

# SMART FASHION RECOMMENDER APPLICATION

Mr. S. Karan<sup>1</sup> , Mr. E. Pradeesh<sup>2</sup> , Mr. R. Sathyasadhan<sup>3</sup> , Ms. A. Deepika<sup>4</sup> , Ms .R. Devadharshini<sup>5</sup>

Department of computer science and engineering, N. S. N college of engineering and technology, Karur, Tamil Nadu. iamkaranhari@gmail.com, pradeesh95141791@gmail.com, sathyasudhan2000@gmail.com, deepiannadurai2@gmail.com, devadharshini8765@gmail.com.

---

## 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 assistant.

## **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 SURVEY**

### **2.1 EXISTING PROBLEM:**

In existing system only simple web application and their rating has been implemented in existing system, An ecommerce product recommendation engine is a

piece of technology that displays recommended products to shoppers throughout your store. It uses machine learning to get smarter and show increasingly relevant products to shoppers based on their interests and previous browsing behavior.

In existing model is content based filtering scheme has been employed in existing model The content-based filtering method analyzes customer data on the likes and dislikes of each user (cookies allow tracking over multiple visits), then makes recommendations based on the browsing history of that user. The idea behind content-based filtering is that if you enjoy a certain item, you'll likely also enjoy a similar item. An example of a content-based filtering system would be if you were listening to Pandora and consistently 'liked' downtempo jazz music.

The collaborative-filtering method incorporates data from users who have purchased similar products, then combines that information to make decisions about recommendations. The advantage to this filtering method is that it is capable of making complex recommendations on items such as music or movies without having to 'understand' what the item is. This method of filtering operates under the assumption that users will prefer recommendations that are based on purchases they made in the past. Here's an example: If customer A likes a specific line of products that customer B also likes (assuming they have similar interests), then collaborate-filtering would assume that customer A would like other products that customer B purchased and vice versa.

A hybrid method combines the content-based and collaborative-based methods to incorporate group decisions but focuses the output based on the attributes of a specific visitor. An example of a hybrid filtering system would be how Spotify curates its personalized 'Discover Weekly' playlists. If you've ever listened to a personalized Spotify playlist, it's shocking how accurately they're able to recommend songs based on what you like. The secret behind how they pull this off is through a complex hybrid filtering system that aggregates data on your listening habits as well as similar users' listening habits, to create a playlist of unique songs that align with your personal taste.

**2.1.1 TITLE:** Smart Chatbot System for E-Commerce Assistance based on AIML, 2020.**AUTHOR:** Arif Nursetyo<sup>1</sup>

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<sup>1</sup>

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<sup>1</sup>

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<sup>1</sup>

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 Assistance 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.

### **I am (USER)**

Buying Products with the help of bot system .

### **I am Trying To**

Users can ask queries to the bot in this application instead of buying a damaged product or unnecessary things.

**But**

This might be mistake in shopping application, such as a lack of quality control, poor customer service etc.

**Because**

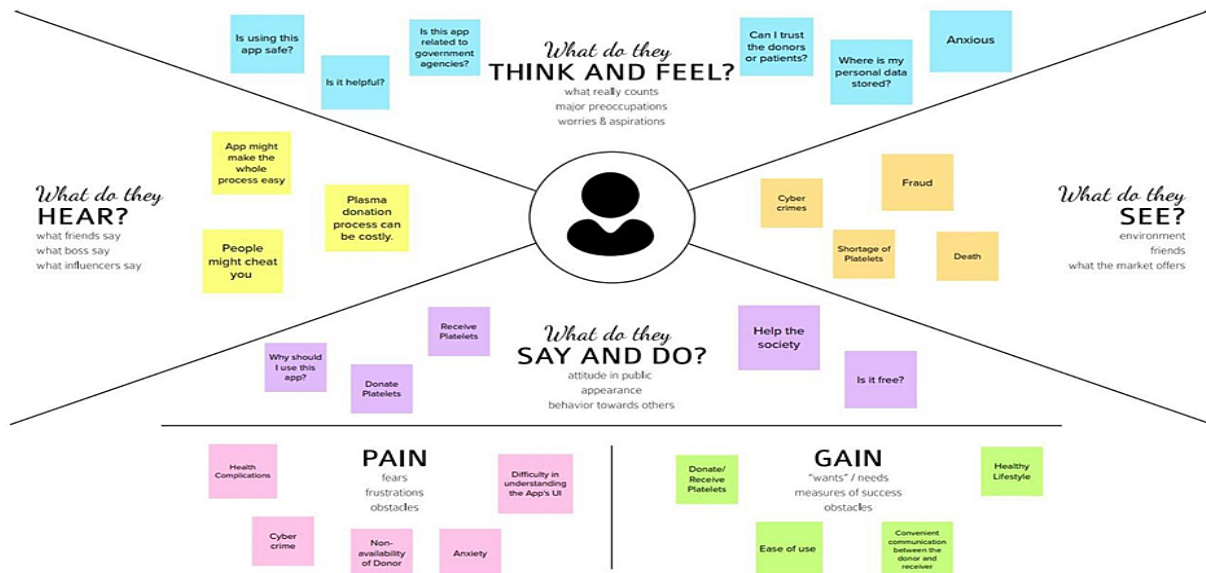
It is inflexible, and there is not a communicate system in any shopping website to order the products.

**Which makes me feel?**

The chatbot system may help to alleviate these issues by automating the assessment of buying products.

### 3.IDEATION & PROPOSED SOLUTION

#### 3.1 EMPATHY MAP CANVAS:





## 3.2 IDEATION & BRAINSTORMING :

2

### Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

#### TIP

You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing

Karan S



Pradeesh E



Sethya sudhan R



Devedhansini R



Deepika A



3

### Group Ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

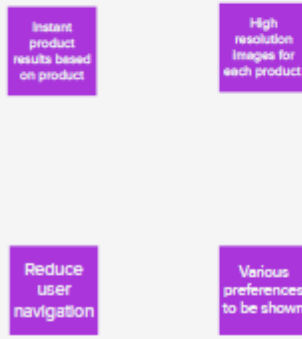
#### TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Group 1



Group 2

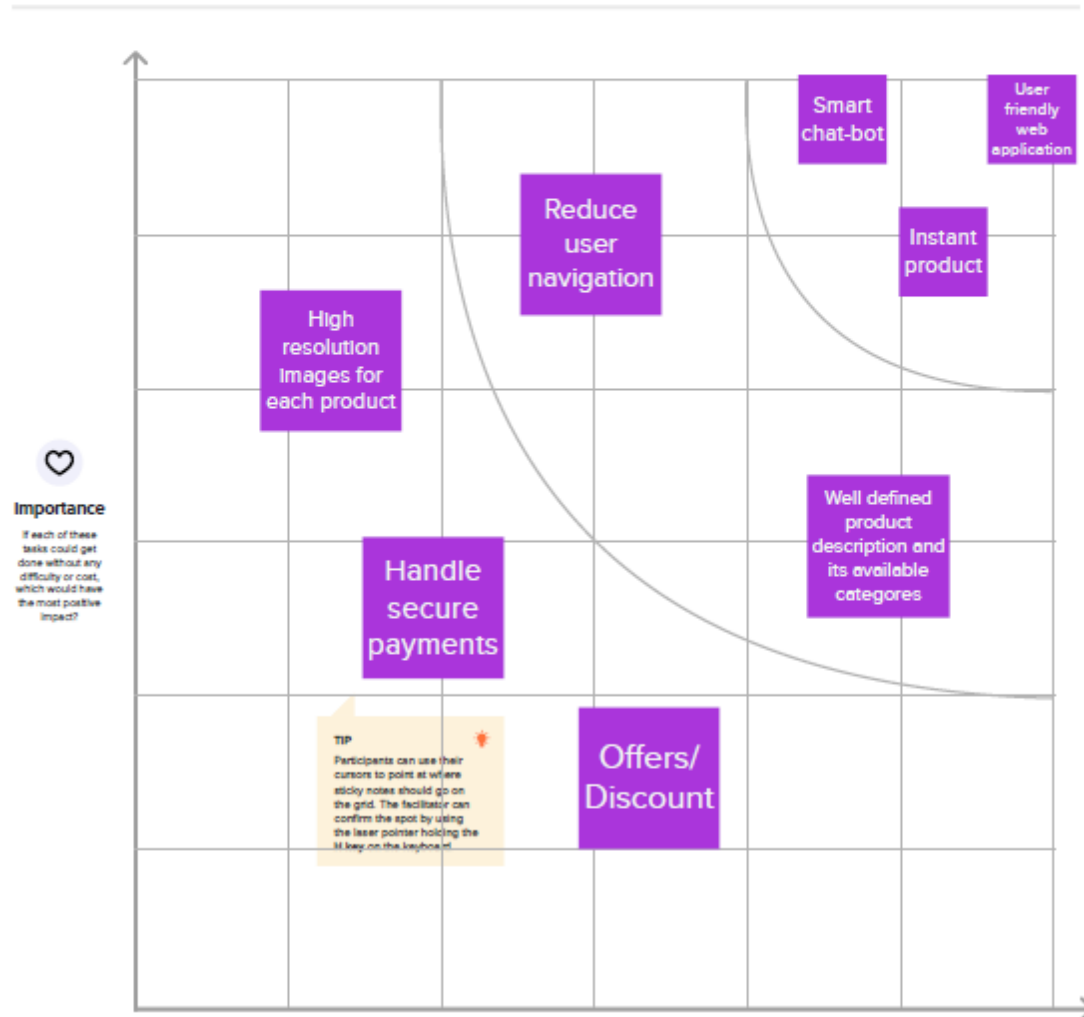


4

**Prioritize**

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

⌚ 20 minutes

**3.3 PROPOSED SOLUTION:**

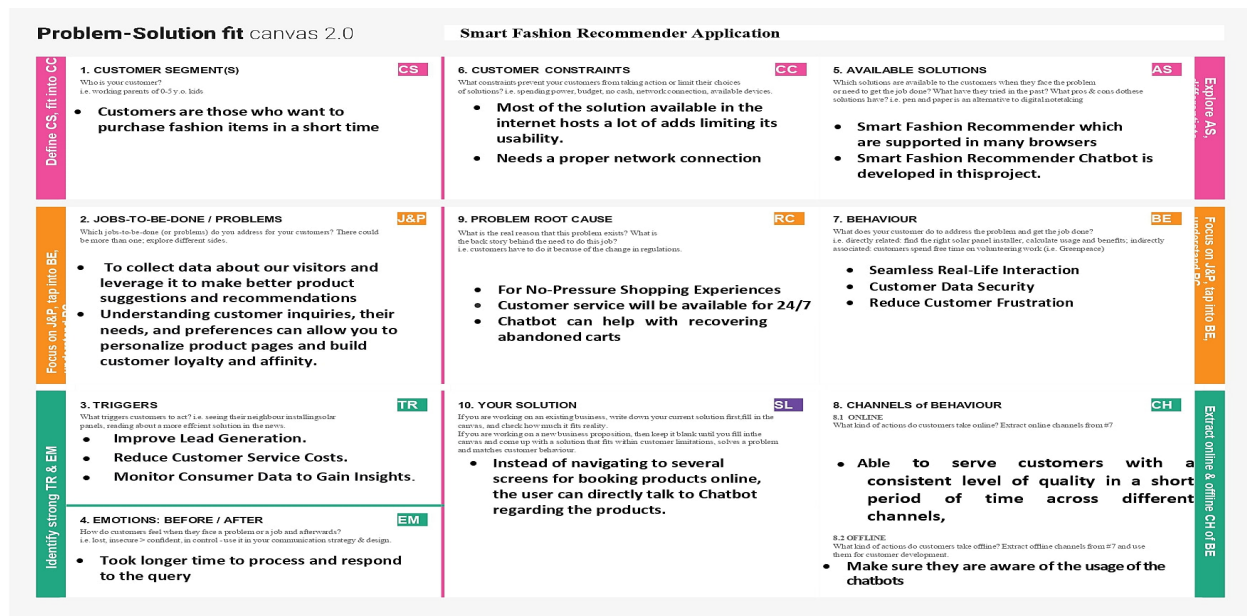
The proposed system is developed after a detailed study about the requirements requested by the user. Proposed system is a computerized one, where all the limitations of manual system are compensated. Product details of online shopping system have simplified the working information and make a user friendly 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. The suggested system was created following a thorough examination of the user's needs. The suggested approach is automated, making up for all the drawbacks of the manual technique. Online shopping system's product details have streamlined the working information and created a user-friendly environment where the user is given a lot of flexibility to manage successfully. The merchant benefits from quicker production of desired reports and improved outcomes. A chatbot system that can react fast and effectively also helps to manage and address client needs.

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	he 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
2.	Idea / Solution description	The bot system is applied to the shopping website might be useful to users and admin to improve the efficiency of website and products.
3.	Novelty / Uniqueness	Providing Chatbot system helps to know customers need
4.	Social Impact / Customer Satisfaction	Recommending products based on users wish makes customer more satisfied and handy.
5.	Business Model (Revenue Model)	-
6.	Scalability of the Solution	Capable of resolve the customers queries

### 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

##### 1. 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.

## **2. Add Products:**

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

## **3. 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.

## **4. Post Questions:**

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

## **5. 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:**

### **1. 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.

### **2. 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.

### **3. 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.

#### **4. Security:**

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

#### **5. 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 requestsubmittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 secondsto appear on the screen.

#### **6. Reliability :**

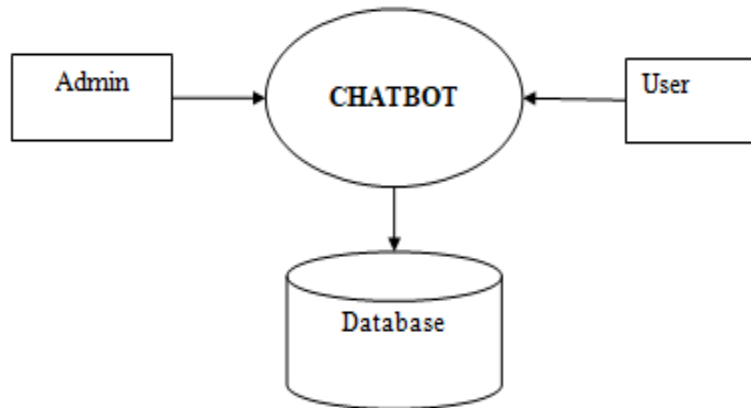
The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week. 24 hours a day.

## **5. PROJECT DESIGN**

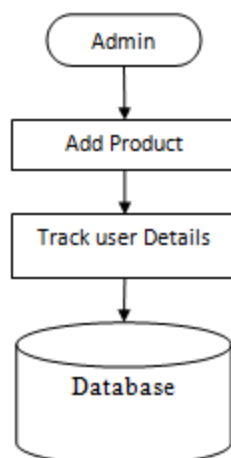
### **5.1 DATA FLOW DIAGRAMS :**

#### **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.

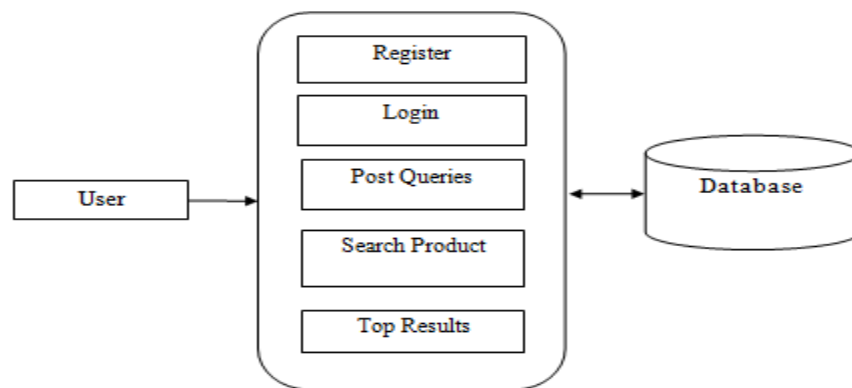
**I****LEVEL 1**

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.



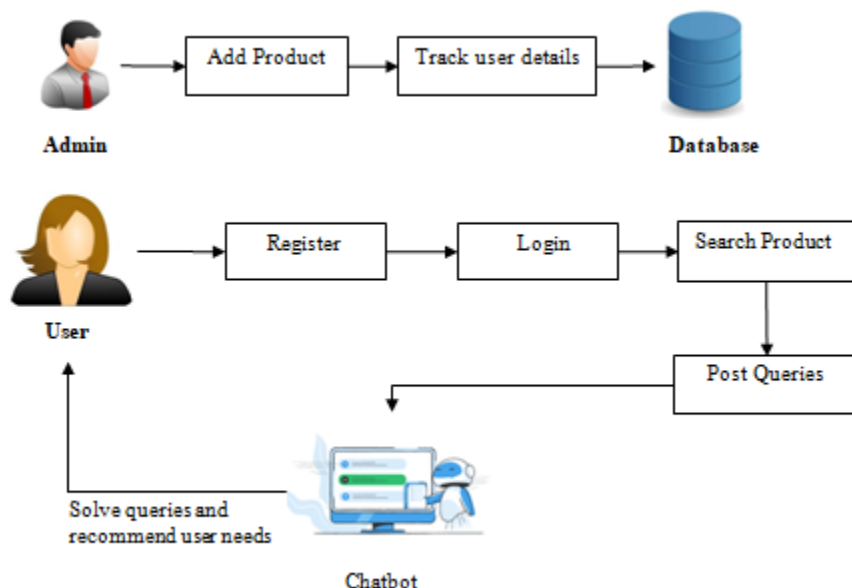
## 5.2 SOLUTION & TECHNICAL ARCHITECTURE :

### 5.2.1 SOLUTION 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).



## 5.2.2 TECHNICAL ARCHITECTURE :



## COMPONENTS & TECHNOLOGIES:

S.No	Component	Description	Technology
1.	User Interface	Web UI	HTML, CSS, JavaScript
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	Chatbot
4.	Application Logic-3	Logic for a process in the application	-
5.	Database	Data Type, Configurations etc.	MySQL
6.	Cloud Database	Database Service on Cloud	Local host
7.	File Storage	File storage requirements	Local host
8.	External API-1	Purpose of External API used in the application	-
9.	External API-2	Purpose of External API used in the application	-
10.	Machine Learning Model	Purpose of Machine Learning Model	Helps the bot to solve the query and recommend

11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local Server Configuration
-----	---------------------------------	---	----------------------------

### APPLICATION CHARACTERISTICS:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	PYCHARM
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	-
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Able to respond the changes in an application
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	The system must always be functional
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Takes no longer time to response

### 5.3 USER STORIES :

A user story is the smallest unit of work in an agile framework. It's an end goal, not a feature, expressed from the software user's perspective. 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. Note that "customers" don't have to be external end users in the traditional sense, they can also be internal customers or colleagues within your organization who depend on your team. User stories are a few

sentences in simple language that outline the desired outcome. They don't go into detail. Requirements are added later, once agreed upon by the team. Stories fit neatly into agile frameworks like scrum and kanban. In scrum, user stories are added to sprints and “burned down” over the duration of the sprint. Kanban teams pull user stories into their backlog and run them through their workflow. It's this work on user stories that help scrum teams get better at estimation and sprint planning, leading to more accurate forecasting and greater agility. Thanks to stories, kanban teams learn how to manage work-in-progress (WIP) and can further refine their workflows. User stories are also the building blocks of larger agile frameworks like epics and initiatives. Epics are large work items broken down into a set of stories, and multiple epics comprise an initiative. These larger structures ensure that the day-to-day work of the development team (on stories) contributes to the organizational goals built into epics and initiatives.

## 6. PROJECT PLANNING & SCHEDULING

### 6.1 SPRINT PLANNING & ESTIMATION :

Sprint	Functional Requirement (Epic)	User Story Number	User Story/ Task	Story Points	Priority
Sprint-1	User Panel	USN-1	The user will login into the website and go through the products available on the website	20	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.	20	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.	20	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	20	High

## 6.2 SPRINT DELIVERY SCHEDULE :

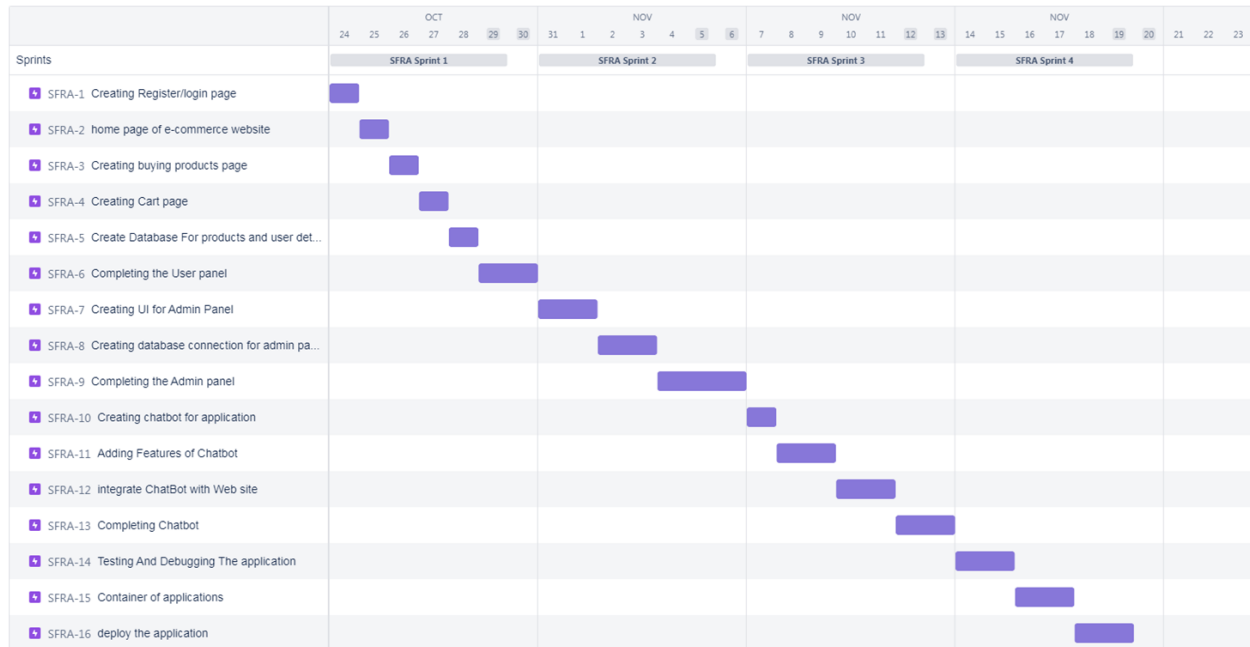
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date(Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022		29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12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

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

### Burndown Chart:



## 7. CODING & SOLUTIONING

### 7.1 FEATURE 1 :

#### DB2

DB2 is a database product from IBM. It is a Relational Database Management System (RDBMS). DB2 is designed to store, analyze and retrieve the data efficiently. DB2 product is extended with the support of Object-Oriented features and non-relational structures with XML.

## 7.2 FEATURE 2 :

### **Flask**

Flask is a web framework that provides libraries to build lightweight web applications in python. It is developed by Armin Ronacher who leads an international group of python enthusiasts (POCCO). It is based on WSGI toolkit and jinja2 template engine. Flask is considered as a micro framework. Flask Tutorial provides the basic and advanced concepts of the Python Flask framework. Our Flask tutorial is designed for beginners and professionals. Flask is a web framework that provides libraries to build lightweight web applications in python. It is developed by Armin Ronacher who leads an international group of python enthusiasts

(POCCO). Flask is a web framework, it's a Python module that lets you develop web applications easily. It's has a small and easy-to-extend core: it's a microframework that doesn't include an ORM (Object Relational Manager) or such features. It does have many cool features like url routing, template engine. It is a WSGI web app framework.

### **Docker**

Docker is a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called container that have everything the software needs to run

including libraries, system tools, code, and runtime. Using Docker, you can quickly deploy and scale applications into any environment and know your code will run. Docker works by providing a standard way to run your code. Docker is an operating system for containers.

## 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:

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

S.NO	Scenario	Input	Excepted output	Actual output
1	User login	User name and password	Login	Login success.

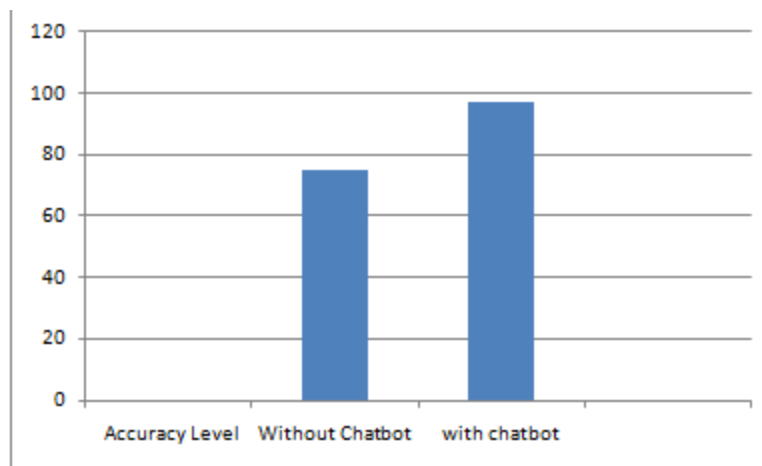
2	Search Product	Show product list	Purchase successfully	User details are stored in a database.
3	Post Queries	Ask questions to chatbot	Get result from chatbot	Details are stored in a database.

## 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



**10.1 ADVANTAGE:**

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

**10.2 DISADVANTAGE:**

1. Difficult to decision making for an administrator to improve
2. Immediate response to the queries is difficult.
3. More stationary use so they are expensive.
4. Manual system is takes more time.
5. 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

### 13.1 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)

```

```

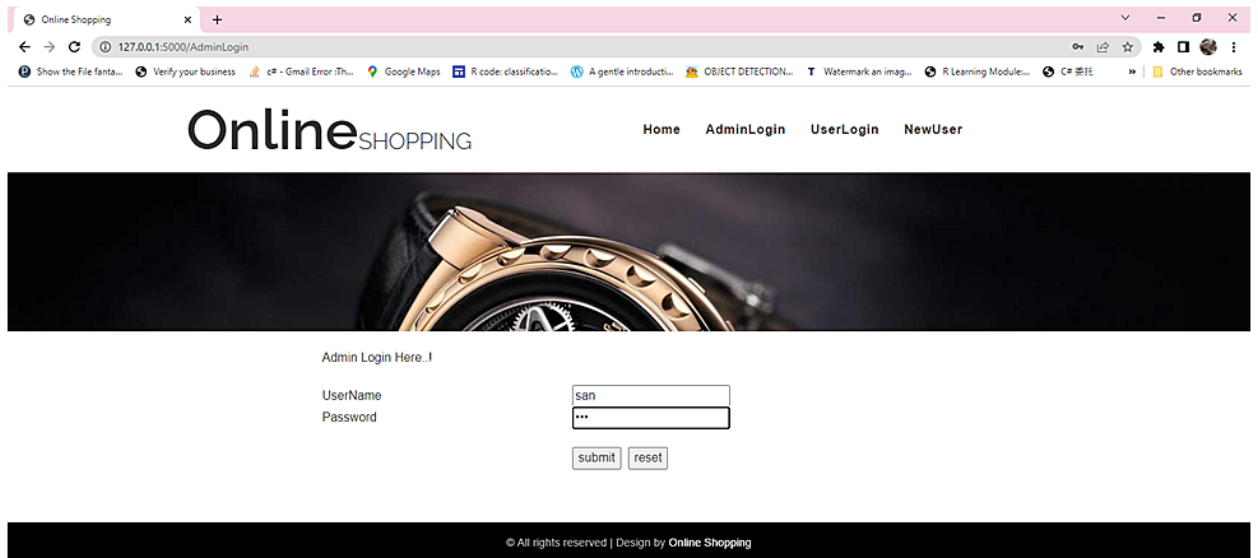
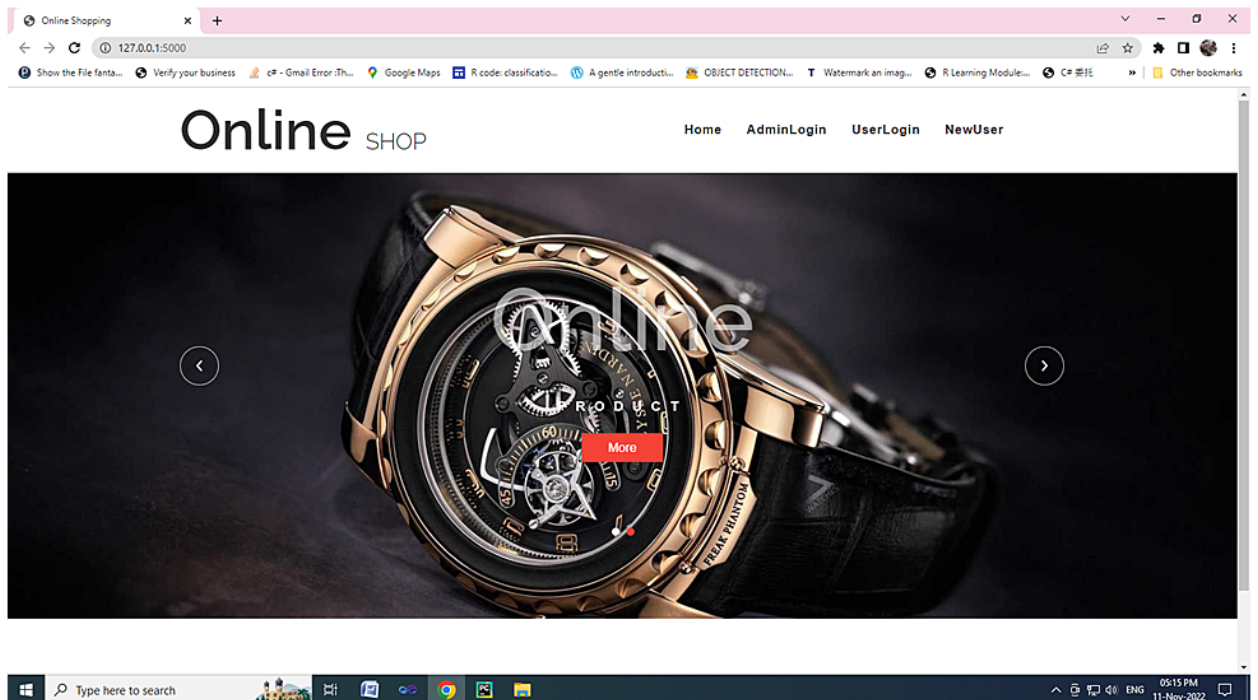
@app.route("/UserHome")
def UserHome():
    uname = session['uname']
    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM regtb where  UserName= '" + uname + "' "
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('booktb1', 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

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

Online Shopping

127.0.0.1:5000/NewProduct



### New Product Registration

ProductId	<input type="text"/>
Gender	<input type="text" value="--Select--"/>
Category	<input type="text" value="--Select--"/>
SubCategory	<input type="text" value="--Select--"/>
ProductType	<input type="text" value="--Select--"/>
Colour	<input type="text"/>
Usage	<input type="text"/>
ProductTitle	<input type="text"/>
Price	<input type="text"/>
Image	<input type="button" value="Choose file"/> No file chosen
<input type="button" value="Submit"/> <input type="button" value="Reset"/>	

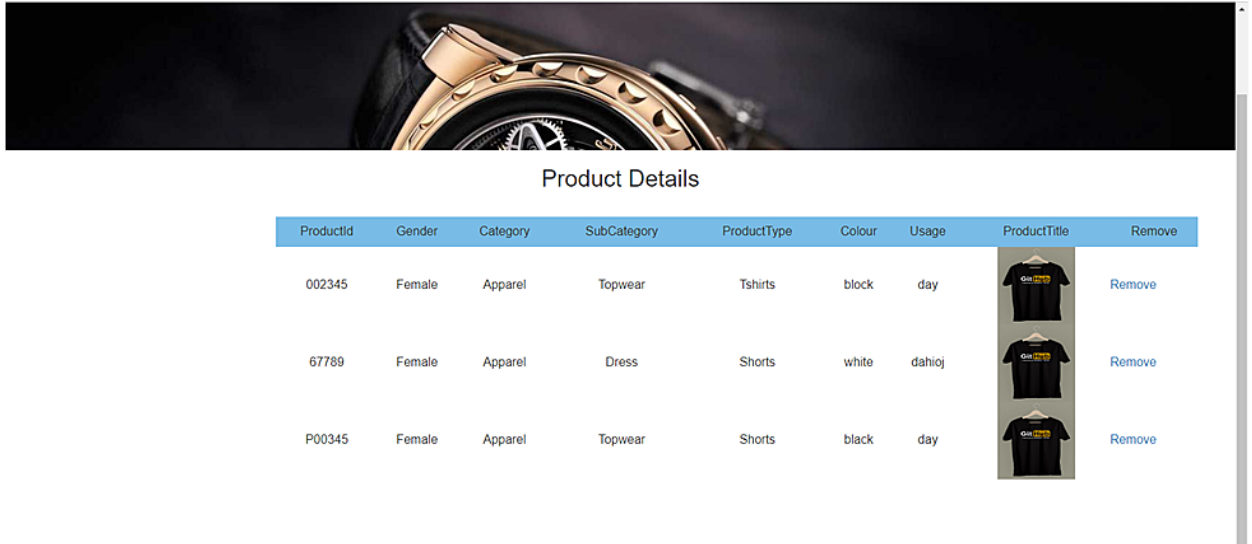
© All rights reserved | Design by Online Shopping

Type here to search




Online Shopping

127.0.0.1:5000/ProductInfo

Show the File fanta... Verify your business Gmail Error .Th... Google Maps R code: classificatio... A gentle introductio... OBJECT DETECTION... Watermark an imag... R Learning Module... C# 委托 Other bookmarks



Product Details

ProductId	Gender	Category	SubCategory	ProductType	Colour	Usage	ProductTitle	Remove
002345	Female	Apparel	Topwear	Tshirts	black	day		<a href="#">Remove</a>
67789	Female	Apparel	Dress	Shorts	white	dahioj		<a href="#">Remove</a>
P00345	Female	Apparel	Topwear	Shorts	black	day		<a href="#">Remove</a>

© All rights reserved | Design by Online Shopping

Type here to search

05:16 PM 11-Nov-2022

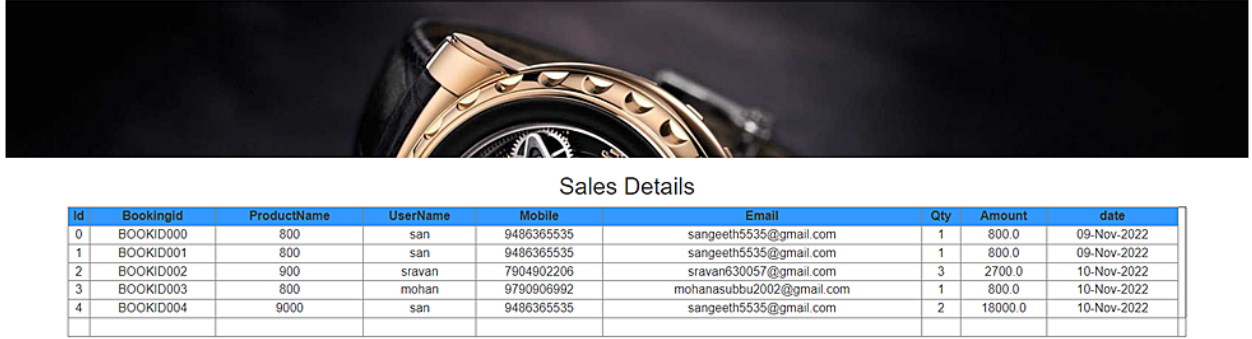
Online

127.0.0.1:5000/ASalesInfo

Show the File fanta... Verify your business Gmail Error .Th... Google Maps R code: classificatio... A gentle introductio... OBJECT DETECTION... Watermark an imag... R Learning Module... C# 委托 Other bookmarks

OnlineSHOPPING

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

05:16 PM 11-Nov-2022

Online Shopping x +

127.0.0.1:5000/NewUser

Show the File fanta... Verify your business C# - Gmail Error.Th... Google Maps R code: classificatio... A gentle introducti... OBJECT DETECTION... Watermark an imag... R Learning Module... C# 委托 Other bookmarks



### New User Registration

Name

Gender ☐ Male ☐ Female

Age

Email Id

Phone Number

Address

User Name

Passwrod

© All rights reserved | Design by Online Shopping

Type here to search

05:16 PM 11-Nov-2022


Online Shopping x +

127.0.0.1:5000/UserLogin

Show the File fanta... Verify your business C# - Gmail Error.Th... Google Maps R code: classificatio... A gentle introducti... OBJECT DETECTION... Watermark an imag... R Learning Module... C# 委托 Other bookmarks

# OnlineSHOPPING

Home AdminLogin UserLogin NewUser



### User Login Here..!

UserName

Password

© All rights reserved | Design by Online Shopping

Type here to search


05:16 PM 11-Nov-2022

Online Shopping

127.0.0.1:5000/userlogin

Home Search OrderInfo Logout

# OnlineSHOPPING



## 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




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®




Online Shopping

127.0.0.1:5000/fullinfo?pid=67789

Show the File fanta... Verify your business Gmail Error:Th... Google Maps R code: classificatio... A gentle introductio... OBJECT DETECTION... Watermark an imag... R Learning Module... C# 委托 Other bookmarks

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


05:18 PM 11-Nov-2022

Online Shopping

127.0.0.1:5000/fullinfo?pid=67789

Show the File fanta... Verify your business Gmail Error:Th... Google Maps R code: classificatio... A gentle introductio... OBJECT DETECTION... Watermark an imag... R Learning Module... C# 委托 Other bookmarks

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

05:18 PM 11-Nov-2022

### 13.1 GITHUB LINK:

<https://github.com/IBM-EPBL/IBM-Project-47912-1660803266>

### DEMO VIDEO LINK:













