SMART FASHION RECOMMENDER

CLOUD APPLICATION DEVELOPMENT

DOMAIN

TEAMID:PNT2022TMID11504

PROJECT REPORT

Submittedby

M.S.R.KOUSHIKHA (910619106026)

V. ABINAYA(910619106001)

M. DEEPIKA(910619106012)

B. GOKULA PRIYA(910619106017)

CR. INDHUJA(910619106023)

ELECTRONICS AND COMMUNICATION ENGINEERING

KLN COLLEGE OF ENGINEERING

POTTAPALAYAM

TEAM MENTOR:

DR.B.BUVANESWARI

INDUSTRYMENTOR:

- KRISHNA CHAITHANYA

EVALUATOR:

MS. L.MEENAKSHI

ANNA UNIVERSITY : CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report **"SMART FASHION RECOMMENDER APPLICATION"** is the bonafide work of

"**KOUSHIKHA M S R (910619106026),**

ABINAYA V (910619106001),

GOKULA PRIYA B (910619106017),

INDHUJA CR (910619106023),

DEEPIKA M (910619106012)",

Who carried out the project work under our supervision.

SIGNATURE	**SIGNATURE**
FACULTY MENTOR	**FACULTY EVALUATOR**
Name	Name
Designation	Designation
ELECTRONICS AND COMMUNICATION ENGINEERING	ELECTRONICS AND COMMUNICATION ENGINEERING
K.L.N COLLEGE OF ENGINEERING	K.L.N COLLEGE OF ENGINEERING

SIGNATURE
DR.V.KEJALAKSHMI
HEAD OF THE DEPARTMENT
ELECTRONICS AND
COMMUNICATION ENGINEERING
K.L.N COLLEGE OF ENGINEERING

ABSTRACT

Fashion applications have seen tremendous growth and are now one of the most used programs inthe e-commerce field. The needs of people are continuously evolving, creating room for innovation among the applications. One of the tedious processes and presumably the mainactivities is choosing what you want to wear. Having an AI program that understands thealgorithm of a specific application can be of great aid.

We are implementing such a chat bot, which is fed with the knowledge of the application's algorithm and helps the user completely from finding their needs to processing the payment and initiating delivery. It works as an advanced filter search that can bring the user what they want with the help of pictorial and named representation.

The application also has two main user interfaces - the user and the admin. The users can interact with the chat bot, search for products, order them from the manufacturer or distributor, make payment transactions, track the delivery, and so on. The admin interface enables the user to upload products, find how many products have been bought, supervise the stock availability and interact with the buyer regarding the product as reviews.

The rapid progress of computer vision, cloud computing and artificial intelligence combined with the current growing urge for online shopping systems opened an excellent opportunity for the fashion industry. As a result, many studies worldwide are dedicated to modern fashionrelated applications such as virtual try-on and fashion synthesis.

Project Report Format

CHAPTER-1

1. INTRODUCTION

1.1 Project Overview :

Instead of searching for products in the search bar and navigating to individual products to find required preferences, this project leverages the use of chatbots to gather all required preferences and recommend products to the user. The solution is implemented in such a way as to improve the interactivity between customers and applications. The chat-bot sends messages periodically to notify offers and preferences. Forsecurity concerns, this application uses a token to authenticate and authorize users securely. The token has encoded user id and role. Based on the encoded information, access to the resources is restricted to specific users. User is divided based on their roles. The roles comprises of admin and user. Admin is provided access with adding new products, update a product track user details. Users of the application get periodic recommendation via chat-bot based on their preference and search. The main purpose of this application is to make process of searching, filtering and ordering of products simple and quickly that enhances the overall user experience. The chatbot is trained by providing different categories of product information and its related details.

1.2 Purpose of the Project:

Fashion applications have seen tremendous growth and are now one of the most used programs in the e-commerce field. The needs of people are continuously evolving, creating room for innovation among the applications. One of the tedious processes and presumably the main activities is choosing what youwant to wear. Having an AI program that understands the algorithm of a specific application can be of great aid. We are implementing such a chatbot, which is fed with the knowledge of the application's

algorithm and helps the user completely from finding their needs to processing the payment and initiating delivery. It works as an advanced filter search that can bring the user what they want with the help of pictorial and named representation. The application also has two main user interfaces - the user and the admin. The users can interact with the chatbot, search for products, order them from the manufacturer or distributor, make payment transactions, track the delivery, and so on. The admin interface enables the user to upload products, find how many products have been bought, supervise the stock availability and interact with the buyer regarding the product as reviews. In E-commerce websites, users need to search for products and navigate across screens to view the product, add them to the cart, and order products.

The smart fashion recommender application leverages the use of a chat bot to interact with the users, gather information about their preferences, and recommend suitable products to the users. This application has two predefined roles assigned to the users. The roles are customer and admin. The application demands redirection of the user to the appropriate dashboard based on the assigned role. Admin should be able to track the number of different products and admin should be assigned the responsibility to create products with appropriate categories. The user should be able to mention their preferences using interacting with chat bots. The user must receive a notification on order confirmation/failure. The chat bot must gather feedback from the user at the end of order confirmation. The main objective of this application is to provide better interactivity with the user and to reduce navigating pages to find appropriate products.

CHAPTER-2

LITERATURE SURVEY

2.1 Existing Problem:

On E-commerce websites, users need to search for products and navigate across screens to view the product, add them to the cart, and order products. The smart fashion recommender application leverages the use of a chatbot to interact with the users, gather information about their preferences, and recommend suitable products to the users. This application has two predefined roles assigned to the users. The roles are customer and admin. The application demands redirection of the user to the appropriate dashboard based on the assigned role. Admin should be able to track the number of different products and admin should be assigned the responsibility of creating products with appropriate categories. The user should be able to mention their preferences using interacting with chatbots. The user must receive a notification on order confirmation/failure. The chatbot must gather feedback from the user at the end of order confirmation. The main objective of this application is to provide better interactivity with the user and to reduce navigating pages to find appropriate products.

2.2 References:

i) Paper Title: A COMPREHENSIVE REVIEW ON ONLINE FASHION RECOMMENDATION

Publication: December 2020

Author name: Samit Chakraborty

Methodology: Auto Regression (AR) and Linear Regression Model. Auto Regression (AR) and Linear Regression Model Using photos pulled from social

media, online fashion magazines, well-known
e-commerce sites, fashion site blogs, and discussion forums, (Ngai et al., 2018) employed the autoregressive (AR) model (or ARMAX) to forecast style or trends. Due to the data patterns being obtained over a set amount of time, it makes precise trend prediction possible (Fung, Wong, Ho, & Mignolet, 2003). These forecasting models' detailed theoretical contents were demonstrated in two separate studies by Liu et al. (2013) and Nenni, Giustiniano, & Pirolo (2013), which also included severalgeneral approach forms. Because they were straightforward, quick, wellinformed, and simple to understand, statistical techniques including auto-regression, exponential smoothing, ARIMA, and SARIMA were frequently employed to assess the sales of clothing. A technique for forecasting retail products was proposed by Demerit (2018). weekly using linear regression models in multi-processing groups with both positive and negative commodities. The introduction of dynamic pricing models to support markdown choices in multi-item group predictions has since followed. In order to prevent overfitting, grouping items in predictive models can be seen as a way of variable selection. They then exhibited regression results from multiple-item groupings on the real-world dataset provided by a clothing company in addition to the findings from the single-item regression model. They also revealed the results of markdown optimization for single items and groups of multiple items that serve as the foundation for multi-item forecasting models. The results suggested that regression models provide better estimates in many categories than the one-item model.

ii) Paper Title: Fashion Recommendation Systems

Author name: Samit Chakraborty , Md. Saiful Hoque, Naimur Rahman Jeem, Manik Chandra Biswas, Deepayan Bardhan and Edger Lobaton.

Methodology: Fast fashion has grown significantly over the past few years, which has had a significant impact on the textile and fashion industries.

An effective recommendation system is needed in e-commerce platforms where there are many options available to sort, order, and effectively communicate to user's pertinent product content or information.

Fast fashion retailers have paid a lot of attention to image-based fashion recommendation systems (FRSs), which offer customers a customised purchasing experience.

There aren't many academic studies on this subject, despite its enormous potential. The studies that are now accessible do not conduct a thorough analysis of fashion recommendation systems and the accompanying filtering methods.

This review also looks at many potential models that might be used to create future fashion suggestion systems.

iii) Paper Title: A Review on Clothes Matching and Recommendation System Based on User Attributes.

Author name: Atharv Pandit , Kunal Goel , Manav Jain , Neha Katre

Methodology: It's crucial to dress adequately while venturing out into the real world. The confidence of the individual is raised and a very positive impression is made when they are dressed appropriately in clothing that exhibits some degree of style and is worn in a way that complies with societal norms. The goal of the study is to make it easier for customers to locate the best-fitting outfits by taking into account fine elements like style, patterns, colors, and textures, as well as user characteristics like age, skin tone, and favorite colors. It seeks to assist the user in organizing their closet and making stylish clothing selections.

It makes an effort to assist the user in dressing appropriately for the occasion and in finding clothing that compliments their personal style.

In order to create a robust system that discovers the user's matching outfits and provides recommendations, an in-depth analysis of numerous systems that are built for various aspects is undertaken in this research. Systems created to propose clothing using various methodologies have been researched, with both their benefits and drawbacks highlighted.

It has also been investigated how to make clothing detecting systems user-friendly while accepting feedback from the user.

2.3 Problem statement:

Problem Statement Definition:

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love.
A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face.

Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

How to write a problem statement?

A good problem statement can be created by identifying and answering several questions related to the problem,

1. Identify the Problem
2. Begin were statement with where ideal situation
3. Describe current gaps
4. State the consequence of the problem
5. Propose addressing the problem.

KEY ELEMENTS OF PROBLEM STATEMENT:

- Reality - It will also describe when and where the problem was identified.
- Consequences - Problem statements should identify the consequences of the problem.
- Proposal - The proposal section of a problem statement may contain several possible solutions to the problem

CHAPTER -3

3.IDEATION & PROPOSED SOLUTION

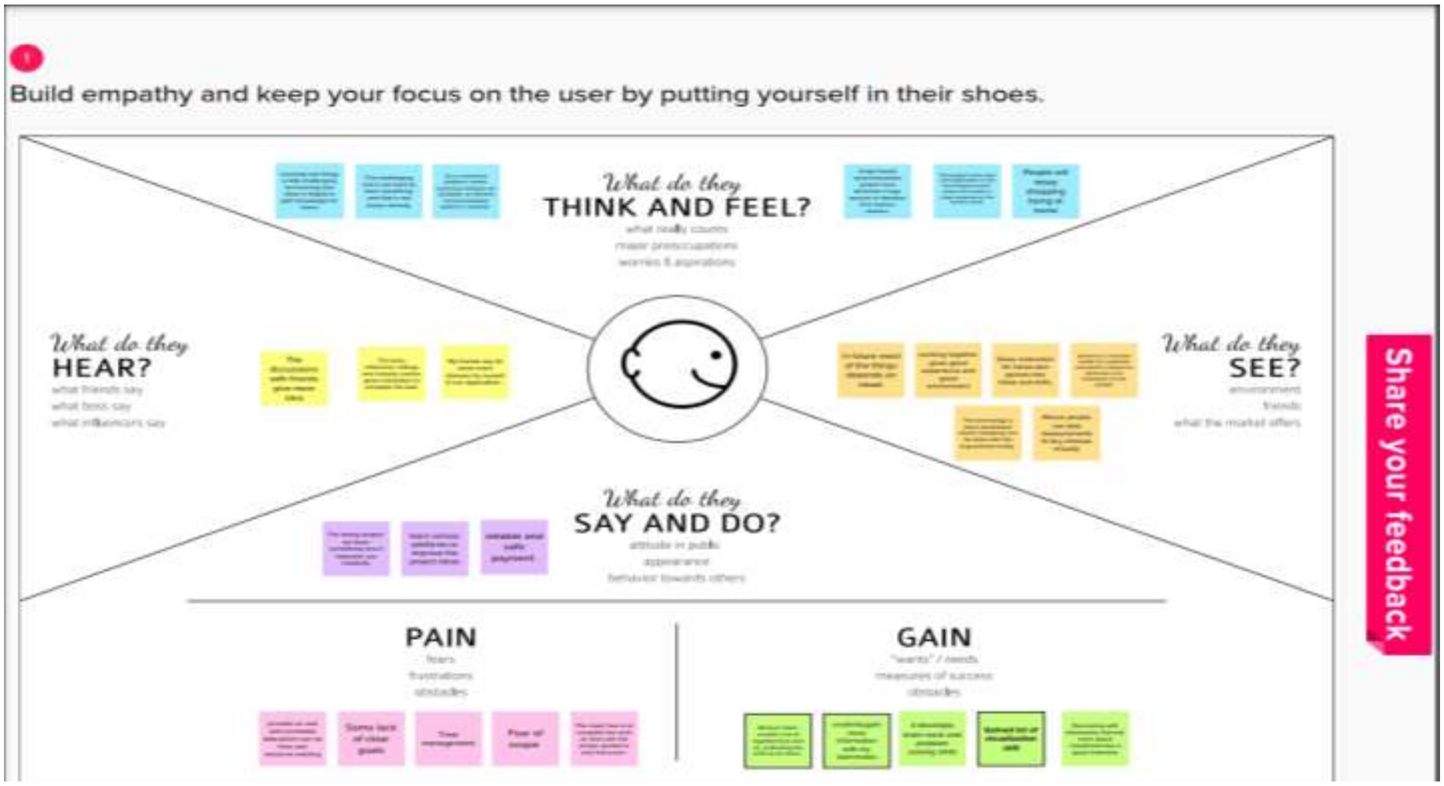
3.1 Empathy Map:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it.

The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

Benefits of empathy map:

- It is simple and quick to complete an empathy map. There is no need to spend a lot of time or money on the undertaking. All you require is a team and basic tools. As an alternative, you can use expensive but cutting-edge software, even on a tight budget.
- The visualizing technique allows for widespread participation. You invite managers, designers, tech and non-tech experts, and product owners. The more opinions and ideas you have, the better.
- It aids creators in expanding their thinking and understanding that their ideal product vision may vary from the viewpoint of the customers.
- These resources are all-purpose and can be used in any field, sector, or type of business.
- Companies that engage in empathy mapping outperform those that do not.


Build empathy and keep your focus on the user by putting yourself in their shoes.
What do they THINK AND FEEL?
What do they HEAR?
What do they SEE?
What do they SAY AND DO?
PAIN
GAIN
Share your feedback
Figure 3.1 Empathy map

3.2 Brainstorming- Idea prioritization

Brainstorming provides a free and open environment that encourages everyone within a team toparticipate in the creative thinking process that leads to problem solving.

Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creativesolutions.

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

There are some brainstorming approaches:

STEP 1: Prepare the group step:

How much background knowledge or planning is required for your team to generate problem-solving ideas? It's vital to keep in mind that preparation is necessary, but toomuch of it might hinder or even ruin a brainstorming session.

Once everyone is present, designate one person to take notes on the suggestions madeduring the session.

It's difficult to record and contribute at the same time, thus this individual shouldn't necessarily be the team manager.

Use a computer with a data projector, flip charts, or whiteboards to post notes whereeveryone can see them.

Step 2: Present the Problem

Lay up any requirements that must be met as well as the problem that you are trying to solve. Make it crystal clear that the goal of the meeting is to create as many ideas as you can.

Allow everyone plenty of quiet time to come up with as many original ideas as they can at the beginning of the session.

Ask them to share or present their thoughts after that, while ensuring that everyone has an equal chance to contribute.

Step 3: Guide the Discussion

Start a group conversation once everyone has given their views a chance to be heard in order to develop existing ideas and generate new ones.

One of the most beneficial features of group brainstorming is building on others' ideas.

Encourage everyone—even the most reserved individuals—to participate and develop ideas, and forbid anybody from critiquing others' ideas.

If you have any ideas, you should share them as the group facilitator, but otherwise focus your time and efforts on helping your team and facilitating the conversation.

Keep each conversation to itself and bring the group back on track if it wanders off.

Don't spend too much time thinking along the same lines. Make sure to come up with lots of different ideas and thoroughly examine each one.

Give a team member the freedom to pursue an idea alone if they need to "tune out" for it.

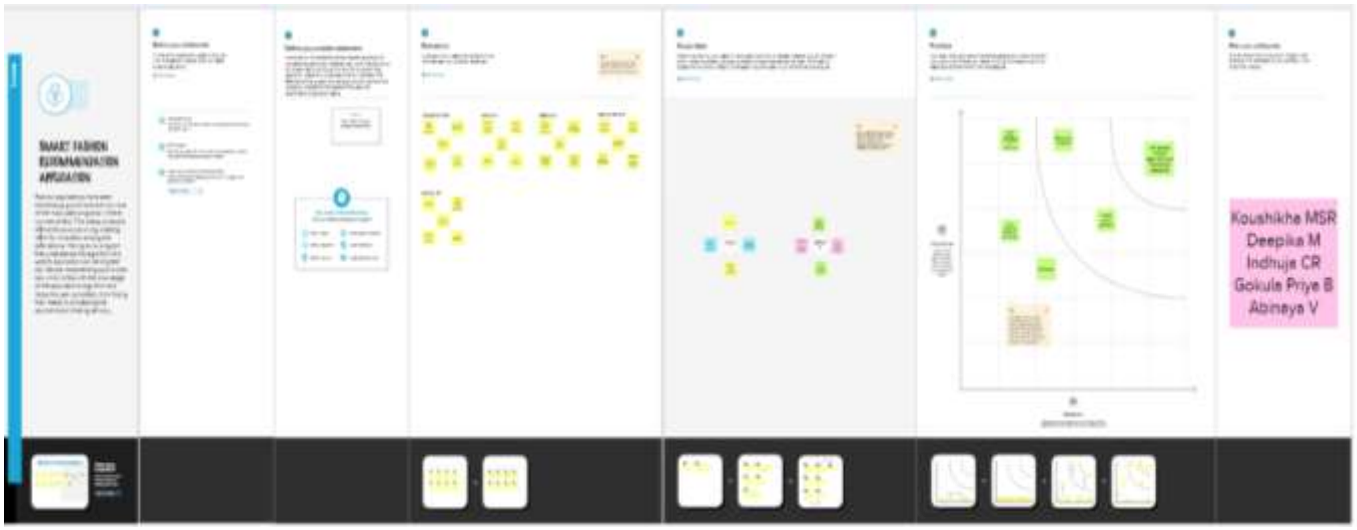


Figure 3.2 Brainstorming and polarization

3.3 Proposed Solution

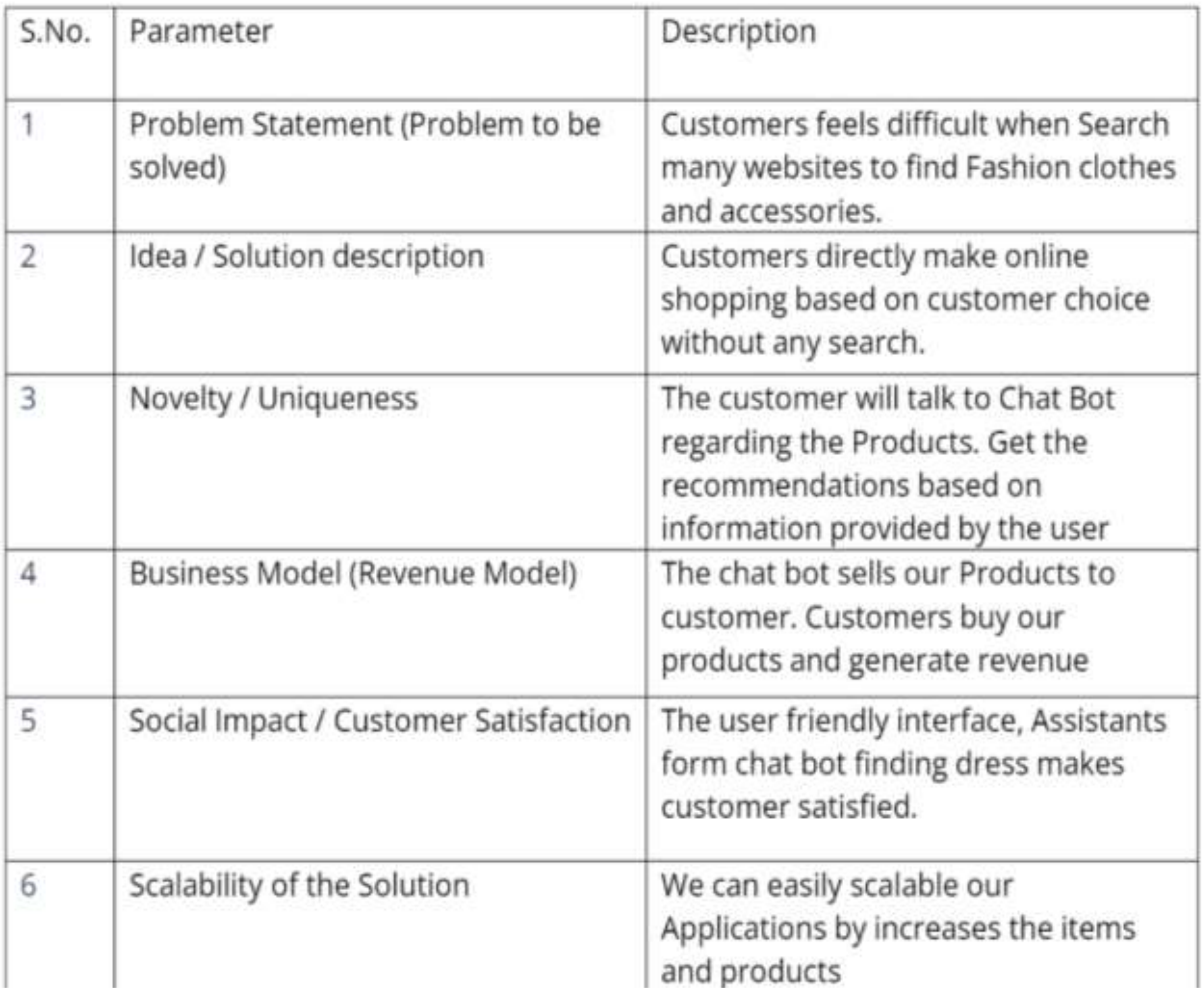
Definition:

Proposed Solution refers to the technical response that the Implementation agency will offer inresponse to the Project's requirements and objectives.

How to write a problem statement:

1. Describe how things should work.
2. Explain the problem and state why it matters.
3. Explain your problem's financial costs.
4. Back up your claims.
5. Propose a solution.
6. Explain the benefits of your proposed solution(s).
7. Conclude by summarizing the problem and solution.

The main goal of presenting a business proposal is to provide a solution to a problem faced by apotential buyer. This section should be as comprehensive as possible, and able to address all theneeds that we have pointed out in the first section.



S.No.	Parameter	Description
1	Problem Statement (Problem to be solved)	Customers feels difficult when Search many websites to find Fashion clothes and accessories.
2	Idea / Solution description	Customers directly make online shopping based on customer choice without any search.
3	Novelty / Uniqueness	The customer will talk to Chat Bot regarding the Products. Get the recommendations based on information provided by the user
4	Business Model (Revenue Model)	The chat bot sells our Products to customer. Customers buy our products and generate revenue
5	Social Impact / Customer Satisfaction	The user friendly interface, Assistants form chat bot finding dress makes customer satisfied.
6	Scalability of the Solution	We can easily scalable our Applications by increases the items and products

Table 3.3 proposed solution

3.4Problem Solution fit:

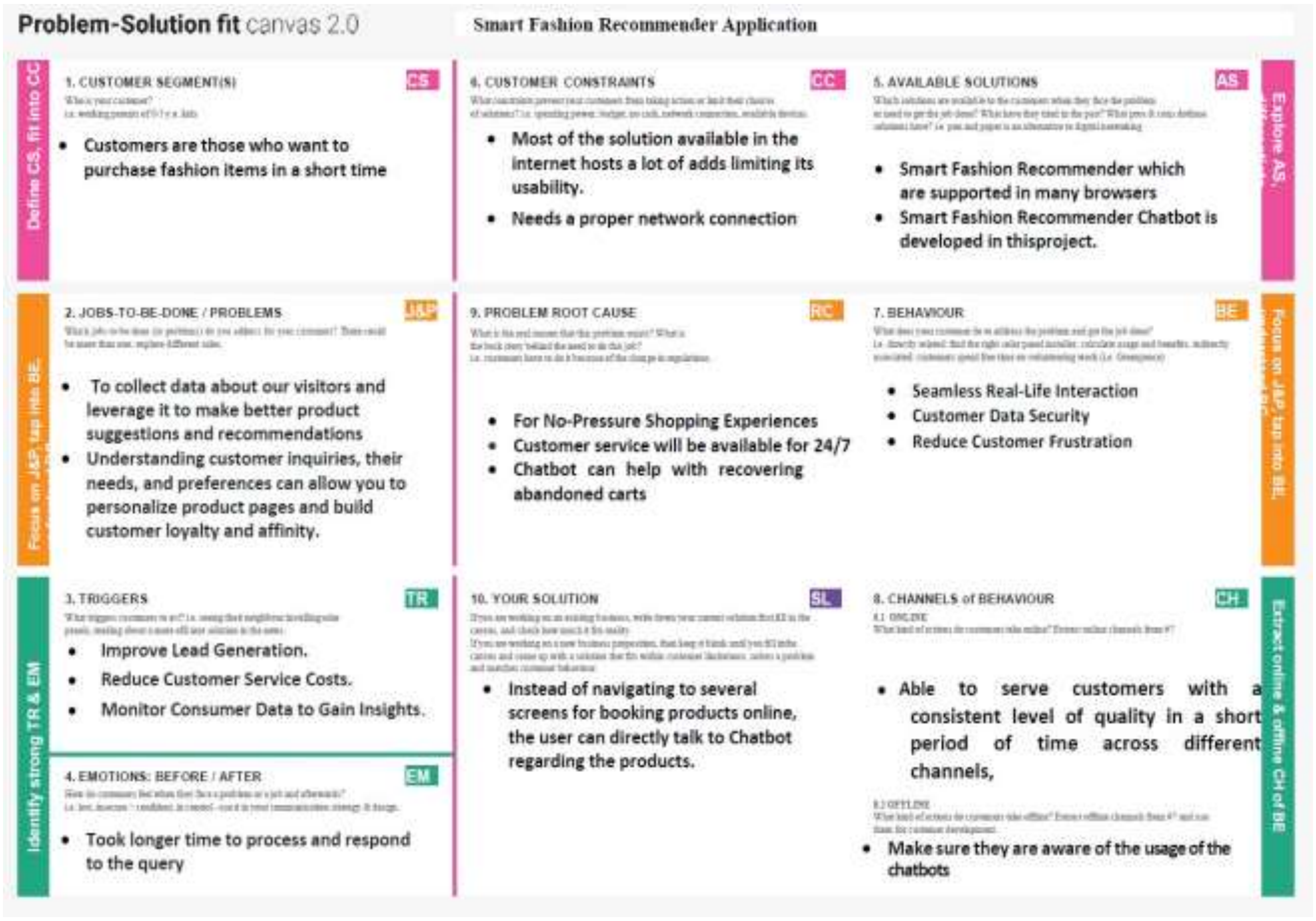
The Lean Startup, LUM (Lazy User Model), and User Experience design tenets serve as the foundation for the Problem-Solution Fit canvas. It aids in the identification of behavioural patterns by business innovators, marketers, and entrepreneurs.

• It serves as a template for identifying solutions with the best prospects of being adopted, cutting down on testing time, and getting a clearer picture of the situation as it stands.

My objective was to develop a tool that transforms a problem into a solution while considering client behavior and the surrounding circumstances.

With the help of this template, we will be able to thoroughly examine problem solving and take important information into account earlier.

• It increases our chances of finding a product-market fit and a problem-solution fit.



Problem-Solution fit canvas 2.0

Smart Fashion Recommender Application

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

What is your customer?
[illegible]

- Customers are those who want to purchase fashion items in a short time

6. CUSTOMER CONSTRAINTS CC

[illegible]

- Most of the solution available in the internet hosts a lot of adds limiting its usability.
- Needs a proper network connection

5. AVAILABLE SOLUTIONS AS

[illegible]

- Smart Fashion Recommender which are supported in many browsers
- Smart Fashion Recommender Chatbot is developed in thisproject.

Explore AS, differentiate

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

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

[illegible]

- To collect data about our visitors and leverage it to make better product suggestions and recommendations
- Understanding customer inquiries, their needs, and preferences can allow you to personalize product pages and build customer loyalty and affinity.

9. PROBLEM ROOT CAUSE RC

[illegible]

- For No-Pressure Shopping Experiences
- Customer service will be available for 24/7
- Chatbot can help with recovering abandoned carts

7. BEHAVIOUR BE

[illegible]

- Seamless Real-Life Interaction
- Customer Data Security
- Reduce Customer Frustration

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

Identify strong TR & EM

3. TRIGGERS TR

[illegible]

- Improve Lead Generation.
- Reduce Customer Service Costs.
- Monitor Consumer Data to Gain Insights.

4. EMOTIONS: BEFORE / AFTER EM

[illegible]

- Took longer time to process and respond to the query

10. YOUR SOLUTION SL

[illegible]

- Instead of navigating to several screens for booking products online, the user can directly talk to Chatbot regarding the products.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
[illegible]

- Able to serve customers with a consistent level of quality in a short period of time across different channels,

8.2 OFFLINE
[illegible]

- Make sure they are aware of the usage of the chatbots

Extract online & offline CH of BE

Figure 3.4 problem solution fit

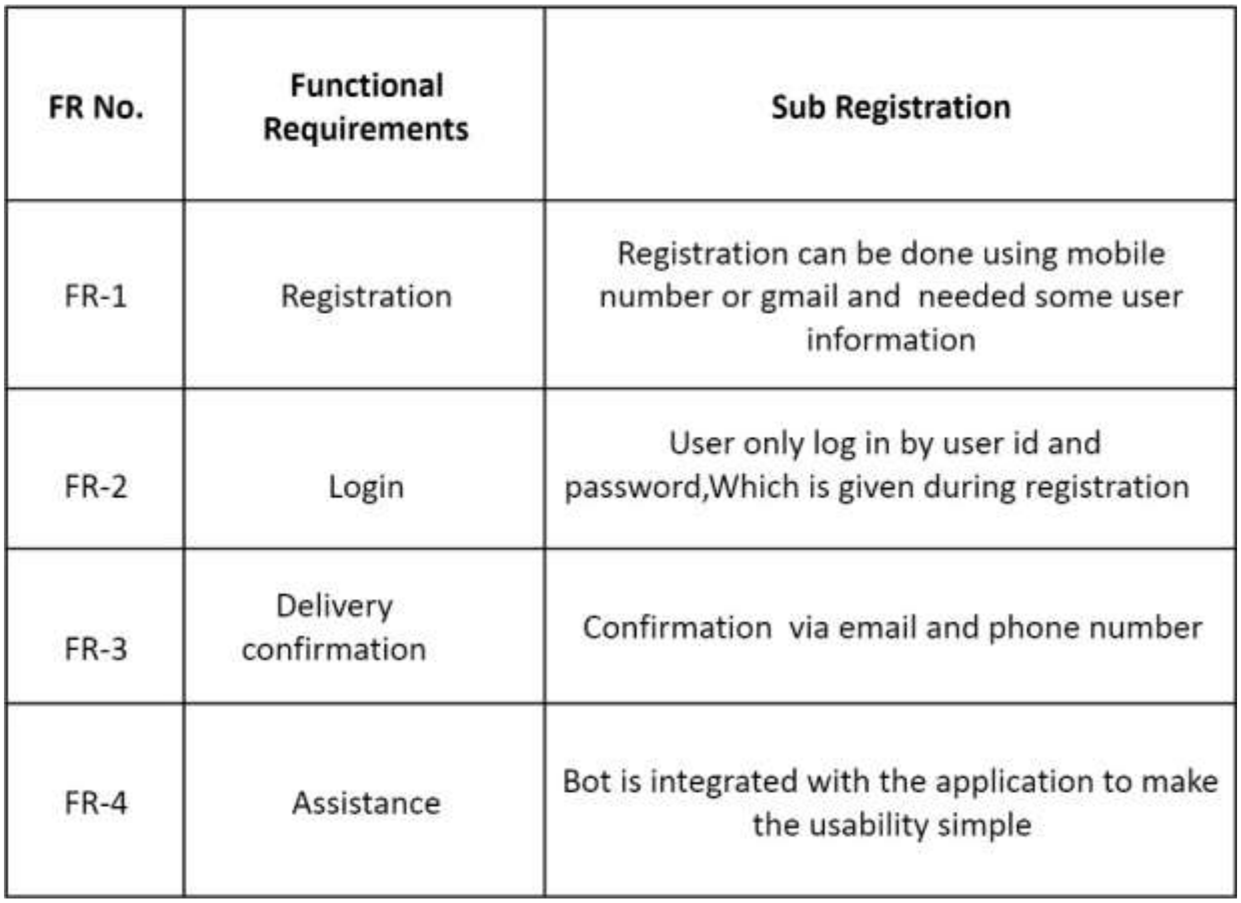
CHAPTER-4

4. REQUIREMENT ANALYSIS:

4.1Functional Requirements:

The functional requirements of the application are:

- Redirect users to their respective dashboards
- Allow admin to track sales of individual products
- Allow admin to manage orders made by a particular customer.
- Allow users to interact with the chatbot.
- Manage users' choices and charges using the chatbot.
- Promote the best deals and offers.
- Store customer details and orders.
- Send Notifications to customers if the order is confirmed.
- Collect user feedback.
- Recommend products based on user preference.
- Enable online payment features.
- Generate reports for order summary and order histories.



FR No.	Functional Requirements	Sub Registration
FR-1	Registration	Registration can be done using mobile number or gmail and needed some user information
FR-2	Login	User only log in by user id and password,Which is given during registration
FR-3	Delivery confirmation	Confirmation via email and phone number
FR-4	Assistance	Bot is integrated with the application to make the usability simple

Table 4.1 Functional requirements

4.2Non-Functional Requirements

Following are the Non-Functional requirements of the proposed solution.

Performance Requirements

- The response should not take longer than 5 seconds to appear on the client side.
- The client application should lazy load images of the product to minimize network calls over
- the network.

- The responses from the server should be cached on the client side.

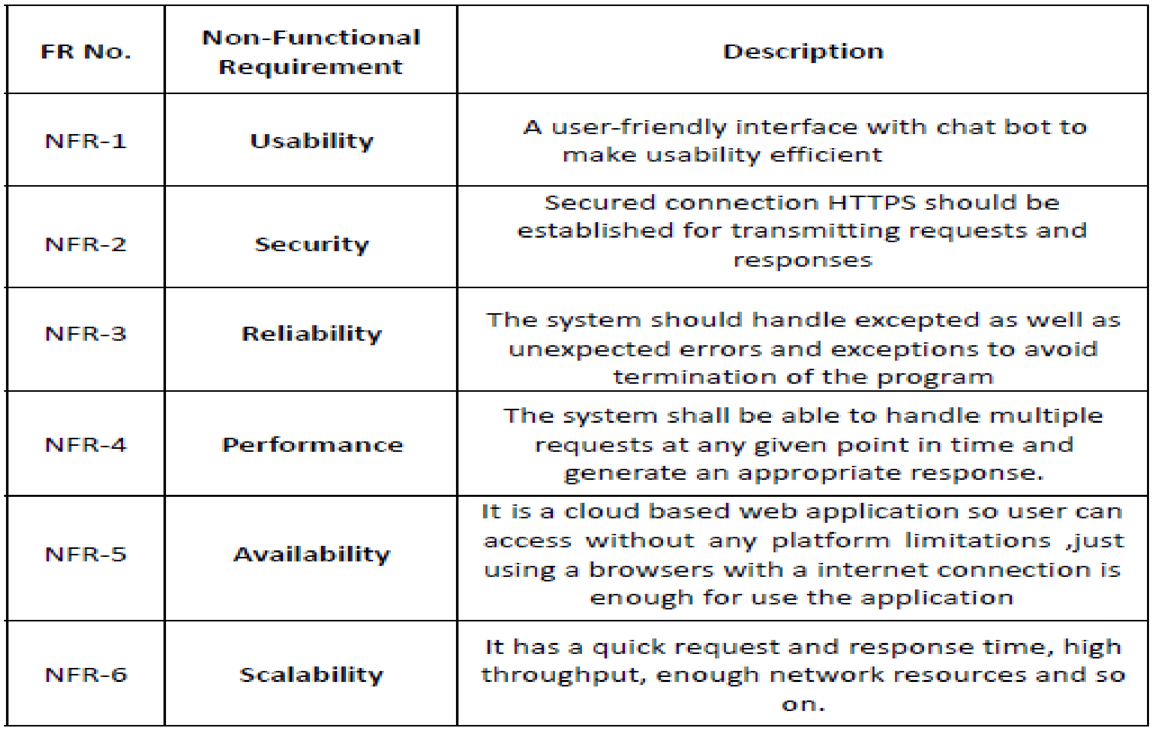
Security Requirements

- Credentials and secrets should be stored securely and should not be leaked.
- Secured connection HTTPS should be established for transmitting requests and responses between client and server.
- The system has different roles assigned to a user and every user has access constraints.
- User access token should be valid for a shorter period and needs to be refreshed periodically.
- Clients should implement mechanisms to prevent XSS attacks.
- The server should restrict access to the resources for the particular client domain.

Error Handling

The system should handle expected as well as unexpected errors and exceptions to avoid termination of the program.

Appropriate error messages should be generated and displayed to the client.



FR No.	Non-Functional Requirement	Description
NFR-1	Usability	A user-friendly interface with chat bot to make usability efficient
NFR-2	Security	Secured connection HTTPS should be established for transmitting requests and responses
NFR-3	Reliability	The system should handle excepted as well as unexpected errors and exceptions to avoid termination of the program
NFR-4	Performance	The system shall be able to handle multiple requests at any given point in time and generate an appropriate response.
NFR-5	Availability	It is a cloud based web application so user can access without any platform limitations ,just using a browsers with a internet connection is enough for use the application
NFR-6	Scalability	It has a quick request and response time, high throughput, enough network resources and so on.

Table 4.2 Non-Functional Requirements

CHAPTER -5

5.PROJECT DESIGN

Data Flow Diagram

A data flow diagram (DFD) is a graphical or visual representation that describes how data is moved through an organization's operations using a standardized set of symbols and notations.

In formal methodologies like the Structured Systems Analysis and Design Method, they are frequently included. DFDs may initially resemble flowcharts or the Unified Modeling Language.

1. Advantages of data flow diagram:

- It aids in describing the boundaries of the system.
- It is beneficial for communicating existing system knowledge to the users.
- A straightforward graphical technique which is easy to recognise.

- DFDs can provide a detailed representation of system components.
- It is used as the part of system documentation file.
- DFDs are easier to understand by technical and nontechnical audiences
- It supports the logic behind the data flow within the system.

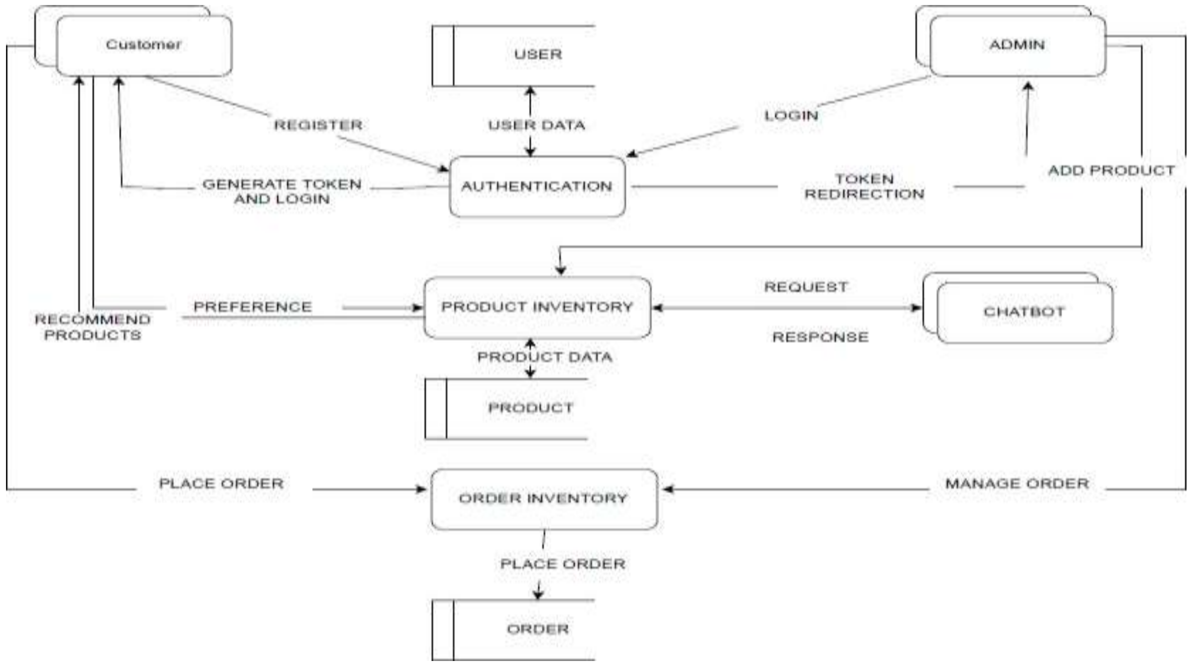


Figure 5.1 Data flow diagram

5.2Solution & Technical Architecture:

The system architecture defines the hardware, software and network environment of the structure. The system will be web-based meaning that the users need to run the URL in order to run the system. The system will run both horizontally and vertically. The architecture used in the system is shown horizontally where the Model View Controller . The high-level part of the system is looked at using the vertical way.

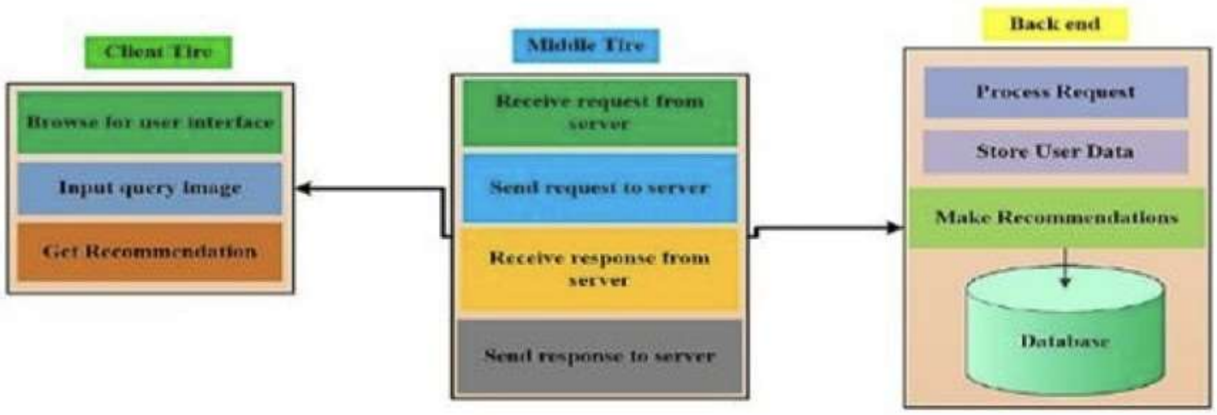


Figure 5.2.1 Three tire architecture

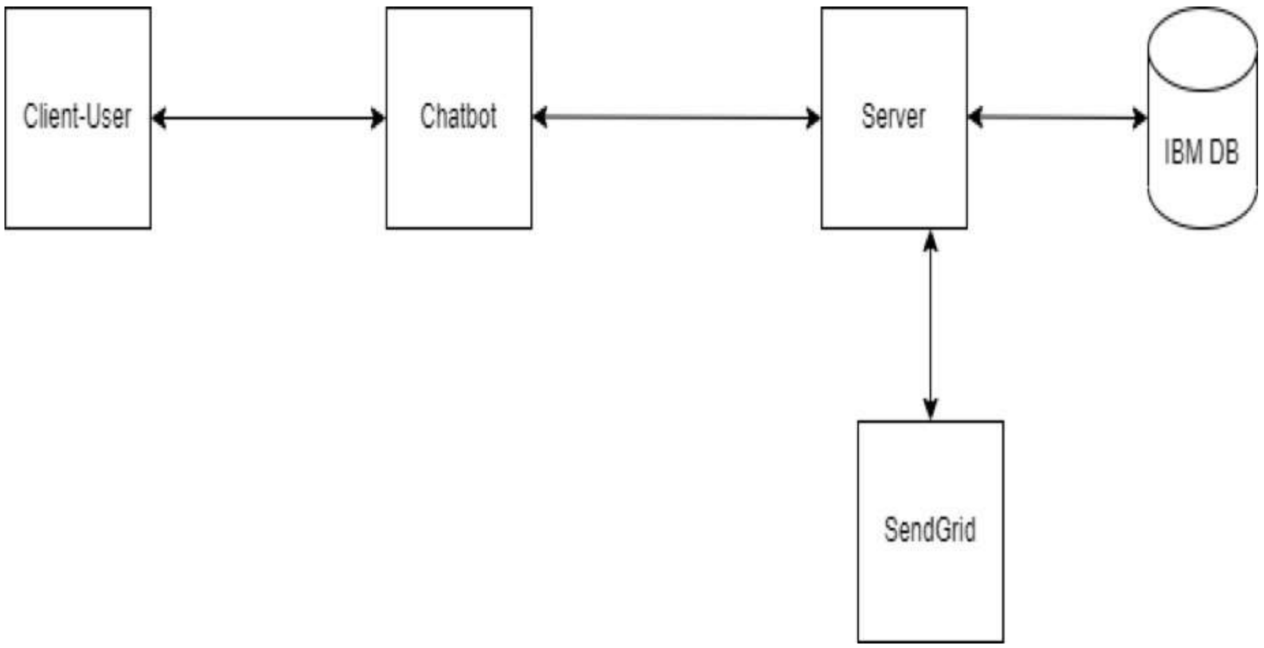


Figure 5.2.2 Intergration of Chat-bot with IBM BD and Sendgrid

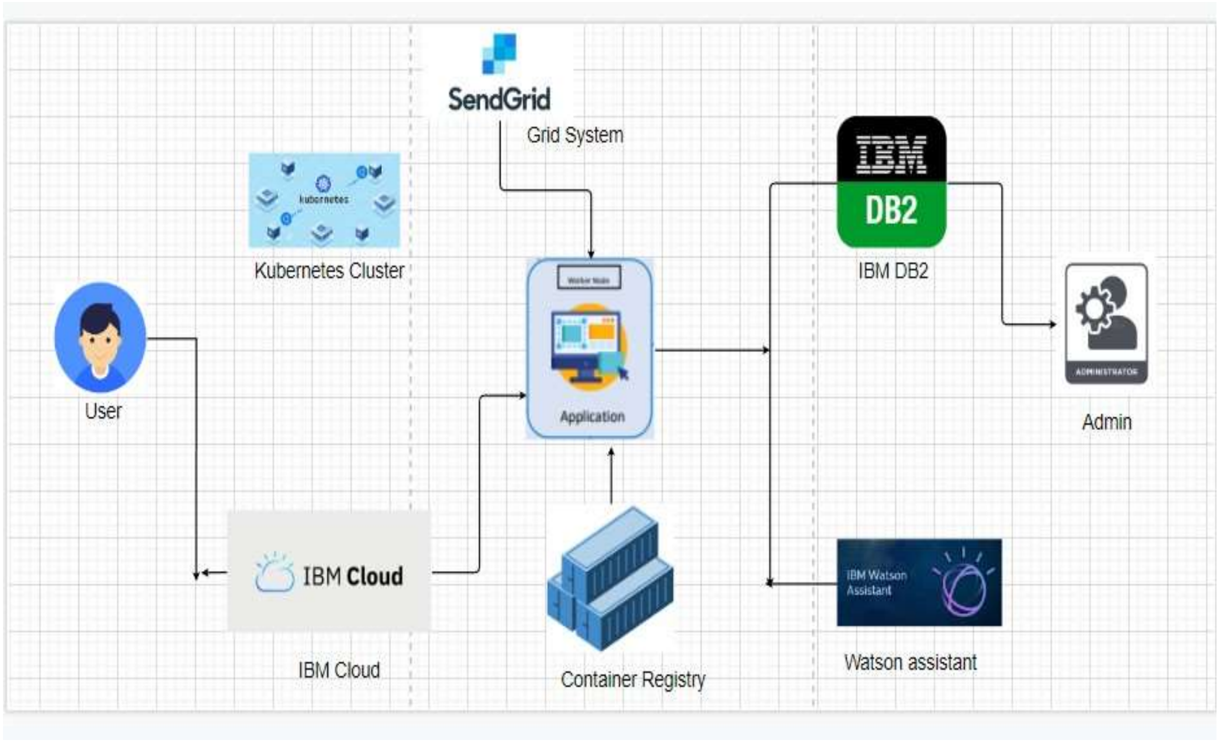


Figure 5.2.3 integration of total website with IBM Cloud ,Container Register ,Watson assistant

5.3 User Stories

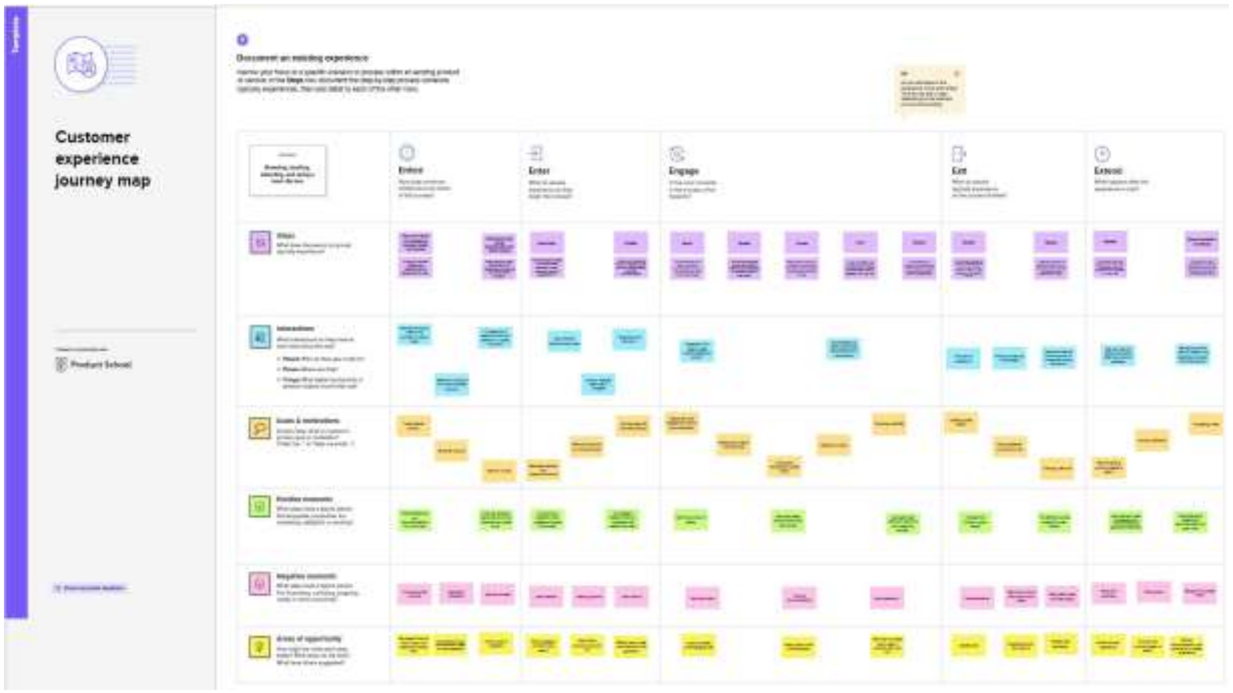


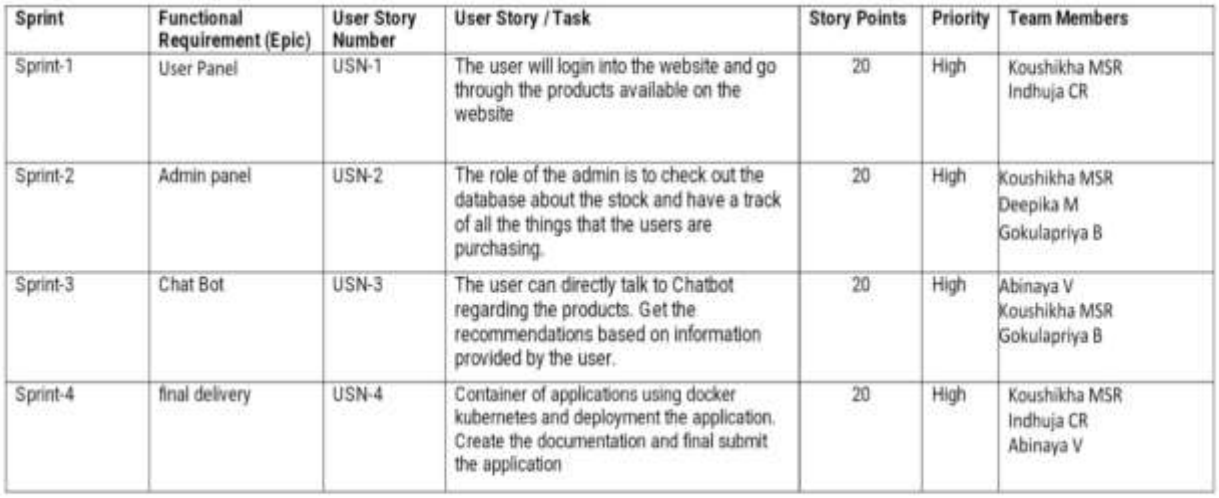
Figure 5.3 Customer Experience Journey Map

CHAPTER-6

6.PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

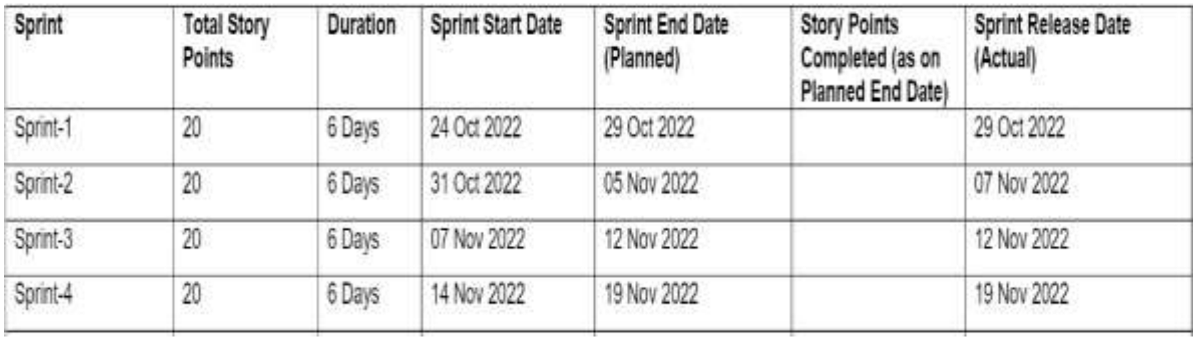
Use the below template to create product backlog and sprint schedule



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Panel	USN-1	The user will login into the website and go through the products available on the website	20	High	Koushikha MSR Indhuja CR
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	Koushikha MSR Deepika M Gokulapriya B
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	Abinaya V Koushikha MSR Gokulapriya B
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	Koushikha MSR Indhuja CR Abinaya V

Table 6.1 Sprint planning

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		07 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

Table 6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

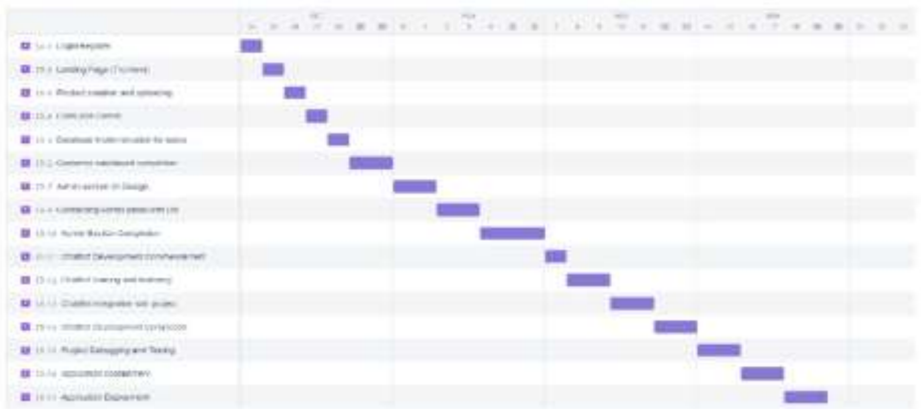


Chart 6.3 Jira file

CHAPTER-7

7.Coding and solutioning

Backend

i) app.py

```
from datetime import datetime

from flask import Flask,request,send_from_directory

from .lib import db,validation_error

import ibm_db

from flask_cors import CORS


import os

from .api import auth_bp,category_bp,product_bp,cart_bp,order_bp

app = Flask(_name_)

app.config['MAX_CONTENT_LENGTH'] = 1024 * 1024

app.config['UPLOAD_EXTENSIONS'] = ['.jpg', '.png', '.gif']

app.config['UPLOAD_PATH'] = 'uploads'


CORS(app) # This will enable CORS for all routes


db.get_db()


app.register_blueprint(auth_bp.auth_bp,url_prefix="/api/v1/auth")

app.register_blueprint(category_bp.category_bp,url_prefix="/api/v1/category")

app.register_blueprint(product_bp.product_bp,url_prefix="/api/v1/product")

app.register_blueprint(cart_bp.cart_bp,url_prefix="/api/v1/cart")
app.register_blueprint(order_bp.order_bp,url_prefix="/api/v1/order")
```

The API source files are given as follows:

ii) auth_bp.py

```
from flask import Blueprint,jsonify,g,request

import ibm_db

from passlib.hash import sha256_crypt

import jwt



from ..lib import validation_error

from ..lib import exception

 from ..lib import db



auth_bp = Blueprint("auth",__name__)








@auth_bp.route("/",methods=["GET"])

def check():

    print(g.get("db"))

    return jsonify({"msg":"hi"})



@auth_bp.route('/register',methods=['POST'])

def reg():

    try:
        data = request.get_json()

        name=data['name']

        email=data['email']

        password=data['password']

        mobile_no=data['mobileNo']

        print(email,password,name,mobile_no)

        insert_sql="INSERT INTO USER(name,email,password,role,mobilenumber) VALUES(?,?,?,?,?)"

        prep_stmt = ibm_db.prepare(db.get_db(), insert_sql)

        ibm_db.bind_param(prep_stmt,1,name)

        ibm_db.bind_param(prep_stmt,2,email)

        ibm_db.bind_param(prep_stmt,3,sha256_crypt.encrypt(password))

        ibm_db.bind_param(prep_stmt,4,"user")

        ibm_db.bind_param(prep_stmt,5,mobile_no)

        ibm_db.execute(prep_stmt)

        return {"message":'Created'},201


    except Exception as e:

        return exception.handle_exception(e)




@auth_bp.route('/me',methods=['GET'])

def getMe():

    try:

        token = request.headers['Authorization']

        if (not token):
            return validation_error.throw_validation("Please login",401)
        decoded = jwt.decode(token,"secret",algorithms=["HS256"])
        select_sql = "SELECT * FROM USER WHERE ID=?"
        prep_stmt = ibm_db.prepare(db.get_db(), select_sql)
        ibm_db.bind_param(prep_stmt,1,decoded['id'])
        ibm_db.execute(prep_stmt)
        isUser=ibm_db.fetch_assoc(prep_stmt)
        return  isUser
    except Exception as e:
            return exception.handle_exception(e)



@auth_bp.route('/login',methods=['POST'])
def auth_log():
    try:
        data = request.get_json()
        print(data)
        email=data['email']
        password=data['password']
        select_sql = "SELECT * FROM USER WHERE EMAIL=?"
        prep_stmt = ibm_db.prepare(db.get_db(), select_sql)
        ibm_db.bind_param(prep_stmt,1,email)
        ibm_db.execute(prep_stmt)
        isUser=ibm_db.fetch_assoc(prep_stmt)
        print(isUser)
        if not isUser:

            return validation_error.throw_validation("Invalid Credentials",400)

        if not sha256_crypt.verify(password,isUser['PASSWORD']):

            return validation_error.throw_validation("Invalid Credentials",400)

        encoded_jwt = jwt.encode({"id":isUser['ID'],"role":isUser['ROLE']},"secret",algorithm="HS256")

        isUser["token"] = encoded_jwt

        return  isUser

    except Exception as e:

            return exception.handle_exception(e)

Footer
```

iii) cart_bp.py

```
from flask import Blueprint,request

import ibm_db

from ..lib import validation_error

from ..lib.auth import check_auth

from ..lib import exception

from ..lib import db




cart_bp = Blueprint("cart",__name__)




@cart_bp.route("/",methods=['POST'])
```

```
def add_cart():
 try:
  user_id =check_auth(request)
  data=request.get_json()
  product=data['product']
  select_sql = "SELECT * FROM PRODUCT WHERE ID=?"
  prepare_select =ibm_db.prepare(db.get_db(),select_sql)
  ibm_db.bind_param(prepare_select,1,product)
  ibm_db.execute(prepare_select)
  is_product = ibm_db.fetch_assoc(prepare_select)

  print(is_product)

  if not is_product:
   return validation_error.throw_validation("No Product found",404)

  if(is_product['STOCK']<=0):
   return validation_error.throw_validation("No Stock found",404)

  print("Hey")
  insert_sql="INSERT INTO CART(user,product) VALUES(?,?)"
  prep_stmt = ibm_db.prepare(db.get_db(), insert_sql)
  ibm_db.bind_param(prep_stmt,1,user_id)
  ibm_db.bind_param(prep_stmt,2,product)
  ibm_db.execute(prep_stmt)
```

```
    print("heyy")

    update_sql="UPDATE PRODUCT SET stock=? WHERE ID=?"
    update_stmt = ibm_db.prepare(db.get_db(), update_sql)
    ibm_db.bind_param(update_stmt,1,is_product['STOCK']-1 or 0)
    ibm_db.bind_param(update_stmt,2,product)
    ibm_db.execute(update_stmt)



    print("sdd")
    return {"message":'Created'},201
  except Exception as e:
    return exception.handle_exception(e)

@cart_bp.route("/",methods=['DELETE'])
def delete_user_cart():
  try:
    user_id =check_auth(request)
    insert_sql="DELETE FROM CART WHERE USER=?"
    prep_stmt = ibm_db.prepare(db.get_db(), insert_sql)
    ibm_db.bind_param(prep_stmt,1,user_id)

    ibm_db.execute(prep_stmt)
    return {"message":'Deleted'},201
```

```
 except Exception as e:

   return exception.handle_exception(e)





@cart_bp.route("/",methods=['GET'])

def get_cart():

 try:

   user_id =check_auth(request)

   insert_sql="SELECT PRODUCT.ID AS product_id,cart_id,
category,category_name,product_name,description,price,stock,image,brand,specificity,CART.user as user
FROM CART JOIN PRODUCT ON CART.PRODUCT=PRODUCT.ID JOIN CATEGORY ON
PRODUCT.CATEGORY = CATEGORY.ID WHERE CART.USER=?"

   prep_stmt = ibm_db.prepare(db.get_db(), insert_sql)

   ibm_db.bind_param(prep_stmt,1,user_id)


   ibm_db.execute(prep_stmt)

   products=[]

   product=ibm_db.fetch_assoc(prep_stmt)

   while(product != False):

   products.append(product)

     product = ibm_db.fetch_assoc(prep_stmt)

   print(products)

   return products or [],200


 except Exception as e:
   return exception.handle_exception(e)



@cart_bp.route("/<product>/<id>",methods=['DELETE'])

def delete_cart(product,id):

 try:

   user_id =check_auth(request)

   print(product,id,user_id)


   select_sql = "SELECT * FROM PRODUCT WHERE ID=?"

   prepare_select =ibm_db.prepare(db.get_db(),select_sql)

   ibm_db.bind_param(prepare_select,1,product)

   ibm_db.execute(prepare_select)

   is_product = ibm_db.fetch_assoc(prepare_select)


   print(is_product)


   if not is_product:

     return validation_error.throw_validation("No Product found",404)


   print("ff")

   insert_sql="DELETE FROM CART WHERE CART_ID=? AND user=?"

   prep_stmt = ibm_db.prepare(db.get_db(), insert_sql)

   ibm_db.bind_param(prep_stmt,1,id)

   ibm_db.bind_param(prep_stmt,2,user_id)
   ibm_db.execute(prep_stmt)

   print("aa")

   update_sql="UPDATE PRODUCT SET stock=? WHERE ID=?"

   update_stmt = ibm_db.prepare(db.get_db(), update_sql)

   ibm_db.bind_param(update_stmt,1,is_product['STOCK']+1)

   ibm_db.bind_param(update_stmt,2,product)

   ibm_db.execute(update_stmt)

   return {"message":'Deleted'},200

 except Exception as e:

   return exception.handle_exception(e)

Footer
```

iv) category_bp.py

```
from flask import Blueprint,request,jsonify

import ibm_db

from ..lib import exception

from ..lib import db



category_bp = Blueprint("category",__name__)




@category_bp.route("/get",methods=["GET"])

def get_category():

  try:
    select_sql = "SELECT * FROM CATEGORY WHERE"

    prep_stmt = ibm_db.prepare(db.get_db(), select_sql)

    ibm_db.execute(prep_stmt)

    categories=[]

    category=ibm_db.fetch_assoc(prep_stmt)

    while(category != False):

    categories.append(category)

      category = ibm_db.fetch_assoc(prep_stmt)

    print(categories)

    # return categories,200

    return jsonify(categories),200

  except Exception as e:

    return exception.handle_exception(e)


@category_bp.route("/create",methods=["POST"])

def add_category():

  try:

    data = request.get_json()

    category = data['category']

    insert_sql="INSERT INTO CATEGORY(category_name) VALUES(?)"

    prep_stmt = ibm_db.prepare(db.get_db(), insert_sql)

    ibm_db.bind_param(prep_stmt,1,category)

    ibm_db.execute(prep_stmt)

  return {"message":'Created'},201

  except Exception as e:
    return exception.handle_exception(e)



@category_bp.route("/<id>",methods=["DELETE"])
def get_category_id(id):
  try:
    print(id)
    select_sql = "DELETE FROM CATEGORY WHERE ID=?"
    prep_stmt = ibm_db.prepare(db.get_db(), select_sql)
    ibm_db.bind_param(prep_stmt,1,id)
    ibm_db.execute(prep_stmt)

    return {"message":'Deleted'},200
  except Exception as e:
    return exception.handle_exception(e)
```

v) image_bp.py

```
from datetime import datetime
from flask import Blueprint,request
import ibm_db
import os
from ..lib import exception
from ..lib import db
```

```
image_bp = Blueprint("image",__name__)

@image_bp.route('/image/<id>',methods=['POST'])
def uploadImage(id):
  try:
    uploaded_file = request.files['file']+datetime.date
    if uploaded_file.filename != '':
        uploaded_file.save(os.path.join('/uploads', uploaded_file.filename))
    insert_sql="UPDATE PRODUCT SET image=? WHERE ID=?"
    prep_stmt = ibm_db.prepare(db.get_db(), insert_sql)
    ibm_db.bind_param(prep_stmt,1,uploaded_file)
    ibm_db.bind_param(prep_stmt,2,id)

    ibm_db.execute(prep_stmt)
  return {"message":'Updated'},200
  except Exception as e:
    return exception.handle_exception(e)

@image_bp.route('/<filename>')
def upload(filename):
  try:
    return send_from_directory("/uploads", filename)
  except Exception as e:
    return exception.handle_exception(e
```

v) product_bp.py

```
from flask import Blueprint,request,jsonify

import ibm_db

from ..lib import exception

from ..lib import db




product_bp = Blueprint("product",__name__)


@product_bp.route("/create",methods=['POST'])

def add_product():

  try:

    data = request.get_json()

    product_name=data['product_name']

    category=data['category']

    description = data['description']

    stock=data['stock']

    price = data['price']

    insert_sql="INSERT INTO PRODUCT(product_name,category,description,stock,price) 
VALUES(?,?,?,?,?)"

    prep_stmt = ibm_db.prepare(db.get_db(), insert_sql)

    ibm_db.bind_param(prep_stmt,1,product_name)

    ibm_db.bind_param(prep_stmt,2,category)

    ibm_db.bind_param(prep_stmt,3,description)

    ibm_db.bind_param(prep_stmt,4,stock)
```

```
     ibm_db.bind_param(prep_stmt,5,price)

     ibm_db.execute(prep_stmt)

     return {"message":'Created'},200

  except Exception as e:

    return exception.handle_exception(e)


@product_bp.route("/get",methods=['GET'])

def get_product():

  try:

    # select_sql = "SELECT PRODUCT.ID AS product_id,
category,category_name,product_name,description,price,stock,image FROM PRODUCT JOIN CATEGORY
ON CATEGORY.ID=PRODUCT.CATEGORY"

    select_sql = "SELECT * FROM PRODUCT WHERE"

    prep_stmt = ibm_db.prepare(db.get_db(), select_sql)

    ibm_db.execute(prep_stmt)

    products=[]

    product=ibm_db.fetch_assoc(prep_stmt)

    while(product != False):

    products.append(product)

      product = ibm_db.fetch_assoc(prep_stmt)

    print(products)

    return jsonify(products) or [],200

  except Exception as e:

    return exception.handle_exception(e)
```

```
@product_bp.route("/<id>",methods=['GET'])

def get_product_id(id):

  try:

    # select_sql = "SELECT PRODUCT.ID AS product_id,
category,category_name,product_name,description,price,stock,image FROM PRODUCT JOIN CATEGORY
ON CATEGORY.ID=PRODUCT.CATEGORY WHERE PRODUCT.ID=?"

    select_sql = "SELECT * FROM PRODUCT WHERE PRODUCT.ID=?"

    prep_stmt = ibm_db.prepare(db.get_db(), select_sql)

    ibm_db.bind_param(prep_stmt,1,id)

    ibm_db.execute(prep_stmt)

    product=ibm_db.fetch_assoc(prep_stmt)

    print(product)

    return product or [],200

  except Exception as e:

    return exception.handle_exception(e)




@product_bp.route("/<id>",methods=['PUT'])

def update_product(id):

  try:

     data = request.get_json()

     product_name=data['product_name']

     category=data['category']

     description = data['description']

     stock=data['stock']

     price = data['price']
     insert_sql="UPDATE PRODUCT SET product_name=?,category=?,description=?,stock=?,price=?
WHERE ID=?"

     prep_stmt = ibm_db.prepare(db.get_db(), insert_sql)

     ibm_db.bind_param(prep_stmt,1,product_name)

     ibm_db.bind_param(prep_stmt,2,category)

     ibm_db.bind_param(prep_stmt,3,description)

     ibm_db.bind_param(prep_stmt,4,stock)

     ibm_db.bind_param(prep_stmt,5,price)

     ibm_db.bind_param(prep_stmt,6,id)

     ibm_db.execute(prep_stmt)

     return {"message":'Updated'},200

  except Exception as e:

    return exception.handle_exception(e)




@product_bp.route("/<id>",methods=['DELETE'])

def delete_product(id):

  try:

     insert_sql="DELETE FROM PRODUCT WHERE ID=?"

     prep_stmt = ibm_db.prepare(db.get_db(), insert_sql)

     ibm_db.bind_param(prep_stmt,1,id)

     ibm_db.execute(prep_stmt)

     return {"message":'Deleted'},200

  except Exception as e:

    return exception.handle_exception(e)
```

The Library source code files for the application are given as follows

vi) auth.py

```
import jwt
from . import validation_error


def check_auth(request):
  token = request.headers['Authorization']
  if (not token):
    return validation_error.throw_validation("Please login",401)
  decoded = jwt.decode(token,"secret",algorithms=["HS256"])
  return decoded['id']
```

vii) db.py

```
import ibm_db


conn = None


def get_db():
    global conn
    print(conn)
    if conn == None:
        conn =
ibm_db.connect("DATABASE=bludb;HOSTNAME=9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32459;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=yvp21124;PWD=jNDAkHSrZNopa2oe","","")
```

```
    return conn
```

viii) exception.py

```
from flask import json


def handle_exception(e):

    exception_name = type(e)._name____

    exception_str = e._str_()

    print(exception_name,exception_str.find('803'))

    status_code=500

    response={

        "status":"fail",

        "message":"Something went wrong"

    }

    if exception_name=='KeyError':

        status_code=400

        response["message"] = "Please fill all fields"




    if exception_name=='ValidationException':

        print(e.args)

        status_code=400

        response["message"] = "Please fill all fields"
```

```
 if exception_str.find('803') >=0:

     print('Db')

     status_code=400

     response["message"] = "Invalid value provided"


 if exception_name == 'DecodeError':

     print("token error")

     status_code=401

     response['message']='Token expired.Please login'




 return json.dumps(response), status_code, {'ContentType':'application/json'}
```

ix) sendmail.py

```
import os

from sendgrid import SendGridAPIClient

from sendgrid.helpers.mail import Mail

from ..lib import exception




def sendemail():
  message = Mail(

      from_email='from_email@example.com',

      to_emails='to@example.com',

      subject='ORDER CONFIRMATION',

      html_content='<strong>Your order placed</strong>')

  try:

      sg = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))

      response = sg.send(message)

      print(response.status_code)

      print(response.body)

      print(response.headers)

  except Exception as e:

      return exception.handle_exception(e)
```

x) validation_error.py

```
from flask import json



def throw_validation(msg,code):

  return json.dumps({"status":"fail","message":msg}),code,{'ContentType':'application/json'}
```

Output Screenshots:

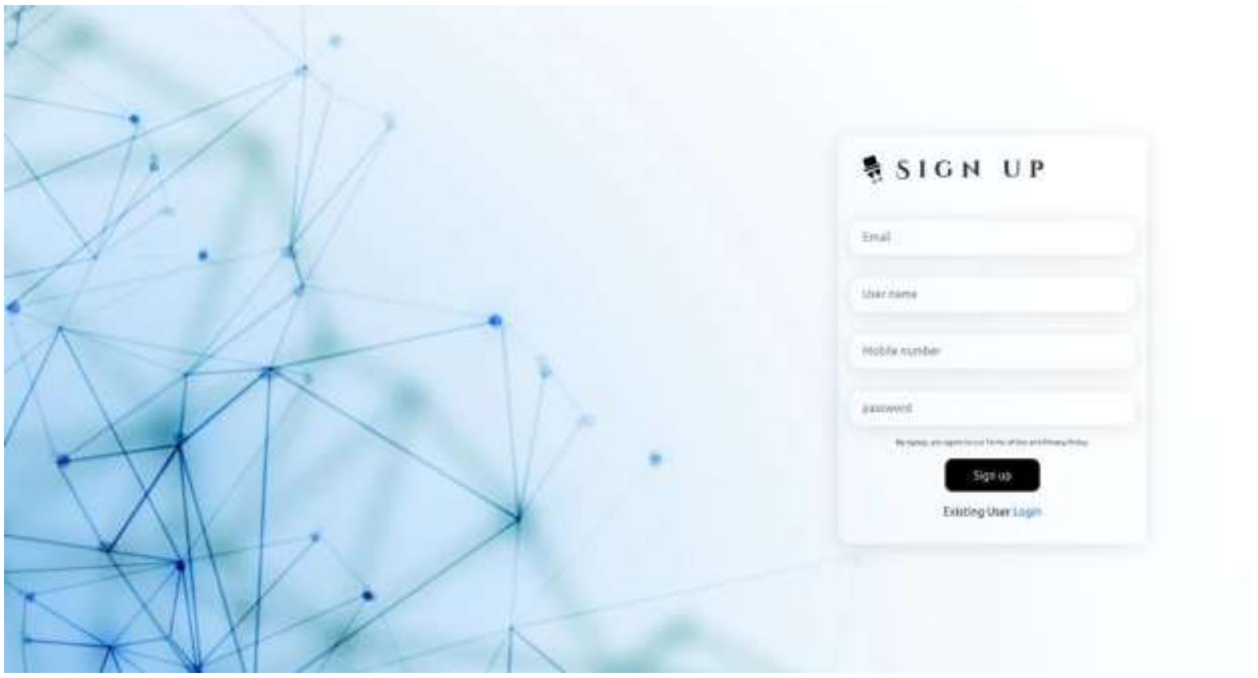


Figure 7.1 Signup page

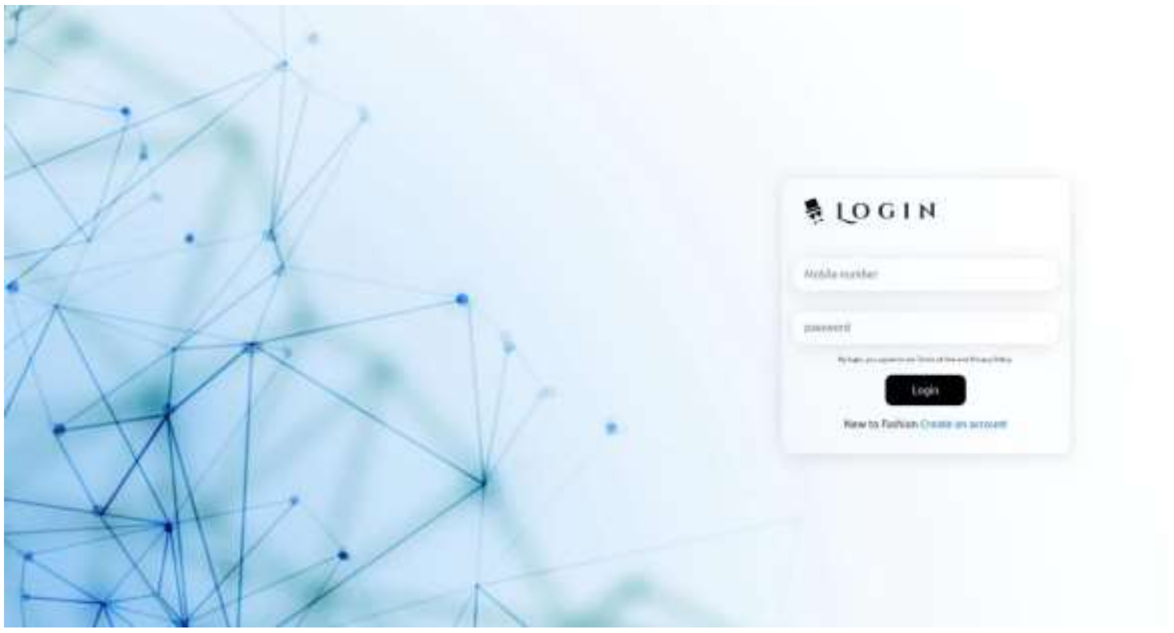


Figure 7.2 Login page

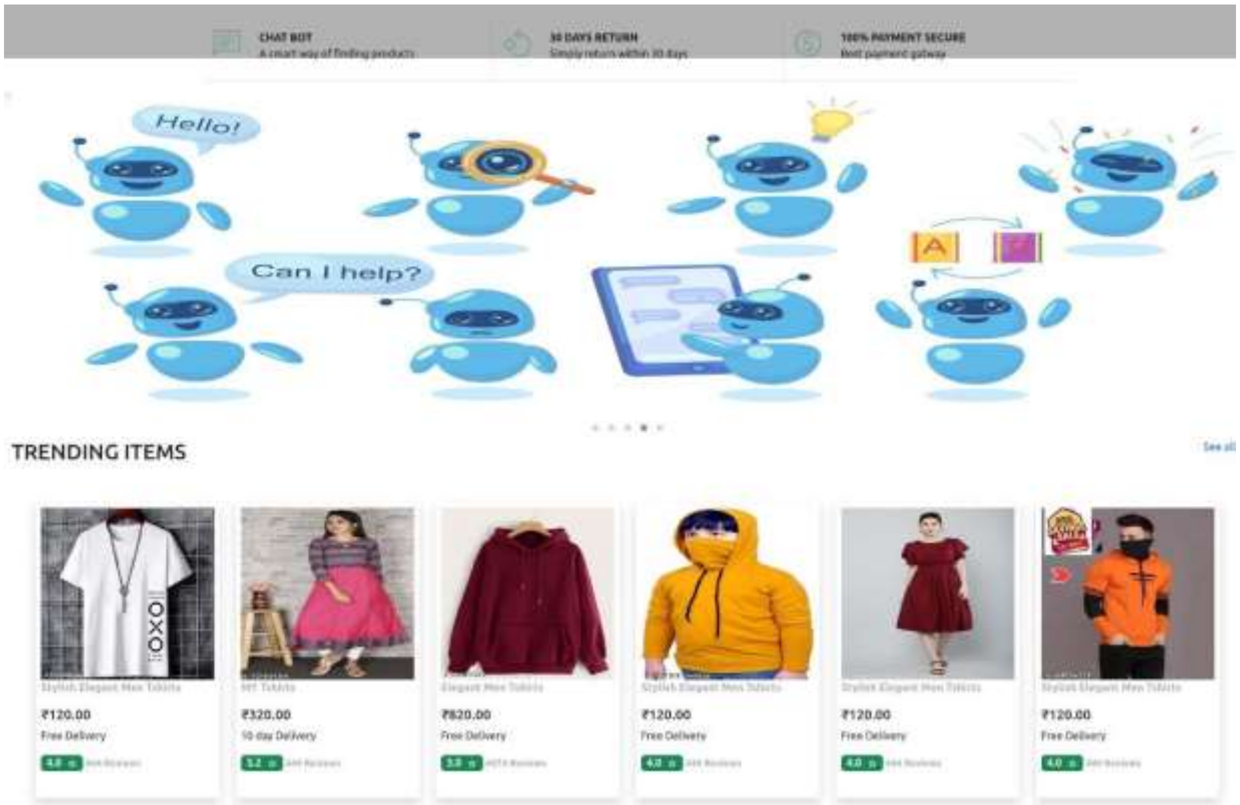


Figure 7.3 chat-bot and Trending images

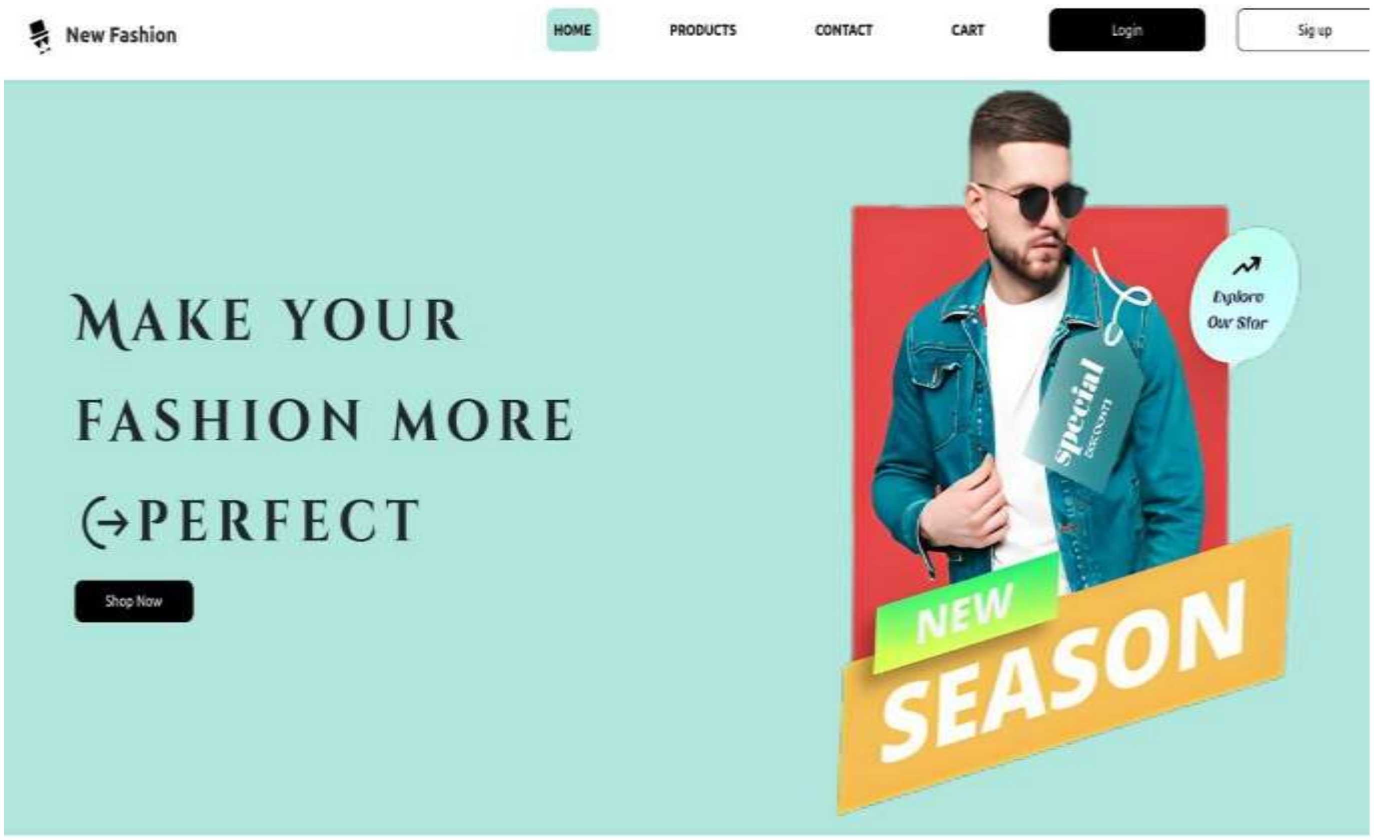


Figure 7.4 home page

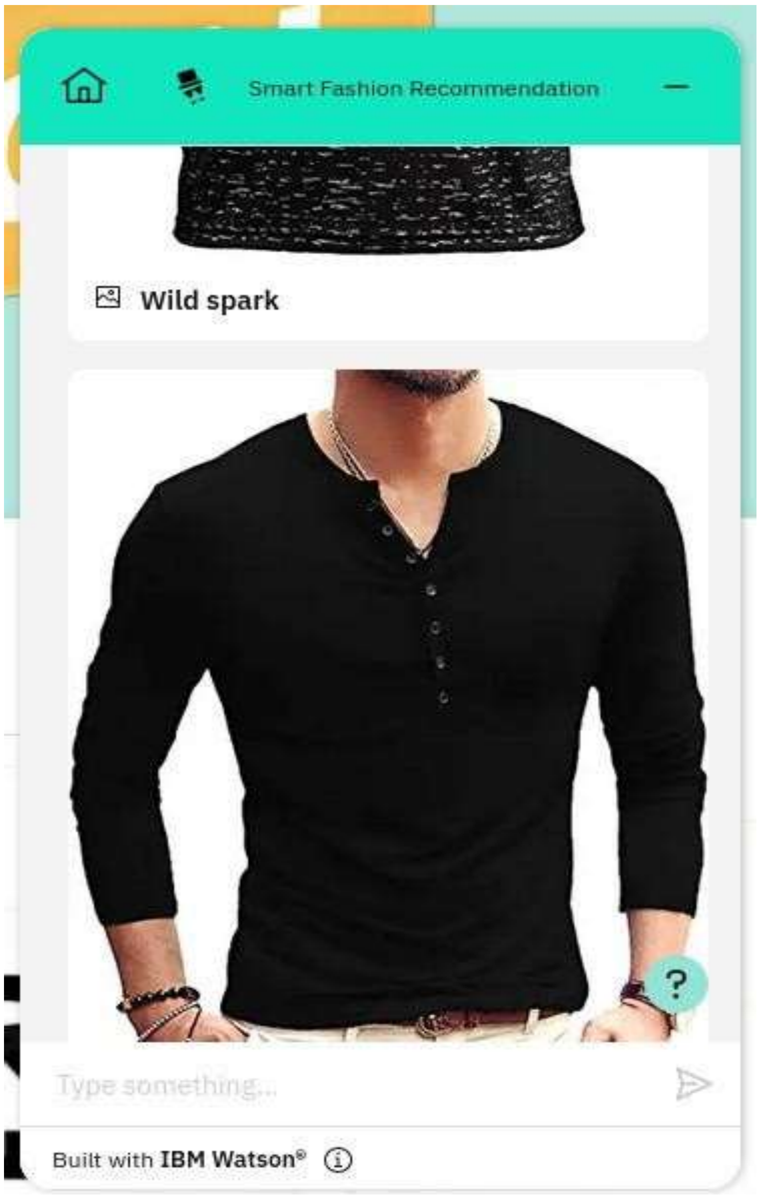


Figure7.5 chat-bot filtering

8.TESTING:

CHAPTER-8

8.1 Test cases

A test case is a series of operations carried out on a system to see if it complies with software requirements and operates properly. A test case's objective is to ascertain whether various system features operate as anticipated and to check that the system complies with all applicable standards, recommendations, and user requirements. The act of creating a test case can also aid in identifying flaws or mistakes in the system.

A test case document includes test steps, test data, preconditions and the post conditions that verify requirements.

Why test case are so important:

Writing test cases is a significant task and is regarded as one of the most crucial components of software testing. The testing team, the development team, and management all use it. We can use a test case as a starting point document if an application doesn't have any documentation.

- In other words, test cases make clear what must be done to test a system. It provides us with the actions we take in a system, the values we provide as input data, and the anticipated outcomes when we run a certain test case.
- Test cases give a precise picture of the expectations that must be met.
- Test cases demonstrate how you addressed and tested each requirement for the product.
- Test cases assist new team members in quickly becoming engaged in the project, learning about your product and test management processes, and running prepared test cases when necessary.
- A solid foundation for developing automated scripts and adding automated

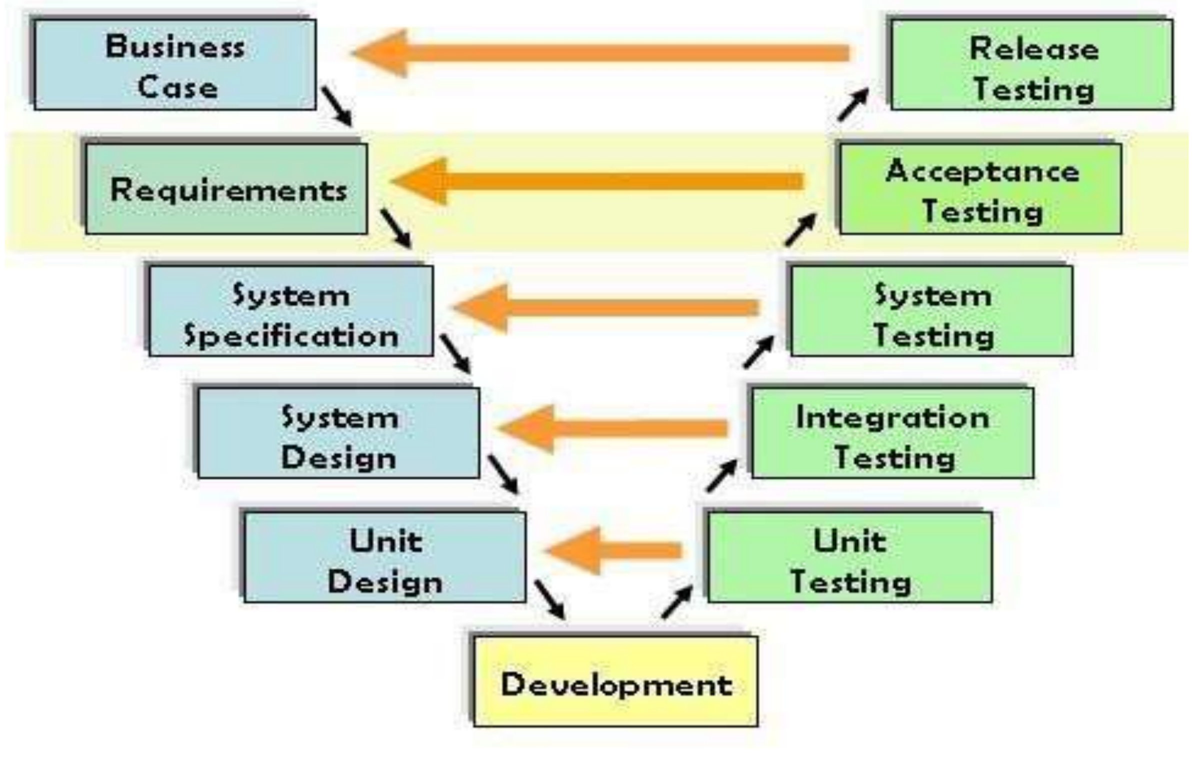
testing to the QA processis detailed manual test cases.

1. **User Acceptance Testing**

User acceptance testing (UAT), also called application testing or end-user testing, is a phase of software development in which the software is tested in the real world by its intended audience.

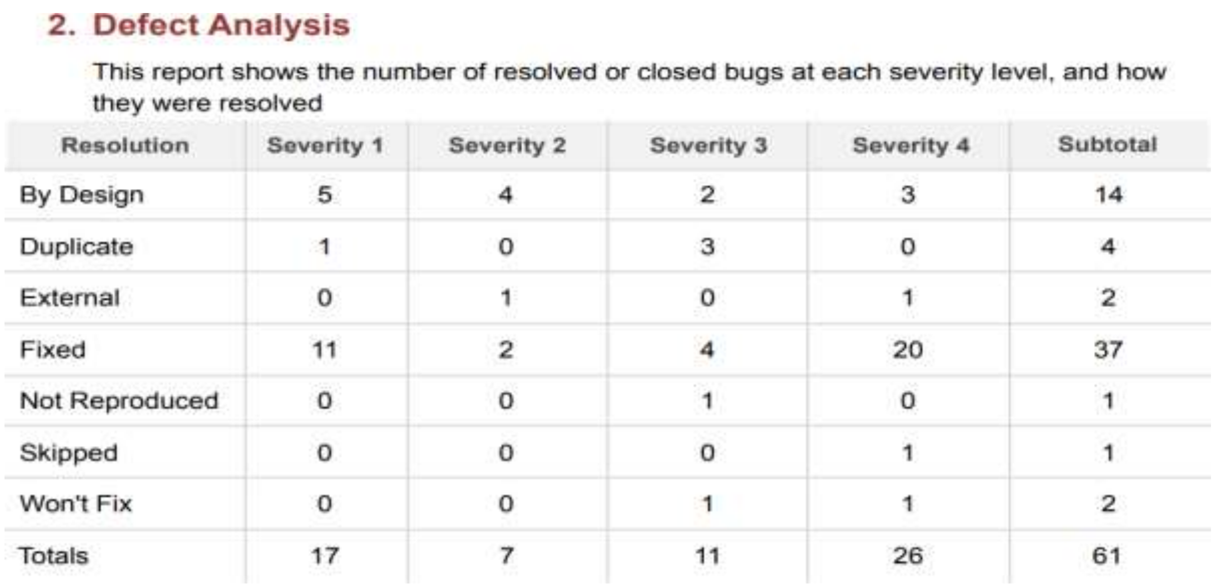
User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done.

Fig 7.1) Testing architecture



CHAPTER-9

Performance Metrics:

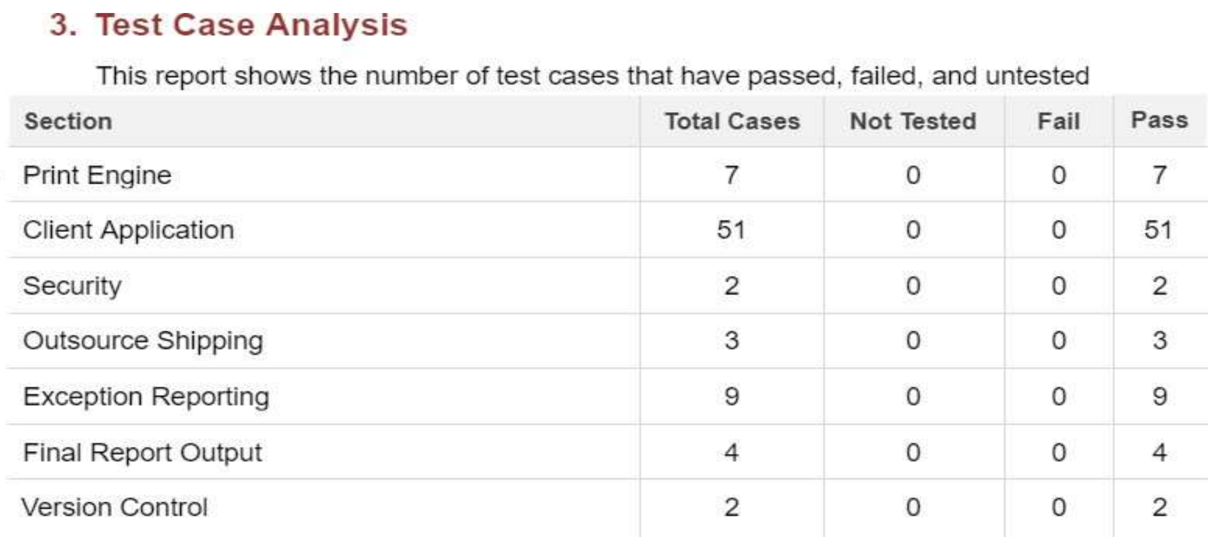


2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	4	2	3	14
Duplicate	1	0	3	0	4
External	0	1	0	1	2
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	0	1	1
Won't Fix	0	0	1	1	2
Totals	17	7	11	26	61

Table 9.1 Defect analysis



3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

Table 9.2 Test Case Analysis

CHAPTER-10

CONCULSION'

The present paper presents the development of a system that recognizes fashion similar images. We accomplish this by implementing an already existing CNN model with transfer
learning for cloth image recognition using different libraries. For this purpose, we created a plan for collecting data and for developing the steps needed for preprocessing and cleaning up the data.

We took into account features like patterns, machine, fabric, style etc. After extensive preprocessing and cleaning of data in a dataset, we constructed the model of stacked CNN to predict the features specific to these attributes and to train the models with the dataset to generate accurate predictions regarding almost all forms of images. A stacked CNN was used and implemented, with the help of this algorithm through which the system can recommend similar images This is the last test to assess if deep learning for style recovery is at a high development and
can be utilized in making fashion choices.