Team ID: PNT2022TMID20463

Project Name: SMART FASHION RECOMMENDER APPLICATION

Team Leader: Mohammed Azeer

Team Member 1: Menati Vasudeva Reddy

Team Member 2: Loghadharshann D

Team Member 3: Rajavignesh M

Abstract

Nowadays, fashion applications and e-commerce are growing more and more.And it also has some problems when finding the customer's wanted product in the web applications. Having a chatbot that understands the algorithm of a specific application can be of great aid. We are implementing such a chat bot in a web application, 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 by getting simple user information and activities. The application also has two main UI interactions: one is the user panel and the other one is the admin panel. Users can interact with the chat bot to search for products, order them from the manufacturer or distributor through chatbot AI, and it can also make payment transactions, track the delivery, and so on. The admin interface enables the user to upload products' details ,user details, orders and find how many products have been bought, supervise the stock availability, and interact with the buyer regarding the product reviews.

Features of Chatbot :

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

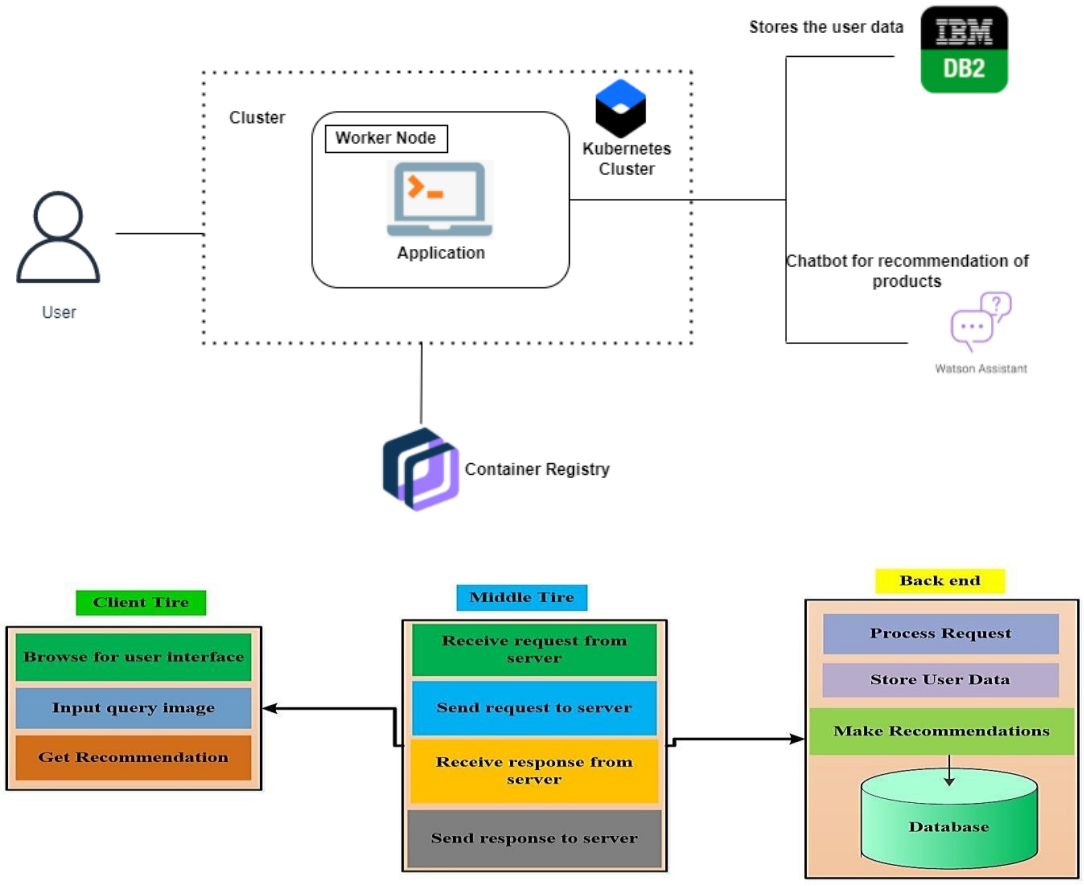
Introduction

During the last years, online shopping has been growing. Both the customer and the business enterprise desire the client to easily discover applicable products or items both throughout search and when they are searching, and this is where recommender systems come into the picture. Recommendation systems make recommendations based on the information they are provided with and in the manner in which they are programmed. Going into details, most of the evaluation applied is independent coming up with a brand-new recommendation algorithm, system, or model. However, different researchers use already existingwork as researchers use an already existingcurrent piece of work to come up with a new diagramor to truly improve the current one. The present analysis model focuses on the use of a current algorithmic program and, consequently, the use of a new research concept comes up with a recommender system.Existing research and fashions have given us some inspirations of how to design fashion recommendation systems. The contribution of the research are follows:

- A scheme for improving a person's clothingstyle by removingthe features he/shedoesn't like from his/herclothing images.
- These attributes served to a similar model to retrievesimilar images as recommendations.
- Combined with more common content-based recommendation systems, our model can help to extend robustness and performance, for example can suit a more pretentious style of a client.

● Proposed system architecture

The system architecture defines the hardware, software and network environment of the structure. The system will be web-based meaning that the users need to run the URL in order to runthe system.The system will run both horizontally and vertically. The architecture used in thesystem is shown horizontally where the Model View Controller is explained as represented in Figure 2. The high-level part of the system is looked at using the vertical way.



The system comprises of the Client tire, which is the front end or View mode, middle tier which is the system controller and the backend tire which is the model. The client side is where the users/customers log in in the system, browse for the system interface, provide input query image to the system, and get recommendation accordingto the input query. The middle true is responsible for communication betweenthe front end and the back end. It receivesuser requests and sends them to the back end and in turn accepts

responses from the backend and sends them to the user.

The back end which involves the data set and recommender algorithm deals with data storage, user input data storage, processing user requests, determining user input similarity, making recommendations and forwarding them to the middle tier which in turn sends them to the respective users. The internet works to provide access to the site with a strong security check, provided by both firewall and password protection policy. Any unauthorized access is detected and prevented by the firewall.

1.2 PURPOSE

Clothing is a kind of symbol that represents people's internal perceptions through their outer appearance. It conveys information about their choices, faith, personality, profession, social status, and attitude towards life. Therefore, clothing is believed to be a nonverbal way of communicating and a major part of people's outer appearance. Recent technological advancements have enabled consumers to track current fashion trends around the globe, which influence their choices [2,3]. The fashion choices of consumers depend on many factors, such as demographics, geographic location, individual preferences, interpersonal influences, age, gender, season, and culture.

CHAPTER 2

LITERATURE REVIEW

[1] devised a parametric distance transformation that assigns a lower distance to garment pairings that fit well than to those that do not. And provided Image-based recommendations on styles and substitutes.

[2] conducted a preliminary investigation into personalised outfit recommendation. To

describe the user-item and item-item interactions, a functional tensor factorization method was presented. They proposed A functional tensor factorization approach. Veit et al.

[4] learned feature transformation for a compatibility measure between pairs of objects using a Siamese CNN architecture. All of these works focused solely on the compatibility of two things. Furthermore, they simply modelled broad matching criteria and ignored the issue of personalisation.

Thombre in [3] used image segmentation and Kalman filter to realize Human detection and tracking. Orrite-Urunuela proposed a statistical model for detection and tracking of human silhouette and the corresponding 3D skeletal structure in gait sequences [5]. How-Lung [6] provided an outdoor aquatic surveillance system for human motion tracking and detection. Ajmani et al.

[7] present a novel method for content based recommendation of media-rich commodities with the use of probabilistic multimedia ontology. Proposed an ontology based personalized garment recommendation system.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

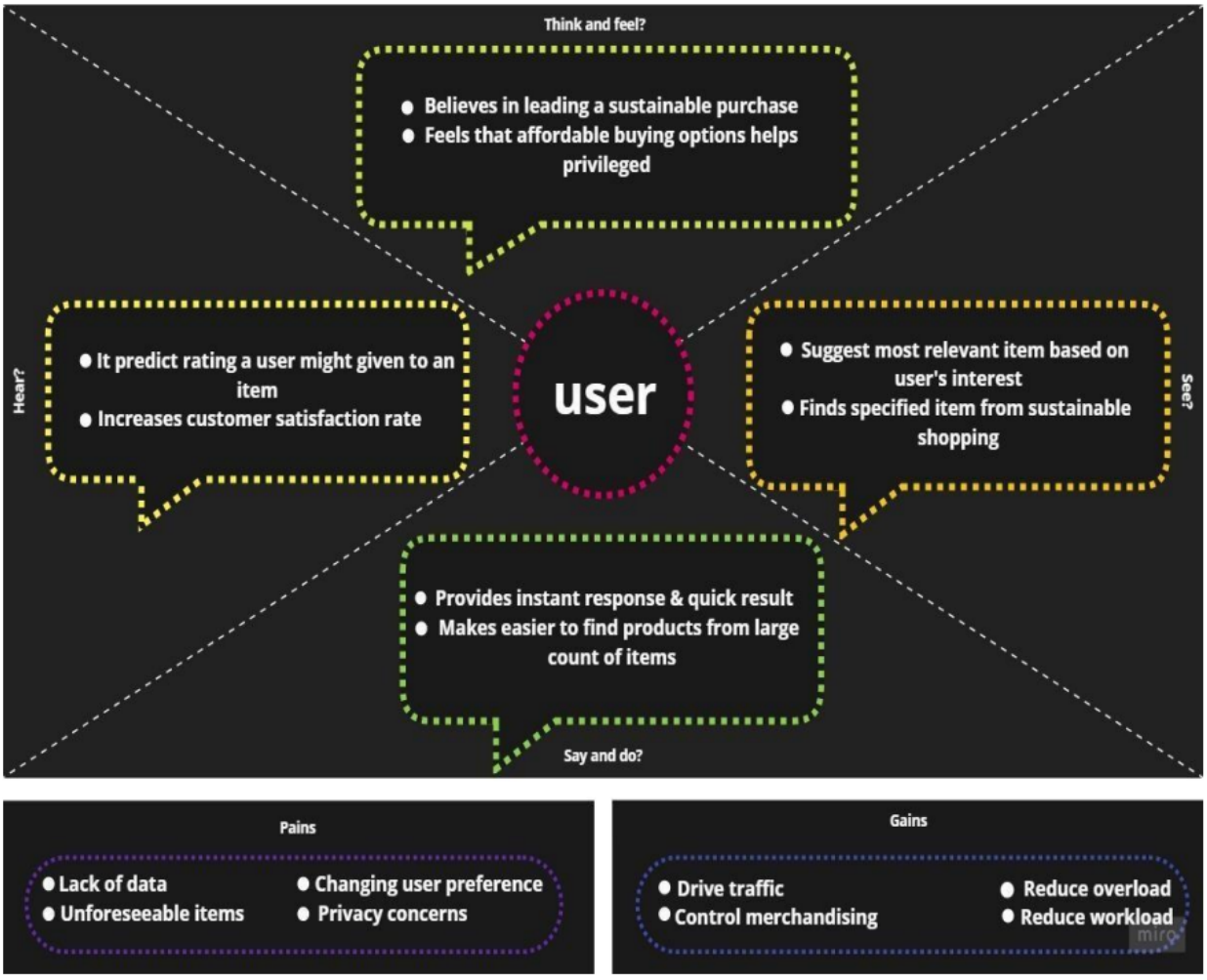
3.1 Empathy Map Canvas

Goals and Objectives

▪ Recommendation system is to suggest relevant items to the user.

▪ To predict user's interest and recommend relevant items that quietly likely interesting

for them.



3.2 Ideation & Brainstorming

Step-1:Team Gathering, Collaboration and select the Problem statement

o Team Members – MOHAMMED AZEER, MENATI VASUDEVA REDDY, LOGHADHARSHANN D, RAJAVIGNESH M

o Goal – sorting data sparsity and lack of data analytics problem

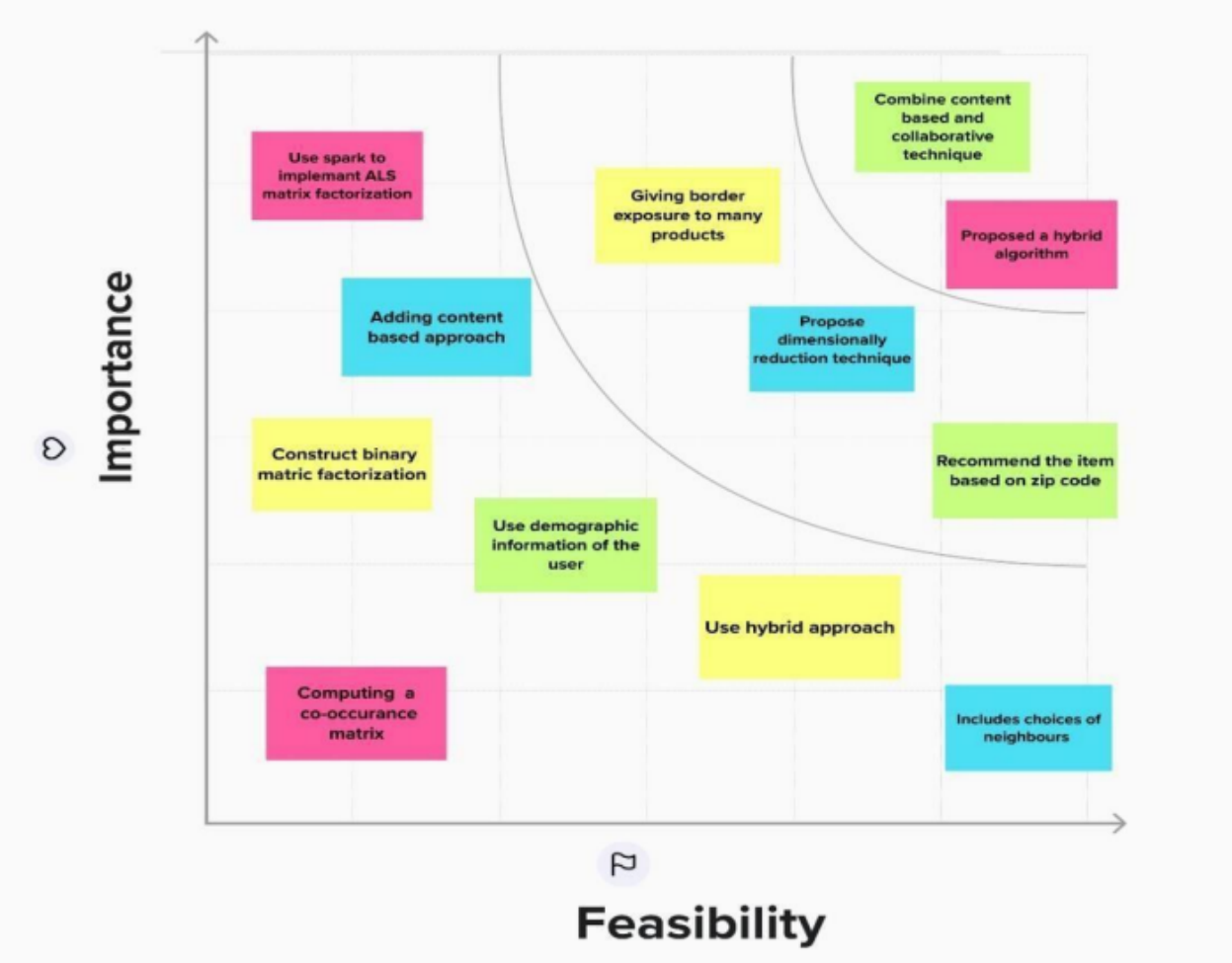
o Problem statement

➢ In recommendation system,sometimes recommend different itemsthanuser's preference.

➢ Privacy with minimum imposition of accuracy loss on the recommendation.

Step-2: Brainstorm, Idea Listing and Grouping

step-3: Idea Prioritization



Brainstorming Ideation

3.3 Proposed Solution

1)Problem Statement (Problem to be solved)

Improving users privacy with minimum imposition of accuracy loss on the recommendation. ➢ In the recommendation system the problem is trying to forecast the opinion of user will have on dissimilar substance and be able to recommend the finest items to each user. ➢ Another problems are data sparsity, gray sheep and scalability. ➢ Lack of data analytic capability.

2. Idea / Solution description

➢ System will recommend the items based on the zip code.

➢ As the user click on link of any item a time session will be started to record how much time he hasspent on particular page.

➢ When the time spent by the user crosses a certain threshold time, rating of the particular item will be increased by some measures.

➢ Distributed aggregation of user's profile.

3. Novelty / Uniqueness

➢ Recommender system are very powerful to help a user find good products or items. the important thing the goals of recommender systems are to serve information

4. Social Impact / Customer Satisfaction

➢ Recommender systems has consistently suggested that customer satisfaction will be highest when the recommendation algorithm is accurate and recommends a diversity of items.

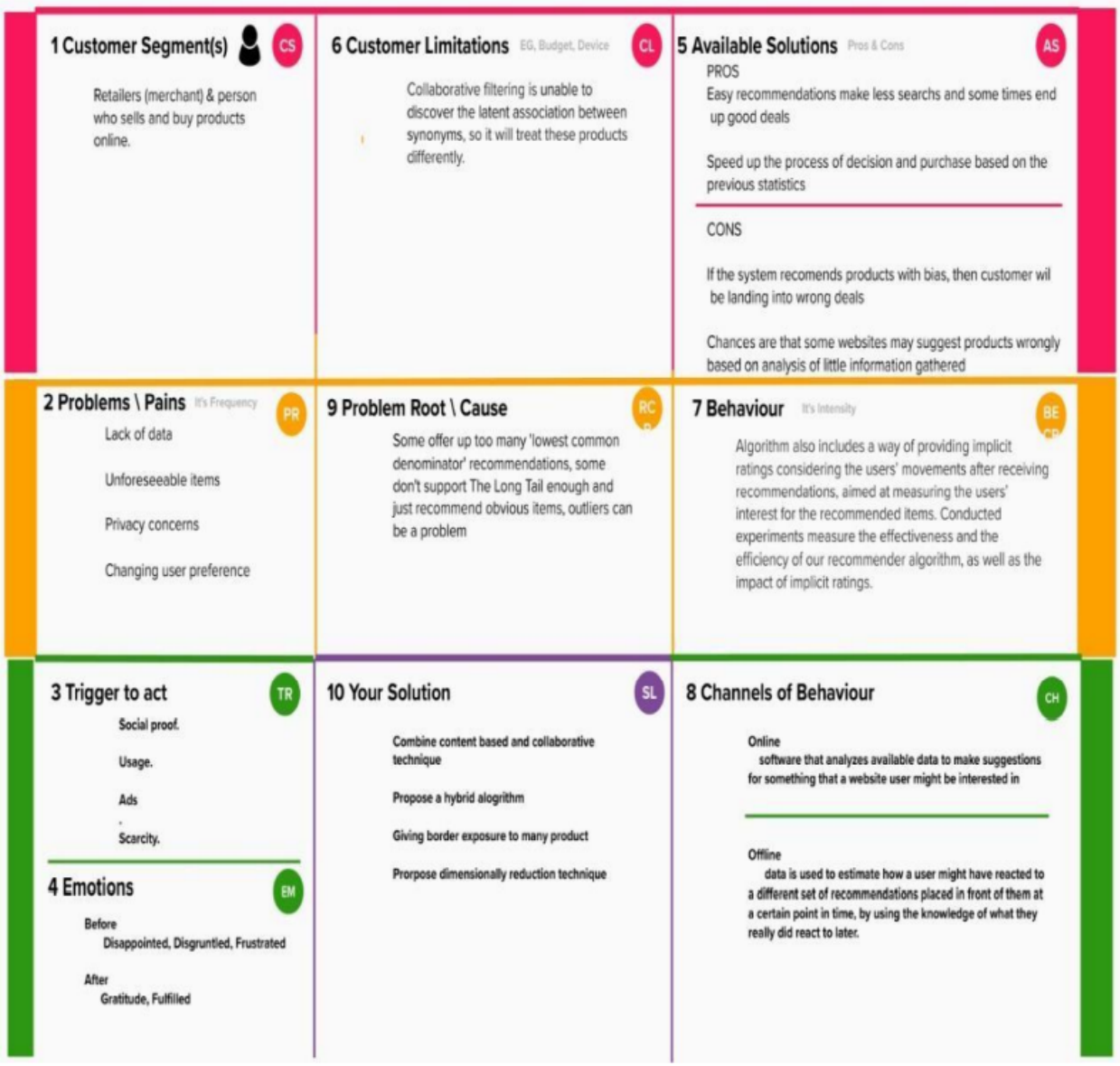
5. Business Model (Revenue Model)

➢ Researchers have studied and generate many algorithms to learn increasing rate for an online customer like Amazon site. Also, These algorithms study the difference between shopping online sites with others using recommendersystems for items to increase revenue by increasing the number ofsales.

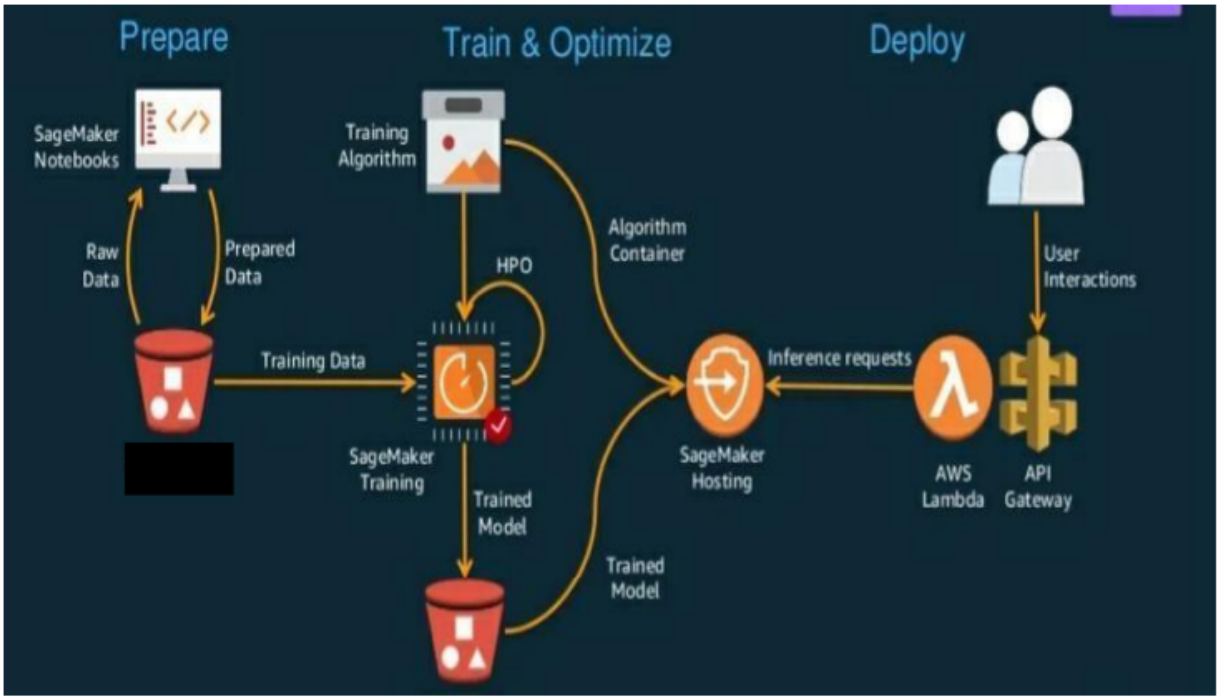
6. Scalability of the Solution

➢ A recommendation technique that is efficient when the number of dataset is limited may be unable to generate satisfactory number of recommendations when the volume of dataset is increased

3.4 Problem Solution fit



Solution architecture



CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional requirement

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 mail Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login using username & Password
FR-4	Personal Details	Personal details through Form Personal details through UI Tab
FR-5	Delivery Confirmation	Confirmation via Email Confirmation via Phone

4.2 Non-Functional requirements

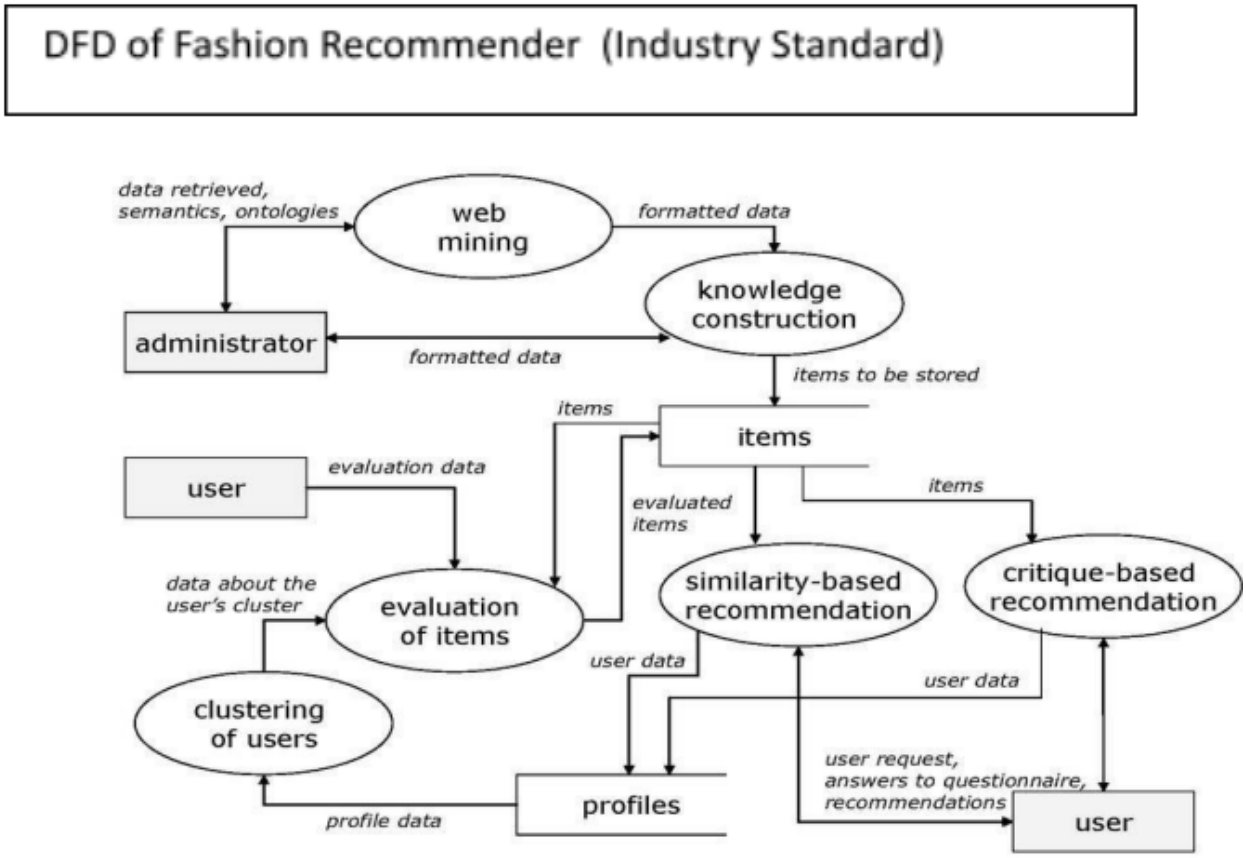
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Ease of use of the application for the user
NFR-2	**Security**	User privacy is the highest priority of the application. Security measures are undertaken for the user
NFR-3	**Reliability**	It can handle more than 2000 users at a time. It can process and initialize most functions.
NFR-4	**Performance**	The application can handle complex tasks and supports multi-tasking.
NFR-5	**Availability**	It is a free web and application available on all platforms.
NFR-6	**Scalability**	With higher workloads the user will experience a 10 to 17% drop in performance.

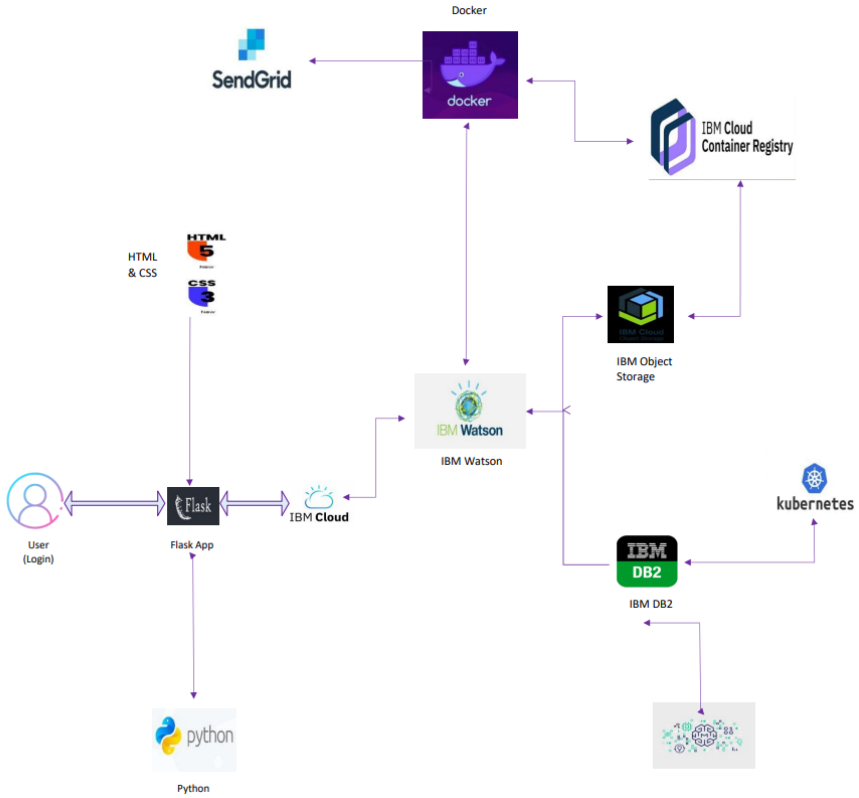
CHAPTER 5

PROJECT DESIGN

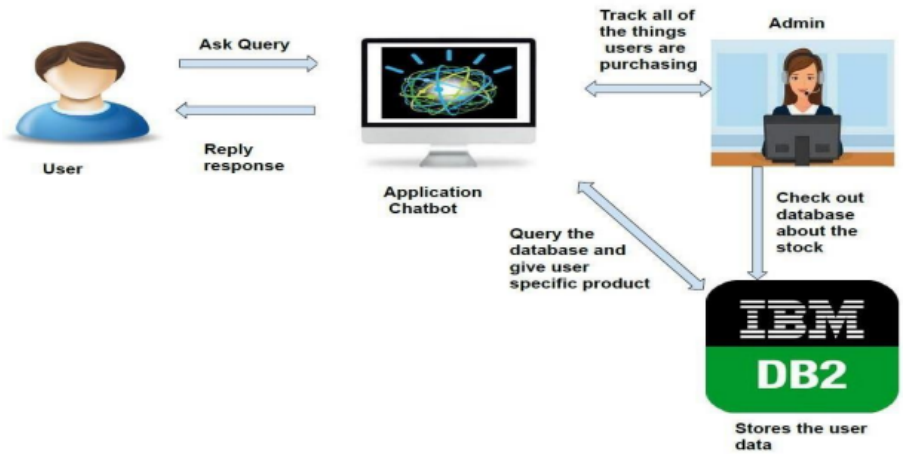
5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories



CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story points	Priority	Team Members
Sprint-1	Setting up App environment	USN-1	As a user, I can register in ICTA Academy and create IBM cloud account.	2	High	MENATI VASUDEVA REDDY MOHAMMAD AZEER
Sprint-1		USN-2	As a user, I will create a flask project	1	Low	LOGHA DHARSHANN D RAJA VIGNESH M
Sprint-1		USN-3	As a user, I will install IBM CloudCLI	2	Medium	LOGHA DHARSHANN D MENATI VASUDEVA REDDY
Sprint-2	Setting up App environment	USN-4	As a user, I can install Docker CLI	1	Low	MOHAMMAD AZEER RAJA VIGNESH M
Sprint-2		USN-5	As a user, I will Create an accountin sendgrid	2	Medium	LOGHA DHARSHANN D MOHAMMAD AZEER
Sprint-3	Implementing web application	USN-6	As a user, I Create UI to interactwith the application	1	High	MENATI VASUDEVA REDDY MOHAMMAD AZEER
Sprint-3		USN-7	As a user, I Create IBM DB2 and connect with Python	3	High	LOGHA DHARSHANN D
Sprint-3	Integrating sendgridservice	USN-8	As a user, I will integrating sendgridwith python code	2	High	MOHAMMAD AZEER
Sprint-3	Developing a chatbot	USN-9	As a user, I have to build a chatbotand Integrate to application	1	Medium	MENATI VASUDEVA REDDY
Sprint-4	Development of App inIBM Cloud	USN-10	As a user, I will Containerize theApp	1	Low	RAJA VIGNESH M
Sprint-4		USN-11	As a user, I will upload image toIBM Container registry	2	Medium	LOGHA DHARSHANN D
Sprint-4		USN-12	As a user, I will deploy App in Kebernetes cluster	3	High	MOHAMMAD AZEER
Sprint-4	User panel		As a user ● Register, Login, Email, Verification ● Manual Search ● Order placement, Order Details	3	High	MENATI VASUDEVA REDDY LOGHA DHARSHANN D RAJA VIGNESH M

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	18	6 Days	24 Oct 2022	29 Oct 2022	24	29 Oct 2022
Sprint-2	18	6 Days	31 Oct 2022	05 Nov 2022	24	05 Nov 2022
Sprint-3	18	6 Days	07 Nov 2022	12 Nov 2022	24	12 Nov 2022
Sprint-4	18	6 Days	14 Nov 2022	19 Nov 2022	24	19 Nov 2022

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Features

```
from cs50 import SQL
from flask_session import Session
from flask import Flask, render_template, redirect, request, session, jsonify
from datetime import datetime

# # Instantiate Flask object named app
app = Flask(__name__)

# # Configure sessions
app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"
Session(app)

# Creates a connection to the database
db = SQL ( "sqlite:///data.db" )



@app.route("/")
def index():
    shirts = db.execute("SELECT * FROM shirts ORDER BY onSalePrice")
```

```
    shirtsLen = len(shirts)
    # Initialize variables
    shoppingCart = []
    shopLen = len(shoppingCart)
    totItems, total, display = 0, 0, 0
    if 'user' in session:
        shoppingCart = db.execute("SELECT samplename, image, SUM(qty), SUM(subTotal), price, id FROM
cart GROUP BY samplename")
        shopLen = len(shoppingCart)
        for i in range(shopLen):
            total += shoppingCart[i]["SUM(subTotal)"]
            totItems += shoppingCart[i]["SUM(qty)"]
        shirts = db.execute("SELECT * FROM shirts ORDER BY onSalePrice ASC")
        shirtsLen = len(shirts)
        return render_template ("index.html", shoppingCart=shoppingCart, shirts=shirts, shopLen=shopLen,
shirtsLen=shirtsLen, total=total, totItems=totItems, display=display, session=session )
    return render_template ( "index.html", shirts=shirts, shoppingCart=shoppingCart, shirtsLen=shirtsLen,
shopLen=shopLen, total=total, totItems=totItems, display=display)



@app.route("/buy/")
def buy():
    # Initialize shopping cart variables
    shoppingCart = []
    shopLen = len(shoppingCart)
    totItems, total, display = 0, 0, 0
    qty = int(request.args.get('quantity'))
    if session:
        # Store id of the selected shirt
        id = int(request.args.get('id'))
        # Select info of selected shirt from database
        goods = db.execute("SELECT * FROM shirts WHERE id = :id", id=id)
        # Extract values from selected shirt record
        # Check if shirt is on sale to determine price
        if(goods[0]["onSale"] == 1):
            price = goods[0]["onSalePrice"]
```

```
        else:
            price = goods[0]["price"]
        samplename = goods[0]["samplename"]
        image = goods[0]["image"]
        subTotal = qty * price
        # Insert selected shirt into shopping cart
        db.execute("INSERT INTO cart (id, qty, samplename, image, price, subTotal) VALUES (:id, :qty,
:samplename, :image, :price, :subTotal)", id=id, qty=qty, samplename=samplename, image=image,
price=price, subTotal=subTotal)
        shoppingCart = db.execute("SELECT samplename, image, SUM(qty), SUM(subTotal), price, id FROM
cart GROUP BY samplename")
        shopLen = len(shoppingCart)
        # Rebuild shopping cart
        for i in range(shopLen):
            total += shoppingCart[i]["SUM(subTotal)"]
            totItems += shoppingCart[i]["SUM(qty)"]
        # Select all shirts for home page view
        shirts = db.execute("SELECT * FROM shirts ORDER BY samplename ASC")
        shirtsLen = len(shirts)
        # Go back to home page
        return render_template ("index.html", shoppingCart=shoppingCart, shirts=shirts, shopLen=shopLen,
shirtsLen=shirtsLen, total=total, totItems=totItems, display=display, session=session )

@app.route("/update/")
def update():
    # Initialize shopping cart variables
    shoppingCart = []
    shopLen = len(shoppingCart)
    totItems, total, display = 0, 0, 0
    qty = int(request.args.get('quantity'))
    if session:
        # Store id of the selected shirt
        id = int(request.args.get('id'))
        db.execute("DELETE FROM cart WHERE id = :id", id=id)
        # Select info of selected shirt from database
        goods = db.execute("SELECT * FROM shirts WHERE id = :id", id=id)
```

```
        # Extract values from selected shirt record
        # Check if shirt is on sale to determine price
        if(goods[0]["onSale"] == 1):
            price = goods[0]["onSalePrice"]
        else:
            price = goods[0]["price"]
        samplename = goods[0]["samplename"]
        image = goods[0]["image"]
        subTotal = qty * price
        # Insert selected shirt into shopping cart
        db.execute("INSERT INTO cart (id, qty, samplename, image, price, subTotal) VALUES (:id, :qty, :samplename, :image, :price, :subTotal)", id=id, qty=qty, samplename=samplename, image=image, price=price, subTotal=subTotal)
        shoppingCart = db.execute("SELECT samplename, image, SUM(qty), SUM(subTotal), price, id FROM cart GROUP BY samplename")
        shopLen = len(shoppingCart)
        # Rebuild shopping cart
        for i in range(shopLen):
            total += shoppingCart[i]["SUM(subTotal)"]
            totItems += shoppingCart[i]["SUM(qty)"]
        # Go back to cart page
        return render_template ("cart.html", shoppingCart=shoppingCart, shopLen=shopLen, total=total, totItems=totItems, display=display, session=session )

@app.route("/filter/")
def filter():
    if request.args.get('typeClothes'):
        query = request.args.get('typeClothes')
        shirts = db.execute("SELECT * FROM shirts WHERE typeClothes = :query ORDER BY samplename ASC", query=query )
    if request.args.get('sale'):
        query = request.args.get('sale')
        shirts = db.execute("SELECT * FROM shirts WHERE onSale = :query ORDER BY samplename ASC", query=query)
    if request.args.get('id'):
        query = int(request.args.get('id'))
```

```
        shirts = db.execute("SELECT * FROM shirts WHERE id = :query ORDER BY samplename ASC",
query=query)
    if request.args.get('kind'):
        query = request.args.get('kind')
        shirts = db.execute("SELECT * FROM shirts WHERE kind = :query ORDER BY samplename ASC",
query=query)
    if request.args.get('price'):
        query = request.args.get('price')
        shirts = db.execute("SELECT * FROM shirts ORDER BY onSalePrice ASC")
    shirtsLen = len(shirts)
    # Initialize shopping cart variables
    shoppingCart = []
    shopLen = len(shoppingCart)
    totItems, total, display = 0, 0, 0
    if 'user' in session:
        # Rebuild shopping cart
        shoppingCart = db.execute("SELECT samplename, image, SUM(qty), SUM(subTotal), price, id FROM
cart GROUP BY samplename")
        shopLen = len(shoppingCart)
        for i in range(shopLen):
            total += shoppingCart[i]["SUM(subTotal)"]
            totItems += shoppingCart[i]["SUM(qty)"]
        # Render filtered view
        return render_template ("index.html", shoppingCart=shoppingCart, shirts=shirts, shopLen=shopLen,
shirtsLen=shirtsLen, total=total, totItems=totItems, display=display, session=session )
    # Render filtered view
    return render_template ( "index.html", shirts=shirts, shoppingCart=shoppingCart, shirtsLen=shirtsLen,
shopLen=shopLen, total=total, totItems=totItems, display=display)


@app.route("/checkout/")
def checkout():
    order = db.execute("SELECT * from cart")
    # Update purchase history of current customer
    for item in order:
        db.execute("INSERT INTO purchases (uid, id, samplename, image, quantity) VALUES(:uid, :id,
```

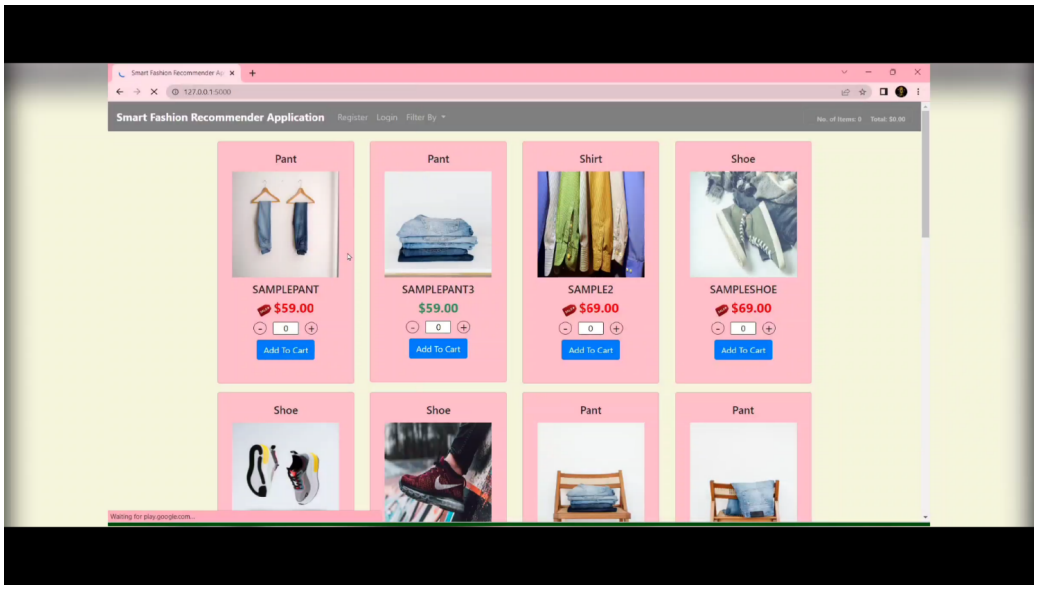
```
:samplename, :image, :quantity)", uid=session["uid"], id=item["id"], samplename=item["samplename"],
image=item["image"], quantity=item["qty"] )
    # Clear shopping cart
    db.execute("DELETE from cart")
    shoppingCart = []
    shopLen = len(shoppingCart)
    totItems, total, display = 0, 0, 0
    # Redirect to home page
    return redirect('/')


@app.route("/remove/", methods=["GET"])
def remove():
    # Get the id of shirt selected to be removed
    out = int(request.args.get("id"))
    # Remove shirt from shopping cart
    db.execute("DELETE from cart WHERE id=:id", id=out)
    # Initialize shopping cart variables
    totItems, total, display = 0, 0, 0
    # Rebuild shopping cart
    shoppingCart = db.execute("SELECT samplename, image, SUM(qty), SUM(subTotal), price, id FROM cart
GROUP BY samplename")
    shopLen = len(shoppingCart)
    for i in range(shopLen):
        total += shoppingCart[i]["SUM(subTotal)"]
        totItems += shoppingCart[i]["SUM(qty)"]
    # Turn on "remove success" flag
    display = 1
    # Render shopping cart
    return render_template ("cart.html", shoppingCart=shoppingCart, shopLen=shopLen, total=total,
totItems=totItems, display=display, session=session )

@app.route("/login/", methods=["GET"])
def login():
    return render_template("login.html")
```

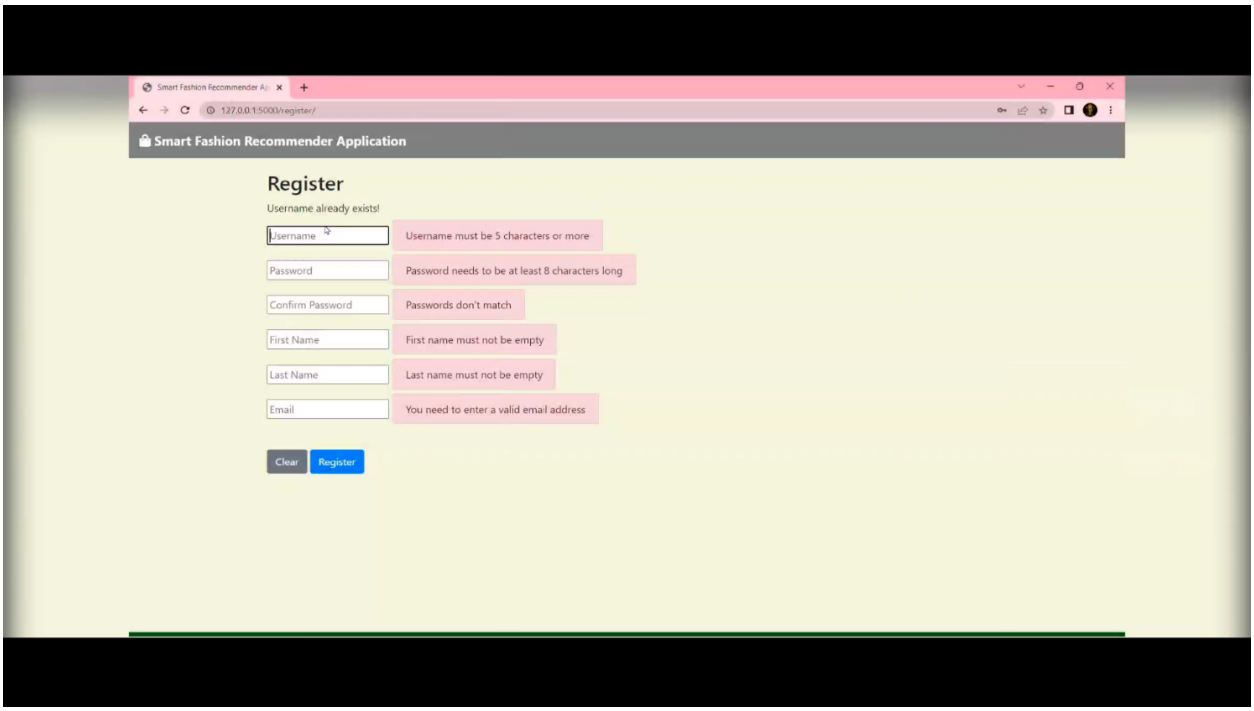
```
@app.route("/new/", methods=["GET"])
def new():
    # Render log in page
    return render_template("new.html")

@app.route("/logged/", methods=["POST"] )
def logged():
    # Get log in info from log in form
    user = request.form["username"].lower()
    pwd = request.form["password"]
    #pwd = str(sha1(request.form["password"].encode('utf-8')).hexdigest())
    # Make sure form input is not blank and re-render log in page if blank
    if user == "" or pwd == "":
        return render_template ( "login.html" )
    # Find out if info in form matches a record in user database
    query = "SELECT * FROM users WHERE username = :user AND password = :pwd"
    rows = db.execute ( query, user=user, pwd=pwd )

    # If username and password match a record in database, set session variables
    if len(rows) == 1:
        session['user'] = user
        session['time'] = datetime.now( )
        session['uid'] = rows[0]["id"]
    # Redirect to Home Page
    if 'user' in session:
        return redirect ( "/" )
    # If username is not in the database return the log in page
    return render_template ( "login.html", msg="Wrong username or password." )

@app.route("/history/")
def history():
    # Initialize shopping cart variables
    shoppingCart = []
    shopLen = len(shoppingCart)
    totItems, total, display = 0, 0, 0
    # Retrieve all shirts ever bought by current user
```

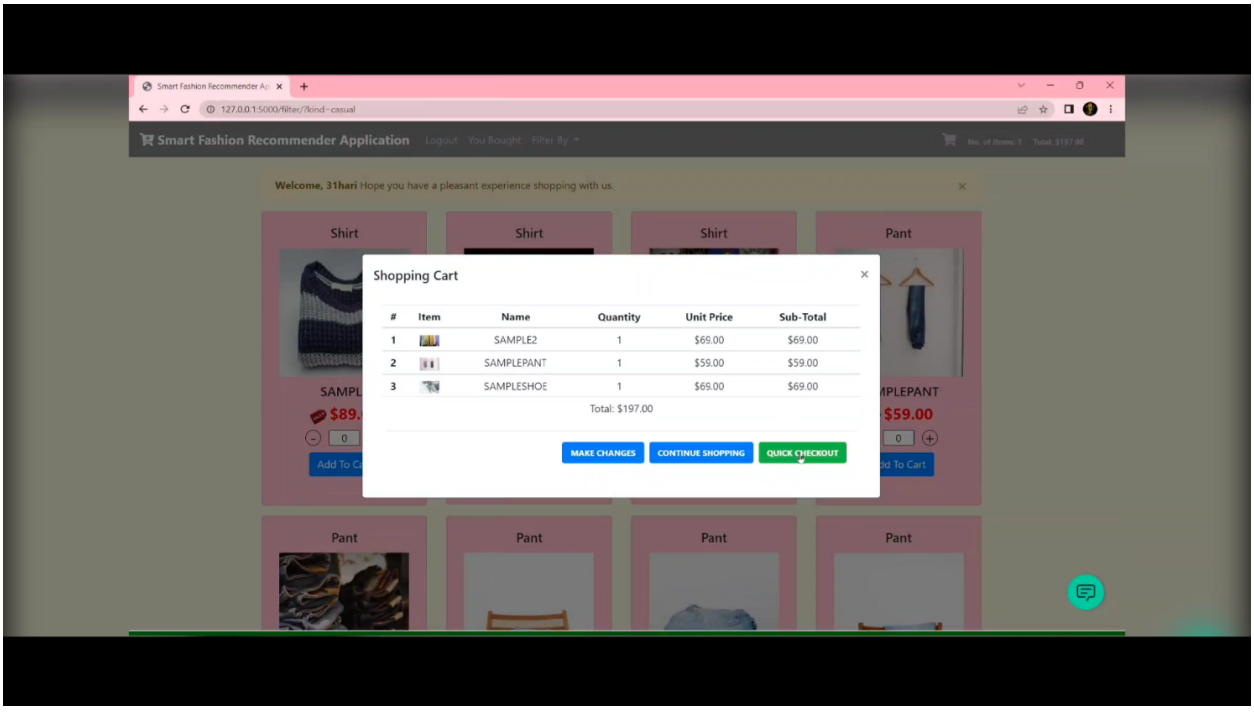
```
    myShirts = db.execute("SELECT * FROM purchases WHERE uid=:uid", uid=session["uid"])
    myShirtsLen = len(myShirts)
    # Render table with shopping history of current user
    return render_template("history.html", shoppingCart=shoppingCart, shopLen=shopLen, total=total,
totItems=totItems, display=display, session=session, myShirts=myShirts, myShirtsLen=myShirtsLen)

@app.route("/logout/")
def logout():
    # clear shopping cart
    db.execute("DELETE from cart")
    # Forget any user_id
    session.clear()
    # Redirect user to login form
    return redirect("/")

@app.route("/register/", methods=["POST"] )
def registration():
    # Get info from form
    username = request.form["username"]
    password = request.form["password"]
    confirm = request.form["confirm"]
    fname = request.form["fname"]
    lname = request.form["lname"]
    email = request.form["email"]
    # See if username already in the database
    rows = db.execute( "SELECT * FROM users WHERE username = :username ", username = username )
    # If username already exists, alert user
    if len( rows ) > 0:
        return render_template ( "new.html", msg="Username already exists!" )
    # If new user, upload his/her info into the users database
    new = db.execute ( "INSERT INTO users (username, password, fname, lname, email) VALUES (:username,
:password, :fname, :lname, :email)",
                username=username, password=password, fname=fname, lname=lname, email=email )
    # Render login template
    return render_template ( "login.html" )
```

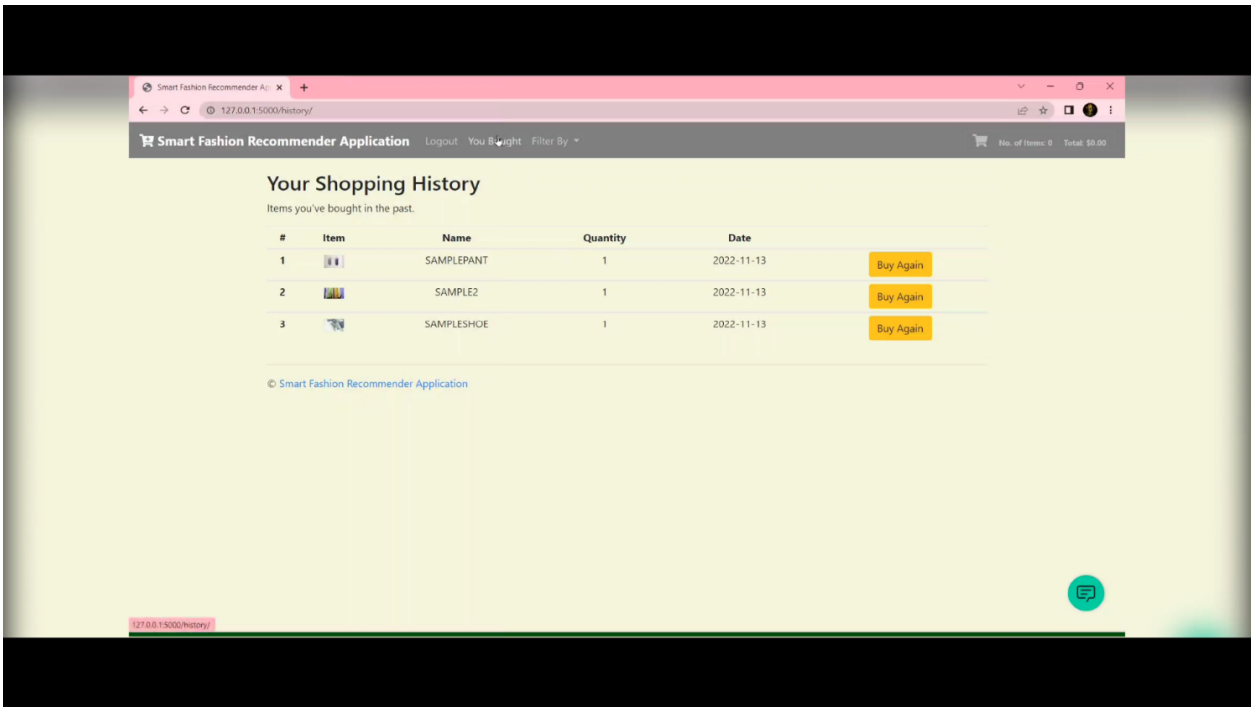
```
@app.route("/cart/")
def cart():
    if 'user' in session:
        # Clear shopping cart variables
        totItems, total, display = 0, 0, 0
        # Grab info currently in database
        shoppingCart = db.execute("SELECT samplename, image, SUM(qty), SUM(subTotal), price, id FROM
cart GROUP BY samplename")
        # Get variable values
        shopLen = len(shoppingCart)
        for i in range(shopLen):
            total += shoppingCart[i]["SUM(subTotal)"]
            totItems += shoppingCart[i]["SUM(qty)"]
    # Render shopping cart
    return render_template("cart.html", shoppingCart=shoppingCart, shopLen=shopLen, total=total,
totItems=totItems, display=display, session=session)
```

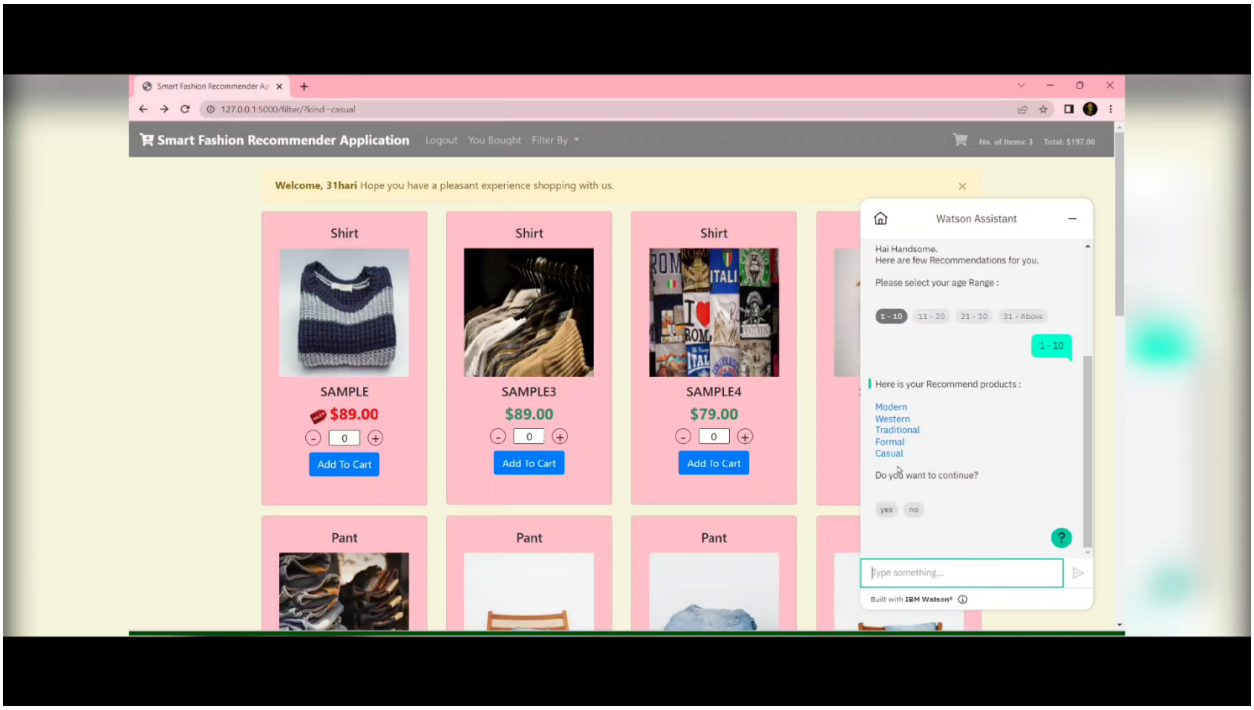
OUTPUT SCREENSHOTS




Smart Fashion Recommender Application
Register
Username already exists!
Username
Username must be 5 characters or more
Password
Password needs to be at least 8 characters long
Confirm Password
Passwords don't match
First Name
First name must not be empty
Last Name
Last name must not be empty
Email
You need to enter a valid email address
Clear
Register


Smart Fashion Recommender Application
Logout
You Bought
Filter By
Welcome, 31hari Hope you have a pleasant experience shopping with us.
Shirt
Shirt
Shirt
Pant
Shopping Cart
Item Name Quantity Unit Price Sub-Total
1 SAMPLE2 1 $69.00 $69.00
2 SAMPLEPANT 1 $59.00 $59.00
3 SAMPLESHOE 1 $69.00 $69.00
Total: $197.00
MAKE CHANGES
CONTINUE SHOPPING
QUICK CHECKOUT
$59.00
Pant
Pant
Pant
Pant


Smart Fashion Recommender Application
Logout
You Bought
Filter By
No. of items: 0
Total: $0.00
Your Shopping History
Items you've bought in the past.
Item Name Quantity Date
1 SAMPLEPANT 1 2022-11-13 Buy Again
2 SAMPLE2 1 2022-11-13 Buy Again
3 SAMPLESHOE 1 2022-11-13 Buy Again
© Smart Fashion Recommender Application

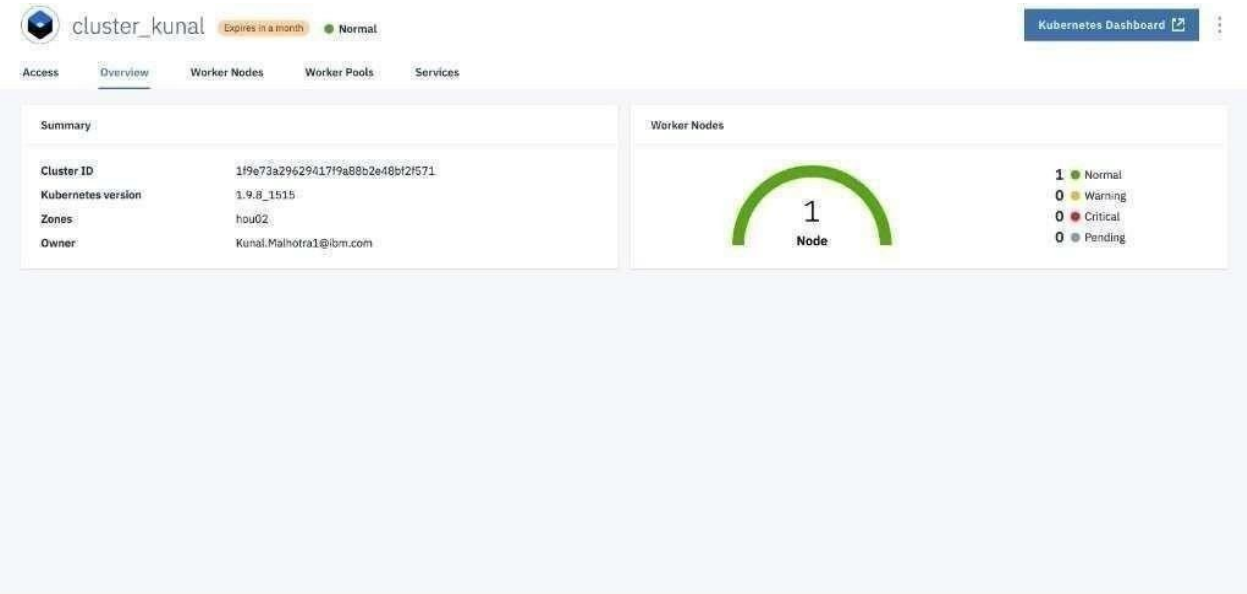

Smart Fashion Recommender Application
Logout
You Bought
Filter By
No. of items: 3
Total: $197.00
Welcome, 31hari Hope you have a pleasant experience shopping with us.
Shirt
SAMPLE
$89.00
Add To Cart
Shirt
SAMPLE3
$89.00
Add To Cart
Shirt
SAMPLE4
$79.00
Add To Cart
Pant
Pant
Pant
Watson Assistant
Hai Handsome.
Here are few Recommendations for you.
Please select your age Range :
1 - 10
11 - 20
21 - 30
31 - Above
1 - 10
Here is your Recommend products :
Modern
Western
Traditional
Formal
Casual
Do you want to continue?
yes
no
Type something...
Built with IBM Watson®

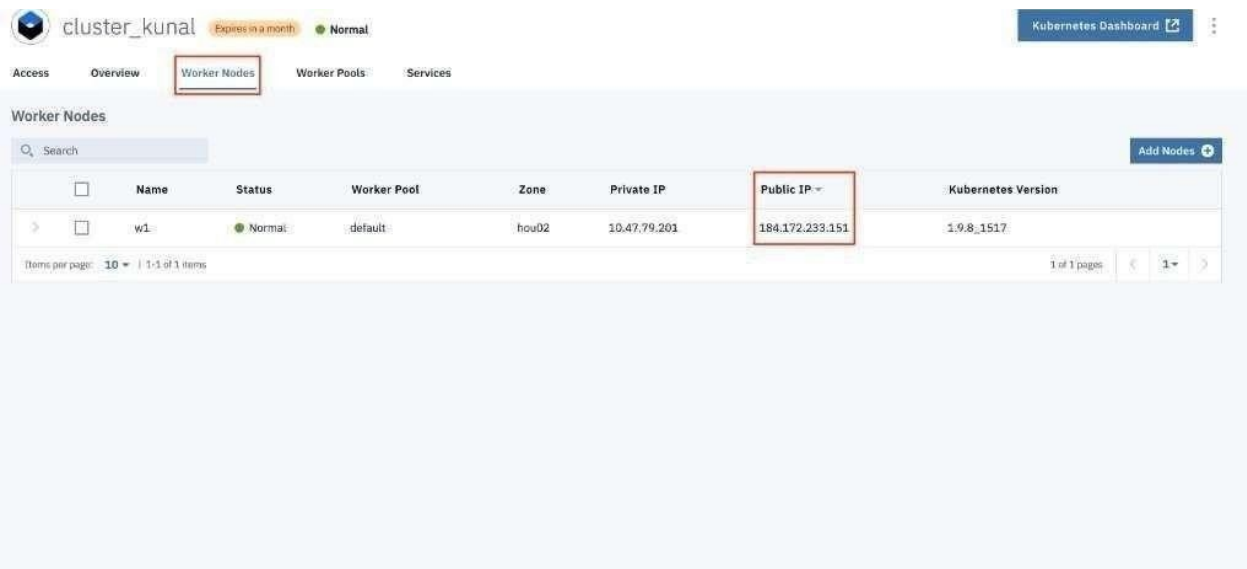
DEPLOY IN KUBERNETES CLUSTER:


```
$ curl http://169.61.18.4:1884


Hostname: echoserver-deployment-859b75d8c4-r6s62

Pod Information:
        node name:      10.73.115.27
        pod name:       echoserver-deployment-859b75d8c4-r6s62
        pod namespace:  default
        pod IP: 172.30.154.209

Server values:
        server_version=nginx: 1.13.3 - lua: 10008

Request Information:
        client_address=195.21
        method=GET
        real path=/
        query=
        request_version=1.1
        request_scheme=http
        request_uri=http://169.61.18.4:8080/

Request Headers:
        accept=*/*
        host=169.61.18.4:1884
        user-agent=curl/7.54.0

Request Body:
        -no body in request-
```

RESULTS

8.1 Performance Metrics





CHAPTER 9

CONCLUSION

The present paper presents the development of a system that recognizes fashion similar images. We accomplish this by implementing an already existingCNN model with transfer learningfor cloth image recognition using different libraries. For this purpose,we created a plan for collecting data and for developing the steps needed for preprocessing and cleaning up the data. We took into account featureslike patterns, machine,fabric, style etc. After extensive preprocessing and cleaning of data in a dataset,we constructed the model of stacked CNN to predict the features specific to these attributes and to train the models with the dataset to generate accuratepredictions 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 andcan be utilized in making fashion choices.

CHAPTER 10

FUTURE SCOPE

Our project has a wide scope in future which indeed results in large revolution in E-commerce platform . In our project we have to improvise with machine learning which results in easy way of interaction and using IBM cloud , suggestions for purchasing and chat bots for customer convenience . This would help in targeting audience who are interested in which specific types of products and also help in tempting others those who don’t believe in online shopping this would increase the rate of online E-commerce users and thus directly proportional to profit of E-Commerce as well as customer satisfaction.

REFERENCES

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J. & Devin, M. (2016). *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*, pp. 265–283. arXiv preprint arXiv:1603.04467.

2. Alkhawlani, M., Elmogy, M. & EL Bakry, H. (2015). *Text-based, content-based, and semantic-based image retrievals: a survey*. Int. J. Comput. Inf. Technol, 4, pp. 58-66.

3. Beel, J., Langer, S., Genzmehr, M., Gipp, B., Breitinger, C. & Nürnberger (2013). A. *Research paper recommender system evaluation: a quantitative literature survey*. Proceedings of the International Workshop on Reproducibility and Replication in Recommender SystemsEvaluation, 2013, pp. 15-22.

4. Bobadilla, J., Hernando, A., Ortega, F. & Bernal, J. (2011). *A framework for collaborative filtering recommender systems*. ExpertSystems with Applications, 38, pp. 14609-14623.

5. Burke, R. (2002). *Interactive critiquing forcatalog navigation in e-commerce*. Artificial Intelligence Review, 18, pp. 245-267.

6. Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. & Sartin,M. (1999). *Combing content-based and collaborative filters in an online newspaper.* Proc. of Workshop on Recommender Systems-Implementation and Evaluation, 1999, pp. 6-22.

7. Guo, G., Zhang, J. & Thalmann, D. (2014). *Merging trust in collaborative filtering to alleviatedata sparsity and cold start*. Knowledge-Based Systems, 57, pp. 57-68.

8. Iglesias, F. & Kastner, W. (2013). *Analysis of similarity measures in times series clustering for the discovery of building energy patterns*. Energies, 6(2), pp. 579-597.

9. Isinkaye, F. O., Folajimi, Y. & Ojokoh,B. A. (2015). *Recommendation systems: Principles,methodsand evaluation.* Egyptian informatics journal, 16, pp. 261-273.

10. Jannach, D. & Friedrich, G. (2013). *Tutorial: recommender systems*. International Joint Conference on Artificial Intelligence Beijing, 2013, pp. 1-144.

11. Krizhevsky, A., Sutskever, I. & Hinton,G. E. (2017). *ImageNet classification with deep convolutional neural networks*. Communications of the ACM, 60, pp. 84-90.

12. Lee, J. S. (2012).*Survey of Recommender Systems*. Citeseer.

13. Massa, P. & Bhattacharjee, B. (2004). *Using trust in recommender systems: an experimentalanalysis*. International conference on trust management, 2004. Springer, Berlin, Heidelberg,pp. 221-235.

14. Melville, P. & Sindhwani, V. (2010). *Recommender systems*. Encyclopedia of machinelearning, vol. 1, pp. 829-838.

15. Mirescu, S. V. & Maiorescu, T. (2010). The premises and the evolution of electronic commerce.Journal of knowledge management, economics and information technology, vol.1(1), pp. 44-56.

16. Park, D. H., Choi, I. Y., Kim, H. K. & Kim, J. K. (2011). *A review and classification of recommender systemsresearch*. International Proceedings of Economics Development & Research, 5 (1), p. 290.

17. Pazzani, M. J. (1999). *A framework for collaborative, content-based and demographic filtering*.Artificial intelligence review, vol. 13, pp. 393-408.

18. Pine, I. (1993). BJ (1993). *Mass customization: The new frontier in business competition*,Vol. 17, issue 2,pp. 271-283.

19. Reshma, C. & Patil, A. (2012). *Content based image retrieval using color and shape features*. International Journal of Advanced Researchin Electrical, Electronics and InstrumentationEngineering, 1, pp. 386-392.

20. Schafer, J. B., Konstan,J. & Riedl,J. (1999). *Recommender systemsin e-commerce.*Proceedings of the 1st ACMconference on Electronic commerce, 1999, pp. 158-166.