

1.INTRODUCTION

1.1 Project Overview

With an increase in the standard of living, peoples' attention gradually moved towards fashion that is concerned to be a popular aesthetic expression. Humans are inevitably drawn towards something that is visually more attractive. This tendency of humans has led to the development of the fashion industry over the course of time. However, given too many options of garments on the e-commerce websites, has presented new challenges to the customers in identifying their correct outfit. Thus, in this project, we proposed a personalized Fashion Recommender system that generates recommendations for the user based on an input given. Unlike the conventional systems that rely on the user's previous purchases and history, this project aims at using an image of a product given as input by the user to generate recommendations since many-a-time people see something that they are interested in and tend to look for products that are similar to that. We use neural networks to process the

images from Fashion Product Images Dataset and the Nearest neighbour backed recommender to generate the final recommendations.

1.2Purpose

Humans are inevitably drawn towards something that is visually more attractive. This tendency of humans has led to development of fashion industry over the course of time. With introduction of recommender systems in multiple domains, retail industries are coming forward with investments in latest technology to improve their business. Fashion has been in existence since centuries and will be prevalent in the coming days as well. Women are more correlated with fashion and style, and they have a larger product base to deal with making it difficult to take decisions. It has become an important aspect of life for modern families since a person is more often than not judged based on his attire. Moreover, apparel providers need their customers to explore their entire product line so they can choose what they like the most which is not possible by simply going into a cloth store.

2.LITERATURE SURVEY

2.1 Existing problem

The work of this project is based on combining two deep learning models to detect the type and color of the clothing in the given image. The recommendation algorithm however is written by us. Hence, it is safe to say that no existing system has been proposed but work has been done in detecting objects, types and colors of clothes by using public datasets and applying machine learning techniques. The related work is presented as follows:

In deep learning classification of clothing apparel was discussed. The approach followed in this system was using Convolutional Neural Networks (CNNs). The Deep Learning technique Inception v-3 was used for different object detection problems. A wide variety of clothing types were used in the paper about 13 different classes, some of which were: Coat, Poncho, Blouse, Dress, Shirt, Vest, Lingerie, T-shirt, Uniform,

Suit, Sweater, Jacket, Sports sweater. The recognition rate is about 70%.

Out of all these classes the ones of utmost importance are only Shirt and T-shirt. Both of which have accuracy rates of 50 and 60% respectively. There are two more classes of interest namely Pant and Shoes that were not included in this study.

We need a model with better accuracy and capable of handling the two missing classes as well. Another drawback is there is no standalone web application that makes it easy for the users to use. Hence, we cannot use this in our system.

The drawback is there is no standalone application that can be used by the patient to use this model.

In paper, deep learning techniques were used to detect the type and color of the clothing. Hence, Convolutional Neural Networks were used. The resultant accuracy of the resultant model was around 86%. The intention of this paper was to create model that does proper labelling of clothes. Hence it is not a Fashion Recommendation System that recommends clothing choices to the user. We can improve the accuracy and also create a user friendly website for ease of access. The paper also does not provide any wardrobe for the

2.2 References

https://www.researchgate.net/publication/353485380_Fashion_Recommendation_Systems_Models_and_Methods_A_Review

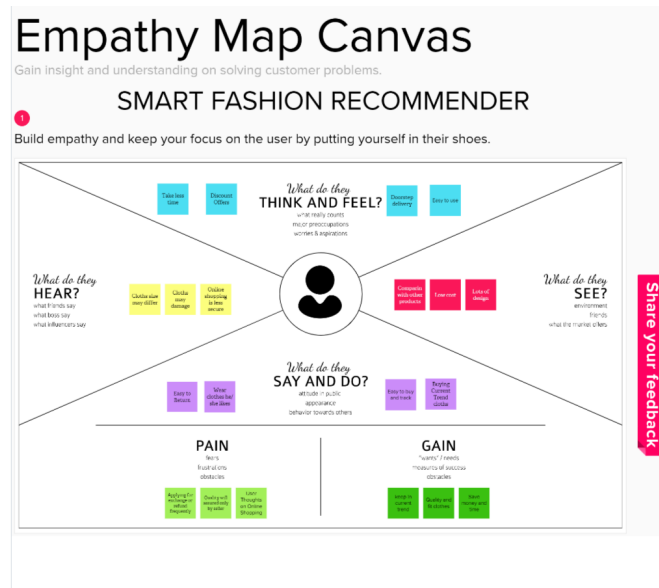
2.3 Problem Statement Definition

The era of recommendation systems originally started in the 1990s based on the widespread research progress in Collective Intelligence. During this period, recommendations were generally provided to consumers based on their rating structure . The first consumer-focused recommendation system was developed and commercialized by Goldberg, Nichols, Oki and Terry in 1992. Tapestry, an electronic messaging system was developed to allow users only to rate messages as either a good or bad product and service .However, now there are plenty of methods to obtain information about the consumer's liking for a product

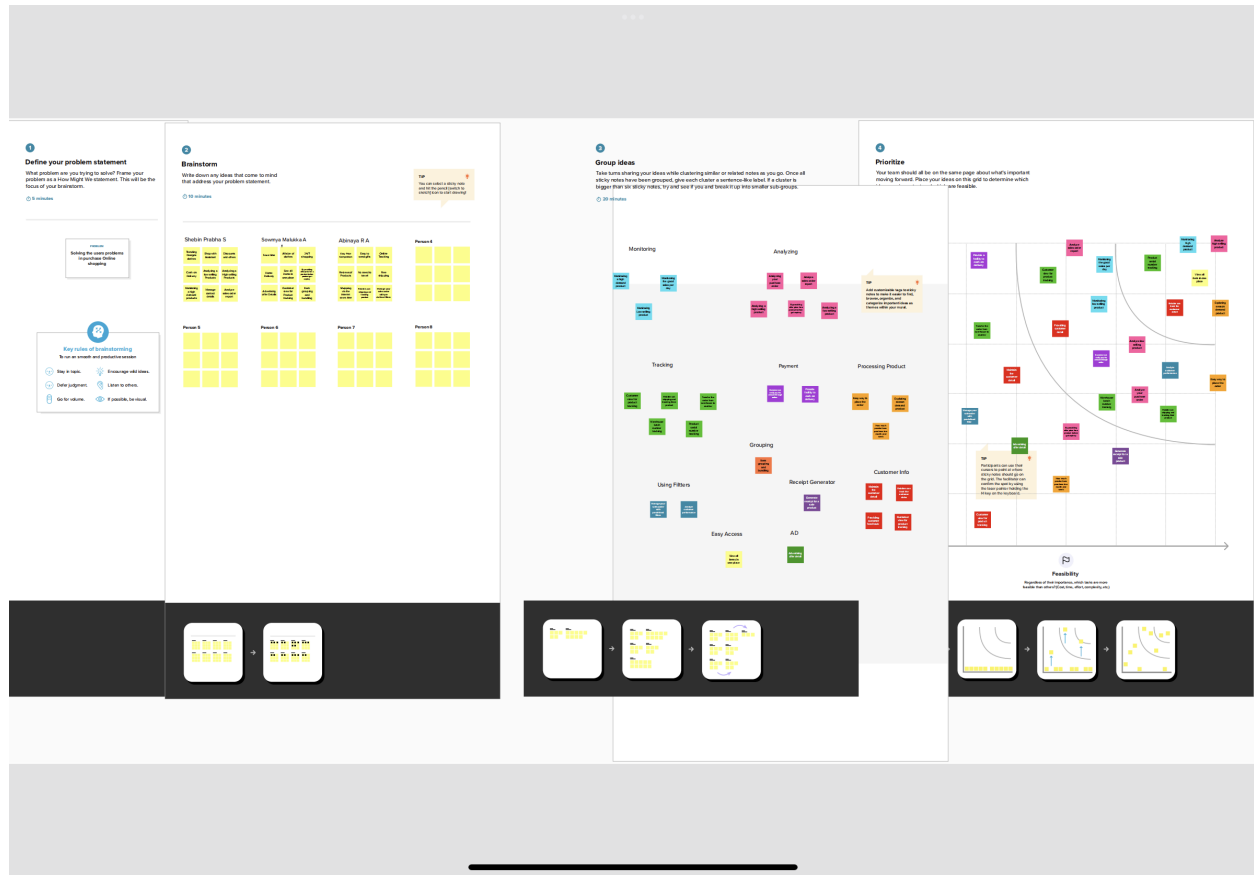
through the Internet. These data can be retrieved in the forms of voting, tagging, reviewing and the number of likes or dislikes the user provides. It may also include reviews written in blogs, videos uploaded on YouTube or messages about a product. Regardless of communication and presentation, medium preferences are expressed in the form of numerical values presents the history of the progress of fashion recommendation systems over the last few decades.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



3.3 Proposed Solution

Proposed Solution Template:

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.	Social Impact / Customer Satisfaction	The user friendly interface, Assistants form chat bot finding dress makes customer satisfied.
5.	Business Model (Revenue Model)	The chat bot sells our products to customer. Customers buy our products and generate revenue.
6.	Scalability of the Solution	We can easily scalable our applications by increases the items and products.

Activate Windows

3.4 Problem Solution Fit

Project Title: Smart Fashion Recommender Application

Project Design Phase-I - Solution Fit Template

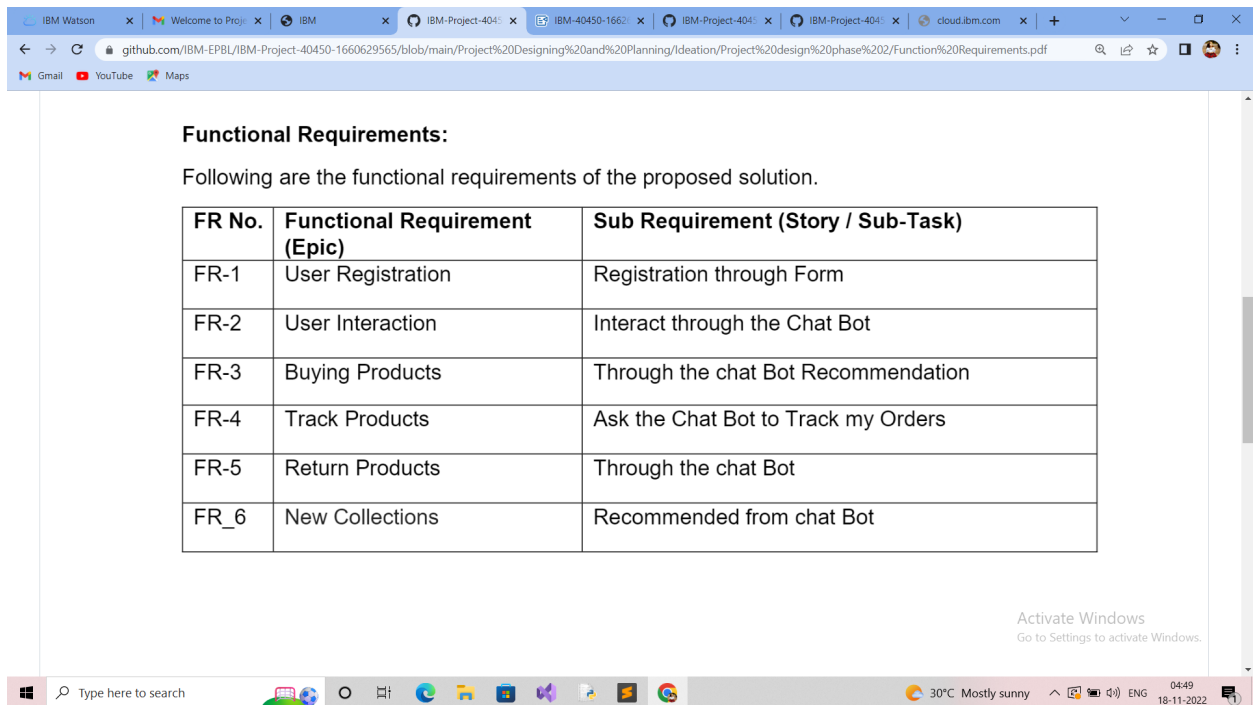
Team ID: PNT2022TMD51358

<div>Define CS, RP, Info CC</div> <div>1. CUSTOMER SEGMENT(S) <small>Who are your customers? Are you targeting a niche or a mass market?</small></div> <div>The Customers are Adults and children</div>	<div>5. CUSTOMER CONSTRAINTS <small>What are the constraints on your customers? What are the constraints on your customers? What are the constraints on your customers?</small></div> <div>Money and Network Connection</div>	<div>6. AVAILABLE SOLUTIONS <small>What are the available solutions? What are the available solutions? What are the available solutions?</small></div> <div>Online shopping gives New Collections pros: Easy to use cons: customer confused when have lost of collection</div>	<div>Export AS information</div>
<div>Focus on JAR to use CS, customer RP</div> <div>2. JOBS-TO-BE-DONE / PROBLEMS <small>What jobs do your customers want to get done? What jobs do your customers want to get done? What jobs do your customers want to get done?</small></div> <div>Users hard to find Trending Fashion Clothes.</div>	<div>9. PROBLEM ROOT CAUSE <small>What is the root cause of the problem? What is the root cause of the problem? What is the root cause of the problem?</small></div> <div>Customers need to be with new fashions for current trends</div>	<div>7. BEHAVIOUR <small>What are the behaviours of your customers? What are the behaviours of your customers? What are the behaviours of your customers?</small></div> <div>Customers spend the time to find the new fashion clothes</div>	<div>Focus on JAR to use CS, customer RP</div>
<div>Identify existing RP & CS</div> <div>3. TRIGGERS <small>What triggers your customers to act? What triggers your customers to act? What triggers your customers to act?</small></div> <div>Seeing neighbor Dressing Styles</div>	<div>10. YOUR SOLUTION <small>What is your solution? What is your solution? What is your solution?</small></div> <div>Make a Chat Bot Assistant for shopping with customers and send notifications when new collections arrived</div>	<div>8. CHANNELS OF BEHAVIOUR <small>What are the channels of behaviour? What are the channels of behaviour? What are the channels of behaviour?</small></div> <div>ONLINE: Customers buy the new clothes OFFLINE: Customers will use the clothes</div>	<div>Import AS information</div>
<div>4. EMOTIONS: BEFORE / AFTER <small>What are the emotions of your customers? What are the emotions of your customers? What are the emotions of your customers?</small></div> <div>Felling Sad and Frustration > Self confident</div>			

Activate Windows
Go to Settings to activate Windows.

4.REQUIREMENT ANALYSIS

4.1 Functional Requirements



The screenshot shows a web browser window with multiple tabs. The active tab displays a GitHub repository page for 'IBM-EPBL/IBM-Project-40450-1660629565'. The page content is titled 'Functional Requirements:' and lists the functional requirements of the proposed solution. Below the text, there is a table with three columns: 'FR No.', 'Functional Requirement (Epic)', and 'Sub Requirement (Story / Sub-Task)'. The table contains six rows of data. The browser's address bar shows the URL 'github.com/IBM-EPBL/IBM-Project-40450-1660629565/blob/main/Project%20Designing%20and%20Planning/Ideation/Project%20design%20phase%202/Function%20Requirements.pdf'. The Windows taskbar is visible at the bottom, showing the search bar, taskbar icons, and system tray with weather and date information.

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
FR-2	User Interaction	Interact through the Chat Bot
FR-3	Buying Products	Through the chat Bot Recommendation
FR-4	Track Products	Ask the Chat Bot to Track my Orders
FR-5	Return Products	Through the chat Bot
FR_6	New Collections	Recommended from chat Bot

Activate Windows
Go to Settings to activate Windows.

4.2 Non Functional Requirements

Non-functional Requirements:

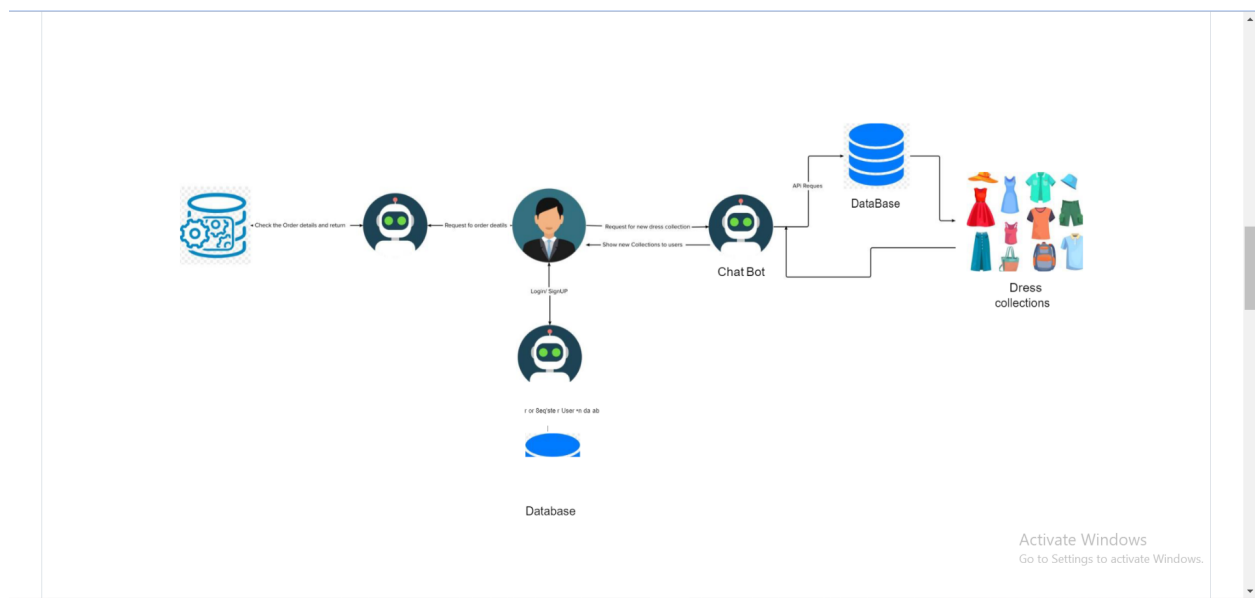
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Using Android or IOS or windows applications.
NFR-2	Security	The user data is stored securely in IBM cloud.
NFR-3	Reliability	The Quality of the services are trusted.
NFR-4	Performance	Its Provide smooth user experience.
NFR-5	Availability	The services are available for 24/7.
NFR-6	Scalability	Its easy to scalable size of users and products.

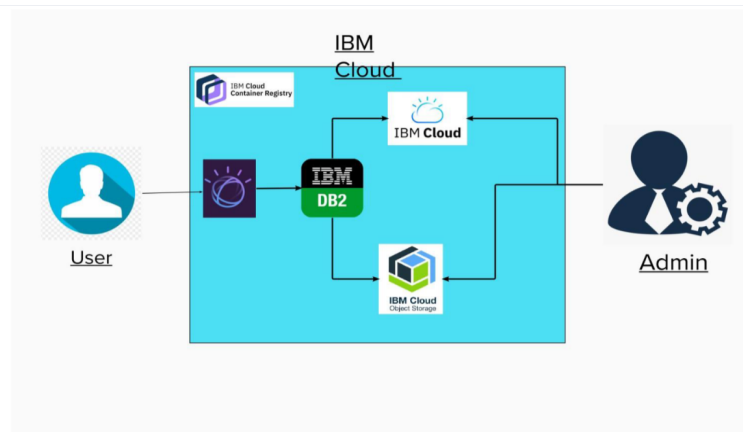
Activate Windows
Go to Settings to activate Windows.

5.PROJECT DESIGN

5.1 Data Flow Diagram



5.2 Solution & Technical Architecture



Technical Architecture:

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access my data by login	High	Sprint-1
	Dashboard	USN-6	As a user , I can view the dashboard and by products		High	Sprint -2
Customer (Web user)	Registration / Login	USN-7	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard		Sprint -1
Customer Care Executive	Contact with Customers	USN-8	As a Customer customers care executive, I solve the customer Requirements and feedback	I can receive calls from customers	High	Sprint-1
Administrator	Check stock and Price , orders	USN_9	As a Administrator , I can Check the database And stock details and buying and selling prices	I am the administrator of the company	High	Sprint -2

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

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	SHEBIN PRABHA S SOWMYA MALUKKA A ABINAYA R A
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	SHEBIN PRABHA S SOWMYA MALUKKA A ABINAYA R A
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	SHEBIN PRABHA S SOWMYA MALUKKA A ABINAYA R A
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	SHEBIN PRABHA S SOWMYA MALUKKA A ABINAYA R A

7.CODING & SOLUTIONING

```
from flask import redirect, flash, render_template,
url_for, request
from fashionshop.forms import
RegistrationForm, LoginForm, InforForm, UserForm
from fashionshop import db, bcrypt, app, es
from fashionshop.models import *
from flask_login import current_user, login_user,
login_required, logout_user
from product_recommender import *
from chatbot import bot
```

```
@app.route("/", methods=['GET', 'POST','PUT'])
```

```
@app.route("/home")
```

```
def home():
```

```
    session.permanent = True
```

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

```
@app.route("/about")
```

```
def about():
```

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

```
@app.route("/register", methods=['GET', 'POST'])
```

```
def register():
```

```
    if current_user.is_authenticated:
```

```
        redirect(url_for('home'))
```

```
form = RegistrationForm()
if form.validate_on_submit():
    password =
bcrypt.generate_password_hash(form.password.data).
decode('utf-8')
    user = User(username = form.username.data,
email = form.email.data, password = password)
    db.session.add(user)
    db.session.commit()
    flash(f'Your account have been created!', 'success')
    return redirect(url_for('login'))
return render_template('register.html', form = form,
title = 'Register')
@app.route("/login", methods=['GET','POST'])
def login():
    if current_user.is_authenticated:
        redirect(url_for('home'))
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email =
form.email.data).first()
        if user and
bcrypt.check_password_hash(user.password,
form.password.data):
```

```

        login_user(user, remember =
form.remember.data)
        flash(f'Login successfully!', 'success')
        return redirect(url_for('home'))
    else:
        flash(f'Please check the information!','danger')
    return render_template('login.html', form = form,
title = 'Login')
@app.route('/contact', methods=['GET', 'POST'])
def contact():
    session['firstname'] = request.form.get('firstname')
    session['lastname'] = request.form.get('lastname')
    session['email'] = request.form.get('email')
    session['subject'] = request.form.get('subject')
    session['message'] = request.form.get('message')
    print(session['firstname'], session['lastname'],
session['email'], session['subject'],session['message'])
    if session['firstname'] != None:
        flash(f'Message sent!', 'success')
    return render_template('contact.html', tilte =
'Contact')

@app.route('/categories')
def categories():

```

```

    page = request.args.get('page', 1, type = int)
    products =
Product.query.order_by(Product.title.desc()).paginate(p
age = page, per_page = 8)
    categories = Category.query.all()
    return render_template('categories.html', title =
'Categories', products = products, categories =
categories)
@app.route('/sort/<string:category_name>', methods =
['POST','GET'])
def soft_products(category_name):
    page = request.args.get('page', 1, type = int)
    print(request.form.get("sort"))
    if 'sort' in session and request.form.get("sort") ==
None:
        sort = session['sort']
    else:
        sort = int(request.form.get("sort"))
        session['sort'] = sort
    print(sort)
    sorts = {0 : Product.title, 1: Product.title,2:
Product.date_posted, 3: Product.price}
    categories = Category.query.all()
    if category_name != 'All':

```

```

        category = Category.query.filter_by(name =
category_name).first_or_404()
        products = Product.query.filter_by(type =
category).order_by(sort[sort].desc()).paginate(page =
page, per_page = 8)
    else:
        category = None
        products =
Product.query.order_by(sort[sort].desc()).paginate(pa
ge = page, per_page = 8)
    return render_template('sort.html', title =
'Categories', products = products, categories =
categories, category = category, category_name =
category_name)
@app.route('/categories/<string:category_name>')
def category(category_name):
    page = request.args.get('page', 1, type = int)
    categories = Category.query.all()
    category = Category.query.filter_by(name =
category_name).first_or_404()
    products = Product.query.filter_by(type =
category).paginate(page = page, per_page = 8)
    return render_template('category.html', title =
'Categories', category = category, products = products,

```

```

categories = categories,category_name =
category_name)
@app.route('/product/<int:product_id>',
methods=['POST','GET'])
def product(product_id):
    product = Product.query.get_or_404(product_id)
    comments = Comment.query.filter_by(product =
product).order_by(Comment.date_posted.desc())
    print("product", product.title)
    if request.method == 'POST':
        quantity = int(request.form.get('quantity'))
        if "cart" in session:
            if not any(product.title in d for d in
session['cart']):
                session['cart'].append({product.title: quantity})
            elif any(product.title in d for d in session['cart']):
                for d in session['cart']:
                    if product.title in d:
                        d[product.title] += quantity
        else:
            session['cart'] = [{product.title: quantity}]
    print(session['cart'])
    session['quantity'] = 0
    for k in session['cart']:

```

```
        for d in k:
            session['quantity'] += k[d]
        flash(f'Adding to shopping cart successfully!',
'success')
        recommended_index_products =
recommender(product_id - 1)
        recommended_products = []
        for id in recommended_index_products:
            p = Product.query.get(id + 1)
            recommended_products.append(p)
        if 'content_error' in session:
            content_error = session['content_error']
        else:
            content_error = None
        return render_template('product.html', title =
'Product', product = product, recommended_products
= recommended_products, comments = comments,
content_error = content_error)
@app.route('/product/<int:product_id>/new_comment',
methods = ['POST', 'GET'])
@login_required
def new_comment(product_id):
    content = request.args.get('content')
    product = Product.query.get_or_404(product_id)
```

```
author = current_user
content_chatbot = str(bot.get_response(content))
comment = Comment(content = content, author =
author, product = product, content_chatbot =
content_chatbot)
db.session.add(comment)
db.session.commit()
flash('Adding new comment successfully!')
return redirect(url_for('product', product_id =
product.id))
@app.route('/checkout', methods=['GET','POST'])
def checkout():
    total = session['total']
    subtotal = session['subtotal']
    shipping = session['shipping']
    form = InforForm()
    if form.validate_on_submit():
        country = request.form.get('country_select')
        infor = Infor(name = form.name.data, address =
form.address.data, country = country, city =
form.city.data, postcode = form.postcode.data, phone
= form.phone.data, total_price = total)
        db.session.add(infor)
        db.session.commit()
```



```
item = []
if 'order' in session:
    for k in session['order']:
        product = Product.query.filter_by(title =
k).first()
        c1 = CartItem(product_id = product.id,
quantity = session['order'][k][2])
        item.append(c1)
    infor.cartitems.extend(item)
    db.session.add_all(item)
    db.session.commit()
    flash(f'You ordered successully!', 'success')
    session.pop('cart')
    session.pop('order')
    return redirect(url_for('home'))
    return render_template('checkout.html', title =
'Check Out', form = form, total = total, subtotal =
subtotal, shipping = shipping)
@app.route('/cart', methods=['POST','GET'])
def cart():
    subtotal = 0
    order = {}
    session['quantity'] = 0
    if 'cart' in session:
```

```
print(session['cart'])
for d in session['cart']:
    for k in d:
        product = Product.query.filter_by(title =
k).first()
        order[product.title] = []
        order[product.title].append(product.price)

order[product.title].append(product.image_file)
        order[product.title].append(d[k])
        order[product.title].append(product.price *
d[k])
        subtotal += product.price * d[k]
        session['quantity'] += d[k]
print(order)
session['order'] = order
print('You have your own items!')
shipping = 10
coupon = 0
total = subtotal + shipping
session['shipping'] = shipping
session['subtotal'] = subtotal
session['total'] = total
return render_template('cart.html', title = 'Cart', total
```

```
= total, subtotal = subtotal, shipping = shipping, order  
= order)
```

```
@app.route('/cart/remove/<string:product_title>',  
methods=['POST'])
```

```
def remove_from_cart(product_title):
```

```
    print('before', session['cart'])
```

```
    for i in session['cart']:
```

```
        if product_title in i:
```

```
            i.pop(product_title)
```

```
    print('after', session['cart'])
```

```
    return redirect(url_for('cart'))
```

```
@app.route('/cart/deleteall', methods=['POST'])
```

```
def delete_all():
```

```
    session.pop('cart')
```

```
    return redirect(url_for('cart'))
```

```
@app.route('/cart/update', methods=['POST'])
```

```
def update_cart():
```

```
    qty = request.form.get('update_qty')
```

```
    p = request.form.get('update_p')
```

```
    print(p, qty)
```

```
    print("update_cart")
```

```
    for i in session['cart']:
```

```
        if p in i:
```

```
            i.update({p:int(qty)})
```

```
    print(session['cart'])
    return redirect(url_for('cart'))
@app.route('/search', methods=['POST','GET'])
def search():
    index_name = 'fashionshop'
    doc_type = 'product'
    query = request.form.get('query')
    print(query)
    query = es.search(index = index_name,
body={'query':{'match': {'title': query}}})
    found = query['hits']['total']['value']
    products = []
    print(query['hits']['hits'])
    for item in query['hits']['hits']:
        product= Product.query.filter_by(title =
item['_source']['title']).first()
        print(product)
        products.append(product)
    print(products)
    return render_template('search.html', products =
products, found = found)
@app.route('/logout')
def logout():
    logout_user()
```

```
    return redirect(url_for('home'))
@app.route('/user', methods = ['GET','POST'])
@login_required
def user():
    form = UserForm()
    if form.validate_on_submit():
        current_user.username = form.username.data
        current_user.email = form.email.data
        current_user.birthday = form.birthday.data
        current_user.gender = form.gender.data
        db.session.commit()
        print('Save successfully!')
        flash(f'Your account information has
updated!','success')
    elif request.method == 'GET':
        form.username.data = current_user.username
        form.email.data = current_user.email
        form.birthday.data = current_user.birthday
        form.gender.data = current_user.gender
    return render_template('user.html', title='User',
```

8.TESTING

8.1 Test Cases

A test case is a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.

8.2 User Acceptance Testing

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.

9.RESULTS

10.ADVANTAGES & DISADVANTAGES

Advantages

For customers, recommender systems can help them find items which they are interested in. For enterprises, recommender systems can improve the loyalty of their customers by enhancing the user experience and further convert more browsers to consumers.

Disadvantages

Perhaps the biggest issue facing recommender systems is that they need a lot of data to effectively make recommendations.

11.CONCLUSION

In this project, we have presented a novel framework for fashion recommendation that is driven by data, visually related and simple effective recommendation systems for generating fashion product images. The proposed approach uses a two-stage phase. Initially, our proposed approach extracts the features of the image using CNN classifier ie., for instance allowing the customers to upload any random fashion image from any E-commerce website and later generating similar images to the uploaded image based on the features and texture of the input image. It is imperative that such research goes forward to facilitate greater recommendation accuracy and improve the overall experience of fashion exploration for direct and indirect consumers alike.

12. FUTURE SCOPE

The textile and apparel industries have grown tremendously over the last years. Customers no longer have to visit many stores, stand in long queues, or try on garments in dressing rooms as millions of products are now available in online catalogs. However, given the plethora of options available, an effective recommendation system is necessary to properly sort, order, and communicate relevant product material or information to users. Effective fashion RS can have a noticeable impact on billions of customers' shopping experiences and increase sales and revenues on the provider-side. The goal of this survey is to provide a review of recommender systems that operate in the specific vertical domain of garment and fashion products. We have identified the most pressing challenges in fashion RS research and created a taxonomy that categorizes the literature according to the objective they are trying to accomplish (e.g., item or outfit recommendation, size recommendation, explainability, among others) and type of side-information (users, items, context). We have also identified the most important evaluation goals and perspectives (outfit generation, outfit recommendation, pairing recommendation, and fill-in-the-blank outfit compatibility prediction) and the most commonly used datasets and evaluation metrics.

13. APPENDIX

GitHub link

<https://github.com/IBM-EPBL/IBM-Project-40450-1660629565>

Project Demo Link

<https://drive.google.com/file/d/1UWPFwugjDvkw4ZXYmeEVYGMSnmlcoPxd/view>