# SMART FASHION RECOMMENDER


## Abstract:

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.


## Introduction

### Problem Statement

Sometimes we see someone on the street wearing the clothes which looks good and appealing. We may want to find more clothing with similar fashion style or even more to find out where we can purchase them. This thesis provides a clothing recommendation system, people can use their iPhone take a photo of the clothes then search it using the App we developed. The recommendation results including 12 clothing images with similar fashion style and the fashion category of the query image (e.g. elegant). In this system we only focus on upper-body clothing. When designing this clothing recommendation system, we want to maximize the amount of code can be reused when we extending the current system. For example, when we need the mobile App not only work on iPhone but also on Android phone, we don't want to redesign the whole system and rewrite every piece of code of current system. In addition to code reusability, we also want the system easy to maintain. In order to archive these requirements, the thesis designs the clothing recommendation system using a decoupled architecture (i.e. the Model-View-Presenter architecture pattern).

**Project Description:**

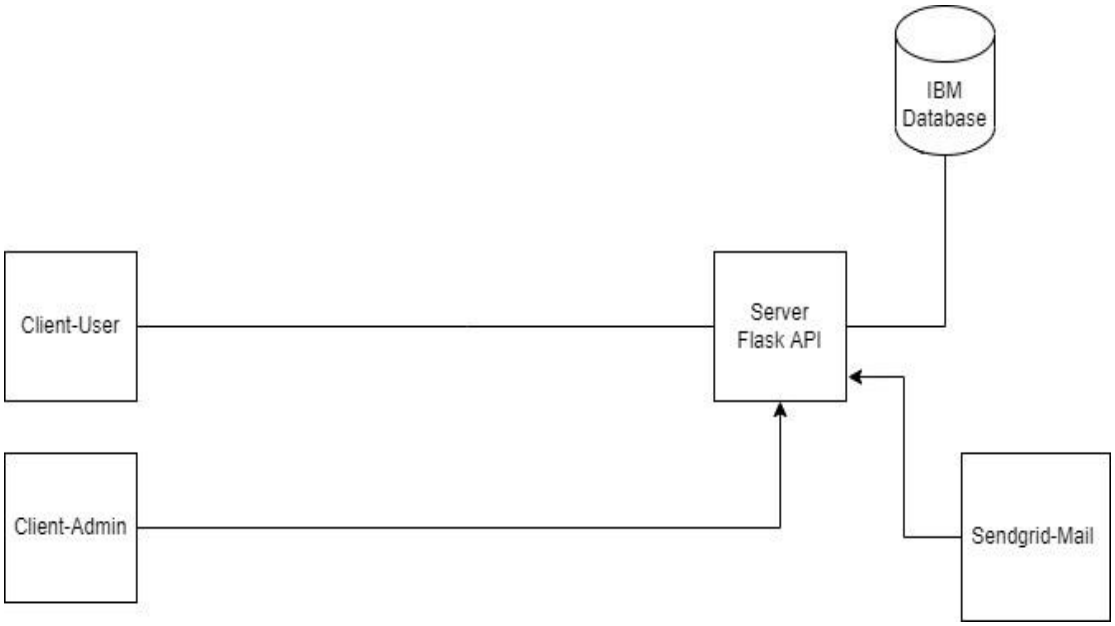
We have developed a new innovative solution through which you can directly do your online shopping based on your choice without any search. It can be done by using the chat bot. In this project you will be working on two modules:

- Admin and
- User

**The methodology used in this solution**

Instead of searching products in the search bar and navigating to individual products to find required preferences, this project leverages the use of chat bots to gather all required preferences and recommend products to the user. The solution is implemented in such a way as to improve the interactivity between customers and applications. The chat bot sends messages periodically to notify offers and preferences. For security concerns, this application uses a token to authenticate and authorize users securely. The token has encoded user id and role. Based on the encoded information, access to the resources is restricted to specific users.

**Overall Architecture:**

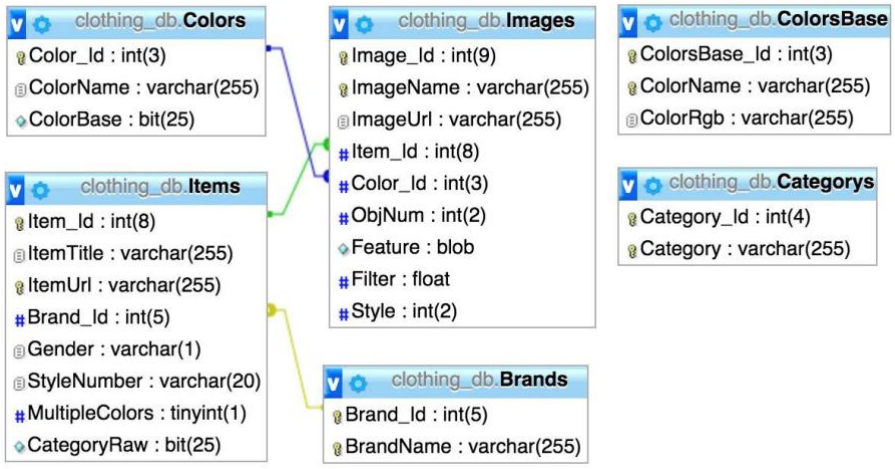


**CLOTHING DATABASE AND CLOTHING FASHION STYLE CATEGORIES :**

data (e.g. image Uniform Resource Locator (URL), image title, page URL, clothing category etc.) After getting all the image URLs (i.e. the links), we use bash script to automatic download all clothing images and put them into specific folder according to the clothing category. (a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) Figure 2.1: Sample images for each clothing category. (a)Women's Dresses (b)Women's Coats (c)Women's Tops (d)Women's Sweaters (e)Women's Jackets (f)Men's Suits (g)Men's Casual Shirts (h)Men's Polos (i)Men's T-Shirts (j)Men's Sweaters (k)Men's Coats We use a relational database MySQL to store the image file path on the machine and all other textual information we collected along with the image. A relational database is a structured collection of data which stores data in separate but related tables instead of putting all the data in one big storeroom.

## Clothing Fashion Categories

Chapter 2.1 describes 11 clothing categories, while this chapter will define 12 fashion categories for clothing image. These 12 fashion categories will be used for fashion style classification.




**Avant-garde**: Avant-garde fashions are characterized by their abnormality. Nontraditional cuts, patterns, and/or colors. Lots of drapey, dark clothing. Abnormal proportions. Often leather and other materials with interesting textures are used.

**Elegant:** Traditional proportions, enhancing a traditionally attractive figure. Refined, sleek, tasteful, often dressier clothing. Often subdued color schemes, though bright colors can considered..

**Folk:** Clothing influenced by rural life and other cultures. Nature references, often less constricting cuts. Classic patterns such as plaids, simple florals, and stripes. Not urban or modern. Hippie-styled

clothing falls within this category. Earth tones and bright colors, usually not subdued color schemes. Flowy layers. Mixed patterns. Worn-in clothing.

**Leisurely**: Relaxed, casual clothing. Athletic wear falls in this category. Can be any color. Looser cuts, comfortable looking usually. Soft or cozy materials. Not dressy. Soft textures, clothing that will not constrict movement. Clothing with sporting team logos or references usually fall within the leisurely category. Sneakers, sweatpants, sweatshirts, etc.

**Modern:** Minimal, trend-following clothing. Slim and skinny cuts, intermingling of casual and more formal looks. Sleek. Black and white and rich hues used. Shiny and smooth textures.

Clothing can be more sexual, though typically not super suggestive. Urban, classy feel.

**Neutral:** Clothing that does not fit strongly into any category. Simple clothing, neutral colors, usually not patterned. Simple jeans, tees, shirts, pants, jackets, etc.

**Renascent:** Clothing reminiscent of earlier times, especially fashions from the first half of the 19th century. Classier clothing, not showing lots of skin, modest, etc. Similar to Romantic style but feels more dated and less embellished. Clothing with conservative cuts, colors, patterns etc.

Much classical menswear will fall in this category (non-slim blazers, waistcoats, etc.)

**Romantic:** Traditionally flattering clothing. Womens clothing accentuates girlish figure, mens clothing emasculating. Not especially modern, more conservative and taking cues from fashions of the past. Wide hems and nipped waists for women. Cutesy/frilly things for women, lace, etc.

Pastel colors especially for women, traditional neutral shades more for men.

**Sexy:** Overtly sexual clothing meant to make the wearer look as desirable as possible. Shows lots of skin, emphasizes legs, chest, back for women, primarily chest and arms for men. Often flashy, shiny, or attention grabbing in some other way. Bright colors or loud patterns common.

**Splendid:** Clothing that is very embellished, adorned, embroidered, fanciful. Often appears quite expensive. Rich colors, intricate patterns, complex clothing. Cuts can vary, but tend to be traditionally flattering. Over-the-top detailing.

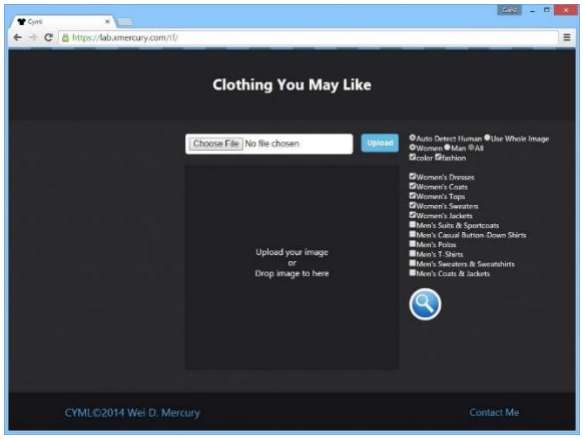
**Technology:** Fashion influenced by technology. Things that look futuristic. Fashions made of technical materials or that incorporate new technologies into their designs. Clothing that is made to interface well with other technology (cell phones, etc.). Overly functional clothing with excessive zippers, closures, straps, etc.

**Young:** Clothing made to embrace youthfulness or to make the wearer appear younger. Fun clothing, things with more juvenile, trendy designs. Lots of prints and bright colors. Basic clothing like tee shirts, jeans, simple dresses. Lacks embellishment beyond colors and patterns.

## System Component:

The CFSRS contains many parts, from front end to back end. In front end, CFSRS has Graphical User Interface (GUI) on both iOS App and Web Application. Because the cross-platform compatibility of Web Application, CFSRS can running on whatever platform supported HTML5, CSS and JavaScript, which means all major Operating System (OS), such as Linux, Unix, Mac OS and Windows along with Android, iOS and Windows Phone can meet the requirements to running this clothing recommendation application. While for the iPhone platform, there is a redesigned iOS App which can provide much more friendly GUI than the Web Application when using CFSRS on iPhone or iPad. Through the GUI, the user can upload query images, select search options and check recommendation results. Figure 3.1 shows the screenshot of Web Application running on Windows 8.1 using Chrome browser. In the back end, CFSRS has the Daemon program, Database server, Web server and PHP. All these programs are running on a Linux server — a physical computer.

## Clothing Fashion Style Recommendation System OVERVIEW:




In general, CFSRS has a client — the GUI and a server — the Linux server. Details of each component will be discussed in Chapter 4 from a different perspective. Then Chapter 5 will cover the Linux server.

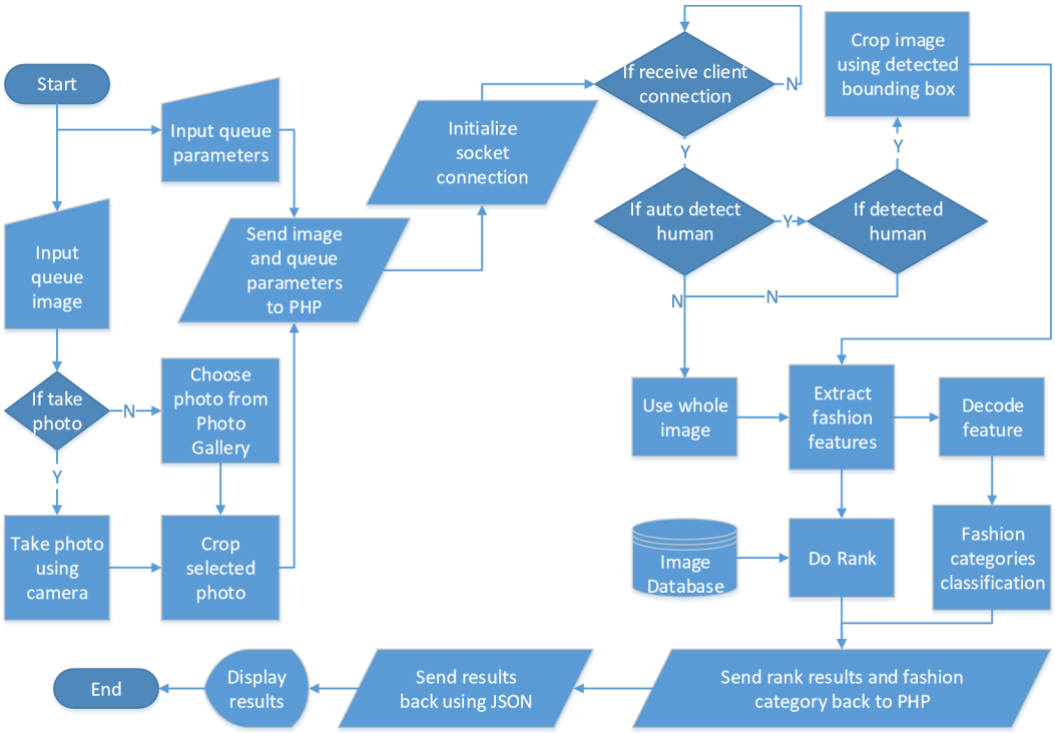
## System Flowchart:

Flowchart is a diagrammatic representation of algorithm, workflow or process. It is used to design complex program and can help other people understand what is going on.

The flow starts at the GUI, either Web Application or iOS App, when user starts choosing query image. The query image can be taken from camera or choose from existing photos from local

device. If using the iOS App, users can crop the image right after selection. Before submitting the search request, users can set the search options, such as gender, clothing categories and whether auto detect human or not.

When clicking or pressing the search button, the GUI will submit the Hypertext Transfer Protocol (HTTP) request to the PHP and the Web Server on the Linux server. The selected image along with the search options will be uploaded to Linux server via HTTP. PHP code is executed when the HTTP request is received by Web server and forwarded to the PHP. The PHP code will parse the search request from GUI, then initialize a local network socket connection with the Daemon server, search options sent by GUI are translated and enclosed in the socket.

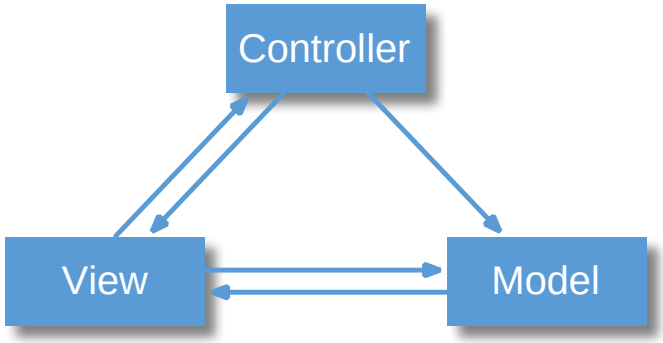



## Model-View-Controller and Model-View-Presenter :

MVC is one of the software architecture pattern. It divides the given application into three interconnected components — the model, the view and the controller. The model is the central component MVC[3], it directly accesses the database, manages the data and implements the logic of the application.

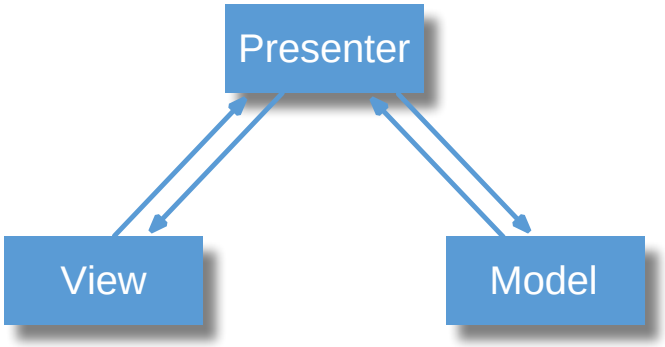
Components of MVC and Their Interactions. The solid line represents a direct association, the dashed line represents indirect association.

The controller processes user input then sends commands to the model to change data or the view to update view. The model will response the data query from view and implement the data change command from controller. The model can also notify the view when data is changed.




**Model-View-Presenter Architecture Pattern**

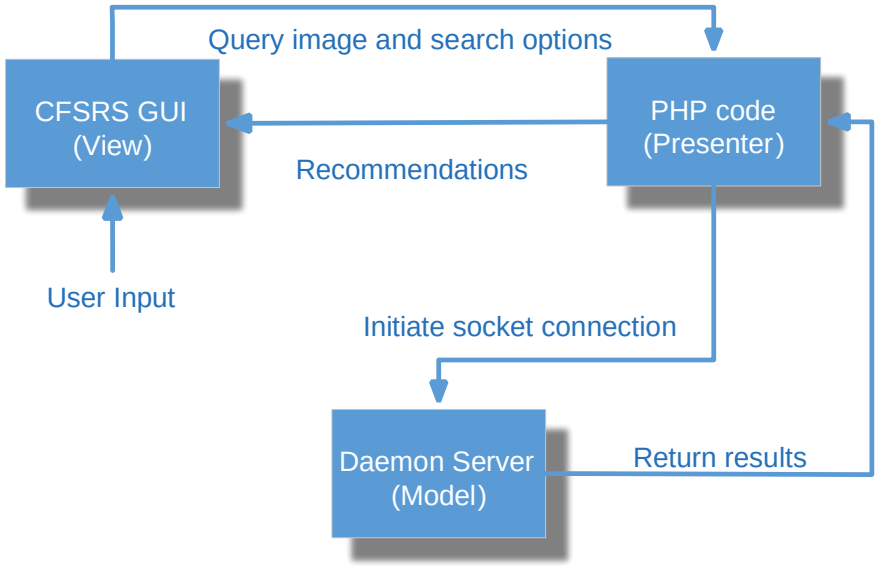
MVP is another software architecture pattern which is a derivation of traditional MVC. MVP also contains three components — the model, the view and the presenter. The major different between traditional MVC and MVP is that inMVP model and view do not communicate directly, which means model and view is completely decoupled.[5] In MVP, presenter is the component of intermediary between the model and the view. All display logic is handled by presenter.[6]



**MVP Architecture Pattern in CFSRS:**

MVC, the view hands over the user input to presenter. However, the view can only been updated by presenter. The presenter handles and processes the user input, then sends corresponding commands to to change or require data from model.

The CFSRS is designed using MVP architecture pattern. The model in CFSRS is the Daemon server along with all other supporting services (e.g. Database). CFSRS has two different views: the Web Application GUI and the iOS App GUI. The last part, the presenter, is the PHP code which act as a intermediary between the GUI and the Daemon server. Figure 4.3 shows how MVP applies in CFSRS. As can be seen in the figure, there is no direct connection between the CFSRS GUI and the Daemon server.



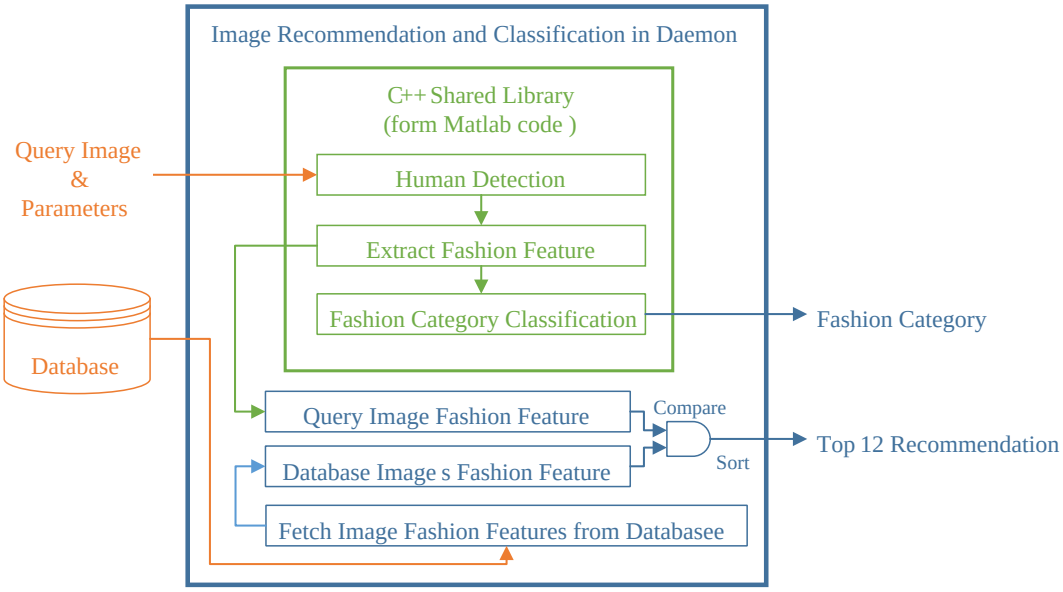

Core Code in Daemon :

The input, output and internal logical of the image recommendation and classification core code in Daemon program. As shown in figure, input of core code are the query image and parameters received from socket. Another input is the database which stored the fashion features extract from the clothing dataset collected from Internet. This is the only place that can access database, which conform the MVP design in CFSRS.

The green rectangle in the figure represents a C++ shared library (is known as a *.so* file on Linux or a *.dll* file on Windows) generated by Matlab code. Matlab programming language is good for matrix computations and image processing; however, pure Matlab code is hard to run as a Daemon program and access database. So in CFSRS, we first developed Matlab code to do the image processing, then compile the Matlab code into a C++ shared library which can be called by the Daemon C++ program. In this way, the input query image and parameters is hand over to Matlab C++ shared library directly, after processing and computing, the shared library return the fashion category and the query image fashion features.

The returned query image fashion features will be used to compare with the database image features by the C++ code. The Daemon C++ program will fetch a subset of database image fashion features based on the parameters from the presenter (i.e. clothing categories), then compare each image fashion features from the subset with the query image fashion features.

The compare results is measure by cosine similarity of the query image fashion



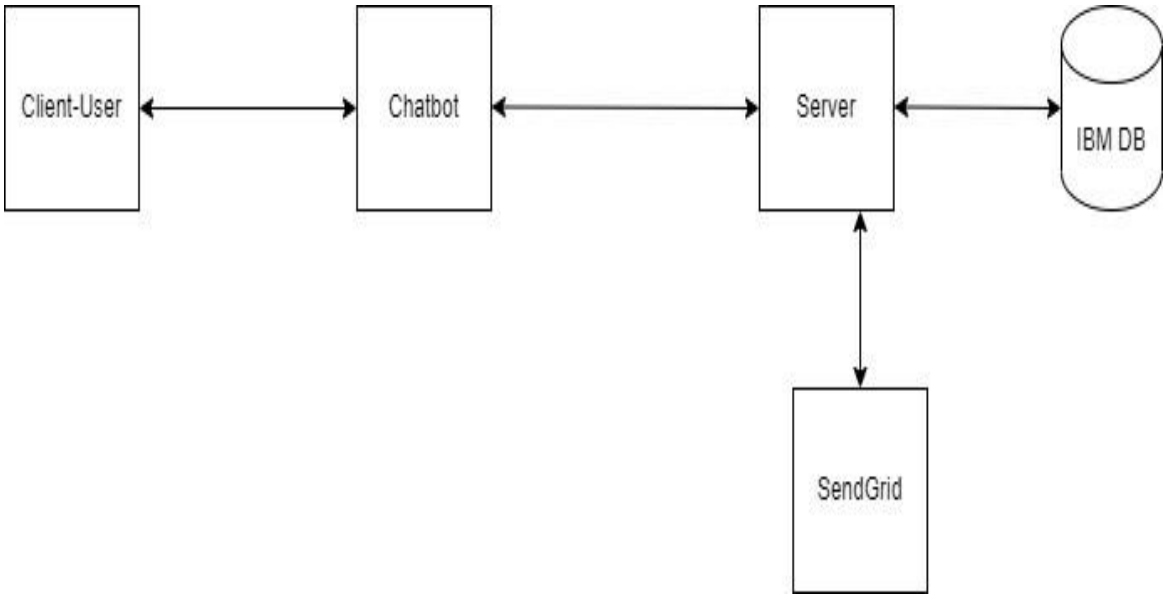

The compare results is measure by cosine similarity of the query image fashion features and database image fashion features. Let $f_q$ denote the fashion features of the query image, and $f_i$ denote the *i*-th image fashion features in the database. The similarity between $f_q$ and $f_i$ is:

$$s_{i,q} = \frac{(f_i \cdot f_q)}{||f_i|| \cdot ||f_q||} \quad (4.1)$$

.

After sorting the compare results $s_{i,q}$ of each $i$ in the selected subset, the top 12 ranking image is send back to socket client along with the fashion category. Chapter 4.2.3 will cove the Matlab program within the C++ shared library

**Customer:**




## Features:

- Using chat bot we can manage users' choices and orders.
- The chat bot can give recommendations to 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 chat bot will send a notification to customers if the order is confirmed.
- Chat bots can also help in collecting customer feedback.

**METHODOLOGY PROPOSED:**

In traditional e-commerce websites, the users need to search for their required product using a search bar or go through the whole effects of their search. It will take a lot of users' time and it will create a lot of flawed user experiments. This approach will create bad marketing for the product. Later when the user comes again to purchase the product it will create a bad impression on the user. Even though the product is good the user will not buy the product.
This type of search will create miss matched products when the product has a different name. Let's say we search for oranges on amazon. Sometimes it will show orange colour or sometimes it will show orange fruit. In recent times, fashion systems have been integrated with artificial intelligence and deep learning. These approaches provide a rich recommendation, but in most cases, it is prone to product mismatch. Even Though the recommender system recommends products based on the user's preference, this system lacks a chat bot that improves user experience by interacting with users. In most fashion systems, the user needs to navigate across multiple products to find the appropriate product. The users are made to filter products based on the long list of categories present in the system. If this system is integrated with an intelligent bot, it would be able to list out only required categories.

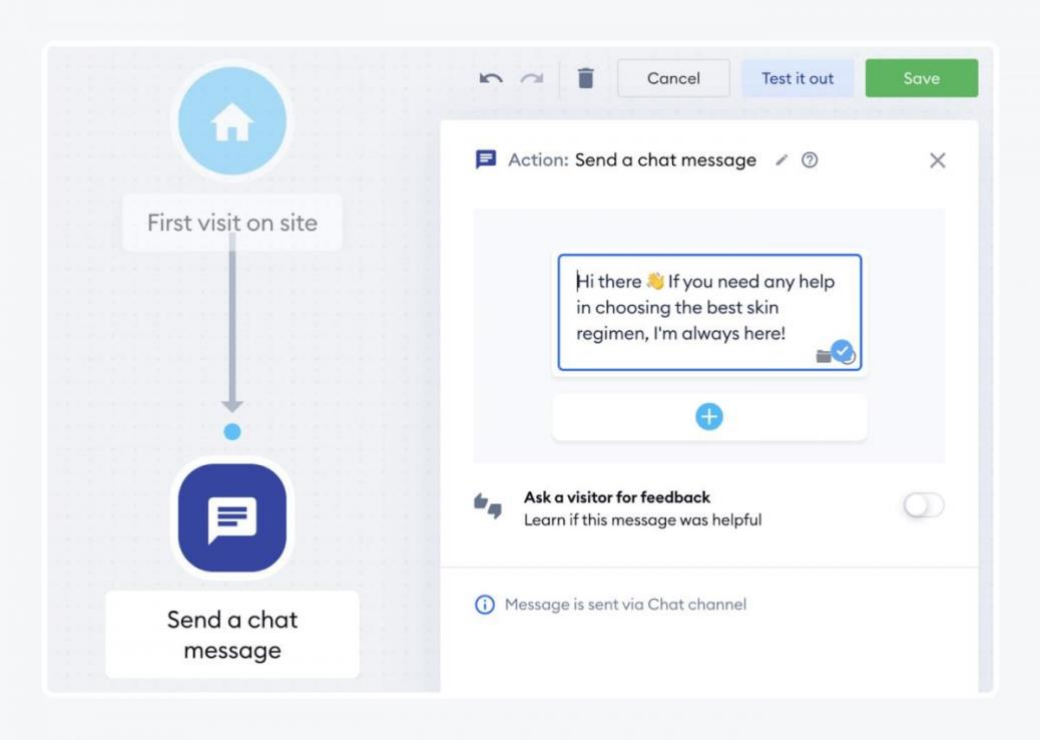
**Results and discussions:**

The smart fashion recommender system should recommend products based on user input. The user input can be any of the following

- Category
- Subcategory
- Product name
- Product brand
- Colour/Size
- Price Range
- Delivery Date/Time

- Discounts/Deals
- New Arrivals

Based on the user input, chat bot displays products sorted by required fields.

**Functional Requirements:**

The functional requirements of the application are:

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

**Non-Functional Requirements**

**Performance Requirements**

The system shall be able to handle multiple requests at any given point in time and generate an appropriate response.

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

**Security Requirements**

- Credentials and secrets should be stored securely and should not be leaked.
- Secured connection HTTPS should be established for transmitting requests and responses
- between client and server.

- The system has different roles assigned to a user and every user has access constraints.

- User access token should be valid for a shorter period and needs to be refreshed periodically.
- Clients should implement mechanisms to prevent XSS attacks.
- The server should restrict access to the resources for the particular client domain.

**Error Handling**

- The system should handle expected as well as unexpected errors and exceptions to avoid termination of the program.
- Appropriate error messages should be generated and displayed to the client.

## Hardware Requirements

- 8GB RAM
- Intel Core i3
- Laptop/Desktop
- Windows/MAC/Linux OS.

**Software Requirements**

- Python
- Flask
- Docker
- Kubernetes
- IBM DB

**Conclusion :**

The smart fashion recommender system uses a chat bot as a primary mechanism to interact with users, collect user interest and recommend products periodically. A chat bot is designed to improve user experience by interacting with users. Users need not navigate between multiple pages to find an appropriate product. The system is designed to minimize the efforts taken by customers to search for the required product. The future enhancements of the chat bot include adding products to the cart, displaying cart items, order history, and payment through the chat bot.

By Team Members:
Parameshwaran VK - 421319104021
Manoj Adhithyan A - 421319104016
Dayanthi S - 421319104003
Mohan Raj M 421319104018