

SMART FASHION RECOMMENDER APPLICATION

IBM – DOCUMENTATION

UNDER THE GUIDANCE OF

INDUSTRY MENTOR(S) NAME : KRISHNA CHAITANYA

FACULTY MENTOR(S) NAME : AJIN M

TEAM ID : PNT2022TMID34632

SUBMITTED BY:

FEMIMA SHELLY A.T	962219104053
ABISHA G	962219104004
BENITTA R.K	962219104036
DELFIN D	962219104046



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ST XAVIER'S CATHOLIC COLLEGE OF ENGINEERING

S.NO	TABLE OF CONTENT	PG.NO
1	INTRODUCTION	4
1.1	PROJECT OVERVIEW	4
1.2	PURPOSE	4
2	LITERATURE SURVEY	5
2.1	EXISTING PROBLEM	5
2.2	REFERENCES	6
2.3	PROBLEM STATEMENT DEFINITION	7
3	IDEATION & PROPOSED SOLUTION	8
3.1	EMPATHY MAP CANVAS	8
3.2	IDEATION & BRAINSTORMING	8
3.3	PROPOSED SOLUTION	10
3.4	PROBLEM SOLUTION FIT	12
4	REQUIREMENT ANALYSIS	13
4.1	FUNCTIONAL REQUIREMENT	13
4.2	NON-FUNCTIONAL REQUIREMENT	14
5	PROJECT DESIGN	15
5.1	DATA FLOW DIAGRAM	15
5.2	SOLUTION & TECHNICAL ARCHITECTURE	16
6	PROJECT PLANNING & SCHEDULING	18
6.1	SPRINT PLANNING & ESTIMATION	18
6.2	SPRINT DELIVERY SCHEDULE	18
6.3	REPORTS FROM JIRA	19
7	CODING & SOLUTIONING	20
7.1	FEATURE 1	20
7.2	DATABASE SCHEMA	27

8	TESTING	28
8.1	TEST CASES	28
8.2	USER ACCEPTANCE TESTING	28
9	RESULTS	29
9.1	PERFORMANCE METRICS	32
10	ADVANTAGES & DISADVANTAGES	33
11	CONCLUSION	34
12	FUTURE SCOPE	35
13	APPENDIX	36
13.1	SOURCE CODE	36
13.2	GITHUB & PROJECT DEMO LINK	83

1. INTRODUCTION

1.1 PROJECT OVERVIEW

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 you want to wear. Having an AI program that understands the algorithm 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.

1.2 PURPOSE

- a) Using chatbot we can manage user's choices and orders.
- b) The chatbot can give recommendations to the users based on their interests.
- c) It can promote the best deals and offers on that day.
- d) It will store the customer's details and orders in the database.
- e) The chatbot will send a notification to customers if the order is confirmed.
- f) Chatbots can also help in collecting customer feedback.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

1. Ajio-Matching Clothes Recommendation:

On selecting a particular item to buy, Ajio automatically suggests a full set of clothes that are matching to the selected item. For example, on selecting a particular t-shirt, the system automatically generates a combination of watches, shoes, pants, etc. that are matching to the selected t-shirt. This system does not take into consideration private qualities of customers like skin color and existing clothes. It will only suggest clothes that already exist in its database.

2. Your Closet:

This is a mobile application that organizes the closet. The application asks customer to input their clothes. It then matches each cloth with other clothes. For example, if there are 4 shirts and 4 pants, the application matches each shirt with each pant and thus provides 16 possibilities. The application does not make matches of clothes depending upon patterns, color and texture of clothes. It also does not have a recommendation system.

3. Magic Closet:

This system aims to retrieve clothes from online stores that are matching to the input clothes. These clothes must be fit to a particular occasion. In this system, the user takes a photo of them specifying if they want to use the top or bottom clothes along with the occasion they want to use it for. The system will search for clothing that matches the user query and satisfies the criterion of wearing aesthetically and wearing properly.

4. Which Clothes to wear confidently?

The basic problem the system addresses is: From the two given images corresponding to a pair of clothes, we have to determine if the pair of clothes matches or not. While there may be several aesthetics espoused by different individuals, it takes a simplistic approach in this problem. An example of shirts and ties is used. Various machine learning method are used to classify if the clothes are matching or not such as Ridge Regression, Standard Neural Network and Siamese Neural Network.

5. Personalized Clothing Recommendation Based on Knowledge Graph:

This system attempts to exploit the knowledge graph for providing clothing recommendations to the user keeping the user context in mind. The recommendation is done by calculating the similarity in the clothing ontology similar to user's collection.

2.2 REFERENCES

1. "A Systematic Study on the Recommender Systems in the E-Commerce"

Electronic commerce or e-commerce includes the service and good exchange through electronic support like the Internet. It plays a crucial role in today's business and users' experience. Also, ecommerce platforms produce a vast amount of information. So, Recommender Systems (RSs) are a solution to overcome the information overload problem. They provide personalized recommendations to improve user satisfaction. The present article illustrates a comprehensive and Systematic Literature Review (SLR) regarding the papers published in the field of e-commerce recommender systems. We reviewed the selected papers to identify the gaps and significant issues of the RSs' traditional methods, which guide the researchers to do future work. So, we provided the traditional techniques, challenges, and open issues concerning traditional methods of the field of review based on the selected papers.

2. Fashion Recommendation Systems :

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.

2.3 PROBLEM STATEMENT DEFINITION

Problem Statement 1:

The User Needs a way to Find Trending Fashion Clothes so that Here find the All Collections.

Problem Statement 2:

The User Needs a way to Find Offers and Discounts so that Here User easy to find Daily Offers.

Problem Statement 3:

The User Needs a way to Assistant for finding Clothes so that Here User got the Chat Bot assistant.

Problem Statement 4:

The Sellers Needs a way to struggling to sells products offline so that Here Sellers will Sell Products via our application.

Problem Statement (PS)

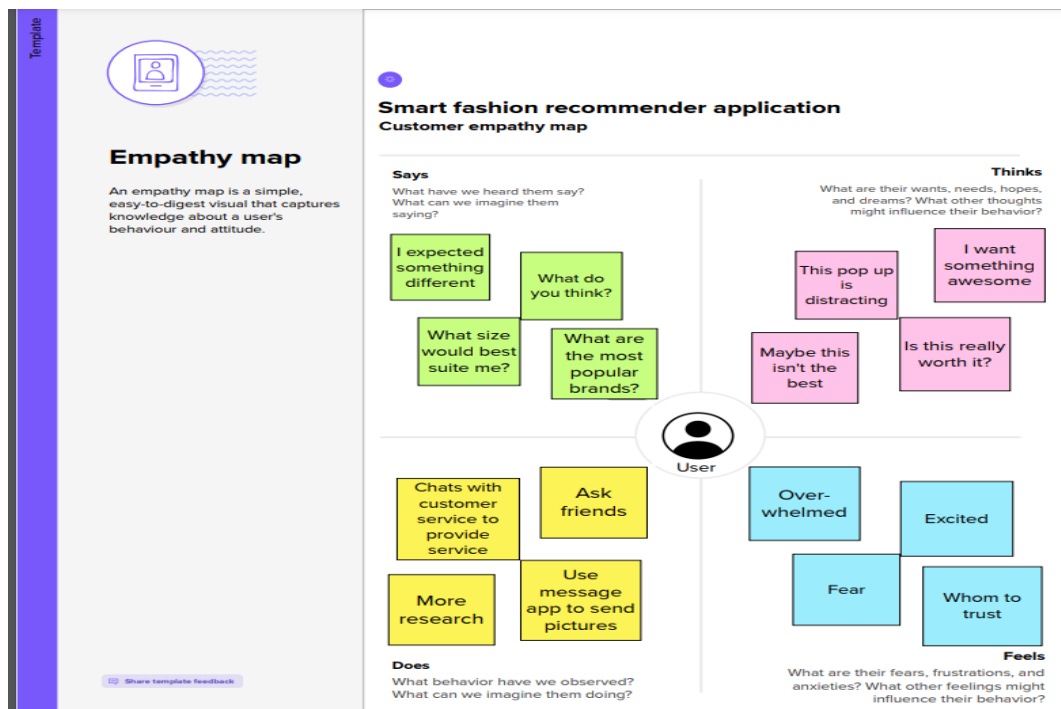


Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Customer	Buy clothes	Unable to buy	Can't find what I wish	Disappointed
PS-2	Software developer	do my best in developing application as customer wish	I can't	Of less time	Guilty

3. IDEATION & PROPOSED SYSTEM

3.1 EMPATHY MAP CANVAS

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to help 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.



3.2 IDEATION & BRAINSTROMING

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions. Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Brainstorm & idea prioritization

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

10 minutes to prepare
1 hour to collaborate
3-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

- Team gathering**
Define who should participate in the session and send an invite. Share relevant information at pre-work ahead.
- Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.
- Learn how to use the Facilitation Guide**
Use the Facilitation Guide to run a happy and productive session.

[Open guide](#)

Define your problem statement

What problem are you trying to solve? Frame your problem as a *How Might We* statement. This will be the focus of your brainstorm.

5 minutes

How might we develop a *How Might We* recommendation application?

Key rules of brainstorming

To run an smooth and productive session:

- Stay in topic
- Encourage wild ideas
- Defer judgment
- Listen to others
- Go for volume
- If possible, be visual

Need some inspiration?

See a finished session and get inspired by the ideas and concepts that others have shared.

[Open examples](#)

Step-2: Brainstorm, Idea Listing and Grouping

Brainstorm

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

10 minutes

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

multiple payment option

brands all over

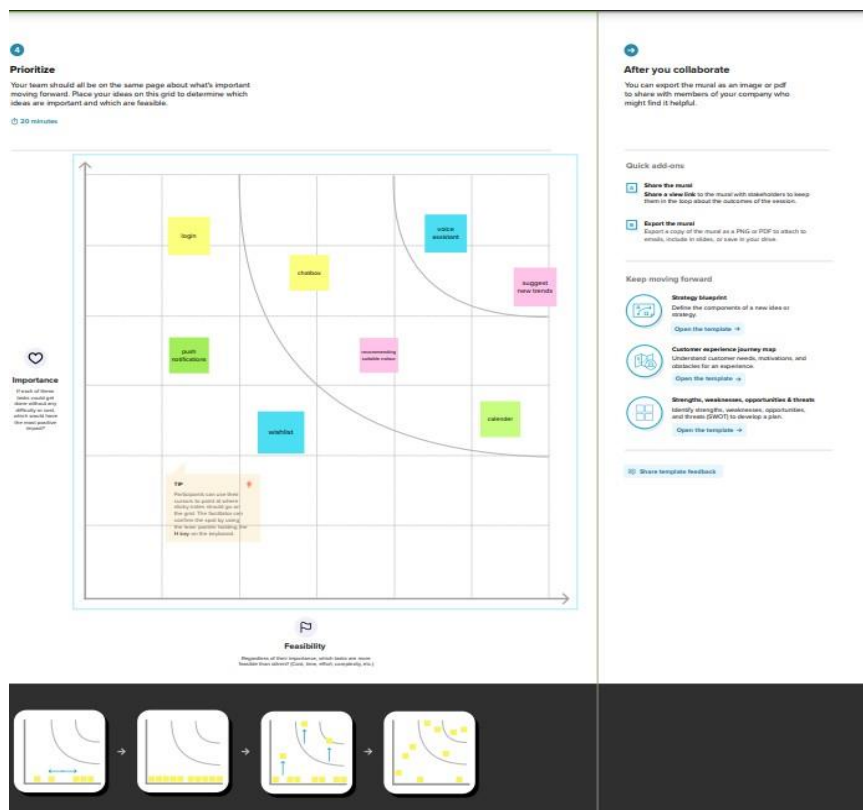
convenience

Need some inspiration?

See a finished session and get inspired by the ideas and concepts that others have shared.

[Open examples](#)

Step-3: Idea Prioritization



3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	<ul style="list-style-type: none"> The smart fashion recommendation application stores the customer's information about the item he/she visited. It perform matching combinations of things according to the customer's needs Instead of using many different apps to keep touch with people, this application handle everything such as communication between the buyer and seller, user queries, product delivery, money transaction, etc. This application further tracks the progress of the order to be delivered.

2.	Idea / Solution description	<ul style="list-style-type: none"> • By using this application, we can recommend clothes based on a person's skin tone, particular season, different occasions, or a festival time. • This recommend matching ties for the shirt selected. • Communication between the buyer and the seller.
3.	Novelty / Uniqueness	<ul style="list-style-type: none"> • The application will recommend dresses and other fashion related things based on the person's age, colour, gender and their personal preferences.
4.	Social Impact / Customer Satisfaction	<ul style="list-style-type: none"> • It helps to maintain with providing clothes based on a person's preference and stay in the current fashion trend for normal people.
5.	Business Model (Revenue Model)	<ul style="list-style-type: none"> • Social media is the best way to develop our application. • Makeup model.
6.	Scalability of the Solution	<ul style="list-style-type: none"> • Good relationship. • Easy access to the application. • User friendly application. • Different clothing combinations are recommended to different aspects of people

3.4 PROBLEM SOLUTION FIT

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) CS <ul style="list-style-type: none"> ✓ Men ✓ Women ✓ Kids ✓ Elderly 	5. CUSTOMER CONSTRAINTS CC <ul style="list-style-type: none"> ✓ Product quality ✓ Hidden cost ✓ Cash budget ✓ Application security ✓ Network connection 	8. AVAILABLE SOLUTIONS AS <ul style="list-style-type: none"> ✓ Multiple contact option ✓ Customer support system ✓ FAQs or help pages ✓ Cash on delivery ✓ Reviews and rating option ✓ Live chat with the seller 	Explore AS, differentiate

Focus on J&P, tap into BE, understand RC	2. JOBS-TO-BE DONE / PROBLEMS J&P <ul style="list-style-type: none"> ✓ Quality issues ✓ Product delivery ✓ Digital payment transaction ✓ Unclear return and warranty policies ✓ Fake products ✓ Lack of physical examination 	6. PROBLEM ROOT CAUSE RC <ul style="list-style-type: none"> ✓ Customers do not get proper and timely information. ✓ Product's price varies for different seller for the same type of product. ✓ Lack of standard weight and measurement. ✓ Delivery of duplicate or incorrect goods. 	9. BEHAVIOUR BE <ul style="list-style-type: none"> ✓ Send email to customer for order confirmation. ✓ Quick to adapt to ensure that the customer have alternate payment methods. ✓ Understand and update the likes of customer. ✓ Social media platform to offer a direct and real time service. 	Focus on J&P, tap into BE, understand RC

Identify strong TR & EM	3. TRIGGERS TR <ul style="list-style-type: none"> ✓ Friends/Family using them. ✓ Recommended in app store. ✓ Ad in other application. ✓ Social media. ✓ Ad while browsing in the web. ✓ Online/ Offline ad. 	7. YOUR SOLUTIONS SL <ul style="list-style-type: none"> ✓ Convenience is a top priority: consumers find convenient is the ability to browse an online store and check out as a guest. ✓ Easy access across all devices: Customers can use various devices such as desktop, a mobile or others without interruption. This multiple device experience is to be expected with personal details on the device the customers finalize their buying. ✓ Personalization: By analyzing the users' history, online retailers can offer products and services that a customer is more likely to be interested in 	10. CHANNELS OF BEHAVIOUR CH <ul style="list-style-type: none"> ✓ ONLINE: <ul style="list-style-type: none"> • The customer can also compare prices with different stores. • Limited product quality standard. • Rely on the product details and reviews available on the site. • Manufacture and expiry date. • Continuous shopping. ✓ OFFLINE: <ul style="list-style-type: none"> • Check the goods personally and there is no scope of one getting disappointed with the quality of the product. • Instant use and no waiting period. • Lack of choices as physical stores have a limitation when it comes to the variety of product choices. 	Identify strong TR & EM
	4. EMOTIONS: BEFORE/ AFTER EM <ul style="list-style-type: none"> ✓ Achievement: People like the feeling of accomplishment. Personal status is a primary motivator for a purchase. ✓ Power: People have a natural desire to be better than others and outperform rivals. ✓ Fear: It refers to missing out on a good opportunity. ✓ Happiness: Customers love buying easy-to-use products that create pleasure in their lives and the sellers likes profit. 			

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIn
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Overall description	The description document consists of product vision, business rules. The requirements might be database requirements, system attributes, and functional requirements.
FR-4	Search and select items	The 'search' feature enable customer to find the specific model, branch, or item.
FR-5	Order item and paying bills	Online customers must have access to the Internet and a valid <u>method of payment</u> in order to complete a transaction.
FR-6	Chat box	Real time chat feature allows customers to chat with the system or the seller.
FR-7	Cart	It allow the consumer to accumulate multiple items and to adjust quantities.

4.2 NON-FUNCTIONAL REQUIREMENTS:

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

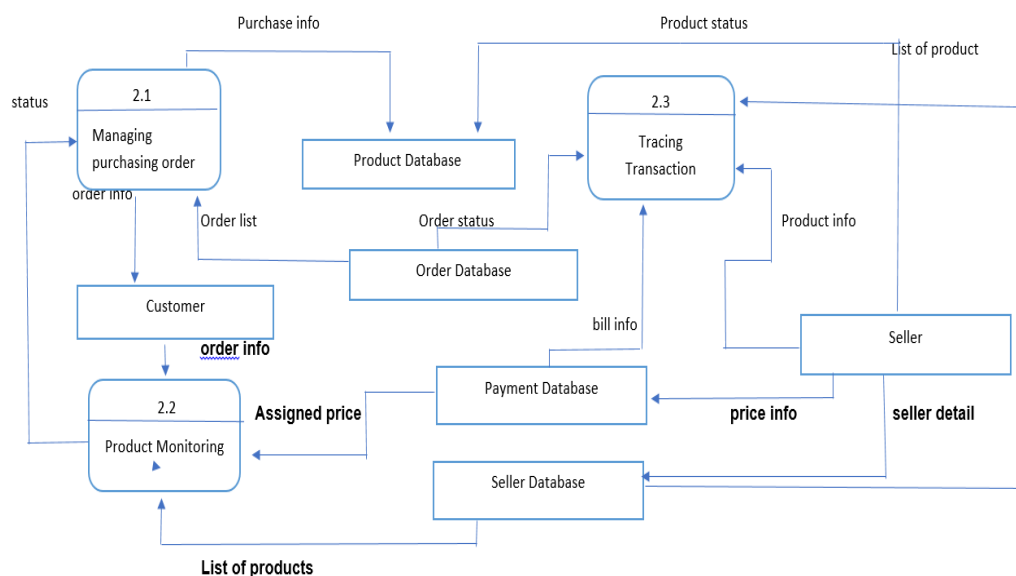
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	It is the degree of ease with which the user will interact with your products to achieve required goals effectively and efficiently.
NFR-2	Security	Security is paramount while dealing with monetary transactions and sensitive data.
NFR-3	Reliability	To achieve high reliability, your team should eliminate all bugs that may influence the code safety and issues with system components.
NFR-4	Performance	Performance describes how your solution behaves when users interact with it in various scenarios.
NFR-5	Availability	The features need to be available to all users, at any time, and from anywhere.
NFR-6	Scalability	This requirement defines how the website can grow and expand its functionality without affecting its performance.

5. PROJECT DESIGN

Project design is an early phase of the project lifecycle where ideas, processes, resources, and deliverables are planned out. A project design comes before a project plan as it's a broad overview whereas a project plan includes more detailed information. There are seven steps involved when creating a project design, including defining goals and using a visual aid to communicate objectives. These visual elements include a variety of methods such as Gantt charts, Kanban boards, and flowcharts. Providing a visual representation of your project strategy can help create transparency between stakeholders and clarify different aspects of the project, including its overall feasibility.

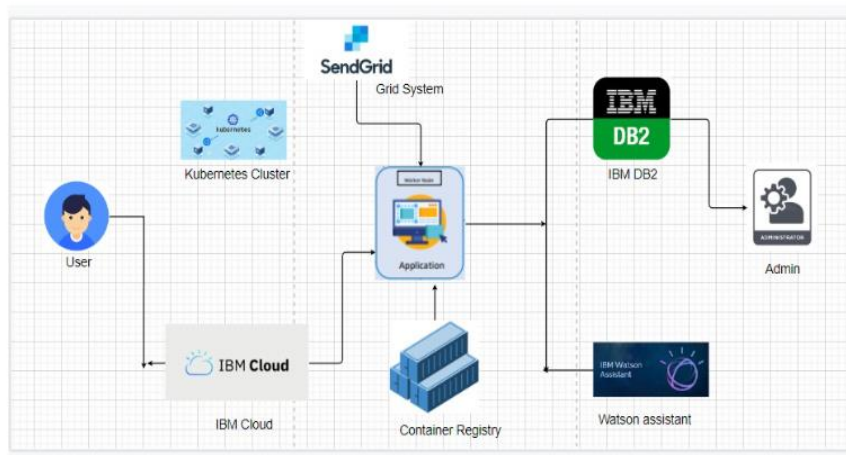
5.1 DATA FLOW DIAGRAMS

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



5.2 SOLUTION & TECHNICAL ARCHITECTURE

SOLUTION ARCHITECTURE



s.no	Components	Description	Technologies
1	User Interface	user interacts with application e.g. Web UI, Mobile App, Chatbot etc	HTML, CSS, JavaScript / Angular Js / React Js etc
2	Requests	How web application communicates with server	Python
3	Application Logic-2	The application includes login where user can login with their credentials and also supports registration where newusers can be added	Python
4	Watson chatbot	The application includes a chatbot which helps the user in recommendation of products.	IBM Watson Assistant
5	IBM cloud DB2	Details of customers and products are stored. Data types are String, Numeric, Date, time, and timestamp distinct types. Act_ sortmem_ limit, auto_ del_ rec_ obj, auto_ maintConfiguration	MySQL, NoSQL, etc.
6	Cloud Db2	A fully managed cloud database with AI capabilities that keep our website	IBM DB2, IBM Cloudant etc

		running 24*7	
7	Kubernetes	Manage the complete process in the stable state If any software crash it automatically restart the work	Kubernetes
8	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration	Local, Cloud Foundry, Kubernetes, etc

Application Characteristics:

S.No	Characteristics	Description	Technology
1	Open-Source Frameworks	Cloud Stack, Eucalyptus. Open Nebula, App Scale, Docker	Docker
2	Security Implementations	Authentication and password management Accountability to authorize and monitor the use anonymous accounts and to remove	Encryptions, Secured Authorization
3	Scalable Architecture	Handles large number users on demand	Container registry, Kubernetes
4	Availability	The application can be accessed at any time. The administrator needs to look up the stock availability in the database	Docker
5	Performance	Speed up the webpage Site optimization based on data analysis	Kubernetes

6. PROJECT PLANNING & SCHEDULING

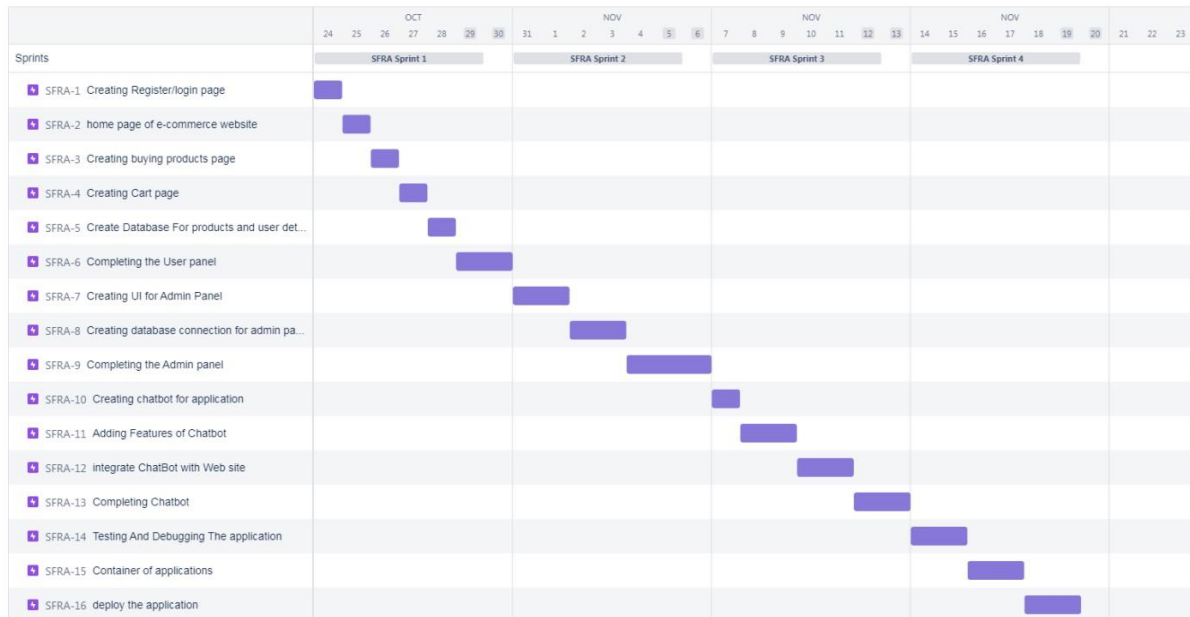
6.1 SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirements	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 available products 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 recommendation based on information provided by the user.	20	HIGH
Sprint– 4	Final Delivery	USN – 4	Container of application using docker kubernetes and deployment of 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	Sprint Release Date(Actual)
Sprint - 1	20	6 days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint – 2	20	6 days	31 Nov 2022	05 Nov 2022	20	05 Nov 2022
Sprint – 3	20	6 days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint - 4	20	6 days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Report from JIRA



Velocity :

Imagine we have a 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)

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

7. CODING & SOLUTIONING

App.py

```

import secrets
from turtle import title
from unicodedata import category
from flask import Flask, render_template, request, redirect, url_for, session
import ibm_db
import bcrypt
import base64
from PIL import Image
import io

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=55fbc997-9266-4331-afd3-
888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31929;SECURIT
Y=SSL;
SSLServerCertificate=DigiCertGlobalRootCA.crt;PROTOCOL=TCPIP;UID=lgs66069;PWD
=xLWZLkQcdBY8onuA;", "", "")
#url_for('static', filename='style.css')

app = Flask(__name__,template_folder = 'template')
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'

@app.route("/",methods=['GET'])
def home():
    if 'email' not in session:
        return redirect(url_for('index'))
    return render_template('home.html',name='Home')
@app.route("/index")
def index():
    return render_template('index.html')
@app.route("/register",methods=['GET','POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        phoneno = request.form['phoneno']

```

```

password = request.form['password']
if not username or not email or not phoneno or not password:
    return render_template('register.html',error='Please fill all fields')
hash=bcrypt.hashpw(password.encode('utf-8'),bcrypt.gensalt())
query = "SELECT * FROM user_detail WHERE email=? OR phoneno=?"
stmt = ibm_db.prepare(conn, query)
ibm_db.bind_param(stmt,1,email)
ibm_db.bind_param(stmt,2,phoneno)
ibm_db.execute(stmt)
isUser = ibm_db.fetch_assoc(stmt)
if not isUser:
    insert_sql = "INSERT INTO user_detail(username, email, phoneno, password) VALUES
(?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, username)
    ibm_db.bind_param(prep_stmt, 2, email)
    ibm_db.bind_param(prep_stmt, 3, phoneno)
    ibm_db.bind_param(prep_stmt, 4, hash)
    ibm_db.execute(prep_stmt)
    return render_template('register.html',success="You can login")
else:
    return render_template('register.html',error='Invalid Credentials')
return render_template('register.html',name='Home')
@app.route("/login",methods=['GET','POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        if not email or not password:
            return render_template('login.html',error='Please fill all fields')
        query = "SELECT * FROM user_detail WHERE email=?"
        stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.execute(stmt)
        isUser = ibm_db.fetch_assoc(stmt)

```

```

print(isUser,password)
if not isUser:
    return render_template('login.html',error='Invalid Credentials')
isPasswordMatch = bcrypt.checkpw(password.encode('utf-8'),isUser['PASSWORD'].encode('utf-8'))
if not isPasswordMatch:
    return render_template('login.html',error='Invalid Credentials')
session['email'] = isUser['EMAIL']
return redirect(url_for('home'))
return render_template('login.html',name='Home')
@app.route("/admin",methods=['GET','POST'])
def adregister():
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        phoneno = request.form['phoneno']
        password = request.form['password']
        if not username or not email or not phoneno or not password:
            return render_template('adminregister.html',error='Please fill all fields')
        hash=bcrypt.hashpw(password.encode('utf-8'),bcrypt.gensalt())
        query = "SELECT * FROM admin_detail WHERE email=? OR phoneno=?"
        stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.bind_param(stmt,2,phoneno)
        ibm_db.execute(stmt)
        isUser = ibm_db.fetch_assoc(stmt)
        if not isUser:
            insert_sql = "INSERT INTO admin_detail(username, email, phoneno, password)
VALUES (?, ?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, username)
            ibm_db.bind_param(prep_stmt, 2, email)
            ibm_db.bind_param(prep_stmt, 3, phoneno)
            ibm_db.bind_param(prep_stmt, 4, hash)
            ibm_db.execute(prep_stmt)

```

```

    return render_template('adminregister.html',success="You can login")
else:
    return render_template('adminregister.html',error='Invalid Credentials')
return render_template('adminregister.html',name='Home')
@app.route("/adminlogin",methods=['GET','POST'])
def adlogin():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        if not email or not password:
            return render_template('adminlogin.html',error='Please fill all fields')
        query = "SELECT * FROM admin_detail WHERE email=?"
        stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.execute(stmt)
        isUser = ibm_db.fetch_assoc(stmt)
        print(isUser,password)
        if not isUser:
            return render_template('adminlogin.html',error='Invalid Credentials')
        isPasswordMatch = bcrypt.checkpw(password.encode('utf-8'),isUser['PASSWORD'].encode('utf-8'))
        if not isPasswordMatch:
            return render_template('adminlogin.html',error='Invalid Credentials')
        session['email'] = isUser['EMAIL']
        return redirect(url_for('home'))

    return render_template('adminlogin.html',name='Home')
@app.route("/addproduct",methods=['GET','POST'])
def addproduct():
    if request.method == 'POST':
        types=request.form['cc']
        name = request.form['name']
        image = request.form['image']
        rate = request.form['rate']
        categorie = request.form['categorie']

```

```

if types == 'shirt':
    insert_sql = "INSERT INTO SHIRT(name, image, categorie,rate) VALUES (?, ?, ?, ?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prepare_stmt, 1, name)
    ibm_db.bind_param(prepare_stmt, 2, image)
    ibm_db.bind_param(prepare_stmt, 3, categorie)
    ibm_db.bind_param(prepare_stmt, 4, rate)
    ibm_db.execute(prepare_stmt)
if types == 'pant':
    insert_sql = "INSERT INTO PANT(name, image, rate) VALUES (?, ?, ?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prepare_stmt, 1, name)
    ibm_db.bind_param(prepare_stmt, 2, image)
    ibm_db.bind_param(prepare_stmt, 3, rate)
    ibm_db.execute(prepare_stmt)
if types == 'watch':
    insert_sql = "INSERT INTO WATCH(name, image, rate) VALUES (?, ?, ?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prepare_stmt, 1, name)
    ibm_db.bind_param(prepare_stmt, 2, image)
    ibm_db.bind_param(prepare_stmt, 3, rate)
    ibm_db.execute(prepare_stmt)
if types == 'ring':
    insert_sql = "INSERT INTO RINGS(name, image, categorie,rate) VALUES (?, ?, ?, ?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prepare_stmt, 1, name)
    ibm_db.bind_param(prepare_stmt, 2, image)
    ibm_db.bind_param(prepare_stmt, 3, categorie)
    ibm_db.bind_param(prepare_stmt, 4, rate)
    ibm_db.execute(prepare_stmt)

return render_template('addproduct.html', success="You can login")

@app.route("/data")
def display():

```



```
shirt_list=[]
pant_list=[]
watch_list=[]
ring_list=[]
#selecting_shirt
sql = "SELECT * FROM SHIRT"
stmt = ibm_db.exec_immediate(conn, sql)
shirt = ibm_db.fetch_both(stmt)
while shirt != False :
    shirt_list.append(shirt)
    shirt = ibm_db.fetch_both(stmt)
print(shirt_list)
#selecting_pant
sql1="SELECT * FROM PANT"
stmt1 = ibm_db.exec_immediate(conn, sql1)
pant=ibm_db.fetch_both(stmt1)
while pant != False :
    pant_list.append(pant)
    pant = ibm_db.fetch_both(stmt1)
print(pant_list)
#selecting_watch
sql2="SELECT * FROM WATCH"
stmt2 = ibm_db.exec_immediate(conn, sql2)
watch=ibm_db.fetch_both(stmt2)
while watch != False :
    watch_list.append(watch)
    watch = ibm_db.fetch_both(stmt2)
print(watch_list)
#selecting_rings
sql3="SELECT * FROM RINGS"
stmt3 = ibm_db.exec_immediate(conn, sql3)
ring=ibm_db.fetch_both(stmt3)
while ring != False :
    ring_list.append(ring)
    ring = ibm_db.fetch_both(stmt3)
```

```
print(ring_list)
#returning to HTML
return render_template('home.html',dictionary=
shirt_list,pants=pant_list,watches=watch_list,rings=ring_list)
@app.route("/home")
def dis():
    watch_list=[]
    sql2="SELECT * FROM WATCH"
    stmt2 = ibm_db.exec_immediate(conn, sql2)
    watch=ibm_db.fetch_both(stmt2)
    while watch != False :
        watch_list.append(watch)
        watch = ibm_db.fetch_both(stmt2)
    print(watch_list)
    return render_template('home.html',watches=watch_list)
@app.route('/logout')
def logout():
    session.pop('email', None)
    return redirect(url_for('login'))
if __name__ == "__main__":
    app.run(debug=True)
```

7.2 DATABASE SCHEMES

The screenshot displays the IBM Db2 on Cloud console interface. The top navigation bar includes tabs for 'Load Data', 'Load History', 'Tables', 'Views', 'Indexes', 'Aliases', 'MQTs', 'Sequences', and 'Application objects'. The 'Tables' tab is currently selected. Below the navigation bar, there is a search bar labeled 'Find schemas or tables' and a 'Refresh' button. The main content area is divided into two panels: 'Schemas' on the left and 'Tables' on the right.

Schemas Panel:

Name	Type	Tables
LGS66069	User	7

Total: 1, selected: 1

Tables Panel:

Name	Schema	Properties
ADMIN_DETAIL	LGS66069	...
PANT	LGS66069	...
SHIRT	LGS66069	...
SHOE	LGS66069	...
SHORTS	LGS66069	...
USER_DETAIL	LGS66069	...
WATCH	LGS66069	...

Total: 7, selected: 0

The bottom of the screenshot shows a Windows taskbar with various application icons, a search bar, and system information including temperature (28°C), time (10:34 PM), and date (26-11-2022).

8. TESTING

8.1 TEST CASES

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

Section	Total Cases	Not Tested	Fail	Pass
Login	5	0	0	5
Registry	5	0	0	5
Home Page	2	0	0	2
Order Page	2	0	0	5
Order Products	5	0	0	5
Final Report Output	2	0	0	2

8.2 USER ACCEPTANCE TESTING

Purpose of Document

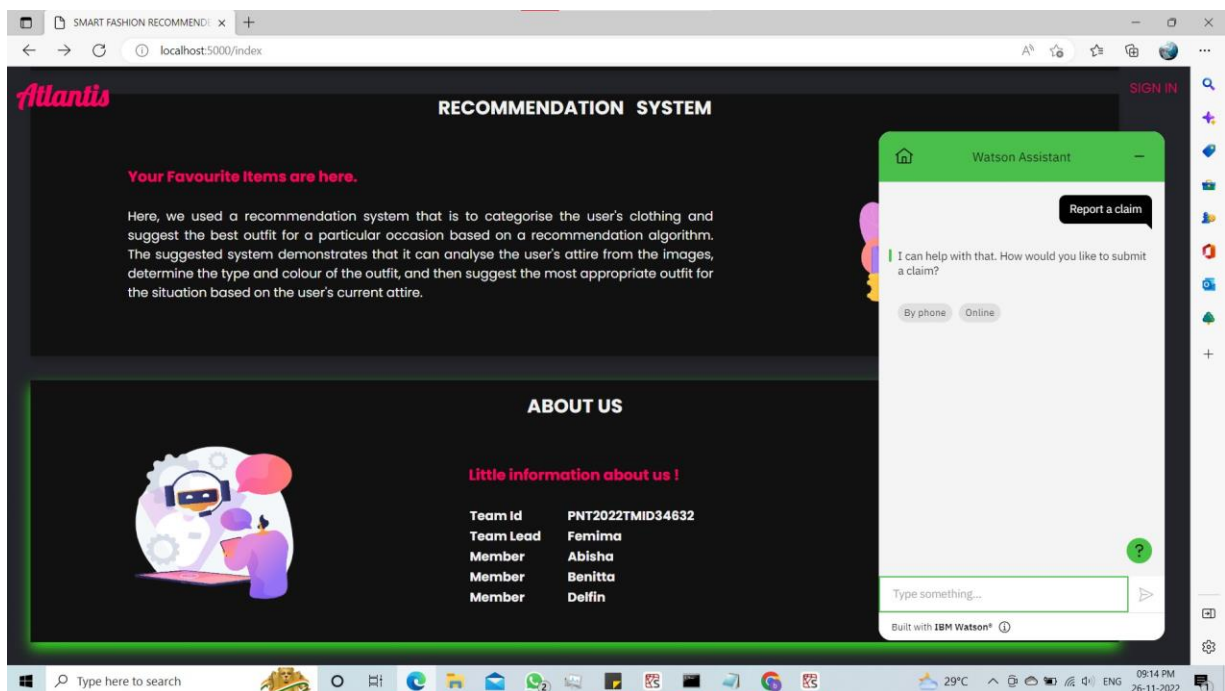
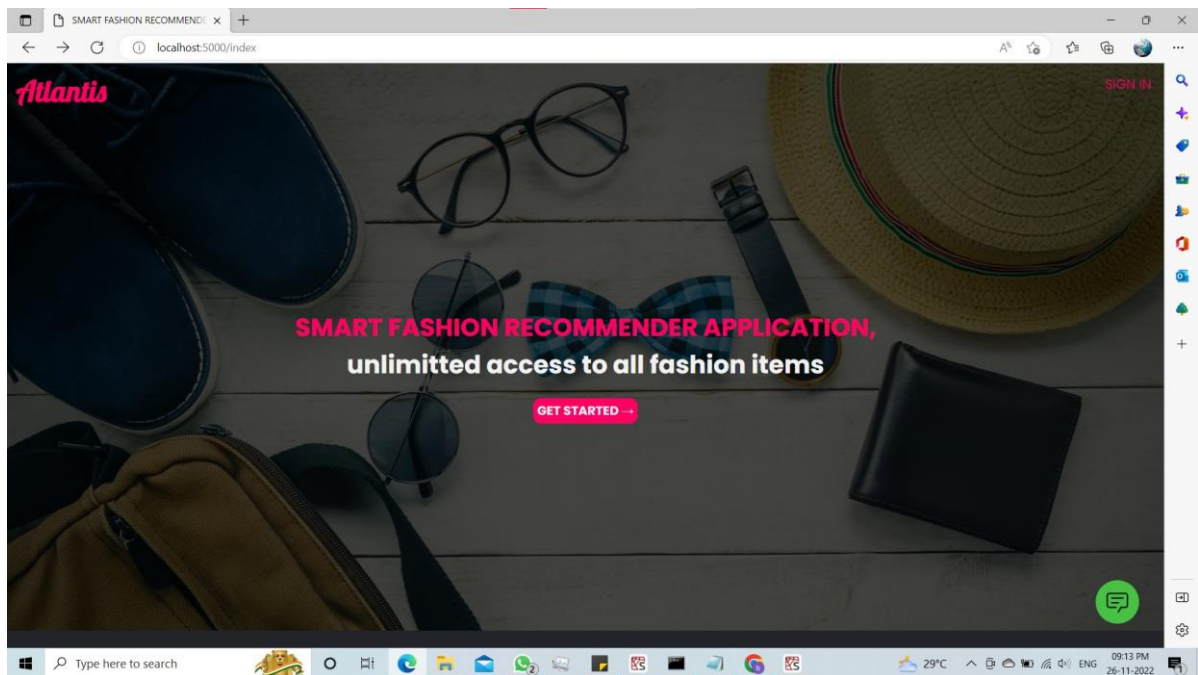
The purpose of this document is to briefly explain the test coverage and open issues of the Smart Fashion Recommender Application project at the time of the release to User Acceptance Testing (UAT).

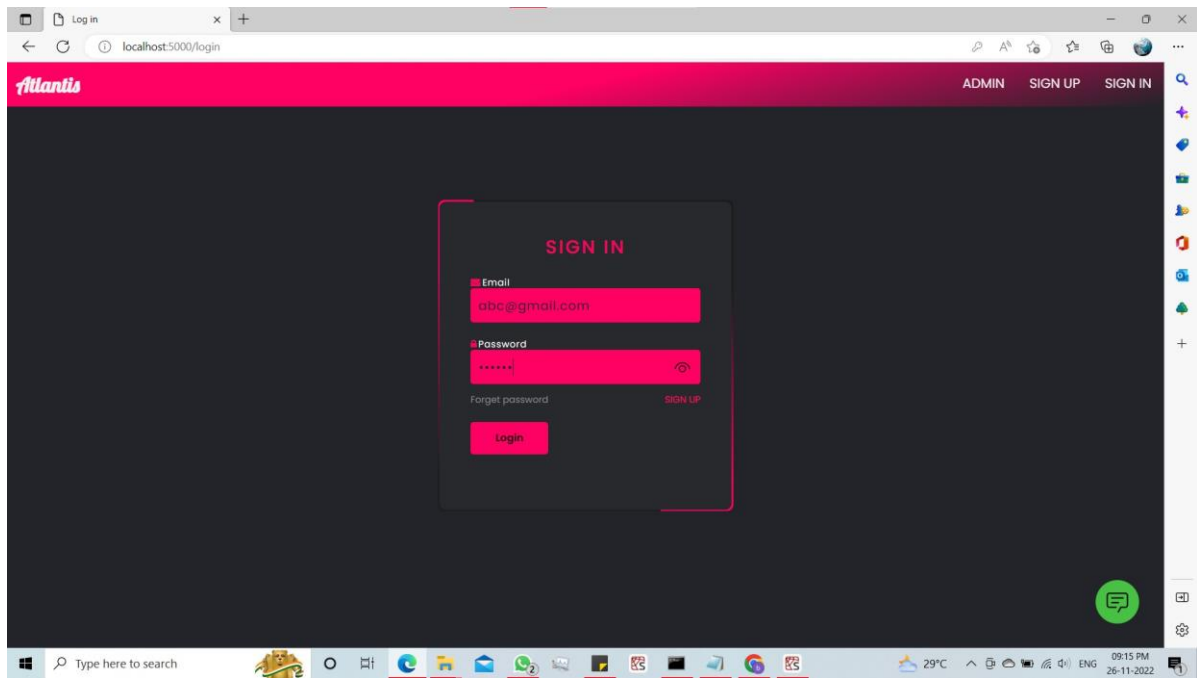
Defect Analysis

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	5	2	3	15
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	10	2	4	15	31
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't fix	0	5	2	1	8
Totals	18	15	13	21	67

9. RESULTS





The screenshot shows a web browser window with the URL `localhost:5000/login`. The page has a dark blue header with the "Atlantis" logo on the left and navigation links "ADMIN", "SIGN UP", and "SIGN IN" on the right. The main content area is dark blue and features a white "SIGN IN" form. The form includes an "Email" field with the value "abc@gmail.com", a "Password" field with masked characters "*****", a "Forgot password" link, a "SIGN UP" link, and a "Login" button. A green chat bubble icon is located in the bottom right corner of the main content area. The Windows taskbar at the bottom shows the search bar, task view button, and several application icons, including Edge, File Explorer, and various utility tools. The system tray on the right displays the temperature as 29°C, the time as 09:15 PM, and the date as 26-11-2022.

Log in

localhost:5000/login

ADMIN SIGN UP SIGN IN

Atlantis

SIGN IN

Email
abc@gmail.com

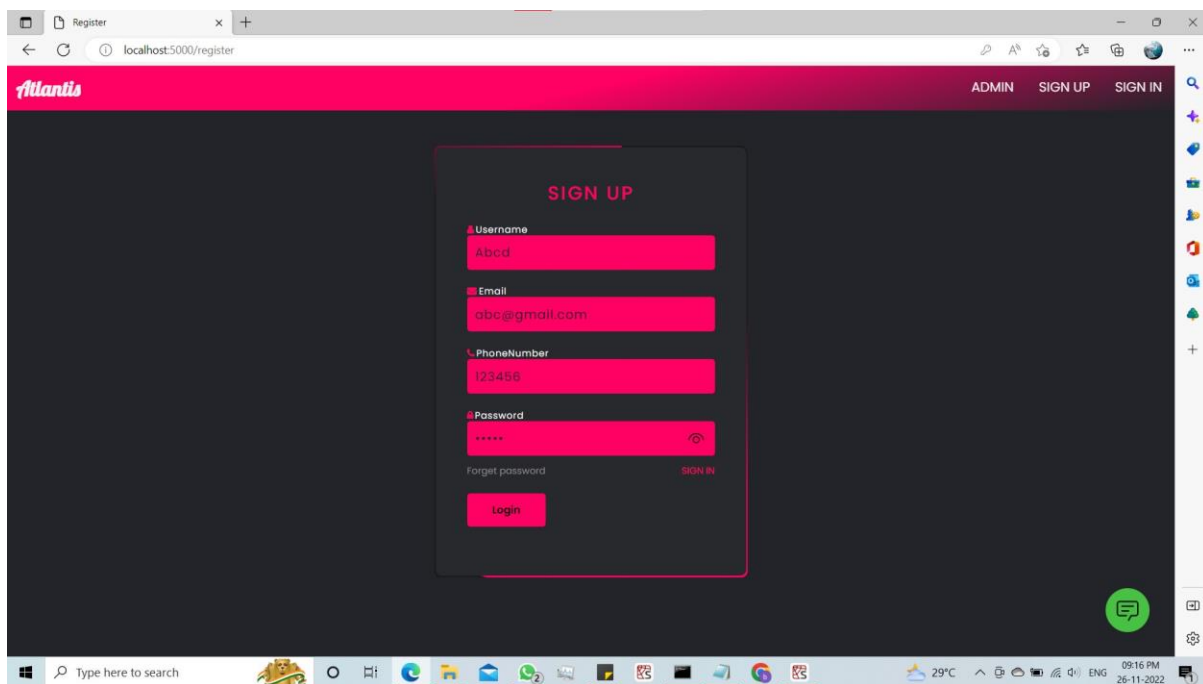
Password

Forgot password SIGN UP

Login

Type here to search

29°C 09:15 PM 26-11-2022



The screenshot shows a web browser window with the URL `localhost:5000/register`. The page has a dark blue header with the "Atlantis" logo on the left and navigation links "ADMIN", "SIGN UP", and "SIGN IN" on the right. The main content area is dark blue and features a white "SIGN UP" form. The form includes a "Username" field with the value "Abcd", an "Email" field with the value "abc@gmail.com", a "PhoneNumber" field with the value "123456", a "Password" field with masked characters "*****", a "Forgot password" link, a "SIGN IN" link, and a "Login" button. A green chat bubble icon is located in the bottom right corner of the main content area. The Windows taskbar at the bottom shows the search bar, task view button, and several application icons, including Edge, File Explorer, and various utility tools. The system tray on the right displays the temperature as 29°C, the time as 09:16 PM, and the date as 26-11-2022.

Register

localhost:5000/register

ADMIN SIGN UP SIGN IN

Atlantis

SIGN UP

Username
Abcd

Email
abc@gmail.com

PhoneNumber
123456

Password

Forgot password SIGN IN


Login

Type here to search


29°C 09:16 PM 26-11-2022

Shop Orders


Welcome, ashwin!




YSL Jacket
Black saint laurant jacket 2020
\$299
Quantity




Crop Top
For new generation fashionista
\$69
Quantity



Watercolor Shirt
summercool
\$500
Quantity



Pink Hoodie
asdfsaf
\$32332
Quantity

Type here to search

28°C
09:24 PM
26-11-2022

Shop Orders

Elite Fashion - Orders

Order No.	Product Id	Product Name	Customer Id	Ordered By	Quantity	User Contact	Delivery Address	Action
12	1	YSL Jacket	4	ashwin	1	9362688687	tiruppur	<input type="button" value="Cancel Order"/>

9.1 PERFORMANCE METRICS

The performance of a recommendation algorithm is evaluated by using some specific metrics that indicate the accuracy of the system. The type of metric used depends on the type of filtering technique. Root Mean Square Error (RMSE), Receiver Operating Characteristics (ROC), Area Under Cover (AUC), Precision, Recall and F1 score is generally used to evaluate the performance or accuracy of the recommendation algorithms.

1. Hours worked : 50 hours
2. Sticking to Timelines : 100%
3. Consistency of the product : 75%
4. Efficiency of the product : 80%
5. Quality of the product : 85%

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

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

DISADVANTAGES:

Smart Fashion recommendation technology has been the most successful recommendation technology so far, but there are two major problems—recommendation quality and scalability. At present, research at home and abroad mainly focuses on recommendation quality, and there is less discussion on scalability. The scalability problem is that as the size of the system increases, the response time of the system increases to a point where users cannot afford it. Existing solutions often result in a significant drop in recommendation quality while reducing recommendation response time. In this paper, the clustering analysis subsystem based on the genetic algorithm is innovatively introduced into the traditional collaborative filtering recommendation system, and its design and implementation are given.

11. CONCLUSION

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 pre-processing and cleaning up the data. We took into account features like patterns, machine, fabric, style etc. After extensive pre-processing 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.

12. FUTURE SCOPE

There has been significant progress recently in fashion recommendation system research, which will benefit both consumers and retailers soon. The use of product and user images, textual content, demographic history, and cultural information is crucial in developing recommendation frameworks. Product attributes and clothing style matching are common features of collaborative and content-based filtering techniques. Researchers can develop more sophisticated hyper personalized filtering techniques considering the correlation between consumers' clothing styles and personalities. The methods based on employing a scoring system for quantifying each product attribute will be helpful in increasing the precision of the model. The use of virtual sales advisers in an online shopping portal would provide consumers with a real time offline shopping experience. Retailers can collect the data on users' purchase history and product reviews from the recommendation system and subsequently use them in style prediction for the upcoming seasons. The integration of different domain information strengthens the deep learning paradigm by enabling the detection of design component variation, which improves the performance of the recommendation system in the long run. Deep learning approaches should be more frequently used to quickly explore fashion items from different online databases to provide prompt recommendations to users or consumers.

13. APPENDIX

13.1 SOURCE CODE

Adminlogin.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Admin Login</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
<script>
  window.watsonAssistantChatOptions = {
    integrationID: "b6892981-e498-43d3-a1fb-36c1a5752cec", // The ID of this integration.
    region: "au-syd", // The region your integration is hosted in.
    serviceInstanceID: "eaca97b3-64b0-4e47-854f-8314f7e3e0ec", // The ID of your service
instance.
    onLoad: function(instance) { instance.render(); }
  };
  setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
  });
</script>
<style>
```

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&displa
y=swap');

@import url('https://fonts.googleapis.com/css2?family=Lobster&display=swap');

*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
}

body{
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  background-color: #23242a;
}

.box{
  position: relative;
  width: 380px;
  height: 400px;
  background-color: #1c1c1c;
  border-radius: 8px;
  overflow: hidden;
}

.box::before{
  content: "";
  position: absolute;
  top: -50%;
  left: -50%;
  width: 380px;
```

```
height: 420px;
background: linear-gradient(0deg, transparent,#FF0063,#FF0063);
transform-origin: bottom right;
animation: animate 6s linear infinite;
}
.box::after{
  content: "";
  position: absolute;
  top: -50%;
  left: -50%;
  width: 380px;
  height: 420px;
  background: linear-gradient(0deg, transparent,#FF0063,#FF0063);
  transform-origin: bottom right;
  animation: animate 6s linear infinite;
  animation-delay: -3s;
}
@keyframes animate {
  0%{
    transform: rotate(0deg);
  }
  100%{
    transform: rotate(350deg);
  }
}
/* Form */
.form{
  position: absolute;
  inset: 2px;
  border-radius: 8px;
  background: #28292d;
  z-index: 10;
```

```
padding: 40px 40px;
display: flex;
flex-direction: column;
}
.form h2{
color: #FF0063;
font-weight: 500;
text-align: center;
letter-spacing: 0.1em;
}
.inputBox{
position: relative;
width: 100%;
margin-top: 35px;
}
.inputBox input{
position: relative;
width: 100%;
padding: 10px 10px 10px;
background: transparent;
border: none;
outline: none;
color: #23242a;
font-size: 1em;
letter-spacing: 0.05em;
z-index: 10;
}
.inputBox span{
position: absolute;
left: 0;
padding: 20px 0px 10px;
font-size: 1em;
```

```
    color: white;
    pointer-events: none;
    letter-spacing: 0.05em;
    transition: 0.5s;
}
.inputBox input:valid ~ span,
.inputBox input:focus ~ span{
    color: #FF0063;
    transform: translateX(0px) translateY(-34px);
    font-size: 0.75em;
}
.inputBox i{
    position: absolute;
    left: 0;
    bottom: 0;
    width: 100%;
    height: 2px;
    background: #FF0063;
    border-radius: 4px;
    transition: 0.5s;
    pointer-events: none;
    z-index: 0;
}
.inputBox input:valid ~ i,
.inputBox input:focus ~ i{
    height: 44px;
}
.links{
    display: flex;
    justify-content: space-between;
}
.links a {
```



```
margin: 10px 0;
font-size: 0.75em;
color: #8f8f8f;
text-decoration: none;
}
.links a:hover,
.links a:nth-child(2){
    color: #FF0063;
}
input[type="submit"]{
    border: none;
    outline: none;
    background: #FF0063;
    padding: 11px 25px;
    width: 100px;
    margin-top: 10px;
    border-radius: 4px;
    font-weight: 600;
    cursor: pointer;
}
input[type="submit"]:active{
    opacity: 0.8;
}
.font{
    font-family: 'Poppins', sans-serif;
    top: 0;
    margin-left: 120%;
    text-align: left;
}
.fa{
    color: aliceblue;
}
```

```
.navbar {
  overflow: hidden;
  background: linear-gradient(-13deg, transparent, #FF0063, #FF0063);
  position: fixed;
  top: 0;
  width: 100%;
  font-size: 3vh;
}

.navbar a {
  float: right;
  display: block;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
}

.navbar a:hover {
  background: #ddd;
  color: black;
}

.title{
  padding: 12px 12px;
  color: #f2f2f2;
}

</style>
</head>
<body>
  <div class="navbar">
    <a href="/login">SIGN IN </a>
    <a href="/register">SIGN UP</a>
    <a href="/adminlogin">ADMIN</a>
```

```

<div class="title">
  <h3 style="right:0;font-family:'Lobster', cursive;">Atlantis</h3>
</div>
</div>
<form method="post">
  <div class="box">
    <div class="form">
      <h2>SIGN IN</h2>
      <div class="inputBox" >
        <input type="text" name="email" required>
        <span class="fa fa-envelope"> <span class="font">Email</span></span>
        <i></i>
      </div>
      <div class="inputBox">
        <input type="password" name="password" required>
        <span class="fa fa-lock"> <span class="font">Password</span></span>
        <i></i>
      </div>
      <p>{{success}}</p>
      <p style="color: red">{{error}}</p>
      <div class="links">
        <a href="#">Forget password</a>
        <a href="/admin">SIGN UP</a>
      </div>
      <input type="submit" value="Login">
    </div>
  </div>
</form>
</body>
</html>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Admin Register</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
  <link rel="stylesheet" href="{{url_for('static', filename='styles/style.css')}}">
<script>
  window.watsonAssistantChatOptions = {
    integrationID: "b6892981-e498-43d3-a1fb-36c1a5752cec", // The ID of this integration.
    region: "au-syd", // The region your integration is hosted in.
    serviceInstanceID: "eaca97b3-64b0-4e47-854f-8314f7e3e0ec", // The ID of your service
instance.
    onLoad: function(instance) { instance.render(); }
  };
  setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
  });
</script>
<style>
  @import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&displa
y=swap');
  @import url('https://fonts.googleapis.com/css2?family=Lobster&display=swap');
*{

```

```
margin: 0;
padding: 0;
box-sizing: border-box;
font-family: 'Poppins', sans-serif;
}
body{
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  background-color: #23242a;
}
.box{
  position: relative;
  width: 380px;
  height: 570px;
  background-color: #1c1c1c;
  border-radius: 8px;
  overflow: hidden;
}
.box::before{
  content: "";
  position: absolute;
  top: -50%;
  left: -50%;
  width: 380px;
  height: 420px;
  background: linear-gradient(0deg, transparent,#45ff55,#45ff55);
  transform-origin: bottom right;
  animation: animate 6s linear infinite;
}
.box::after{
```

```
content: "";
position: absolute;
top: -50%;
left: -50%;
width: 380px;
height: 420px;
background: linear-gradient(0deg, transparent,#45ff55,#45ff55);
transform-origin: bottom right;
animation: animate 6s linear infinite;
animation-delay: -3s;
}
@keyframes animate {
  0%{
    transform: rotate(0deg);
  }
  100%{
    transform: rotate(350deg);
  }
}
/* Form */
.form{
  position: absolute;
  inset: 2px;
  border-radius: 8px;
  background: #28292d;
  z-index: 10;
  padding: 40px 40px;
  display: flex;
  flex-direction: column;
}
.form h2{
  color: #45ff55;
```

```
    font-weight: 500;
    text-align: center;
    letter-spacing: 0.1em;
}

.inputBox{
    position: relative;
    width: 100%;
    margin-top: 35px;
}

.inputBox input{
    position: relative;
    width: 100%;
    padding: 10px 10px 10px;
    background: transparent;
    border: none;
    outline: none;
    color: #23242a;
    font-size: 1em;
    letter-spacing: 0.05em;
    z-index: 10;
}

.inputBox span{
    position: absolute;
    left: 0;
    padding: 20px 0px 10px;
    font-size: 1em;
    color: white;
    pointer-events: none;
    letter-spacing: 0.05em;
    transition: 0.5s;
}

.inputBox input:valid ~ span,
```

```
.inputBox input:focus ~ span{
  color: #45ff55;
  transform: translateX(0px) translateY(-34px);
  font-size: 0.75em;
}

.inputBox i{
  position: absolute;
  left: 0;
  bottom: 0;
  width: 100%;
  height: 2px;
  background: #45ff55;
  border-radius: 4px;
  transition: 0.5s;
  pointer-events: none;
  z-index: 0;
}

.inputBox input:valid ~ i,
.inputBox input:focus ~ i{
  height: 44px;
}

.links{
  display: flex;
  justify-content: space-between;
}

.links a {
  margin: 10px 0;
  font-size: 0.75em;
  color: #8f8f8f;
  text-decoration: none;
}

.links a:hover,
```



```
.links a:nth-child(2){
    color: #45ff55;
}
input[type="submit"]{
    border: none;
    outline: none;
    background: #45ff55;
    padding: 11px 25px;
    width: 100px;
    margin-top: 10px;
    border-radius: 4px;
    font-weight: 600;
    cursor: pointer;
}
input[type="submit"]:active{
    opacity: 0.8;
}
.font{
    font-family: 'Poppins', sans-serif;
    top: 0;
    margin-left: 120%;
    text-align: left;
}
.fa{
    color: aliceblue;
}
.navbar {
    overflow: hidden;
    background: linear-gradient(-13deg, transparent,#45ff55,#21e231);
    position: fixed;
    top: 0;
    width: 100%;
```

```
font-size: 3vh;
}
.navbar a {
float: right;
display: block;
color: #f2f2f2;
text-align: center;
padding: 14px 16px;
text-decoration: none;
font-size: 17px;
}
.navbar a:hover {
background: #ddd;
color: black;
}
.title{
padding: 12px 12px;
color: #f2f2f2;
}
</style>
</head>
<body>
<div class="navbar">
<a href="/login">SIGN IN </a>
<a href="/register">SIGN UP</a>
<a href="/adminlogin">ADMIN</a>
<div class="title">
<h3 style="right:0;font-family:'Lobster', cursive;">Atlantis</h3>
</div>
</div>
<form method="post">
<div class="box">
```

```

<div class="form">
  <h2>SIGN UP</h2>
  <div class="inputBox" >
    <input type="text" name="username" required>
    <span class="fa fa-user"> <span class="font">Username</span></span>
    <i></i>
  </div>
  <div class="inputBox" >
    <input type="email" name="email" required>
    <span class="fa fa-envelope"> <span class="font">Email</span></span>
    <i></i>
  </div>
  <div class="inputBox" >
    <input type="text" name="phoneno" required>
    <span class="fa fa-phone"> <span class="font">PhoneNumber</span></span>
    <i></i>
  </div>
  <div class="inputBox">
    <input type="password" name="password" required>
    <span class="fa fa-lock"> <span class="font">Password</span></span>
    <i></i>
  </div>
  <p>{{success}}</p>
  <p style="color: red">{{error}}</p>
  <div class="links">
    <a href="#">Forget password</a>
    <a href="/adminlogin">SIGN IN</a>
  </div>
  <input type="submit" value="Login">
</div>
</div>
</form>

```

```
</body>
```

```
</html>
```

Home.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

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

```
<title>View Product</title>
```

```
<script
```

```
src="https://cdn.jsdelivr.net/npm/@splidejs/splide@2.4.21/dist/js/splide.min.js"></script>
```

```
<link
```

```
rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@splidejs/splide@2.4.21/dist/css/splide.min.css"/>
```

```
<meta content="text/html; charset=iso-8859-2" http-equiv="Content-Type">
```

```
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
```

```
<link rel="stylesheet" href="{{url_for('static',filename='styles/home.css')}}">
```

```
<script>
```

```
    window.watsonAssistantChatOptions = {
```

```
        integrationID: "b6892981-e498-43d3-a1fb-36c1a5752cec", // The ID of this integration.
```

```
        region: "au-syd", // The region your integration is hosted in.
```

```
        serviceInstanceID: "eaca97b3-64b0-4e47-854f-8314f7e3e0ec", // The ID of your service instance.
```

```
        onLoad: function(instance) { instance.render(); }
```

```
    };
```

```
    setTimeout(function(){
```

```
        const t=document.createElement('script');
```

```

    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
  });
</script>
<style>
@import
url('https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,200;0,300;0,40
0;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900
&display=swap');
h1,h2,h3{
color: #fff;
font-family: 'Poppins', sans-serif;
}
</style>
</head>
<body>
<div class="navbar">
<a href="/logout">LOG OUT</a>
<div class="title">
<h3 style="right:0;font-family:'Lobster', cursive; color: #fff;margin:
1%;">InfiniteArt</h3>
</div>
</div>
<div class="co">
<div class="w3-content w3-section" style="max-width:100%;;>



</div>
</div>
<h2>TRENDING SHIRTS</h2>
<div class="splide" style="padding: 0;margin:0">
<div class="splide__track">
<ul class="splide__list">
{% for item in dictionary %}
<li class="splide__slide">
<div class="container">
<div class="card">
<div class="image">

</div>
<div class="descripton">
<h2>{{item.NAME}}</h2>
<div class="size">
<span>7</span>
<span>8</span>
<span>9</span>
<span>10</span>
</div>
<h3>&#8360;{{item.RATE}}</h3>
<div class="color">
<h3>Color: </h3>
<span></span>
<span></span>
<span></span>
</div>
<a href="#">Buy now</a>
</div>
</div>

```

```

</div>
</li>
{% endfor %}
<!-- pant -->
</ul>
</div>
</div>
<H2>TRENDING PANTS</H2>
<div id="pant" class="splide" style="margin:0;background: #131313;" >
<div class="splide__track">
<ul class="splide__list">
{% for item in pants %}
<li class="splide__slide">
<div class="container">
<div class="card">
<div class="image">

</div>
<div class="descripton">
<h2>{{item.NAME}}</h2>
<div class="size">
<span>7</span>
<span>8</span>
<span>9</span>
<span>10</span>
</div>
<h3>&#360;{{item.RATE}}</h3>
<div class="color">
<h3>Color: </h3>
<span></span>
<span></span>
<span></span>

```

```

</div>
<a href="#">Buy now</a>
</div>
</div>
</div>
</li>
{% endfor %}
<!-- pant -->
</ul>
</div>
</div>
<h2>TRENDING WATCHES</h2>
<div id="watch" class="splide" style="padding: 0;margin:0;background: #131313;" >
<div class="splide__track">
<ul class="splide__list">
{% for item in watchs %}
<li class="splide__slide">
<div class="container">
<div class="card">
<div class="image">

</div>
<div class="descripton">
<h2>{{item.NAME}}</h2>
<div class="size">
<span>7</span>
<span>8</span>
<span>9</span>
<span>10</span>
</div>
<h3>&#360;{{item.RATE}}</h3>
<div class="color">

```



```

<h3>Color: </h3>
<span></span>
<span></span>
<span></span>
</div>
<a href="#">Buy now</a>
</div>
</div>
</div>
</li>
{% endfor %}
<!-- pant -->
</ul>
</div>
</div>
<h2>TRENDING RINGS</h2>
<div id="ring" class="splide" style="padding: 0;margin:0;background: #131313;" >
<div class="splide__track">
<ul class="splide__list">
{% for item in rings %}
<li class="splide__slide">
<div class="container">
<div class="card">
<div class="image">

</div>
<div class="descripton">
<h2>{{item.NAME}}</h2>
<div class="size">
<span>7</span>
<span>8</span>
<span>9</span>

```

```

<span>10</span>
</div>
<h3>{{item.RATE}}</h3>
<div class="color">
<h3>Color: </h3>
<span></span>
<span></span>
<span></span>
</div>
<a href="#">Buy now</a>
</div>
</div>
</div>
</li>
{% endfor %}
<!-- pant -->
</ul>
</div>
</div>
<script>
var splide = new Splide( '.splide', {
  type : 'loop',
  perPage : 4,
  autoplay: true,
  });
splide.mount();
document.addEventListener('DOMContentLoaded', function () {
  new Splide('#pant', {
    perPage: 4,
    perMove: 1,
    gap: "30px",
    pagination: false,

```

```

}).mount();
});
document.addEventListener('DOMContentLoaded', function () {
  new Splide('#watch', {
    perPage: 4,
    perMove: 1,
    gap: "30px",
    pagination: false,
  }).mount();
});
document.addEventListener('DOMContentLoaded', function () {
  new Splide('#ring', {
    perPage: 4,
    perMove: 1,
    gap: "30px",
    pagination: false,
  }).mount();
});
</script>
<script>
var splid = new Splide( '.splid', {
  type : 'loop',
  perPage : 4,
  autoplay: true,
  } );
splid.mount();
</script>
<script>
var myIndex = 0;
carousel();
function carousel() {
  var i;

```

```

var x = document.getElementsByClassName("mySlides");
for (i = 0; i < x.length; i++) {
x[i].style.display = "none";
}
myIndex++;
if (myIndex > x.length) {myIndex = 1}
x[myIndex-1].style.display = "block";
setTimeout(carousel, 2000); // Change image every 2 seconds
}
</script>
</body>
</html>

```

Index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
  <link
href="//db.onlinewebfonts.com/c/d8a3c95906aec0c2483082a82e72cb40?family=Wanderlu
stShine-Regular" rel="stylesheet" type="text/css"/>
  <title>SMART FASHION RECOMMENDED APPLICATION</title>
<script>
window.watsonAssistantChatOptions = {
  integrationID: "b6892981-e498-43d3-a1fb-36c1a5752cec", // The ID of this integration.
  region: "au-syd", // The region your integration is hosted in.

```

```

    serviceInstanceId: "eaca97b3-64b0-4e47-854f-8314f7e3e0ec", // The ID of your service
instance.

    onLoad: function(instance) { instance.render(); }
};

setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
});
</script>
<style>
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&displa
y=swap');
@import url('https://fonts.googleapis.com/css2?family=Lobster&display=swap');
@font-face{
    font-family: Wanderlust;
    src: url('/Wanderlust Letters-Font/OTF/WanderlustLetters-Regular.otf');
    font-weight: bold;
}
*{
    margin: 0;
    padding: 0;
    font-family: 'Poppins', sans-serif;
}
body{
    text-align: center;
    background-color: #23242a;
    color: #f2f2f2;
}

```

```
.navbar {  
    overflow: hidden;  
    position: fixed;  
    top: 0;  
    width: 100%;  
}  
.navbar a {  
    float: right;  
    display: block;  
    color: #FF0063;  
    text-align: center;  
    padding: 14px 16px;  
    text-decoration: none;  
    font-size: 17px;  
}  
.navbar a:hover {  
    background: #f2f2f2;  
    color: black;  
}  
.navbar h3{  
    padding: 12px 12px;  
    color: #FF0063;  
    font-size: 5vh;  
    font-weight: 700;  
    top:0;  
    text-align: left;  
}  
.content {  
    width:100%;  
    height:100%;  
    display: table;  
    border:2px solid white;
```

```
}  
.image{  
    width:5%;  
    height: 90%;  
    float: left;  
    padding: 2px;  
}  
.image img{  
    height: 46px;  
    width:80px;  
}  
.banner{  
    width: 100%;  
    height: 730px;  
    background-image: url('{{url_for('static',filename='background.png')}}");  
    background-attachment: fixed;  
    background-size: cover;  
}  
.con{  
    text-align: center;  
    padding: 2%;  
}  
.con a{  
    color: #f2f2f2;  
    font-family: 'Poppins', sans-serif;  
    font-weight: bolder;  
    text-decoration: none;  
    background: #FF0063;  
    padding: 5px;  
    border-radius: 10px;  
}  
.con a:hover{
```

```
background:#f2f2f2;
color: #FF0063;
cursor: pointer;
font-family: 'Poppins', sans-serif;
font-weight: bolder;
}
.get{
margin: 20% 20%;
}
.about,.cart,.chatbot{
margin: 2%;
width: 90%;
height: 300px;
box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2);
background-color: #111111;
transition: 0.3s;
padding: 1%;
overflow-y: scroll;
}
::-webkit-scrollbar{
width:0;
background: transparent;
}
.about:hover,.cart:hover,.chatbot:hover {
box-shadow: 0 8px 16px 0 rgb(52, 230, 28);
}
.left{
width: 10%;
height: 100%;
float: left;
margin: 1%;
}
```



```

.about img,.chatbot img{
    width: 20%;
    height: 85%;
    float: left;
    top: 0;
    margin: 0% 6%;
}

.cart img{
    width: 20%;
    height: 85%;
    float: right;
    top: 0;
    margin: 0% 6%;
}

.txt{
    width: 55%;
    float: right;
    text-align: justify;
    margin: 4% 5%;
}

</style>
</head>
<body>
    <div class="navbar">
        <a href="/login">SIGN IN </a>
        <div class="title">
            <!-- <div class="image"> </div> --
        >
            <div><h3 style="right:0;font-family:'Lobster', cursive;">Atlantis </h3></div>
        </div>
    </div>
    <div class="banner">

```

```

<div class="con">
  <div class="get"><h1><span style="color:#FF0063; ">SMART FASHION
RECOMMENDER APPLICATION,</span> unlimited access to all fashion items</h1><br>
  <a href="/register" name="get started">GET STARTED &#8594;</a></div>
</div>
</div><br>
<div class="about">
  <center><h2>ABOUT</h2></center>
  
  <div class="txt">
    <p><h3 style="color:#FF0063;">A warm welcome to your online fashion
lane.</h3><br>

```

We are aware of how much you value fashion and how much you enjoy online shopping's convenience. When you're on the go, the Atlantis online shopping app makes sure you don't miss out on a chic shopping experience. If you want to shop for clothing for men and women, shoes, accessories, or even the newest electronics and tech gadgets, Atlantis is always there to meet all of your style needs

```

  </p></div>
</div>
<div class="cart">
  <center><h2>RECOMMENDATION&nbsp;SYSTEM</h2></center>
  
  <div class="txt">
    <p><h3 style="color:#FF0063;">Your Favourite Items are here.</h3><br>

```

Here, we used a recommendation system that is to categorise the user's clothing and suggest the best outfit for a particular occasion based on a recommendation algorithm. The suggested system demonstrates that it can analyse the user's attire from the images, determine the type and colour of the outfit, and then suggest the most appropriate outfit for the situation based on the user's current attire.

```

  </p></div>

```

```

</div>
<div class="chatbot">
  <center><h2>ABOUT US</h2></center>
  
  <div class="txt">
    <p><h3 style="color:#FF0063;">Little information about us !</h3><br>
    <table>
      <tr><td><b>Team Id</b></td><td><b>PNT2022TMID34632</b></td></tr>
      <tr><th>Team Lead&emsp;&emsp;</th><th>Femima</th></tr>
      <tr><th>Member</th><th>Abisha</th></tr>
      <tr><th>Member</th><th>Benitta</th></tr>
      <tr><th>Member</th><th>Delfin</th></tr>
    </table>
    </p></div>
  </div>
<div class="developers"></div>
</body>
</html>

```

Login.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Log in</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
<script>
  window.watsonAssistantChatOptions = {
    integrationID: "b6892981-e498-43d3-a1fb-36c1a5752cec", // The ID of this integration.

```

```

    region: "au-syd", // The region your integration is hosted in.
    serviceInstanceId: "eaca97b3-64b0-4e47-854f-8314f7e3e0ec", // The ID of your service
instance.

    onLoad: function(instance) { instance.render(); }
};
setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
});
</script>
<style>
    @import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&displa
y=swap');
    @import url('https://fonts.googleapis.com/css2?family=Lobster&display=swap');
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}
body{
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    background-color: #23242a;
}
.box{

```

```
position: relative;
width: 380px;
height: 400px;
background-color: #1c1c1c;
border-radius: 8px;
overflow: hidden;
}
.box::before{
  content: "";
  position: absolute;
  top: -50%;
  left: -50%;
  width: 380px;
  height: 420px;
  background: linear-gradient(0deg, transparent,#FF0063,#FF0063);
  transform-origin: bottom right;
  animation: animate 6s linear infinite;
}
.box::after{
  content: "";
  position: absolute;
  top: -50%;
  left: -50%;
  width: 380px;
  height: 420px;
  background: linear-gradient(0deg, transparent,#FF0063,#FF0063);
  transform-origin: bottom right;
  animation: animate 6s linear infinite;
  animation-delay: -3s;
}
@keyframes animate {
  0%{
```

```
        transform: rotate(0deg);
    }
    100%{
        transform: rotate(350deg);
    }
}
/* Form */
.form{
    position: absolute;
    inset: 2px;
    border-radius: 8px;
    background: #28292d;
    z-index: 10;
    padding: 40px 40px;
    display: flex;
    flex-direction: column;
}
.form h2{
    color: #FF0063;
    font-weight: 500;
    text-align: center;
    letter-spacing: 0.1em;
}
.inputBox{
    position: relative;
    width: 100%;
    margin-top: 35px;
}
.inputBox input{
    position: relative;
    width: 100%;
    padding: 10px 10px 10px;
```

```
background: transparent;
border: none;
outline: none;
color: #23242a;
font-size: 1em;
letter-spacing: 0.05em;
z-index: 10;
}
.inputBox span{
  position: absolute;
  left: 0;
  padding: 20px 0px 10px;
  font-size: 1em;
  color: white;
  pointer-events: none;
  letter-spacing: 0.05em;
  transition: 0.5s;
}
.inputBox input:valid ~ span,
.inputBox input:focus ~ span{
  color: #FF0063;
  transform: translateX(0px) translateY(-34px);
  font-size: 0.75em;
}
.inputBox i{
  position: absolute;
  left: 0;
  bottom: 0;
  width: 100%;
  height: 2px;
  background: #FF0063;
  border-radius: 4px;
```

```
    transition: 0.5s;
    pointer-events: none;
    z-index: 0;
}
.inputBox input:valid ~ i,
.inputBox input:focus ~ i{
    height: 44px;
}
.links{
    display: flex;
    justify-content: space-between;
}
.links a {
    margin: 10px 0;
    font-size: 0.75em;
    color: #8f8f8f;
    text-decoration: none;
}
.links a:hover,
.links a:nth-child(2){
    color: #FF0063;
}
input[type="submit"]{
    border: none;
    outline: none;
    background: #FF0063;
    padding: 11px 25px;
    width: 100px;
    margin-top: 10px;
    border-radius: 4px;
    font-weight: 600;
    cursor: pointer;
```



```
}  
input[type="submit"]:active{  
    opacity: 0.8;  
}  
.font{  
    font-family: 'Poppins', sans-serif;  
    top: 0;  
    margin-left: 120%;  
    text-align: left;  
}  
.fa{  
    color: aliceblue;  
}  
.navbar {  
    overflow: hidden;  
    background:linear-gradient(-13deg, transparent,#FF0063,#FF0063);  
    position: fixed;  
    top: 0;  
    width: 100%;  
    font-size: 3vh;  
}  
.navbar a {  
    float: right;  
    display: block;  
    color: #f2f2f2;  
    text-align: center;  
    padding: 14px 16px;  
    text-decoration: none;  
    font-size: 17px;  
}  
.navbar a:hover {  
    background: #ddd;
```

```

    color: black;
}
.title{
    padding: 12px 12px;
    color: #f2f2f2;
}
</style>
</head>
<body>
    <div class="navbar">
        <a href="/login">SIGN IN </a>
        <a href="/register">SIGN UP</a>
        <a href="/adminlogin">ADMIN</a>
        <div class="title">
            <h3 style="right:0;font-family:'Lobster', cursive;">Atlantis</h3>
        </div>
    </div>
    <form method="post">
        <div class="box">
            <div class="form">
                <h2>SIGN IN</h2>
                <div class="inputBox" >
                    <input type="text" name="email" required>
                    <span class="fa fa-envelope"> <span class="font">Email</span></span>
                    <i></i>
                </div>
                <div class="inputBox">
                    <input type="password" name="password" required>
                    <span class="fa fa-lock"> <span class="font">Password</span></span>
                    <i></i>
                </div>
            </div>
            <p>{{success}}</p>

```

```

    <p style="color: red">{{error}}</p>
    <div class="links">
        <a href="#">Forget password</a>
        <a href="/register">SIGN UP</a>
    </div>
    <input type="submit" value="Login">
</div>
</div>
</form>
</body>
</html>

```

Register.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
    <link rel="stylesheet" href="{{url_for('static', filename='styles/style.css')}}">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
<script>
    window.watsonAssistantChatOptions = {
        integrationID: "b6892981-e498-43d3-a1fb-36c1a5752cec", // The ID of this integration.
        region: "au-syd", // The region your integration is hosted in.
        serviceInstanceID: "eaca97b3-64b0-4e47-854f-8314f7e3e0ec", // The ID of your service
instance.
        onLoad: function(instance) { instance.render(); }
    };

```

```

setTimeout(function(){
  const t=document.createElement('script');
  t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
  document.head.appendChild(t);
});
</script>
<style>
  @import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&displa
y=swap');
  @import url('https://fonts.googleapis.com/css2?family=Lobster&display=swap');
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
}
body{
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  background-color: #23242a;
}
.box{
  position: relative;
  width: 400px;
  height: 550px;
  background-color: #1c1c1c;
  border-radius: 8px;

```

```
    overflow: hidden;
}
.box::before{
    content: "";
    position: absolute;
    top: -50%;
    left: -50%;
    width: 380px;
    height: 420px;
    background: linear-gradient(0deg, transparent,#FF0063,#FF0063);
    transform-origin: bottom right;
    animation: animate 6s linear infinite;
}
.box::after{
    content: "";
    position: absolute;
    top: -50%;
    left: -50%;
    width: 380px;
    height: 420px;
    background: linear-gradient(0deg, transparent,#FF0063,#FF0063);
    transform-origin: bottom right;
    animation: animate 6s linear infinite;
    animation-delay: -3s;
}
@keyframes animate {
    0%{
        transform: rotate(0deg);
    }
    100%{
        transform: rotate(350deg);
    }
}
```

```
}  
.form{  
  position: absolute;  
  inset: 2px;  
  border-radius: 8px;  
  background: #28292d;  
  z-index: 10;  
  padding: 40px 40px;  
  display: flex;  
  flex-direction: column;  
}  
.form h2{  
  color: #FF0063;  
  font-weight: 500;  
  text-align: center;  
  letter-spacing: 0.1em;  
}  
.inputBox{  
  position: relative;  
  width: 100%;  
  margin-top: 35px;  
}  
.inputBox input{  
  position: relative;  
  width: 100%;  
  padding: 10px 10px 10px;  
  background: transparent;  
  border: none;  
  outline: none;  
  color: #23242a;  
  font-size: 1em;  
  letter-spacing: 0.05em;
```

```
    z-index: 10;
}
.inputBox span{
    position: absolute;
    left: 0;
    padding: 20px 0px 10px;
    font-size: 1em;
    color: white;
    pointer-events: none;
    letter-spacing: 0.05em;
    transition: 0.5s;
}
.inputBox input:valid ~ span,
.inputBox input:focus ~ span{
    color: #FF0063;
    transform: translateX(0px) translateY(-34px);
    font-size: 0.75em;
}
.inputBox i{
    position: absolute;
    left: 0;
    bottom: 0;
    width: 100%;
    height: 2px;
    background: #FF0063;
    border-radius: 4px;
    transition: 0.5s;
    pointer-events: none;
    z-index: 0;
}
.inputBox input:valid ~ i,
.inputBox input:focus ~ i{
```

```
    height: 44px;
}
.links{
    display: flex;
    justify-content: space-between;
}
.links a {
    margin: 10px 0;
    font-size: 0.75em;
    color: #8f8f8f;
    text-decoration: none;
}
.links a:hover,
.links a:nth-child(2){
    color: #FF0063;
}
input[type="submit"]{
    border: none;
    outline: none;
    background: #FF0063;
    padding: 11px 25px;
    width: 100px;
    margin-top: 10px;
    border-radius: 4px;
    font-weight: 600;
    cursor: pointer;
}
input[type="submit"]:active{
    opacity: 0.8;
}
.font{
    font-family: 'Poppins', sans-serif;
```



```
    top: 0;
    margin-left: 120%;
    text-align: left;
}
.fa{
    color: aliceblue;
}
.navbar {
    overflow: hidden;
    background: linear-gradient(-13deg, transparent, #FF0063, #FF0063);
    position: fixed;
    top: 0;
    width: 100%;
    font-size: 3vh;
}
.navbar a {
    float: right;
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 17px;
}
.navbar a:hover {
    background: #ddd;
    color: black;
}
.title{
    padding: 12px 12px;
    color: #f2f2f2;
}
```

```

</style>
</head>
<body>
  <div class="navbar">
    <a href="/login">SIGN IN </a>
    <a href="/register">SIGN UP</a>
    <a href="/adminlogin">ADMIN</a>
    <div class="title">
      <h3 style="right:0;font-family:'Lobster', cursive;">Atlantis</h3>
    </div>
  </div>
  <form method="post">
    <div class="box">
      <div class="form">
        <h2>SIGN UP</h2>
        <div class="inputBox" >
          <input type="text" name="username" required>
          <span class="fa fa-user"> <span class="font">Username</span></span>
          <i></i>
        </div>
        <div class="inputBox" >
          <input type="email" name="email" required>
          <span class="fa fa-envelope"> <span class="font">Email</span></span>
          <i></i>
        </div>
        <div class="inputBox" >
          <input type="text" name="phoneno" required>
          <span class="fa fa-phone"> <span class="font">PhoneNumber</span></span>
          <i></i>
        </div>
        <div class="inputBox">
          <input type="password" name="password" required>

```

```
<span class="fa fa-lock"> <span class="font">Password</span></span>
<i></i>
</div>
<p>{{success}}</p>
<p style="color: red">{{error}}</p>
<div class="links">
  <a href="#">Forget password</a>
  <a href="/login">SIGN IN</a>
</div>
<input type="submit" value="Login">
</div>
</div>
</form>
</body>
</html>
```

13.2 GITHUB LINK:

<https://github.com/IBM-EPBL/IBM-Project-49459-1660819392>