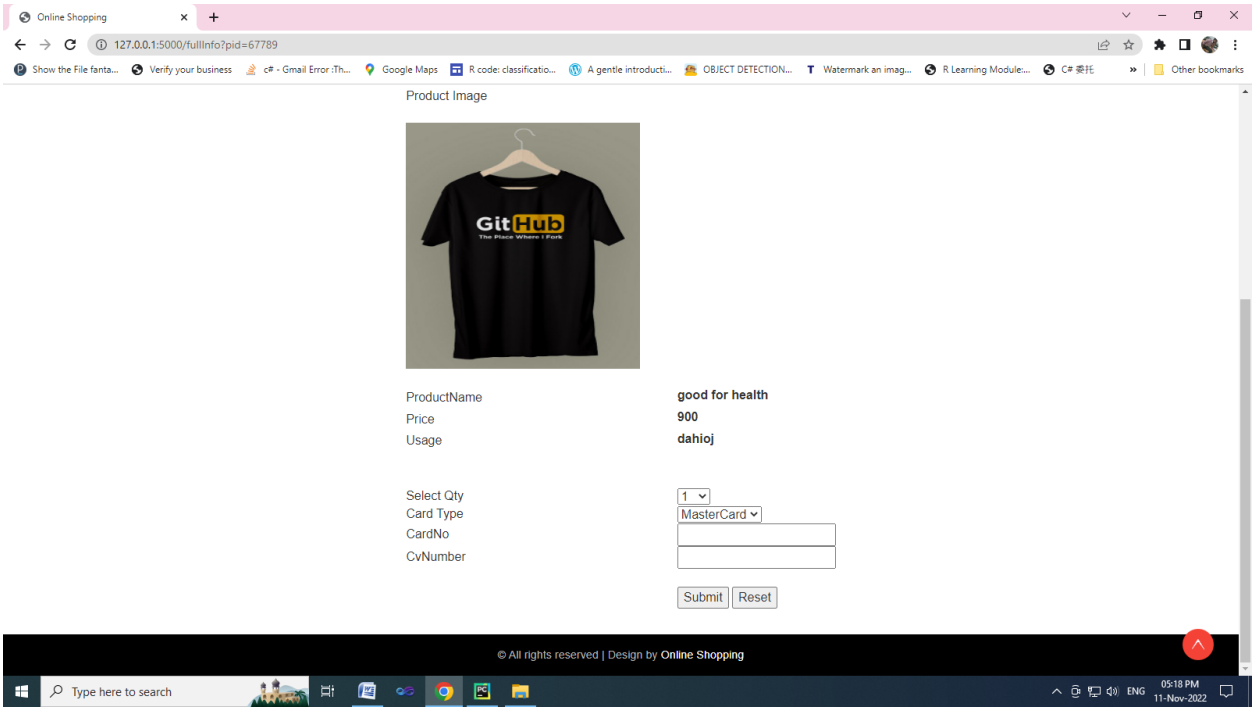

Online Shopping
127.0.0.1:5000/UserLogin
Online SHOPPING
Home
AdminLogin
UserLogin
NewUser
User Login Here..!
UserName
san
Password
submit
reset
© All rights reserved | Design by Online Shopping

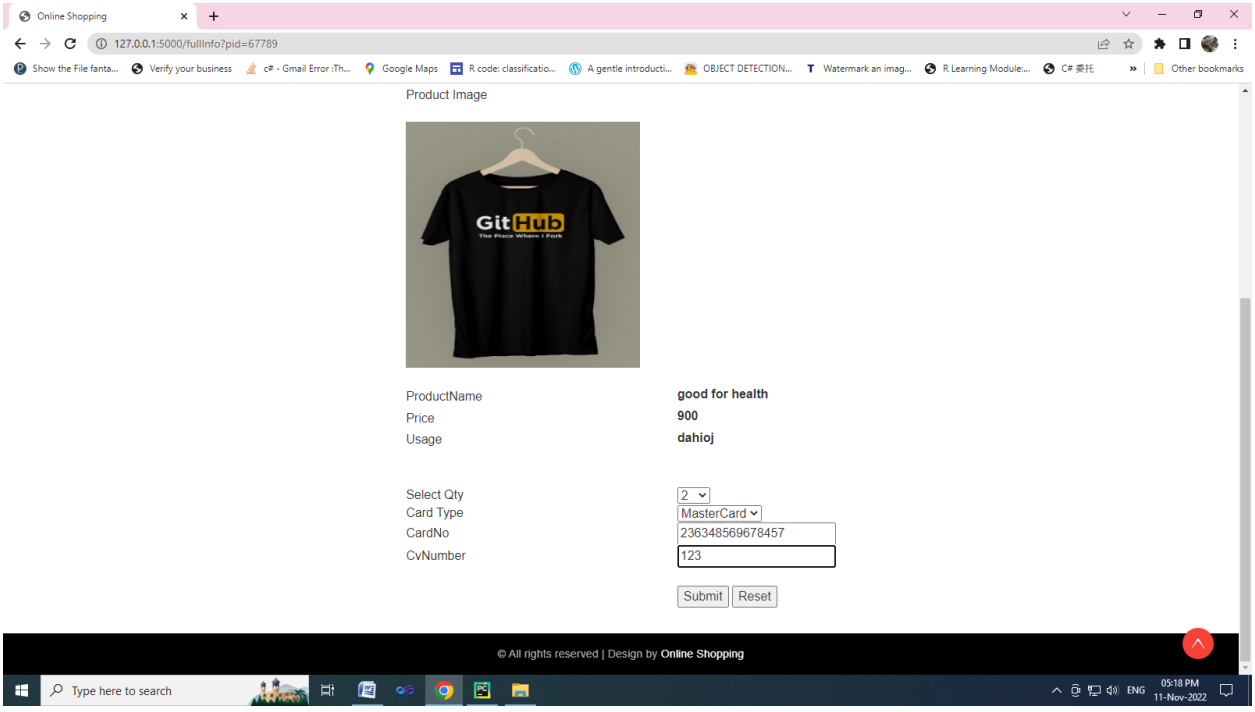
Type here to search
ENG
05:16 PM
11-Nov-2022


Online Shopping
127.0.0.1:5000/userlogin
Online SHOPPING
Home
Search
OrderInfo
Logout
Your Personal Details
Name
Gender
Age
Email
san
male
20
sangeeth5535@gmail.com
san
male
20
sangeeth5535@gmail.com
© All rights reserved | Design by Online Shopping

Type here to search
ENG
05:17 PM
11-Nov-2022


Online Shopping
127.0.0.1:5000/Search
Topwear
800
View Full Information
Dress
900
View Full Information
Watson Assistant
want
what product you want
Topwear
Bottomwear
Dress
Innerwear
Socks
Apparel Set
Shoes
Flip Flops
Sandal
I can do a whole lot of things, from answering your most common questions to connecting you to a live agent.
Try asking me a question and see!
Type something...
Built with IBM Watson®
Type here to search
ENG
05:17 PM
11-Nov-2022


Online Shopping
127.0.0.1:5000/fullInfo?pid=67789
Product Image
ProductName
good for health
Price
900
Usage
dahioj
Select Qty
1
Card Type
MasterCard
CardNo
CvNumber
Submit
Reset
© All rights reserved | Design by Online Shopping
Type here to search
ENG
05:18 PM
11-Nov-2022


Online Shopping
127.0.0.1:5000/fullInfo?pid=67789
Product Image
ProductName
good for health
Price
900
Usage
dahioj
Select Qty
2
Card Type
MasterCard
CardNo
236348569678457
CvNumber
123
Submit
Reset
© All rights reserved | Design by Online Shopping
Type here to search
ENG
05:18 PM
11-Nov-2022

GITHUB & PROJECT DEMO LINK

- **GITHUB LINK :**

  https://github.com/IBM-EPBL/IBM-Project-15181-1659594640

- **PROJECT DEMO LINK**

  https://youtu.be/oR5XHNOpKrw