FYP-2 Final Evaluation Report

Fashion Hunt

Members

Hassam Asif (i170007)

Aimen Inam (i170156)

Yaseen Mughal (i170276)

Supervised by

Dr. Omer Ishaq

FAST School of Computing

National University of Computer and Emerging Sciences

Islamabad, Pakistan

2021

FYP-2 Final Evaluation Report

Anti-Plagiarism Declaration

This is to declare that the above publication produced under the:

Title: Fashion Hunt

Fashion Hunt is the sole contribution of the authors and no part hereof has been reproduced on as it is basis (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. The work presented in the report is our own and it has not been submitted/presented previously to any other institution or organization. We will be responsible and liable for any consequence if violation of this declaration is determined.

Date: June 25th, 2021

Student 1

Name: M. Hassam Asif

Signature: _____

Student 2

Name: Aimen Inam

Signature: _____

Student 3

Name: M. Yaseen Mughal

Signature: _____

Supervisor (Faculty)

Name: Dr. Omer Ishaq

Signature: _____

Abstract

‘Fashion Hunt’ is a web-based application that is designed for searching desired clothing online. User can upload an image that has his/her desired clothing. After uploading, the System displays different segments of clothing e.g., pants, shirts etc. The user can view the segmented items separately and can select the required clothing segment. The System then lists similar, related and recommended links where the selected clothing segment information can be found. The user can then proceed to making a buying decision.

Acknowledgments

We would like to express our deepest sense of gratitude to ALLAH Almighty, the Most Beneficial and the Most Merciful, Who enabled us to complete this Project. And we truly admire our Holy Prophet (P.B.U.H) for being the light of guidance for mankind. We are Thankful to our families and friends for the support and help due to which we carried out our work in a timely manner and in a comfortable environment.

We feel honoured to express our sincere gratitude to our Supervisor Dr. Omer Ishaq for his motivation, enthusiasm and unique ideas. He enabled us to practically implement knowledge which was otherwise just information. Without his support and dedication, our project would not have been possible.

Executive Summary

‘Fashion Hunt’ aims at providing flexibility to search any desired clothing regardless of its brand. This can provide user with multiple choices to choose between similar clothing items of different brands.

‘Fashion Hunt’ is a web-based application that is designed for searching desired clothing online. User can upload an image that has his/her desired clothing. After uploading, the System displays different segments of clothing e.g., pants, shirts etc. The user can view the segmented items separately and can select the required clothing segment. The System then lists similar, related and recommended links where the selected clothing segment information can be found.

This app has a significant advantage in the world of Fashion as there isn’t any Hub that have the dataset of multiple clothing brands where users can search clothing instead of surfing through dozens of brands. The app will benefit a lot to Customers, Clothing Vendor (where Server will be established) and the Brokers.

The app will provide following:

- Reliable search results
- Ease of use
- Correct calculations in fetching and displaying results

The Website is created using MEAN Stack development. User will have to go through the Sign Up and Login process to authenticate and authorize himself/herself.
In the backend, Annotated dataset (with annotations as ‘shirt’, ‘pant’, ‘person’ in the image) is provided to the DeepLabv3 model for training. Overall, the accuracy of predictions is 89 %.

For our last implementation, we did the following:

1. *Predicting the most related clothing:*

   We have used the Siamese Model to train on a given dataset and perform predictions on the test data. By providing the clothing to this model, it predicts most relevant clothing items.

2. *Integrating the Frontend with the backend:*

We integrated the Angular App with the backend using Flask. The Flask server passes the data to and from the backend and Frontend which allows the final results to be displayed on the Web App.

Table of Contents

Introduction **1**

Literature Review **1**

Project Vision **2**

- Problem Statement 2
- Business Opportunity 2
- Objectives 2
- Project Scope 3
- Constraints 3
- Stakeholders Description 3
  - Stakeholders Summary 3
  - Key High-Level Goals and Problems of Stakeholders 4

Software Requirements Specification **4**

- Features 4
- Functional Requirements 5
- Quality Attributes 5
- Non-Functional Requirements 5

High Level Use cases **6**

- Use Case Diagram 10

Iteration Plan **11**

Iteration 1 **12**

- Expanded Use Cases 12
- Activity Diagram 16
- Domain Model 17
- System Sequence Diagram 18
- Operation Contracts 21
- Sequence Diagram 23
- Class Diagram 24

Architecture Diagram (MVC) 25
Iteration 2 **26**
Expanded Use Cases 26
Activity Diagram 31
Domain Model 32
System Sequence Diagram 33
Operation Contracts 36
Sequence Diagram 38
Class Diagram 39
Architecture Diagram (MVC) 40
Iteration 3 **40**
Expanded Use Cases 40
Activity Diagram 45
Domain Model 46
System Sequence Diagram 47
Operation Contracts 50
Sequence Diagram 51
Class Diagram 52
Architecture Diagram (MVC) 53
Iteration 4 **54**
Expanded Use Cases 54
Activity Diagram 57
Domain Model 58
System Sequence Diagram 59
Operation Contracts 61
Sequence Diagram 62
Class Diagram 63
Architecture Diagram (MVC) 64
Implementation Details **65**

11.1 Iteration 1 65

11.2 Iteration 2 68

11.3 Iteration 3 85

11.4 Iteration 4 91

Conclusions **94**

References **94**

Glossary **95**

Table of Implemented Figures

Figure 1. 1: Mask generated by model
Figure 1. 2: This is an image segmented result
Figure 2. 1: child2.jpg is the name of image file and abc.json is the output label file
Figure 2. 2: Defining the class name in labelme
Figure 2. 3: Generating polygon points on the image
Figure 2. 4: Viewing the file in coco format
Figure 2. 5: Drawing bounding boxes on the figure defining each class
Figure 2. 6: Showing the whole representation of the dataset
Figure 2. 7: Deep Lab V3 Layered Architecture
Figure 2. 8: Graph showing Validation accuracy using tensorflow board
Figure 2. 9 : The figure 1 is the actual image, second is the ground truth and third is the predicted output from the model
Figure 2. 10 : Actual Vs Predicted
Figure 2. 11: Actual Vs Predicted
Figure 2. 12 : Actual Vs Predicted
Figure 2. 13: Actual Vs Predicted
Figure 2. 14: Actual Vs Predicted
Figure 2. 15: Login Screen to verify user's account
Figure 2. 16 : Sign Up Screen to register user in database
Figure 2. 17 : Slideshow 1 of Fashion Hunt Web App
Figure 2. 18: Slideshow 2 of Fashion Hunt Web App
Figure 2. 19 : Slideshow 3 of Fashion Hunt Web App
Figure 2. 20 : Optional Selection of Men / Women Collection
Figure 2. 21 : About Us
Figure 2. 22 : shows the gallery. Gallery is added to attract the customers for using the website.
Figure 2. 23 : Contacts section
Figure 3.1 : Siamese Mode
Figure 3.2 : Upload Image to the webpage
Figure 3.3 : Uploaded Image on the Webpage
Figure 3.4 : Segmented Images
Figure 3.5 : Similar Images

Figure 3.6 : Selected Clothing's official site
Figure 4.1: Upload and Segmented Images
Figure 3.6 : Similar Images with Relevant Information
Figure 3.6 : Selected Image's Clothing Site

1 Introduction

Fashion Hunt is a web based application that is designed for searching desired clothing online. It uses deep learning and image detection techniques to detect, train and process images.

User can upload image that has his/her desired clothing. After uploading, the System converts the image into different segments e.g. a person's image will be segmented into shirt, trouser etc. The user can view the segmented items separately and can select the required clothing segment. The System then lists similar, related and recommended links where the selected clothing can be found. Each link will have complete information about its respective clothing e.g. price, availability, size etc. The user can make buying decision after comparing different links based on color, price etc.

2 Literature Review

We are inspired by the idea presented in the paper **"Convolutional Neural Networks for Fashion Classification and Object Detection"** [1] written by Brian Lao bjlao, Karthik Jagadeesh. They have achieved accuracy of 50.2% for clothing style classification and 74.5% for clothing attributes classification. We aim to achieve better accuracy and we have added additional features.

Paper Working:

1. Predict the clothing type through multi-class classification (clothing type classification)
2. Predict the clothing attributes through attribute classification (clothing attribute classification)
3. Find the most similar pieces of clothing in the dataset (clothing retrieval)
4. Determine a set of regions within an image that contains clothing objects

Our Working:

1. Perform Semantic segmentation to crop persons from images just to focus on clothing items
2. Perform semantic segmentation to predict pixel masks of clothing items
3. Predict the clothing type through multi-class classification (clothing type classification)
4. Predict the clothing attributes through attribute classification (clothing attribute classification)
5. Creating an interface to find similar clothing items from scrapped data
6. Giving recommendations of links of websites to users to purchase clothing items

3 Project Vision

3.1 Problem Statement

With a busy Lifestyle nowadays, many people are shifting from traditional shopping to online shopping; But still searching suitable brands and their products is a laborious task. It takes hours and even days to search for a desired product; and even after spending so much time, they still might not be able to get what they want. People are not ready to waste this much time especially when time is precious to them.

"Fashion Hunt" will free people from the trouble of searching aimlessly and wasting time. They will be able to exactly get what they want; All they have to do is to upload an image of their desired clothing and get its related information.

3.2 Business Opportunity

As more and more people are shifting towards online shopping to save their time and effort, still there isn't any app that will help people to search required Pakistani dresses as quickly as possible. All the web and mobile apps of clothing vendors requires customers to search entire catalogues before they can decide anything. Our app can be easily accessible to large population of busy people who have to make quick buying decisions. With this app, they can search their desired clothing's availability, price, size etc. in just few seconds.
Ease of use, interactive UI and the functionalities that we offer, will encourage people to accept and use our app.

3.3 Objectives

- To provide an interactive and easy to use interface
- To allow users to search their desired clothing online
- To provide users a list of ecommerce websites where they can buy their required clothing online
- To provide users similar and recommended clothing websites based on their searches
- To allow users to access their search history as and when required

3.4 Project Scope

Fashion Hunt will allow Users to search Pakistani Dresses that are available in major local brands. They will be able to perform following

- To upload an image consisting their desired clothing
- To view the entire image into clothing segments e.g. Shirt, Pants etc. can be seen in different segments
- To select a particular clothing segment (e.g. shirt) in order to search its availability online
- To view a list of e-commerce websites that consists of similar clothing item and recommended items
- To view the entire search history

3.5 Constraints

- Internet is required to view websites
- Customer never knows the ideal picture size needed to be uploaded. He/she might have to upload many times before proceeding
- Permissions from user to access his/her gallery is required

3.6 Stakeholders Description

Stakeholders will be the following:

- Customers who are interested in clothing searching
- Clothing vendors who may be interested for using this app to produce business value
- Brokers who might use this app for customers to select a clothing item and delivers it to them after buying from its vendors

3.6.1 Stakeholders Summary

All customers will benefit from the app. Customers will easily access their clothing item and place order. Clothing Vendors can use this app to produce their business value. Brokers can benefit for only managing the orders and delivery.

3.6.2 Key High-Level Goals and Problems of Stakeholders

i. High Level Goals

Customers

- To Search Desired Clothing item
- To check and consider the most suitable option after searching e.g. Choosing a shirt with lowest cost than the others that are displayed by search

Clothing Vendors

- To produce business value

Brokers

- To grab market and earn revenue

ii. Problems

Customers

- Customer never knows the ideal picture size needed to be uploaded. He/she might have to upload, re upload before image is accepted
- Customer might have to face situations like '404 not found error', when clicking the displayed links. This is because data was removed from its website and customer can no longer locate it
- Customer can also access items' links that are already sold out, thus wasting his/her time

4 Software Requirements Specification

4.1 Features

1. Giving the user an intuitive interface where they can upload an image as an input and system will output visually similar images and links
2. Users can view a list of ecommerce websites where they can buy desired clothing online
3. Users will be able to view their search history

4.2 Functional Requirements

- User shall be able to Sign Up and create an account
- User shall be able to login to his/her account
- User shall be authenticated and authorized
- User shall be able to upload an image
- User shall select and search a segment of clothing of uploaded image online
- System shall fetch the similar and related image links from database
- System shall display similar and recommended image links
- User shall be able to view information for each image via link
- User shall search his/her search history

4.3 Quality Attributes

- **Reliability**

  Our app will give consistent search results
- **Usability**

  Our app will be user friendly. All options of an app can easily be accessed
- **Correctness**

  Our app will correctly make appropriate calculations when fetching and displaying search results. It can verify each user and can store their respective search results
- **Readability**

  Our system will be well structured with respect to coding, interface etc.
- **Testability**

  It's easy to test and find defects in our system

4.4 Non-Functional Requirements

- The system shall give consistent search history and search results
- The system shall be easy to use
- Users shall authenticate themselves before using their accounts
- The system shall fetch correct and accurate image links as and when required

5 High Level Use cases

Use Case: Sign Up	
Actors	Customer
Type	Primary
Description	The Customer creates a new account by giving his/her name, email and password

Use Case: Login	
Actors	Customer
Type	Primary
Description	The Customer logs into an existing account after authentication and can access its information

Use Case: Upload Image	
Actors	Customer
Type	Primary
Description	The Customer uploads an image from the gallery to be searched online

Use Case: Perform Image Segmentation	
Actors	Fashion Hunt Server (System)
Type	Primary
Description	The System converts the uploaded image into different well defined segments

Use Case: View Segmented Image	
Actors	Customer
Type	Primary
Description	The Customer views the segmented images separately

Use Case: Select Particular Clothing Segment	
Actors	Customer
Type	Primary
Description	The Customer selects one clothing segment out of all the segments, in order to search it online

Use Case: Predict Clothing	
Actors	Fashion Hunt Server (System)
Type	Primary
Description	The System checks the selected clothing segment against its entire dataset to see matches

Use Case: Display Similar Clothing Information	
Actors	Fashion Hunt Server (System)
Type	Primary
Description	The System fetches and displays the matched data to the Customer

Use Case: View Similar Clothing Information	
Actors	Customer
Type	Primary
Description	The Customer views the displayed similar and recommended Image links

Use Case: Select retrieved clothing's URL	
Actors	Customer
Type	Primary
Description	The Customer clicks on one of the displayed links for viewing its information

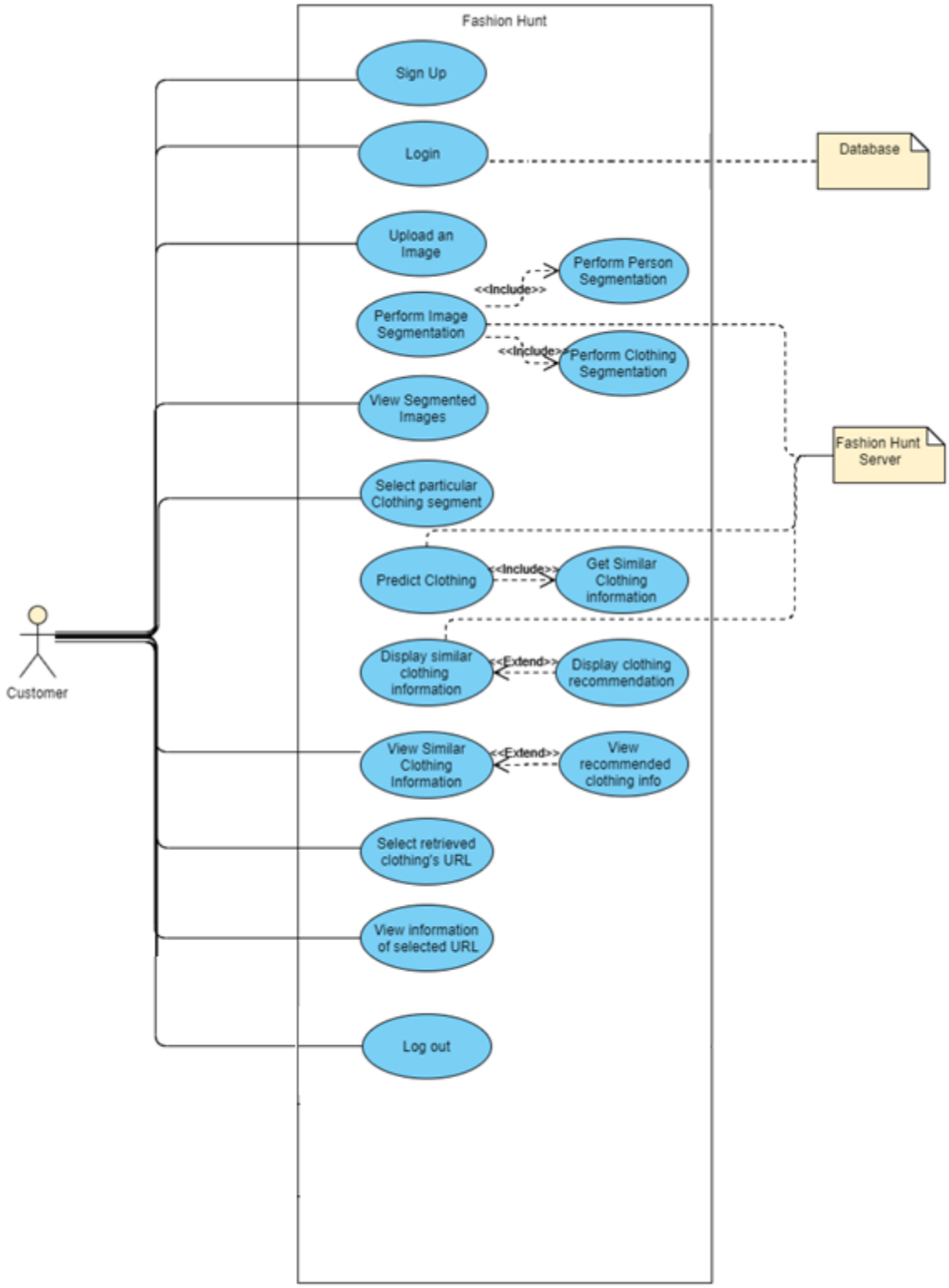
Use Case: View Information of selected URL	
Actors	Customer
Type	Primary
Description	The Customer views the clicked URL's information and can see the details of the image.i.e., its price, size, availability etc.

Use Case: Retrieve User's Search history	
Actors	Database (Server' database)
Type	Primary
Description	The Database fetches the search data of the user and sends it to the web app

Use Case: View Search history	
Actors	Customer
Type	Optional
Description	The Customer views his/her search history

Use Case: Logout	
Actors	Customer
Type	Primary
Description	The Customer logs out from his/her account

5.1 Use Case Diagram



6 Iteration Plan

Sr. No	Iteration no	From	To
1	Iteration 1	August 31st, 2020	October 31st, 2020
2	Iteration 2	November 1st,2020	January 31st, 2020
3	Iteration 3	February 1st, 2021	March 30th, 2021
4	Iteration 4	April 1st, 2021	June 4th, 2021

Iteration 1

- Working on datasets and collecting information from datasets that suits best for our requirements
- Web scraping data of different brand websites and storing the data
- Working on clothing semantic segmentation feature to extract individual clothing item from the given image

Iteration 2

- Continued on web scrapping
- Working on Fashion Hunt website prototype
- Identify the type of clothing, through multiclass classification and segmentation.

Iteration 3

- Similar image retrieval dataset preparation
- Similar Image retrieval model development and Website User Interface work.

Iteration 4

- Integration with website interface showing similar clothing images and displaying links to buy them. (Cloth Retrieval)
- Optimizations in User Interface and in Models
- Report writing

7 Iteration 1

7.1 Expanded Use Cases

Expanded Use Case 1: "Upload an Image"

Name	Upload an Image
Scope	Fashion Hunt Application
Level	User goal
Primary Actor	Customer
Stakeholders and Interests	Customer: Wants to upload an image in order to search online
Pre-Conditions	The Customer must be logged in
Post-Conditions	The uploaded image will be sent to the system for fetching the matched dataset

Main Success Scenario:

Actor Action	**System Responsibility**
1. The Customer Logs in to his/her account. 2. The Customer uploads an image from his/her gallery. 3. The uploaded image is sent to the system.	
	4. The system receives the uploaded image 5. The system accepts the image size. 6. The system stores the image.
7. The Customer has successfully uploaded an image and is proceeded further with search results.	

Extensions	4a. The system couldn't accept the image size. 4a1. The system asks the user to re upload the image of appropriate size
Special Requirements	Response time of system error is 5 seconds
Technology and Data variation List	Angular and NodeJS are used
Frequency of use	Could be nearly continuous
Miscellaneous	What happens if a customer uploads an image and system couldn't receive it because of internet issues?

Expanded Use Case 2: "Perform Image Segmentation (Person)"

Name	Performs Person image segmentation
Scope	Fashion Hunt Application
Level	User goal
Primary Actor	Fashion Hunt Server (System)
Stakeholders and Interests	Customer: Wants to have segmented image in order to select particular clothing segment (after person and clothing segmentation)
Pre-Conditions	The Customer must have successfully uploaded the image
Post-Conditions	The uploaded image will be converted to different well define segments

Main Success Scenario:

Actor Action	**System Responsibility**
1. The Customer uploads an image from his/her gallery. 2. The uploaded image is sent to the system.	3. The system receives the uploaded image 4. The system accepts the image size.

	5. The system performs person segmentation to separate a person from the image. 6. The system has successfully performed Person image segmentation.

Extensions	3a. The system couldn't accept the image size. 3a1. The system asks the user to re upload the image of appropriate size
Special Requirements	Response time of system error is 5 seconds
Technology and Data variation List	Pytorch Encoding is used
Frequency of use	Could be nearly continuous
Miscellaneous	What happens if a user uploads a nonhuman image? How will system performs person segmentation and clothing segmentation?

Expanded Use Case 3: "View Segmented Images"

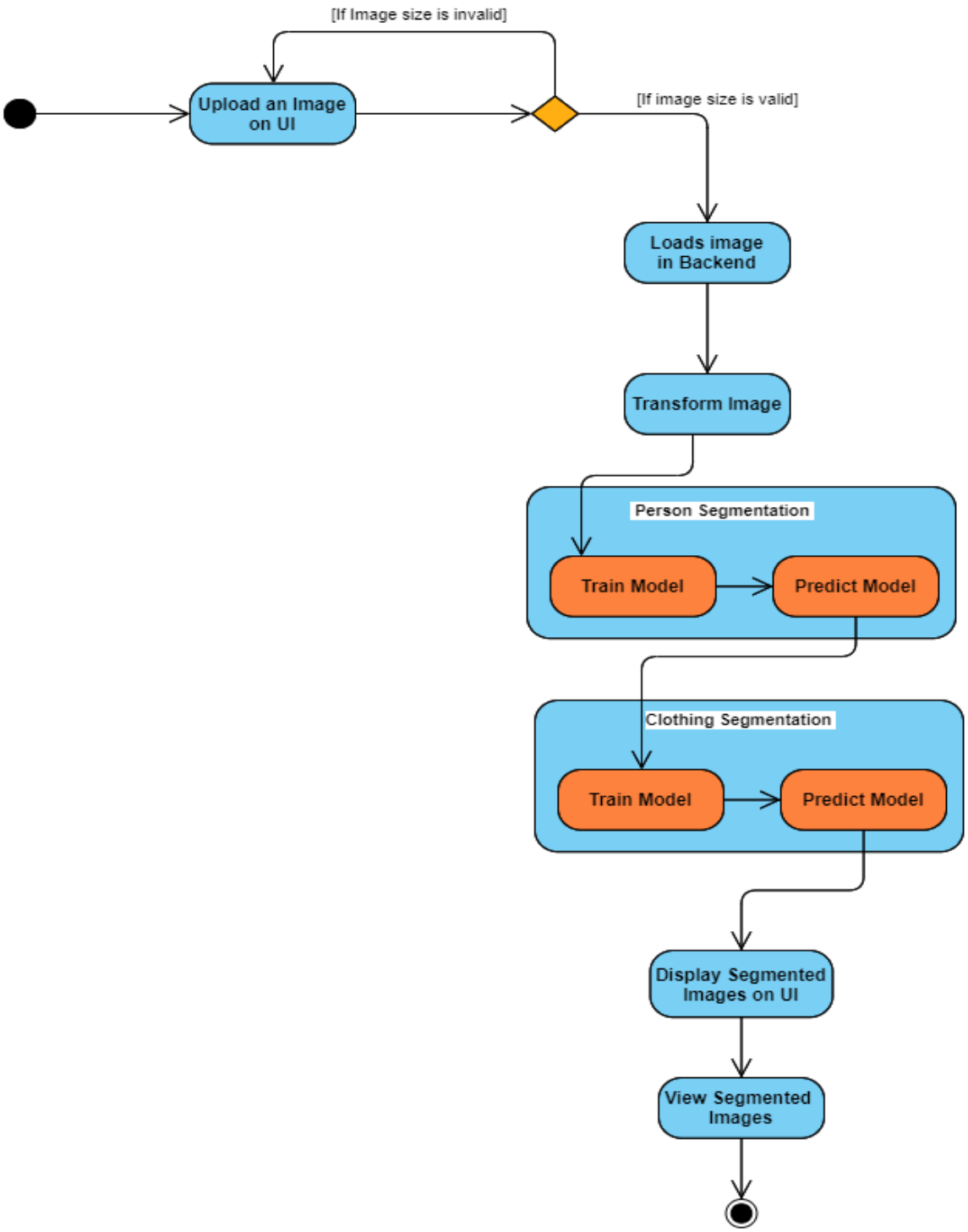
Name	View Segmented Images
Scope	Fashion Hunt Application
Level	User goal
Primary Actor	Customer
Stakeholders and Interests	Customer: Wants to view segmented clothing images so that he/she can search the desired clothing online
Pre-Conditions	The System must have performed image segmentation
Post-Conditions	The user can successfully view the segmented images and can now select any of the segment to proceed searching

Main Success Scenario:

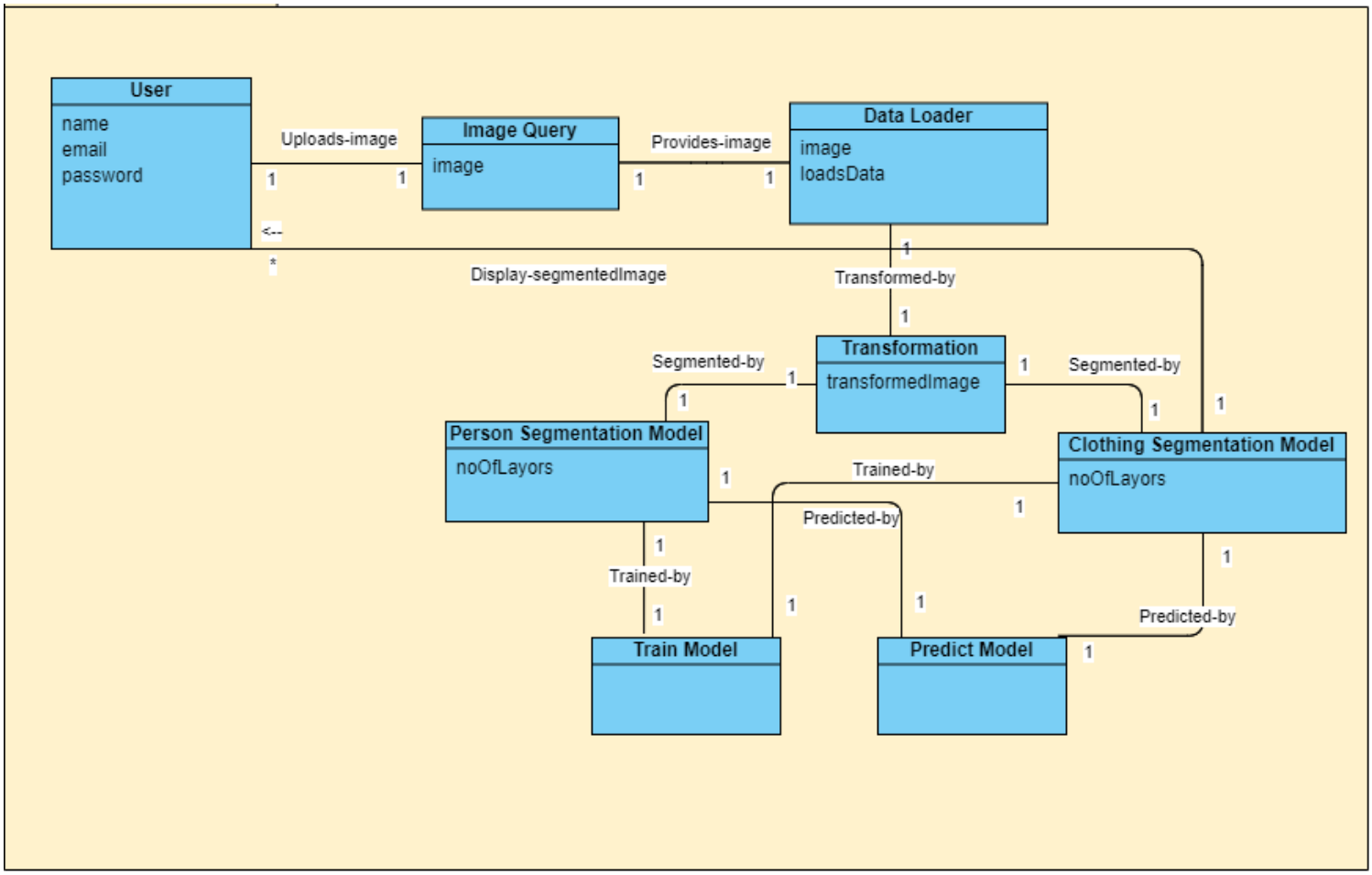
Actor Action	**System Responsibility**
1. The Customer uploads an image from his/her gallery. 2. The uploaded image is sent to the system.	
	3. The system receives the uploaded image 4. The system performs image segmentation. 5. The system displays the segmented images.
6. The Customer has successfully viewed the segmented images	

Extensions	4a. The system couldn't display the segmented images. 4a1. The system asks the user to re upload the image
Special Requirements	Response time of system error is 5 seconds
Technology and Data variation List	Angular, NodeJS and Pytorch encoding are used
Frequency of use	Could be nearly continuous
Miscellaneous	What happens if a system displays the segmented images but customer faces internet issues at his/her side?

7.2 Activity Diagram

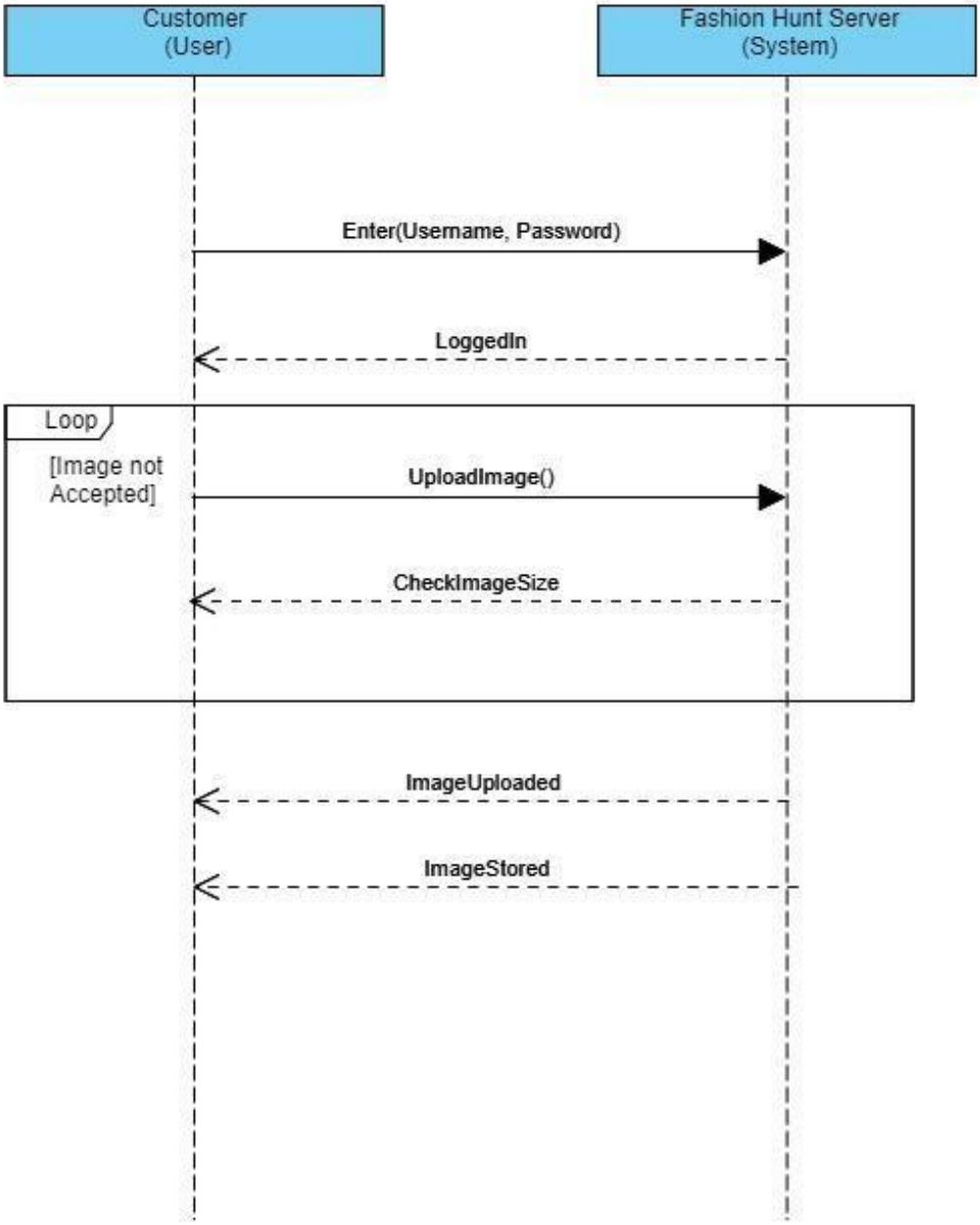


7.3 Domain Model

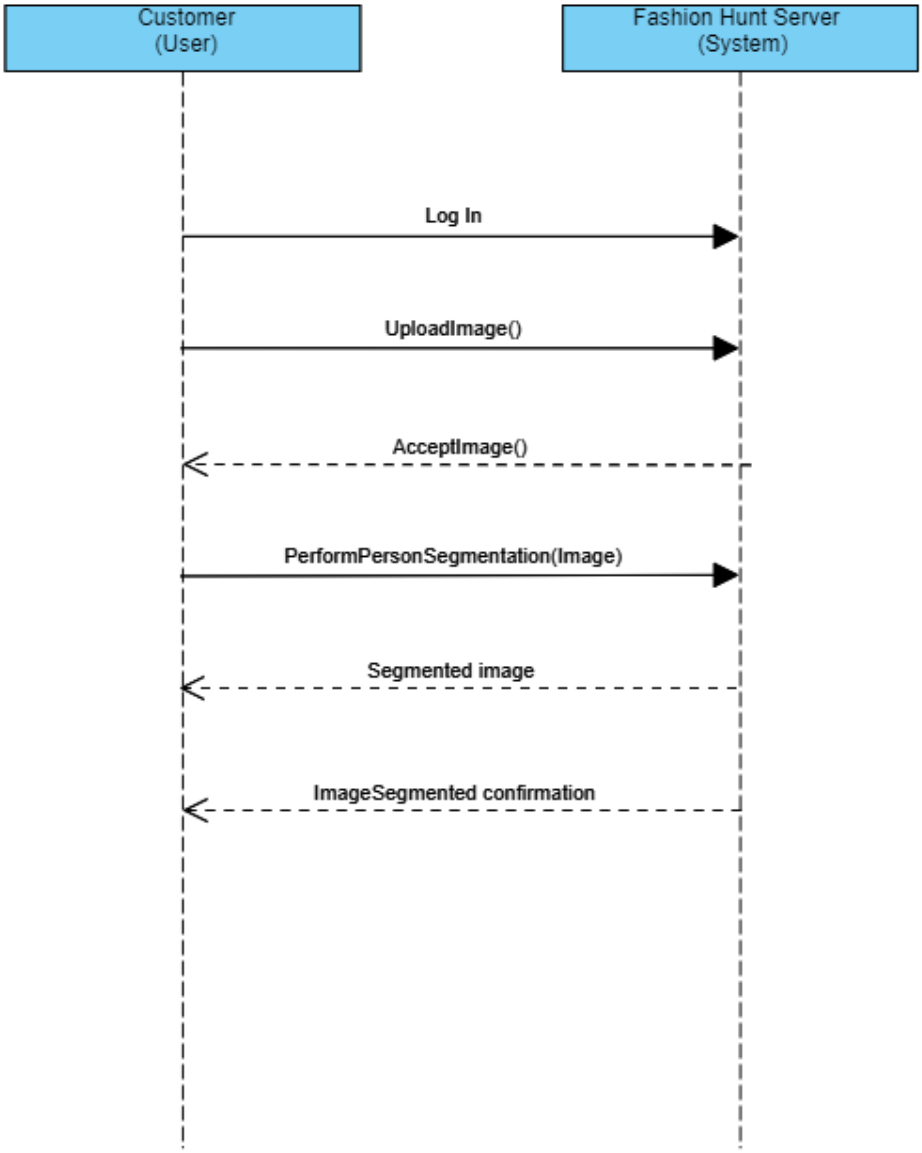


7.4 System Sequence Diagram

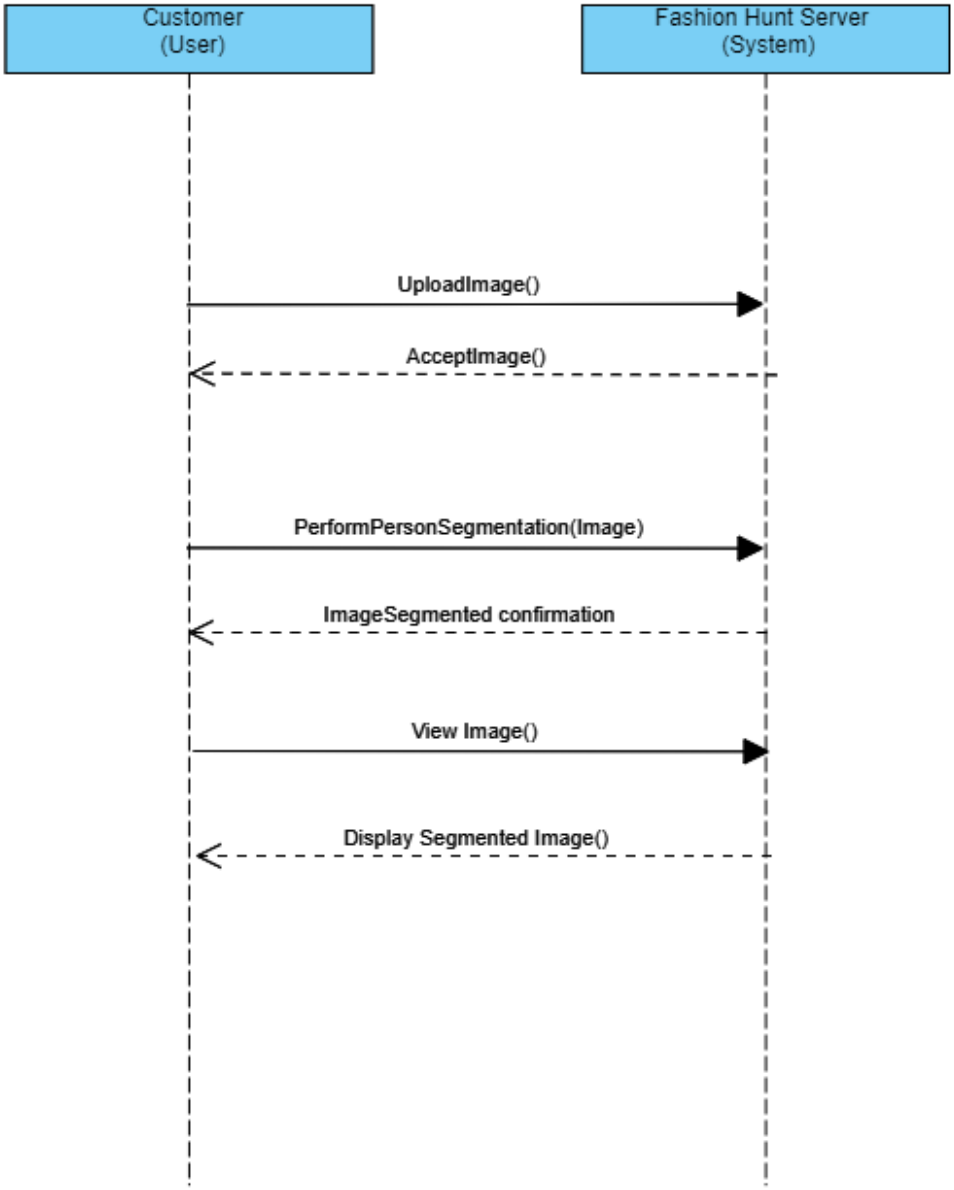
SSD 1: "Upload Image"



SSD 2: “Perform Image Segmentation”



SSD 3: “View Segmented Image”



7.5 Operation Contracts

Operation	Enter (Username, Password)
Use Case	Login
Responsibility	Enters the user credentials
Pre-Condition	User must have valid credentials
Post Condition	1. User instance was created. (Instance creation) 2. User instance was associated with Image Query. (Association Formed)

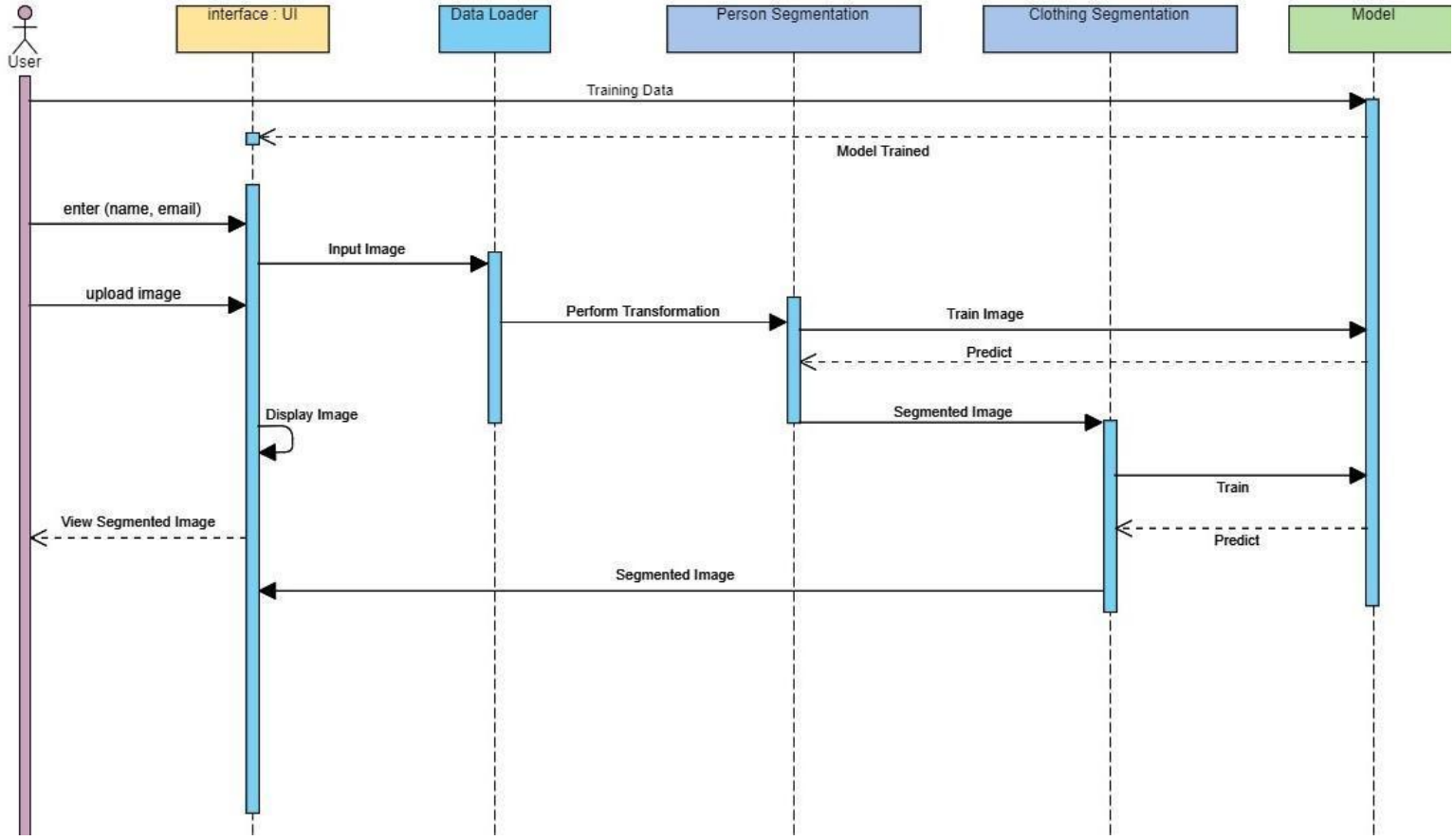
Operation	UploadImage()
Use Case	Upload Image
Responsibility	Uploads the image on the UI to be checked and used by the System
Pre-Condition	The Customer must be logged in
Post Condition	Image Query. Image was set to “Image”. (Attribute modification)

Operation	AcceptImage()
Use Case	Upload Image
Responsibility	Accepts the image size that was uploaded by the user
Pre-Condition	The user must upload an image of correct size and format.
Post Condition	1. Data Loader instance was created. (Instance creation) 2. Data Loader. Image was set to “Image”. (Attribute modification) 3. Image Query was associated with Data Loader. (Association formed)

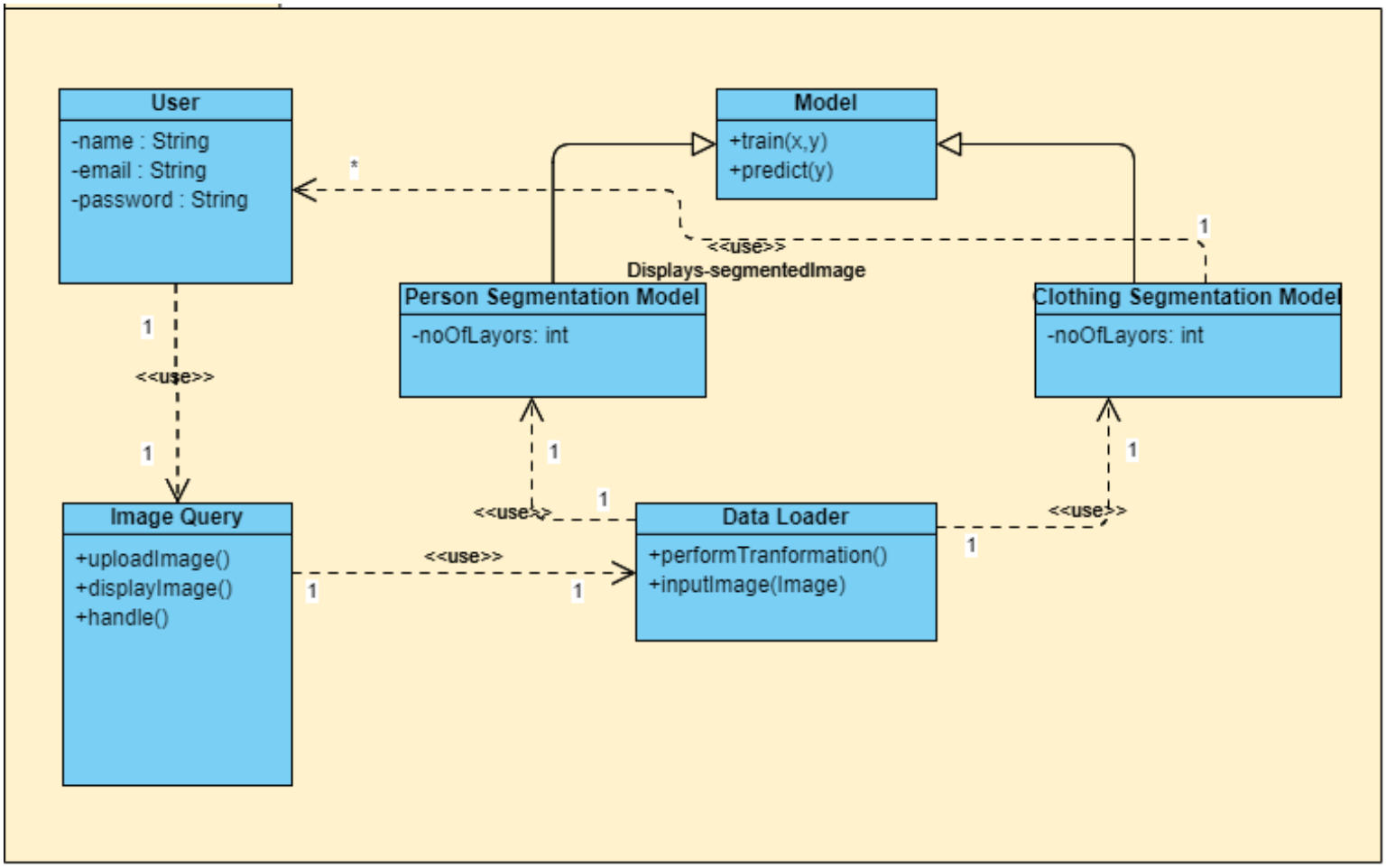
Operation	PerformPersonSegmentation(Image)
Use Case	Perform Image Segmentation
Responsibility	Performs Person Segmentation on the image to separate the person from the image
Pre-Condition	The Customer must have successfully uploaded the image
Post Condition	Clothing Segmentation instance was formed. (Instance creation)

Operation	DisplaySegmentedImage()
Use Case	View Segmented Images
Responsibility	Displays the segmented images back to the UI
Pre-Condition	The System must have performed image segmentation
Post Condition	User. Images [ ] was set to “Images”. (Attribute modification)

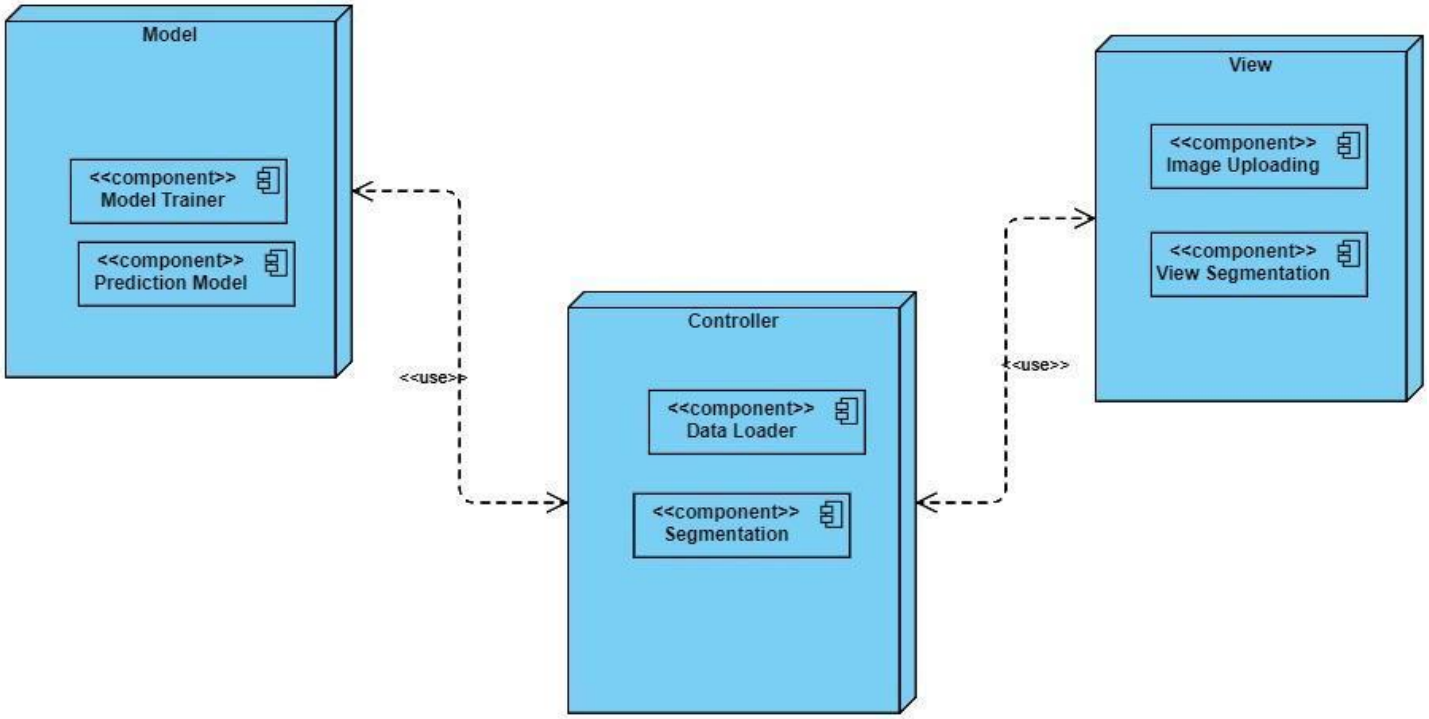
7.6 Sequence Diagram



7.7 Class Diagram



7.8 Architecture Diagram (MVC)



8 Iteration 2

8.1 Expanded Use Cases

Expanded Use Case 1: "Perform Image Segmentation (Clothing)"

Name	Performs Clothing image segmentation
Scope	Fashion Hunt Application
Level	User goal
Primary Actor	Fashion Hunt Server (System)
Stakeholders and Interests	Customer: Wants to have segmented image in order to select particular clothing segment
Pre-Conditions	The Image must be person segmented
Post-Conditions	The uploaded image will be converted to different well defined clickable clothing segments

Main Success Scenario:

Actor Action	**System Responsibility**
1. The Customer uploads an image from his/her gallery. 2. The uploaded image is sent to the system.	
	3. The system receives the uploaded image 4. The system accepts the image size.
	5. The system performs clothing segmentation on the person's image resulting in separate clothing items. 6. The system has successfully performed clothing image segmentation.

Extensions	3a. The system couldn't accept the image size. 3a1. The system asks the user to re upload the image of appropriate size
Special Requirements	Response time of system error is 5 seconds
Technology and Data variation List	Pytorch Encoding is used
Frequency of use	Could be nearly continuous
Miscellaneous	What happens if a user uploads a nonhuman image?

Expanded Use Case 2: "Sign Up"

Name	Sign up to a website to create a personal account
Scope	Fashion Hunt Application
Level	User goal
Primary Actor	Customer
Stakeholders and Interests	Customer: Wants to create an account to use the app
Pre-Conditions	The Customer must have a valid email (Gmail, Yahoo) in order to Sign Up.
Post-Conditions	User's personal account will be created in the app

Main Success Scenario:

Actor Action	**System Responsibility**
1. The Customer enters his/her all credentials (first name, last name, email and password). 2. The Customer clicks 'Sign Up'.	
	3. The system receives User's information. 4. The system validates Customer's credentials.

7. The user has successfully signed up.	5. The system stores User's data in the Database. 6. The system displays a message of 'Successful Signed Up'. 8. The system prompts the user to 'Login' Page.

Extensions	4a. The system displays error message as 'Invalid email format' or 'Minimum 8 characters Password is required'. 4a1. The user reenters the correct information format.
Special Requirements	Response time of system error is 3 seconds
Technology and Data variation List	Angular, NodeJS, Express and Mongo DB
Frequency of use	Could be nearly continuous
Miscellaneous	What happens if a user enters correct information but because of failure (internet issues) at server, data couldn't be stored in database. What error will be displayed to the user?

Expanded Use Case 3: "Login"

Name	Login to a website after authentication
Scope	Fashion Hunt Application
Level	User goal
Primary Actor	Customer
Stakeholders and Interests	Customer: Wants to login in order to use the app
Pre-Conditions	The Customer must have signed up successfully
Post-Conditions	User will be able to access and use the app

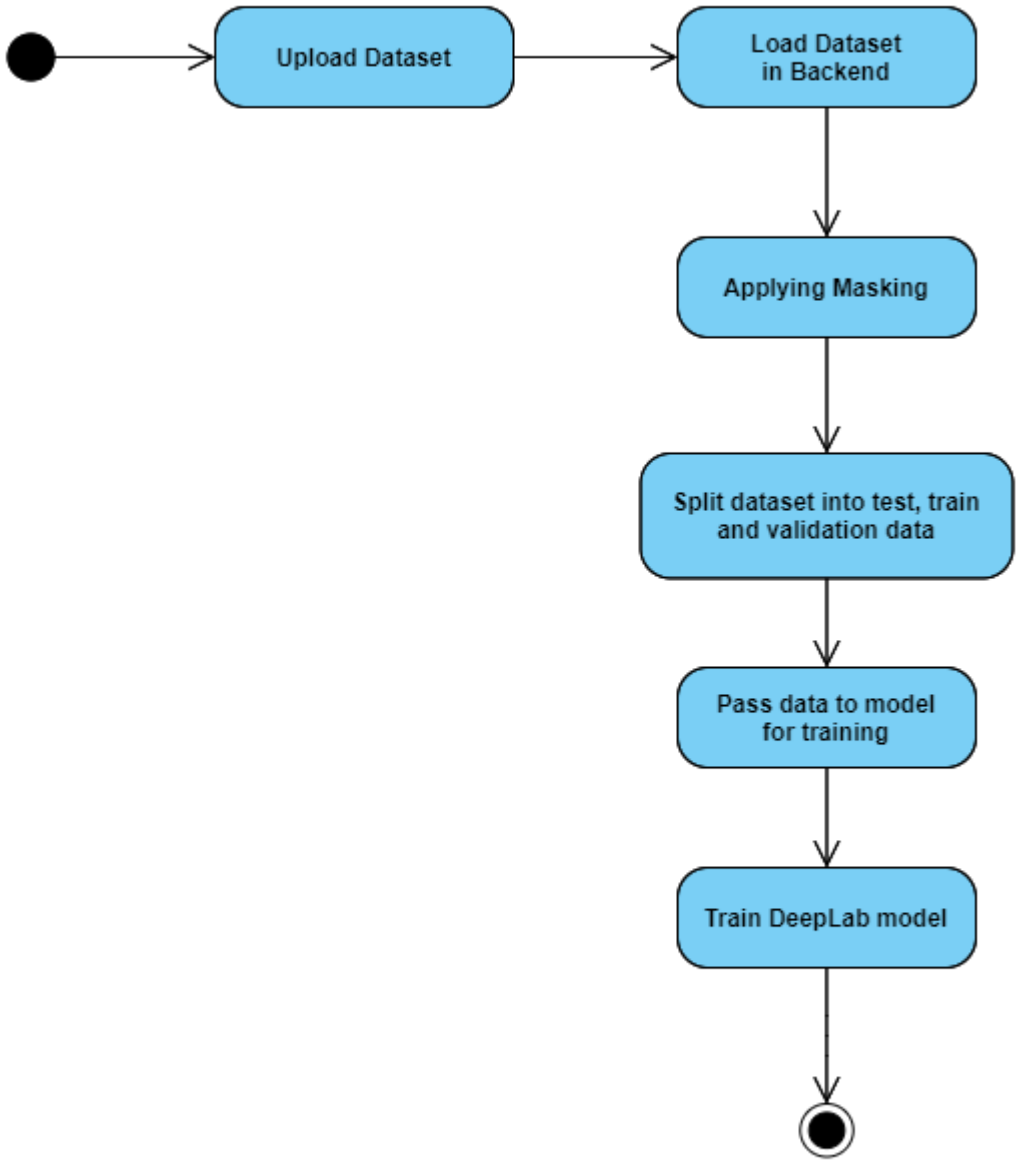
Main Success Scenario:

Actor Action	**System Responsibility**
1. The Customer enters his/her email and password. 2. The Customer clicks 'Login'.	
	3. The system receives User's email and password. 4. The system validates User's email and password'.
5. The User has successfully logged in.	
	6. The system prompts the User to the 'Home' Page.

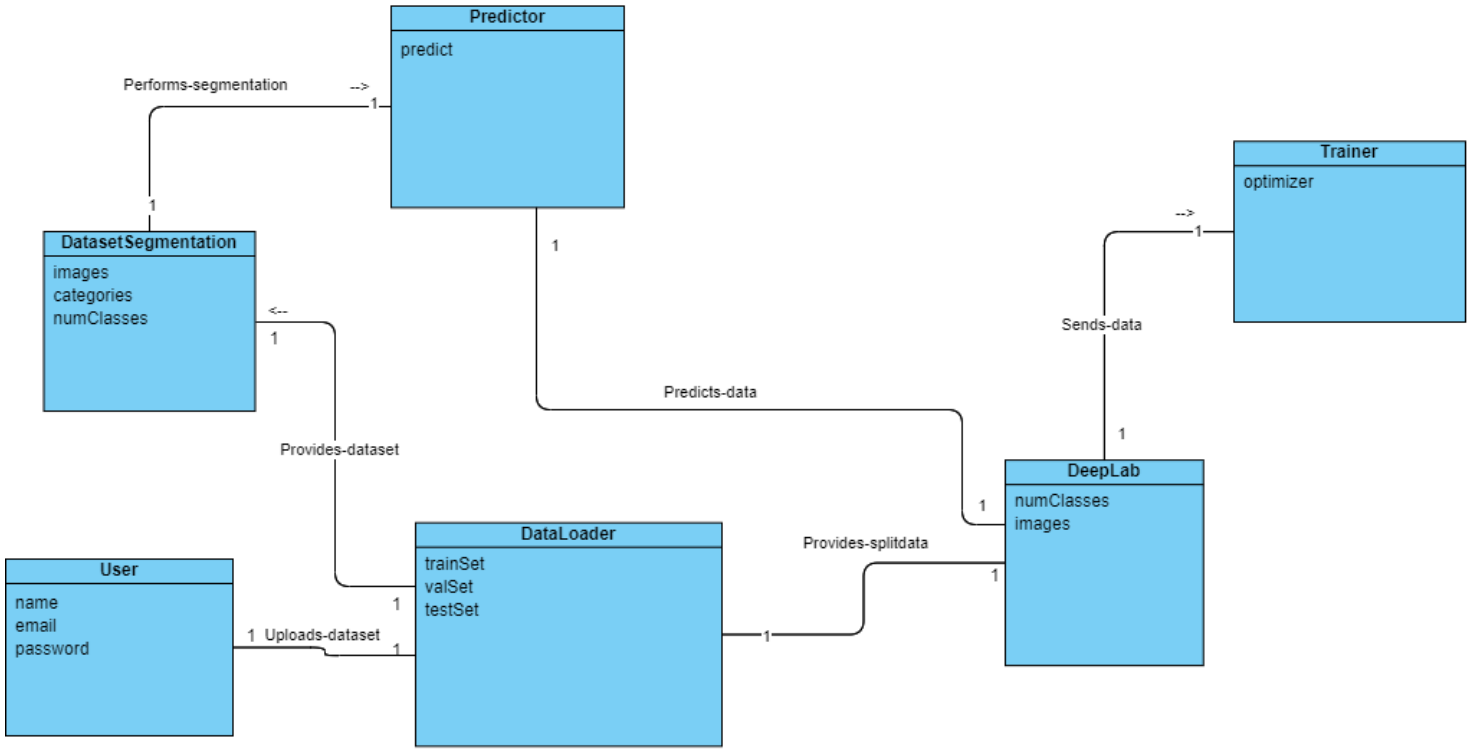
Extensions	4a. The system displays error message as 'Invalid email or password'.

	4a1. The user reenters the correct email or password.
Special Requirements	Response time of system error is 5 seconds
Technology and Data variation List	Angular, NodeJS, Express and MongoDB
Frequency of use	Could be nearly continuous
Miscellaneous	What happens if a user enters correct information but because of failure (internet issues) at the server, the user cannot be properly authenticated. What error will be displayed to the user?

8.2 Activity Diagram

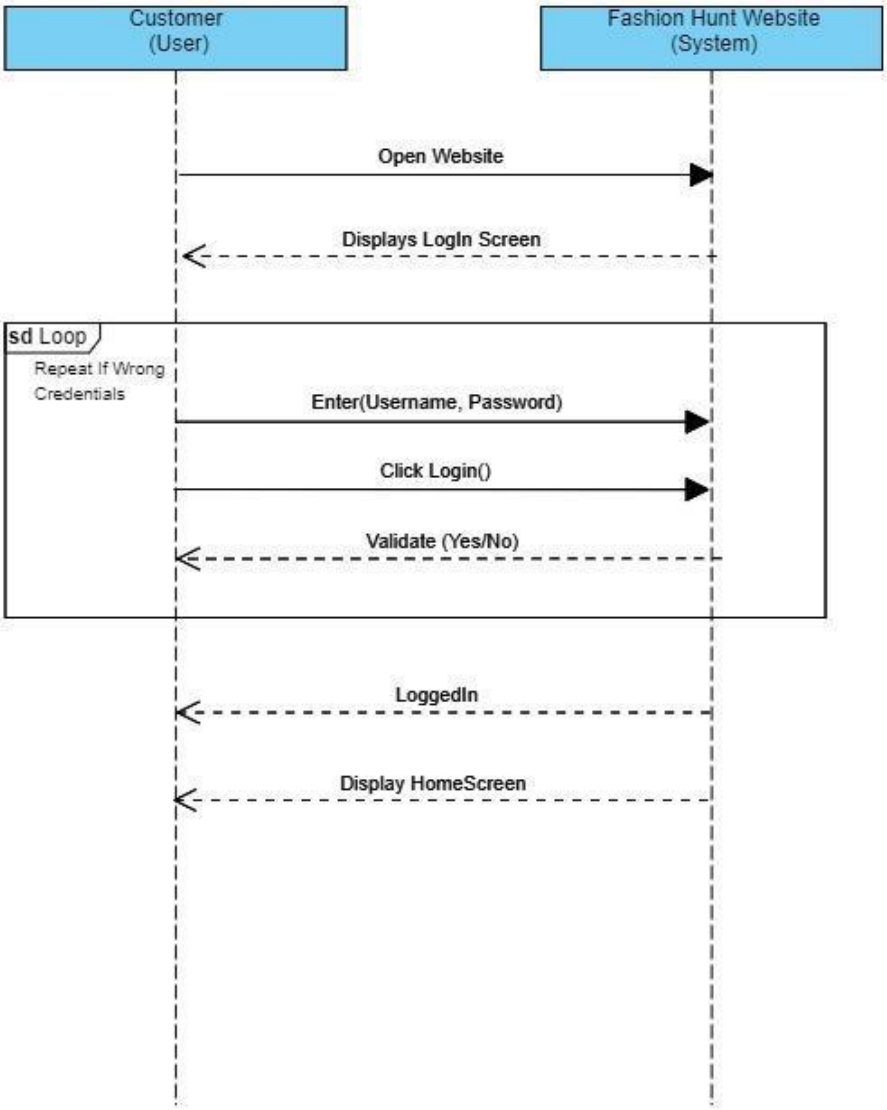


8.3 Domain Model

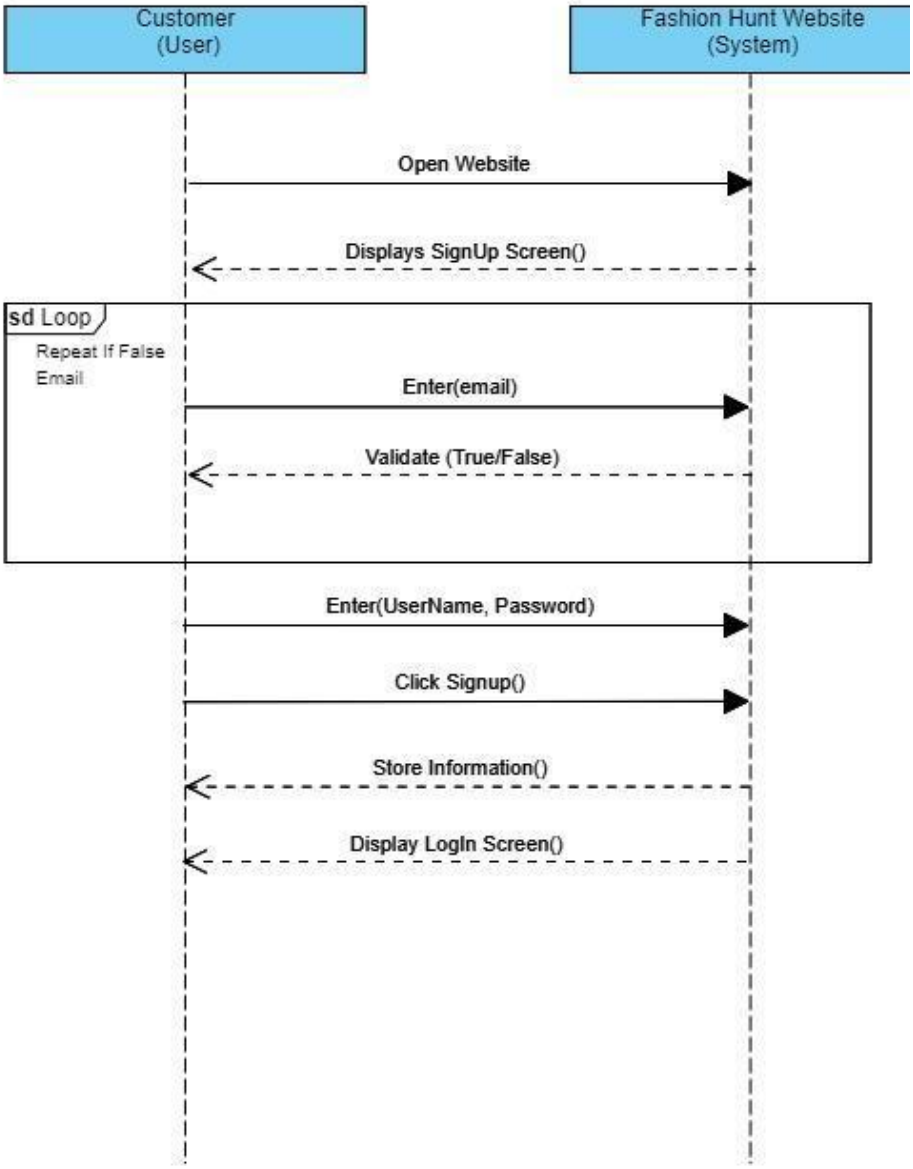


8.4 System Sequence Diagram

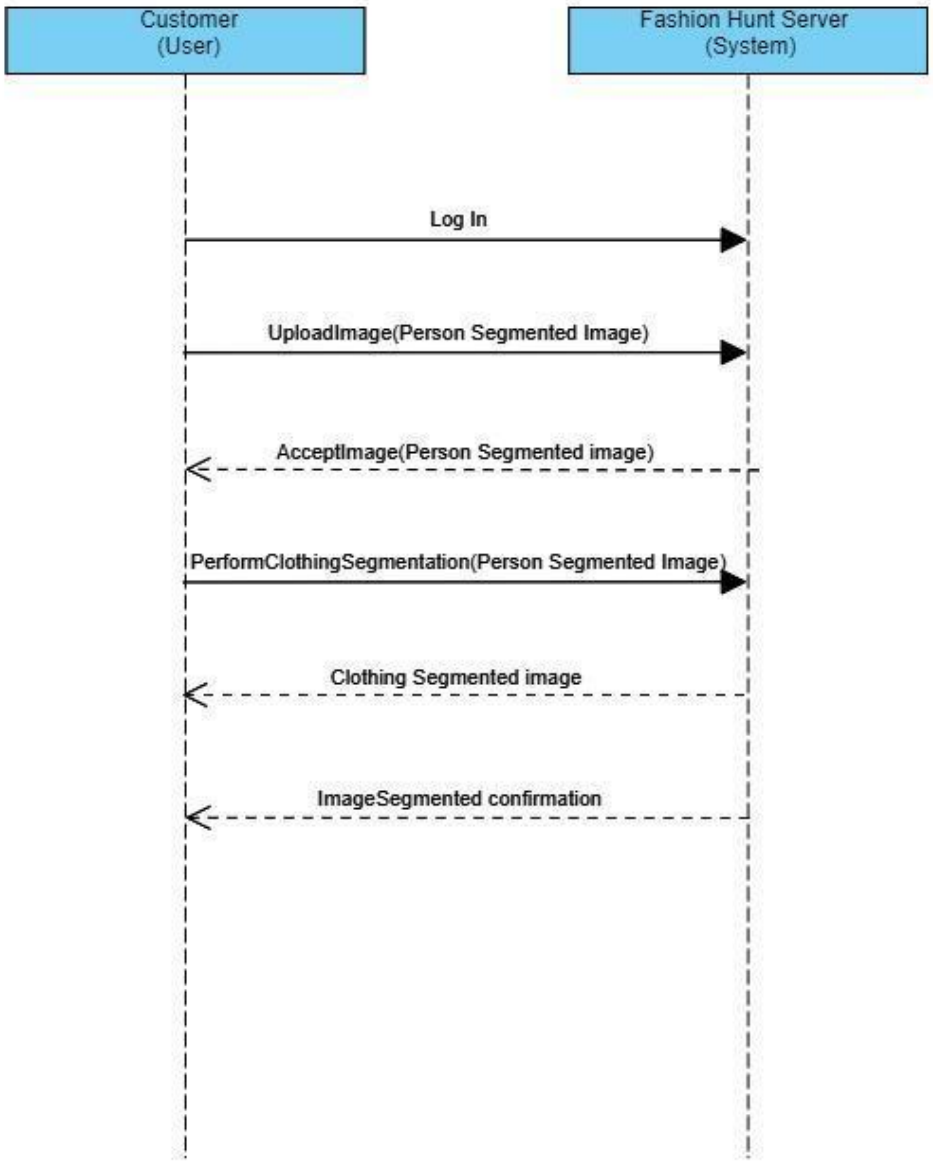
SSD 1: "Login"



SSD 2: “Sign Up”


Customer
(User)
Fashion Hunt Website
(System)
Open Website
Displays SignUp Screen()
sd Loop
Repeat If False
Email
Enter(email)
Validate (True/False)
Enter(UserName, Password)
Click Signup()
Store Information()
Display LogIn Screen()

SSD 3: “Perform Clothing Segmentation”



8.5 Operation Contracts

Operation	Enter (email)
Use Case	Signup
Responsibility	Registration of user in database
Pre-Condition	User must have valid email address
Post Condition	1. User instance will be created. (Instance creation) 2. User instance was associated with Data Loader. (Association Formed)

Operation	Enter (Username, Password)
Use Case	Login
Responsibility	Enters the user credentials
Pre-Condition	User must have valid credentials
Post Condition	1. User.data is set to User's information. (Attribute Modification)

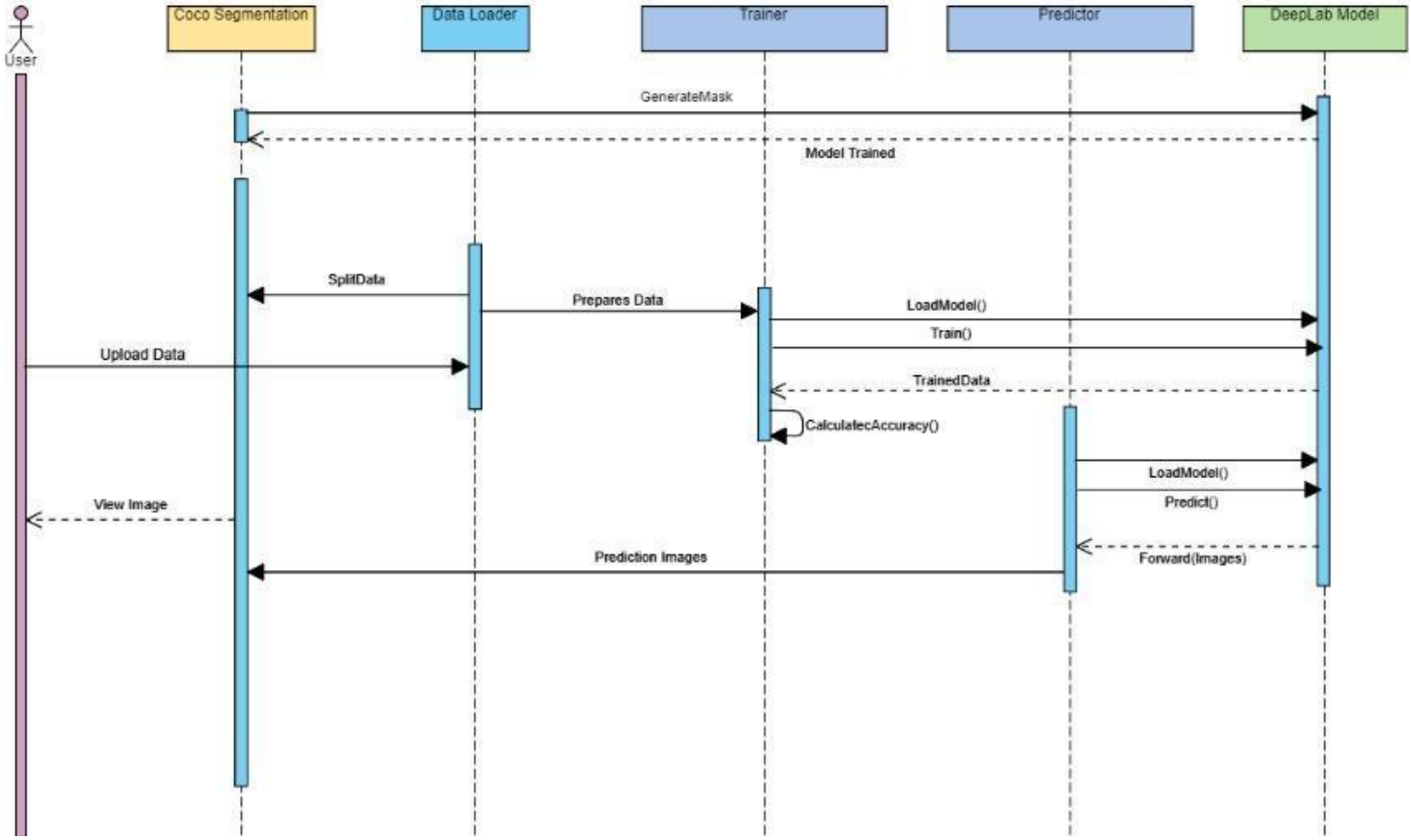
Operation	Validate(Yes/No)
Use Case	LogIn
Responsibility	Check Whether it is valid user
Pre-Condition	The user must try to logIn by entering credentials
Post Condition	User status Will be confirmed by system

Operation	Upload Image (Person Segmented Image)
Use Case	Perform Clothing Segmentation
Responsibility	Send Person segmented image to the model
Pre-Condition	The system must have performed person segmentation
Post Condition	Image Query. Image was set to Clothing segmentation. (Attribute Modification)

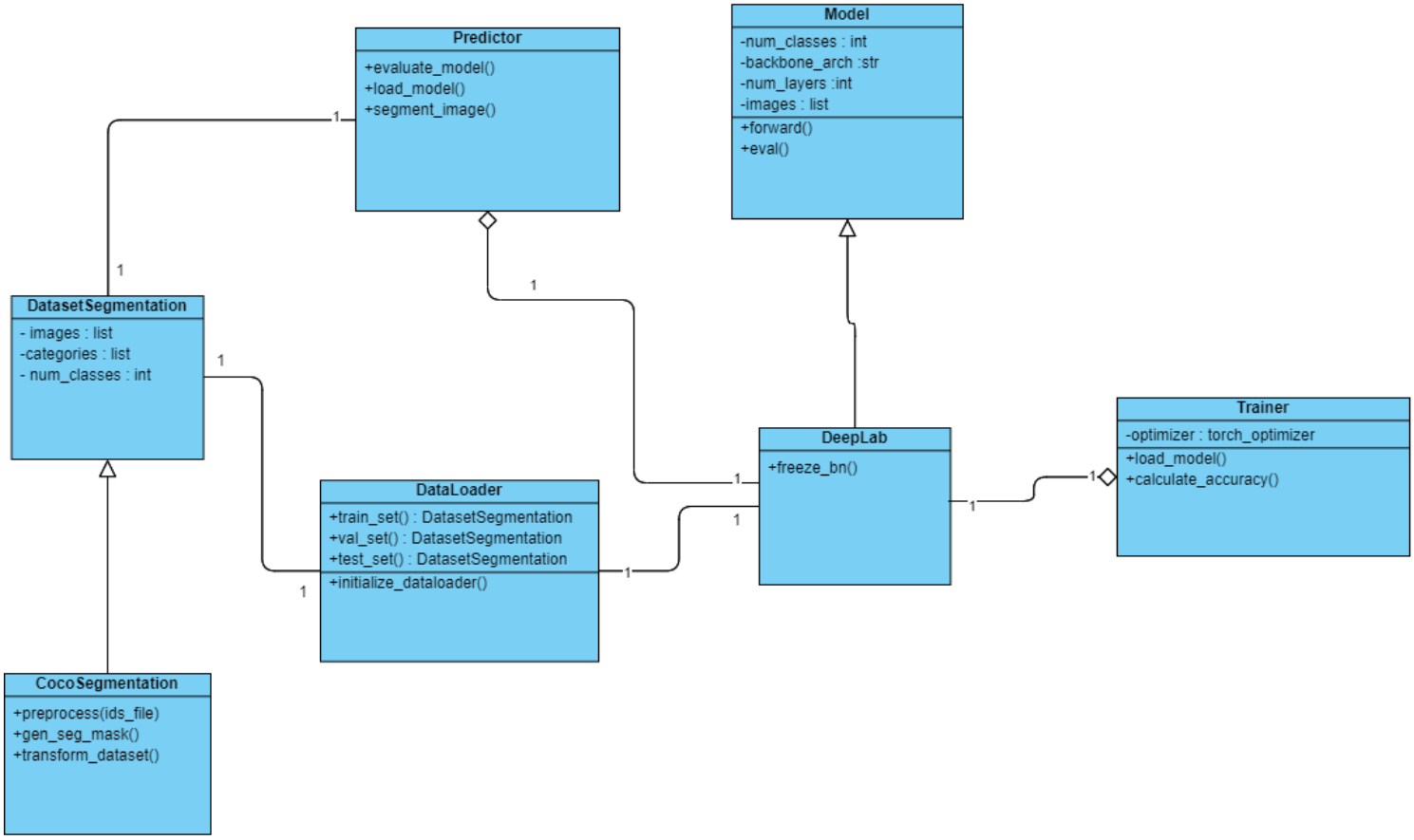
Operation	Accept Image (Person Segmented Images)
Use Case	Perform Clothing Segmentation
Responsibility	Accept Person Segmented Image
Pre-Condition	The user must upload an image with proper masks and correct size.
Post Condition	4. Data Loader instance was created. (Instance creation) 5. Data Loader. Image was set to “Image”. (Attribute modification) 6. Image Query was associated with Data Loader. (Association formed)

Operation	PerformClothingSegmentation (Person Segmented Image)
Use Case	Perform Image Segmentation
Responsibility	Performs Clothing Segmentation on the image to separate the clothing segments (pants, shirts) from the image
Pre-Condition	The Customer must upload proper image
Post Condition	Attribute Classification instance was formed. (Instance creation)

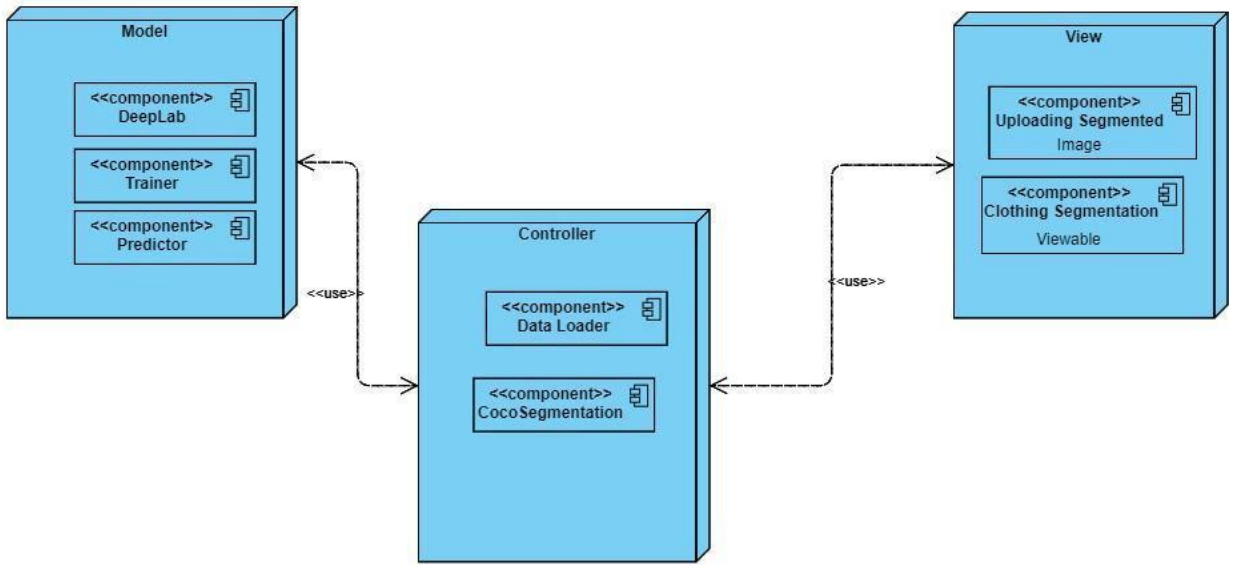
8.6 Sequence Diagram



8.7 Class Diagram



8.8 Architecture Diagram (MVC)



9 Iteration 3

9.1 Expanded Use Cases

Expanded Use Case 1: "Select Particular Clothing Segment"

Name	Select Particular Clothing Segment
Scope	Fashion Hunt Application
Level	User goal
Primary Actor	(User)
Stakeholders and Interests	Customer: Wants to select a particular segmented portion to see similar image results of that portion(pant or shirt).
Pre-Conditions	The Image must be clothing segmented
Post-Conditions	Clickable clothing segment will search similar results

Main Success Scenario:

Actor Action	**System Responsibility**
2. User Selects particular portion i.e. shirt or pants.	1. The System successfully performs clothing segmentation and displays separate clickable portions. 3. The system receives selected image 4. The system accepts the image. 5. The system sends images to the Siamese model for similar image retrieval.

Extensions	3a. The system couldn't accept the image size. 3a1. The system asks the user to re segment the image of appropriate size
Special Requirements	Response time of system error is 5 seconds
Technology and Data variation List	Pytorch Encoding is used
Frequency of use	Could be nearly continuous
Miscellaneous	What happens if a user selects a non segmented image?

Expanded Use Case 2: "Predict Clothing"

Name	Predict Clothing
Scope	Fashion Hunt Application
Level	User goal
Primary Actor	Fashion Hunt Server (System)
Stakeholders and Interests	Customer: Wants to have similar image results in accordance with the image that was uploaded.
Pre-Conditions	The Customer must have selected particular portion(shirt/pant) of image
Post-Conditions	The system will apply model patterns and will return similar image results.

Main Success Scenario:

Actor Action	**System Responsibility**
1. The Customer selects a shirt/pant from the screen.	2. The system gets that image. 3. The system applies the Siamese network on received images. 4. The system finds similar image results from the model. 5. System sends back selected results. 6. The system has successfully performed similar image retrieval.

Extensions	3a. The system couldn't find a similar image. 3a1. The system asks the user to again select an image.
Special Requirements	Response time of system error is 5 seconds

Technology and Data variation List	Pytorch Encoding is used
Frequency of use	Could be nearly continuous
Miscellaneous	What happens if a system displays unmatched images ?

Expanded Use Case 3: "Display Similar Clothing"

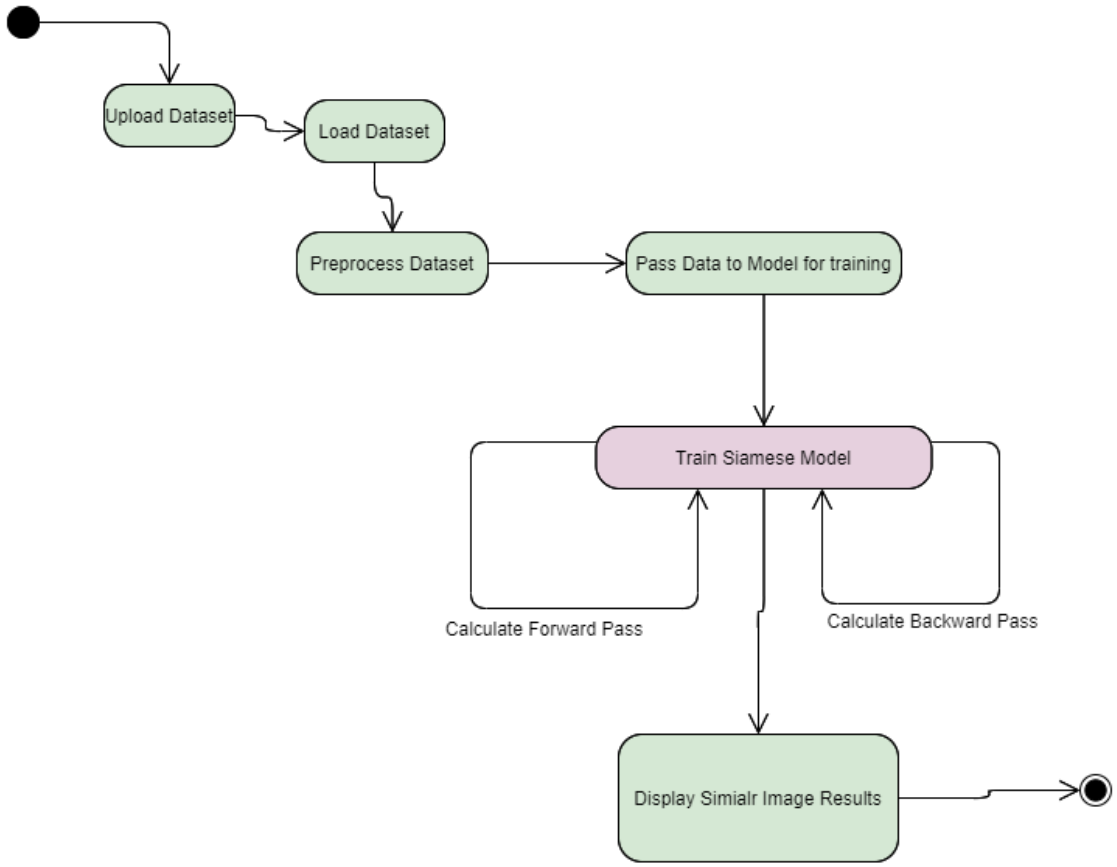
Name	Display Similar Clothing
Scope	Fashion Hunt Application
Level	User goal
Primary Actor	Fashion Hunt Server (System)
Stakeholders and Interests	Customer: Wants to see similar image results on screen.
Pre-Conditions	The System must have found similar images of selected images.
Post-Conditions	The system will display similar image results on screen.

Main Success Scenario:

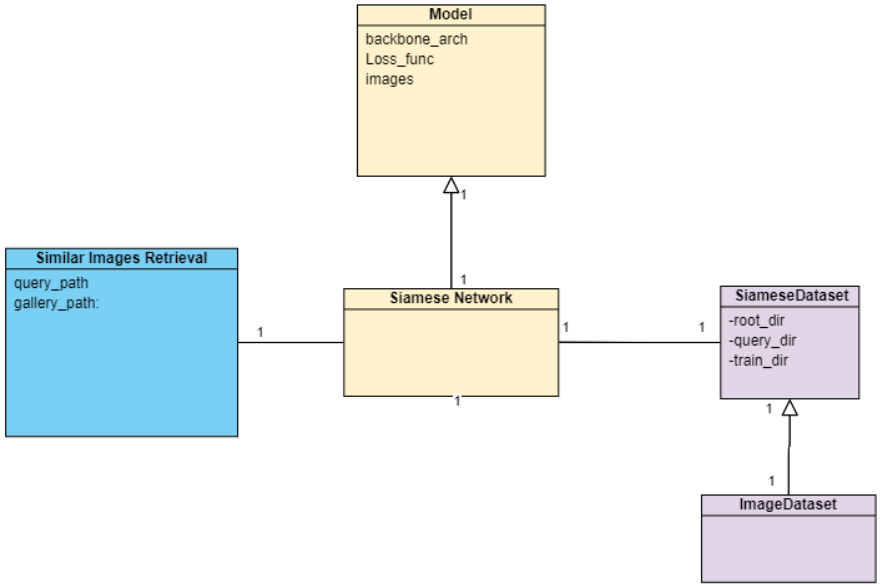
Actor Action	**System Responsibility**
	1. The system finds similar image results from the model. 2. System sends back selected results. 3. The system displayed similar results on screen.
4. The Customer can see and buy any desired clothing.	

Extensions	3a. The system couldn’t fetch results from the Model. 3a1. The system requests again to get results. 3a2. The system displays no result after a certain timeout.
Special Requirements	Response time of system error is 5 seconds
Technology and Data variation List	Pytorch Encoding is used
Frequency of use	Could be nearly continuous
Miscellaneous	What happens if the system gets results but doesn't display them?

9.2 Activity Diagram

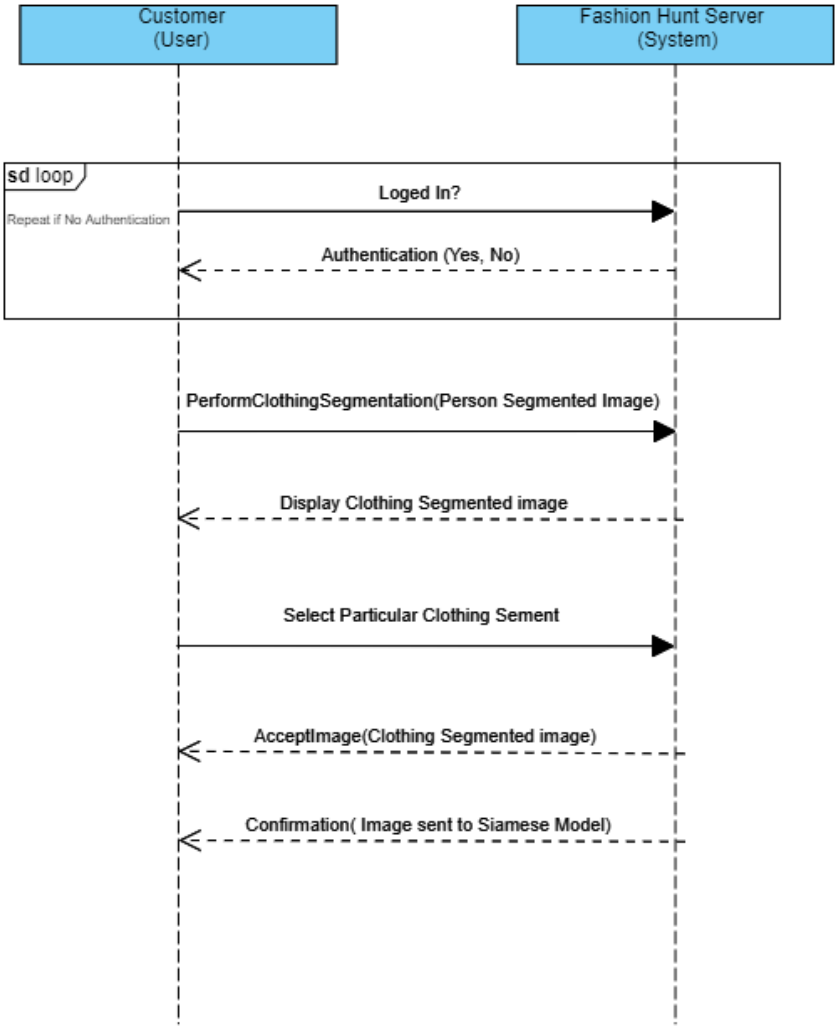


9.3 Domain Model

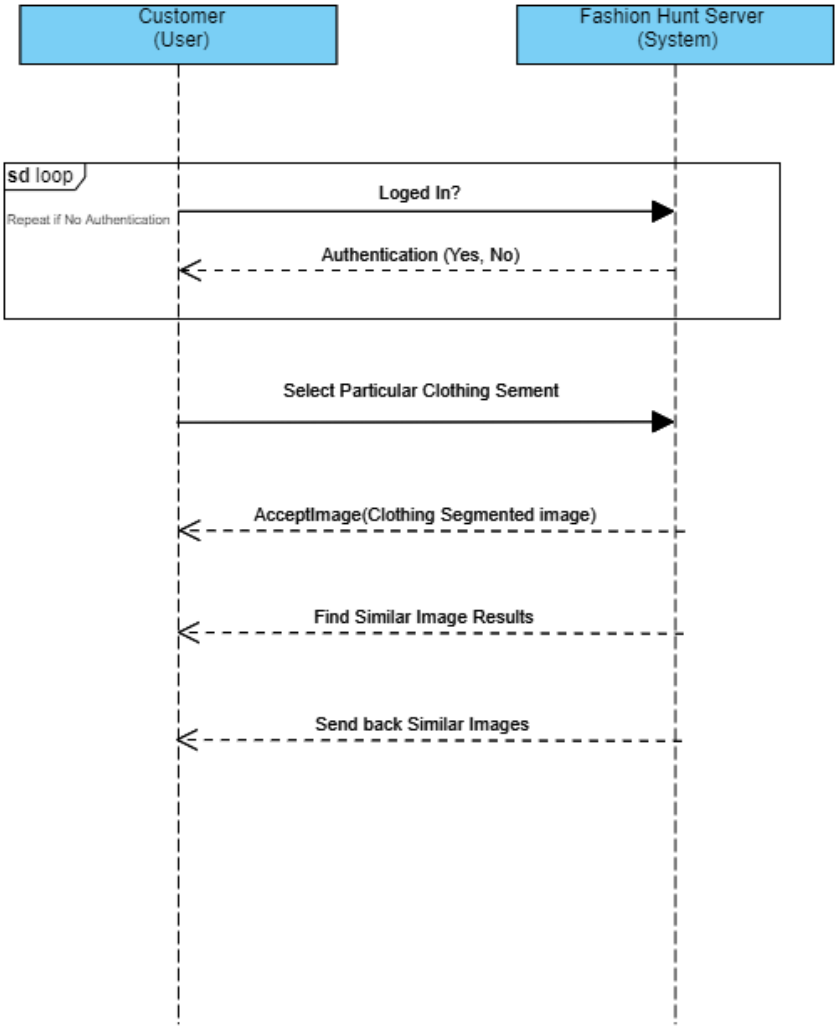


9.4 System Sequence Diagram

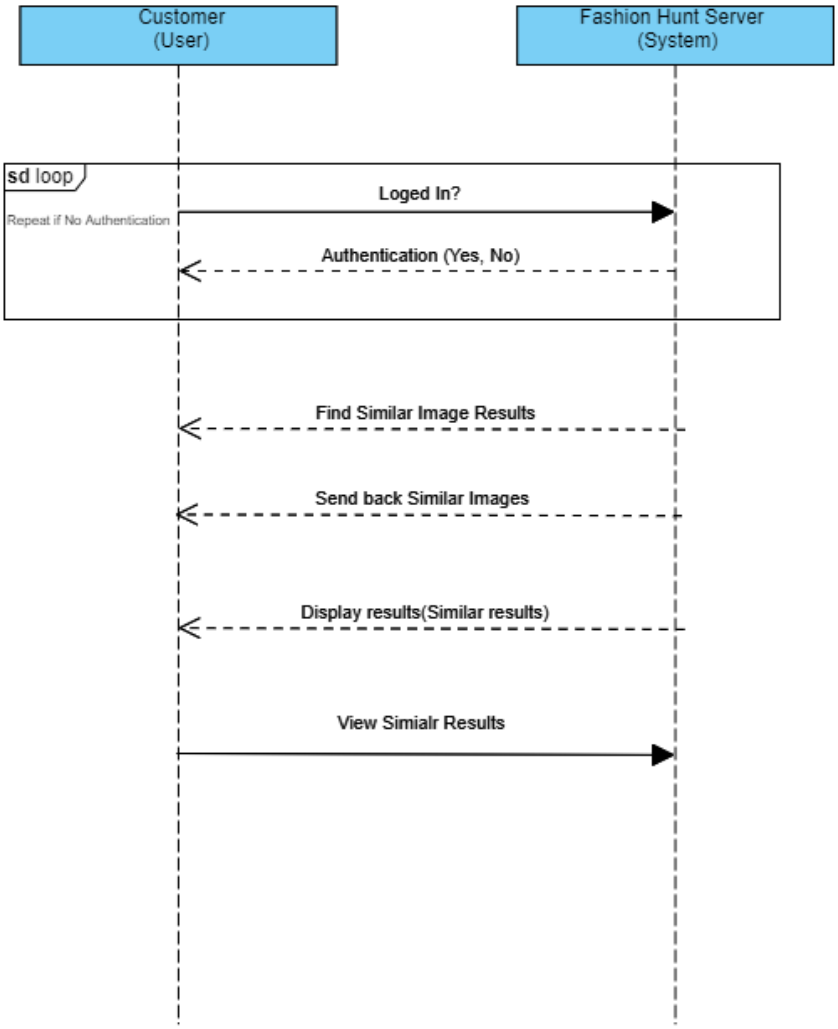
SSD 1: "Select Particular Clothing Segment"



SSD 2: “Predict Clothing”



SSD 3: “Display Similar Clothing”



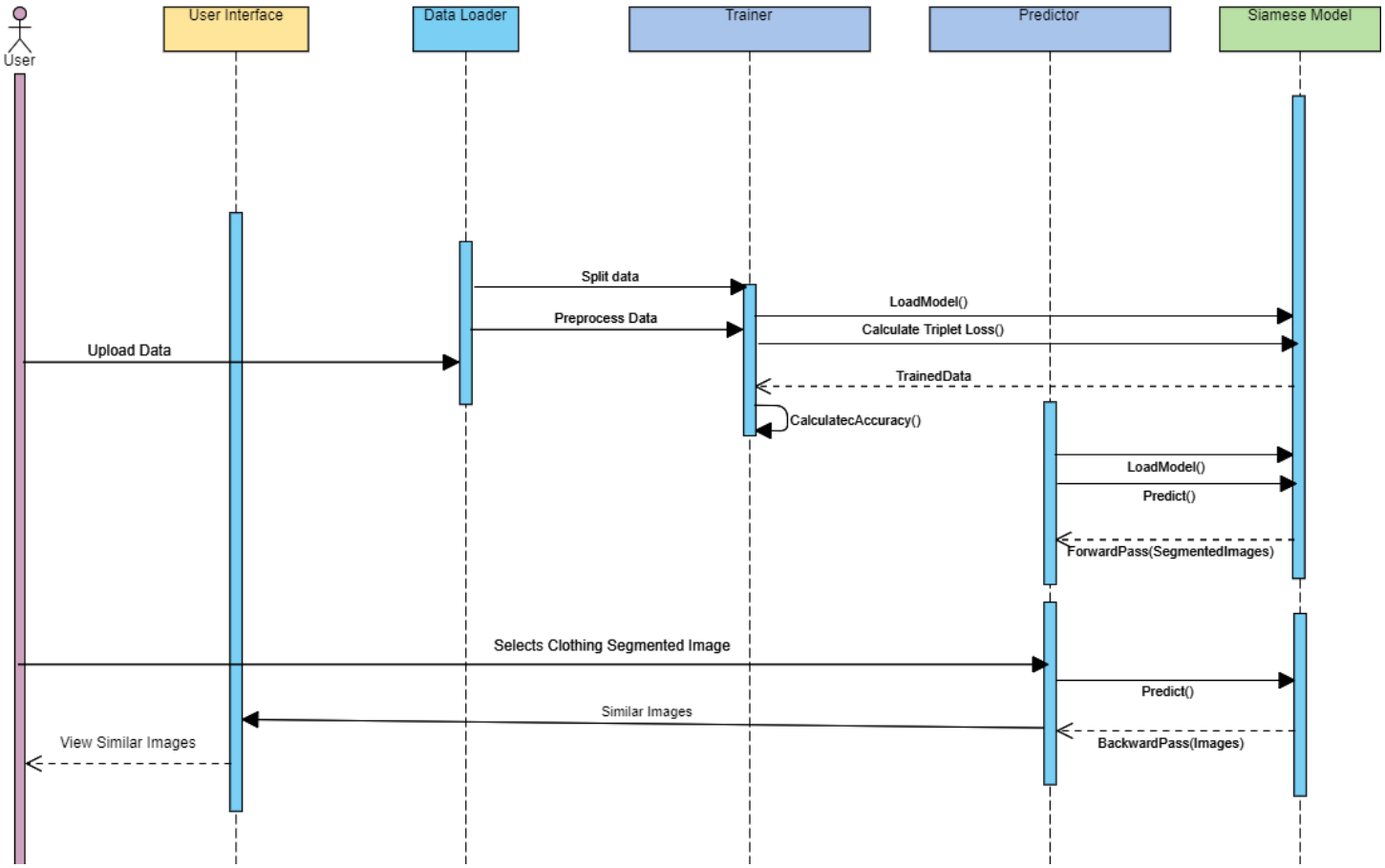
9.5 Operation Contracts

Operation	Accept Image (Clothing Segmented Images)
Use Case	Perform Clothing Segmentation
Responsibility	Accept Clothing Segmented Image
Pre-Condition	The user must upload an image with proper masks and correct size.
Post Condition	7. A Data Loader instance was created. (Instance creation) 8. Data Loader. Image was set to "Image". (Attribute modification) 9. Image Query was associated with Data Loader. (Association formed)

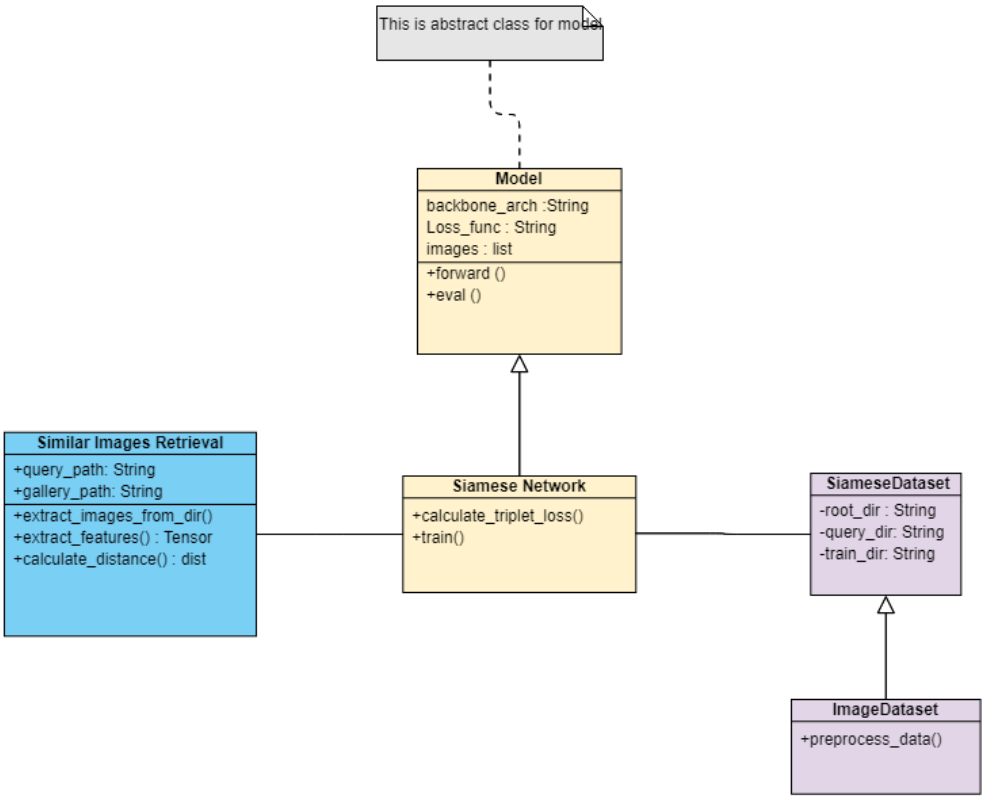
Operation	PerformClothingSegmentation (Person Segmented Image)
Use Case	Perform Image Segmentation
Responsibility	Performs Clothing Segmentation on the image to separate the clothing segments (pants, shirts) from the image
Pre-Condition	The Customer must upload proper image
Post Condition	Attribute Classification instance was formed. (Instance creation)

Operation	Display Results (Similar Results)
Use Case	View Similar Clothing Information
Responsibility	Similar Images are displayed on the UI
Pre-Condition	The Customer must select a segmented Image
Post Condition	SimilarImageRetrieval.gallery_path was set to 'Images'

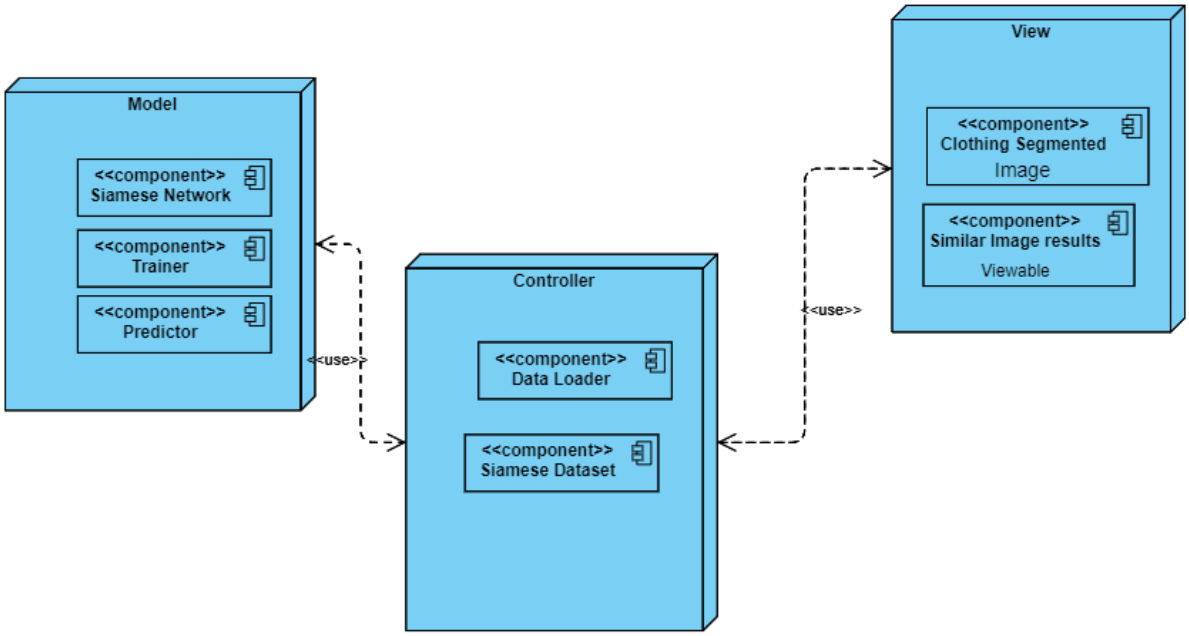
9.6 Sequence Diagram



9.7 Class Diagram



9.8 Architecture Diagram (MVC)



10 Iteration 4

10.1 Expanded Use Cases

Expanded Use Case 1: "Select retrieved clothing's URL"

Name	Select retrieved clothing's URL
Scope	Fashion Hunt Application
Level	User goal
Primary Actor	(User)
Stakeholders and Interests	Customer: Wants to select a particular URL of the image to go onto that website.
Pre-Conditions	Similar results along with URLs must be displayed on screen.
Post-Conditions	User will go onto a specific website for detailed information of the image product.

Main Success Scenario:

Actor Action	**System Responsibility**
2. User Selects particular URL of image.	1. The System successfully displayed similar image results along with their URLs. 3. The system gets a URL request. 4. The system accepts the request. 5. The system reallocates user to specific website.

Extensions	3a. The system couldn't accept the specified URL.

	3a1. The system asks the user to provide valid URL
Special Requirements	Response time of system error is 5 seconds
Technology and Data variation List	Routing and navigation is used.
Frequency of use	Could be nearly continuous
Miscellaneous	What happens if a user selects multiple URLs quickly?

Expanded Use Case 2: "View Information of selected URL"

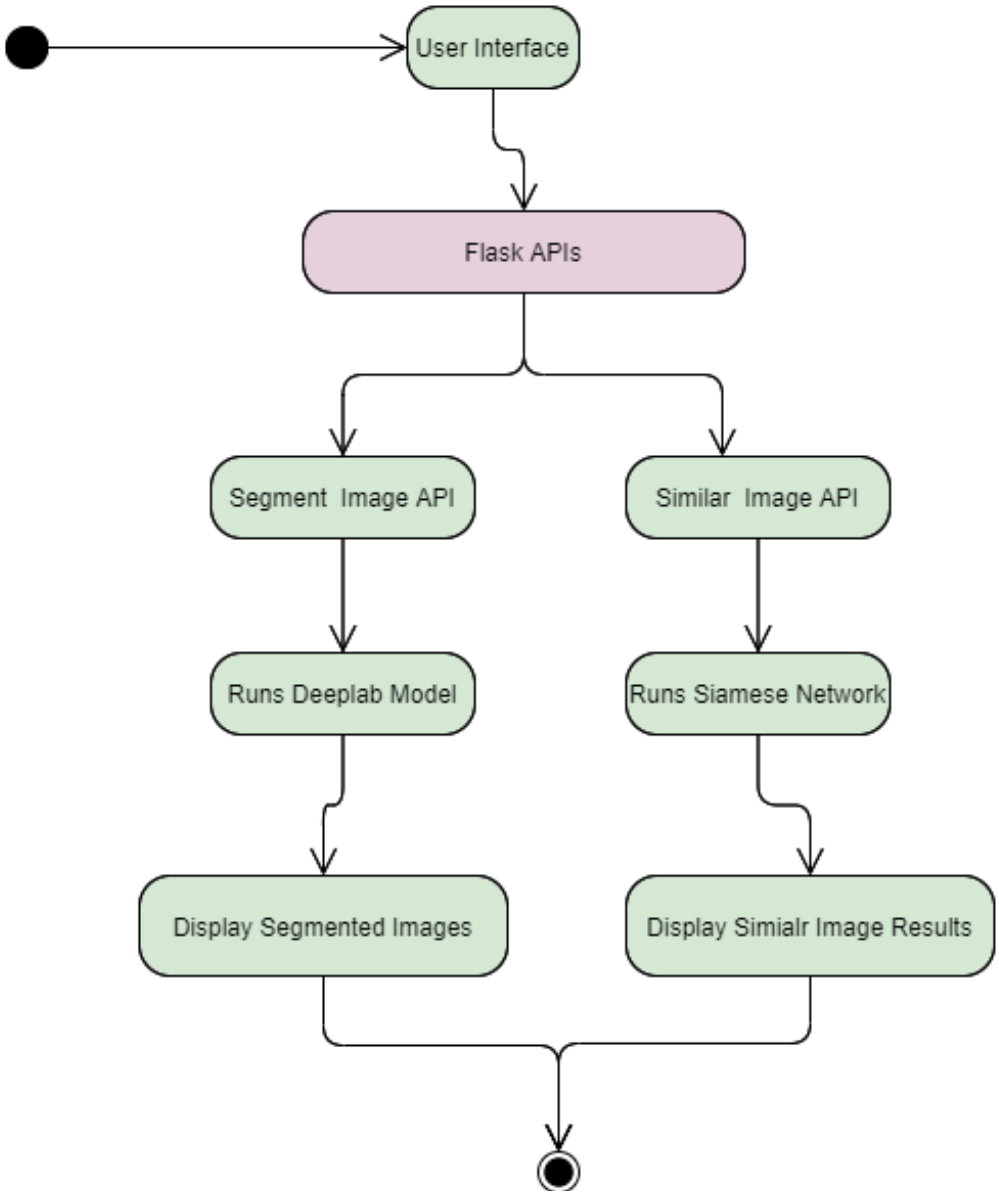
Name	View Information of selected URL
Scope	Fashion Hunt Application
Level	User goal
Primary Actor	User
Stakeholders and Interests	Customer: Wants to see selected URL's information before changing the side.
Pre-Conditions	The user must have selected some URL to look upon in it.
Post-Conditions	The user will get information about the URL.

Main Success Scenario:

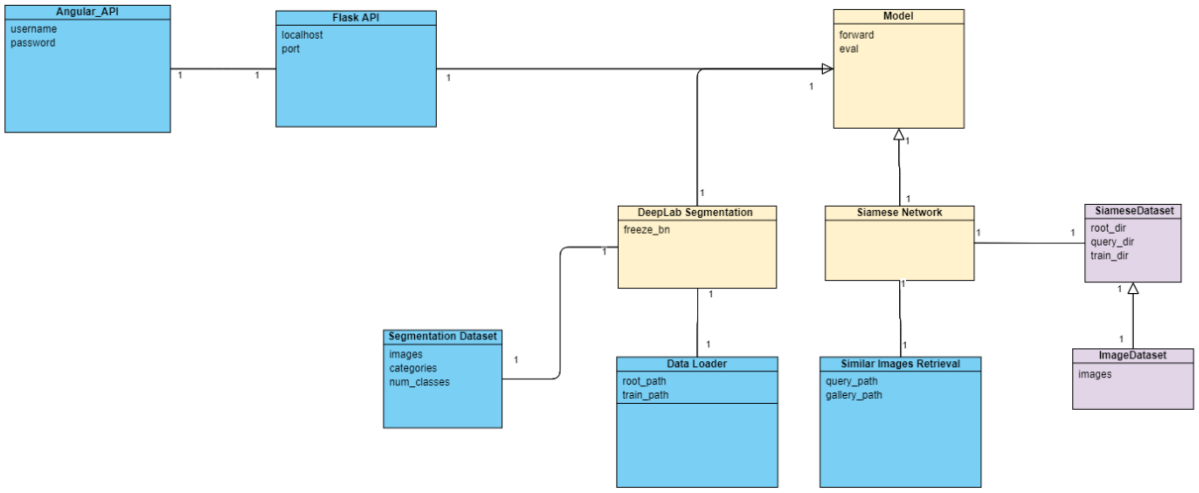
Actor Action	**System Responsibility**
	1. The system displays URLs of similar images
1. The Customer selects a particular URL. 2. The Customer clicks to check information of URL.	
	3. System displays information about that URL.

Extensions	3a. The system couldn’t fetch information about the URL. 3a1. The system asks the user to select again.
Special Requirements	Response time of system error is 5 seconds
Technology and Data variation List	Navigation is used
Frequency of use	Could be nearly continuous
Miscellaneous	What happens if the system gets results but doesn't display them?

10.2 Activity Diagram

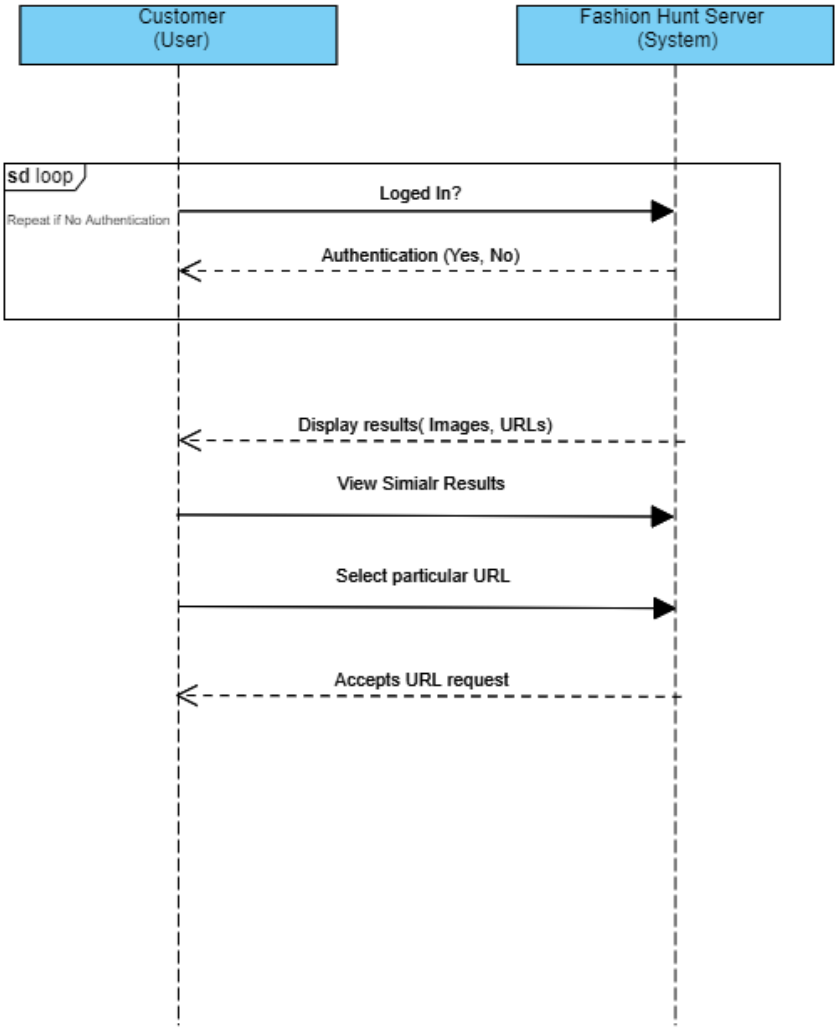


10.3 Domain Model

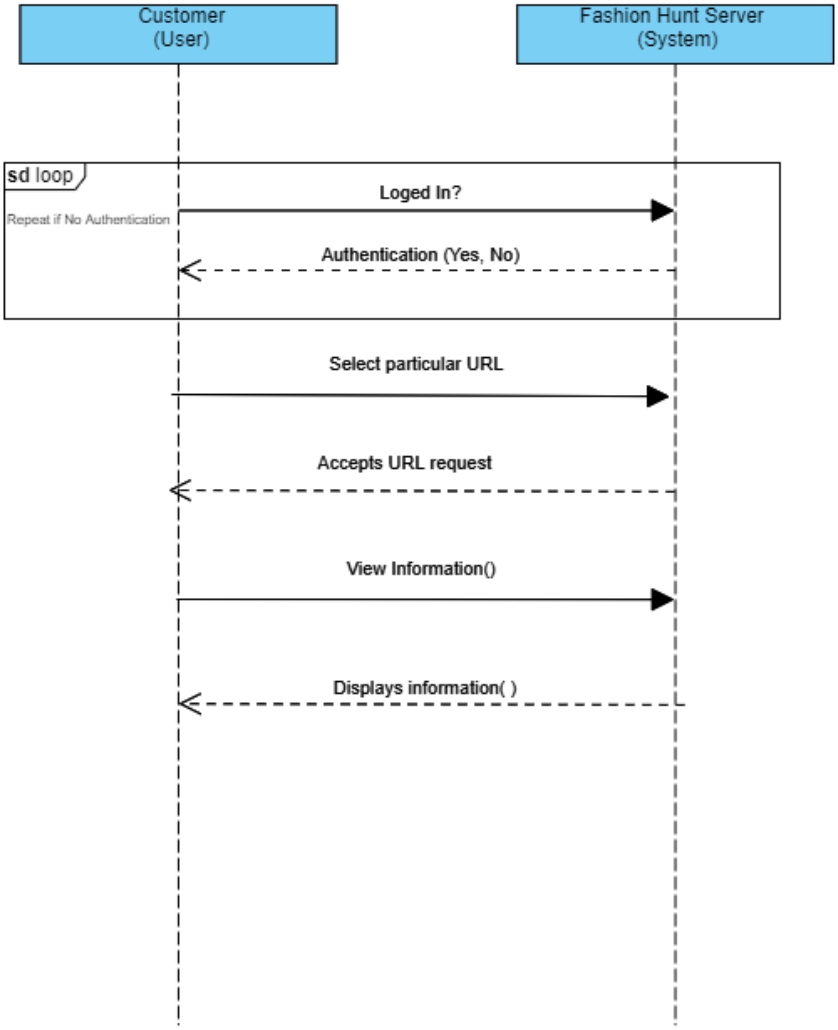


10.4 System Sequence Diagram

SSD 1: "Select retrieved clothing's URL"



SSD 2: “View Information of selected URL”



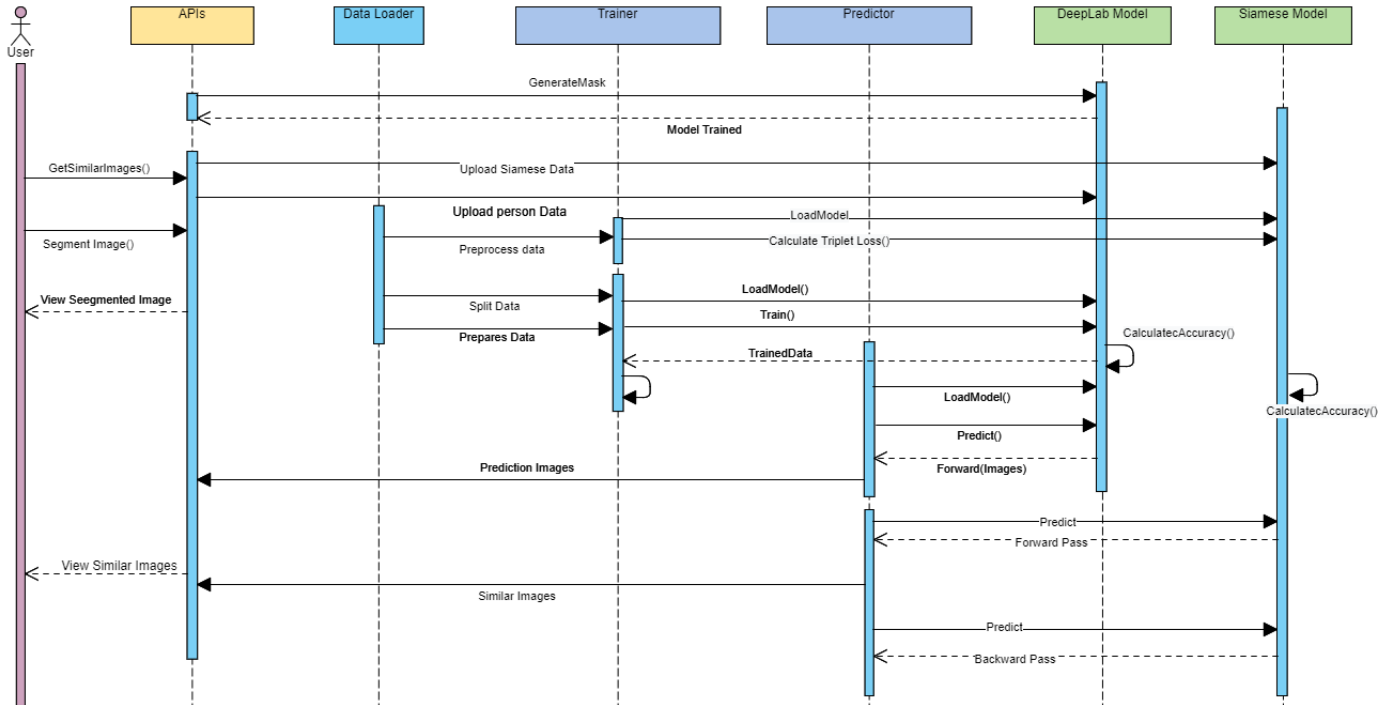
10.5 Operation Contracts

Operation	Display Results (Similar Results)
Use Case	View Similar Clothing Information
Responsibility	Similar Images are displayed on the UI
Pre-Condition	The Customer must select a segmented Image
Post Condition	SimilarImageRetrieval.gallery_path was set to 'Images'

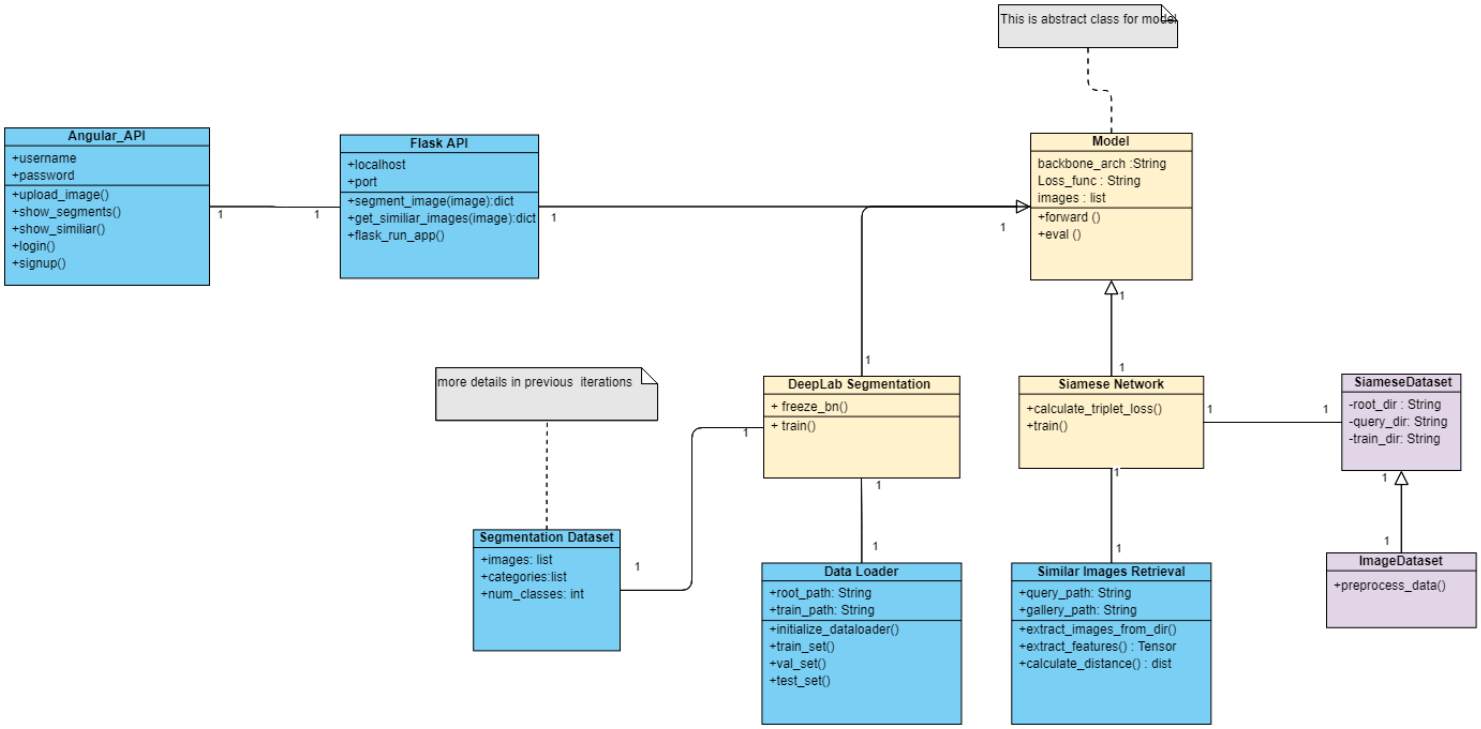
Operation	View Information ()
Use Case	View Information of Selected URL
Responsibility	View Selected Image's information like price, availability, size etc
Pre-Condition	The Customer must select a displayed Image
Post Condition	SimilarImageRetrieval.gallery_path was set to 'Images'

Operation	Displays Information ()
Use Case	View Information of Selected URL
Responsibility	Displays Selected Image's Information like price, availability, size etc
Pre-Condition	The Customer must select a segmented Image
Post Condition	SimilarImageRetrieval.gallery_path was set to 'Images'

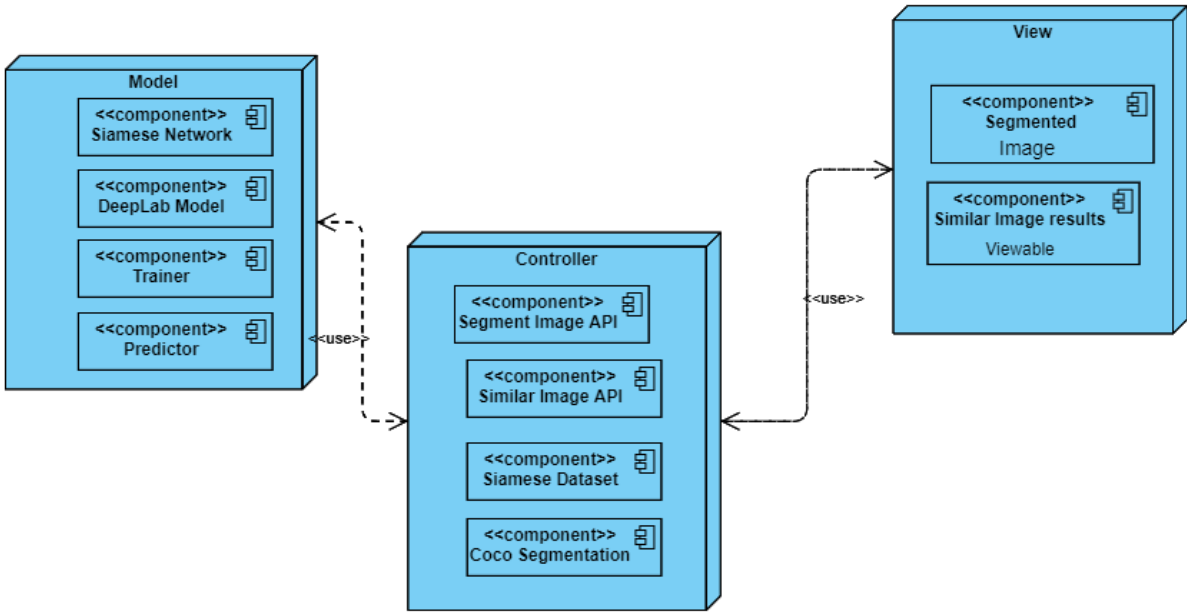
10.6 Sequence Diagram



10.7 Class Diagram

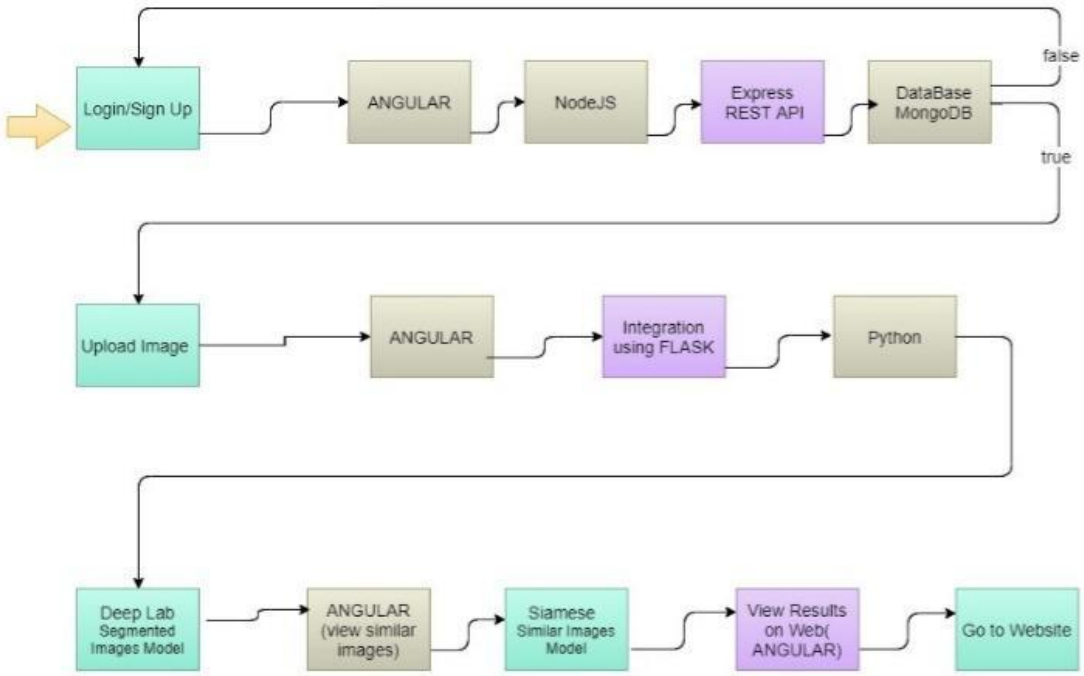


10.8 Architecture Diagram (MVC)



11 Implementation Details

'Complete Workflow'



11.1 Iteration 1

Person Segmentation

Pytorch Encoding [2] library is used to run segmentation models available on the internet. Pre-trained models can be downloaded and run without any training process.

Steps to run on Google Collab

1) Mount the google drive and add the path in path variable
2) Load the model into GPU
3) Download the image from URL that we want to segment

4) To Predict from model, a forward function is called, and it generates mask and output results same as input image size
5) As we want to segment only person class from the image so we will grab the color-code for the person class
6) The output image is saved and loaded again in black and white

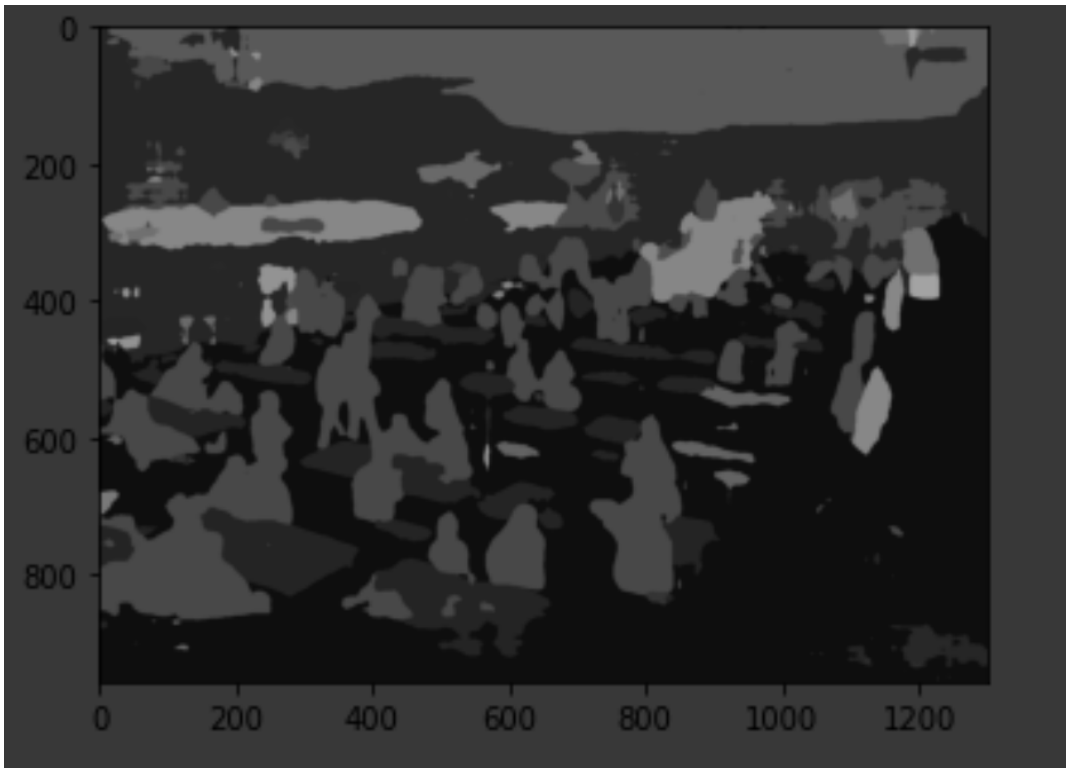


Figure 1.1: Mask Generated By Model

7) All pixels are traversed and if pixels are the same color as person class, only that part from the original picture is redrawn on a new 3 channel colored image exclusively cropping the person and placing it into the new image.

Figure 1.2: This is a Segmented Image Result

Web Scrapping

Tools Required

- Selenium
- Beautiful Soap

Steps to Scrape Data

1) First the required website is opened using Chrome Driver E.g. Opening Khaadi website
2) All the required information of the page is get by using inspect of browser
    a. Get all the links that are specified by navbar (top menu of each website) by inspecting it
    b. Get all images from image links specified on page
    c. Get all URL of all image links displayed on page
    d. Get all the page no links displayed at the bottom of each page (if exists)
3) Store the above information in variables and arrays

4) Perform the following to scrape the data
    a. Run loop for total no of links that exists on the webpage
    b. For each link, scroll manually to load images of the page
    c. Run the driver to fetch the information from the page having the parameters as stored previously
    d. Store the images in the folder where as its links in a text file
    e. Now shift to the next of the same link e.g. page no 2, page no 3…
    f. Repeat the process from 4c) to 4e)
    g. If page no doesn't exist or end of pages has been reached, then move to next link and repeat steps 4b) to 4f)

11.2 Iteration 2

Annotations on customized Dataset

Tool Used: LabelMe

Link: https://github.com/wkentaro/labelme

Image segmentation task requires that we do a proper annotation of the dataset for the machine to know which class the object belongs to and what are the pixel coordinates of the object.

Working:

1) We will take a sample image which wants to be annotated, the image must be placed in the working directory.
2) We'll use the following command to open the labelme environment

```
Ø labelme child2.jpg -O abc.json
```

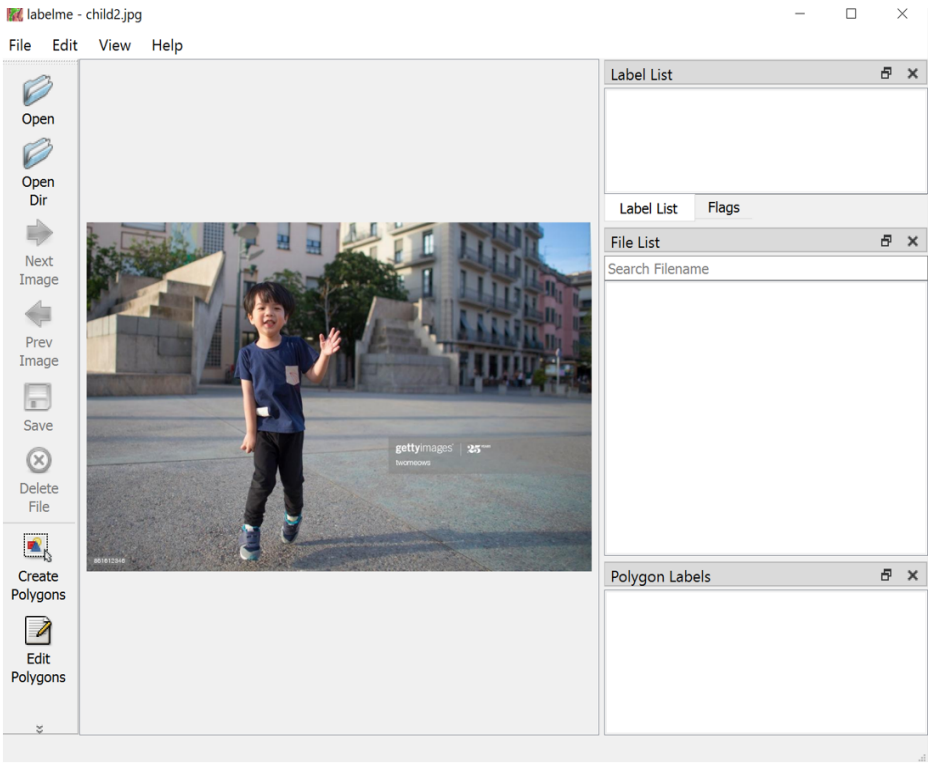


Figure 2. 1: child2.jpg is the name of image file and abc.json is the output label file

3) Now we will create a polygon on the image to separately label each class. For example, the first class is 'person'

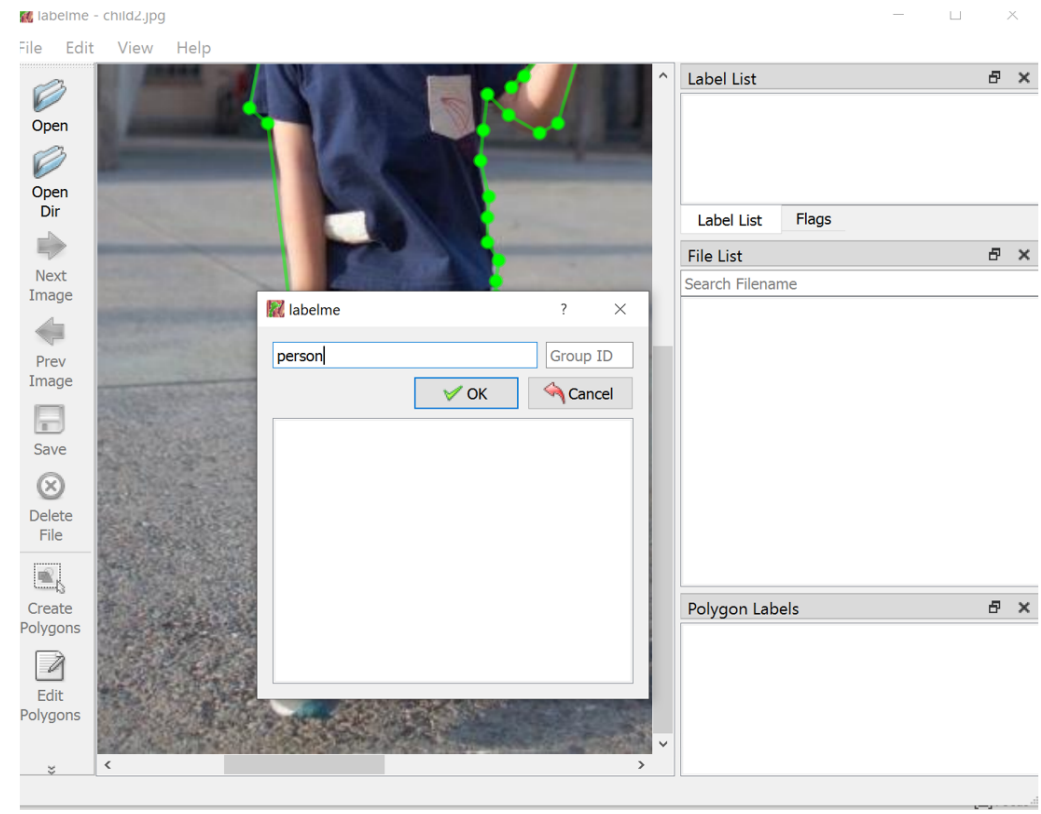


Figure 2. 2: Defining the class name in labelme

4) Similarly, we will use 2 more classes: 'shirt', ‘pant’.

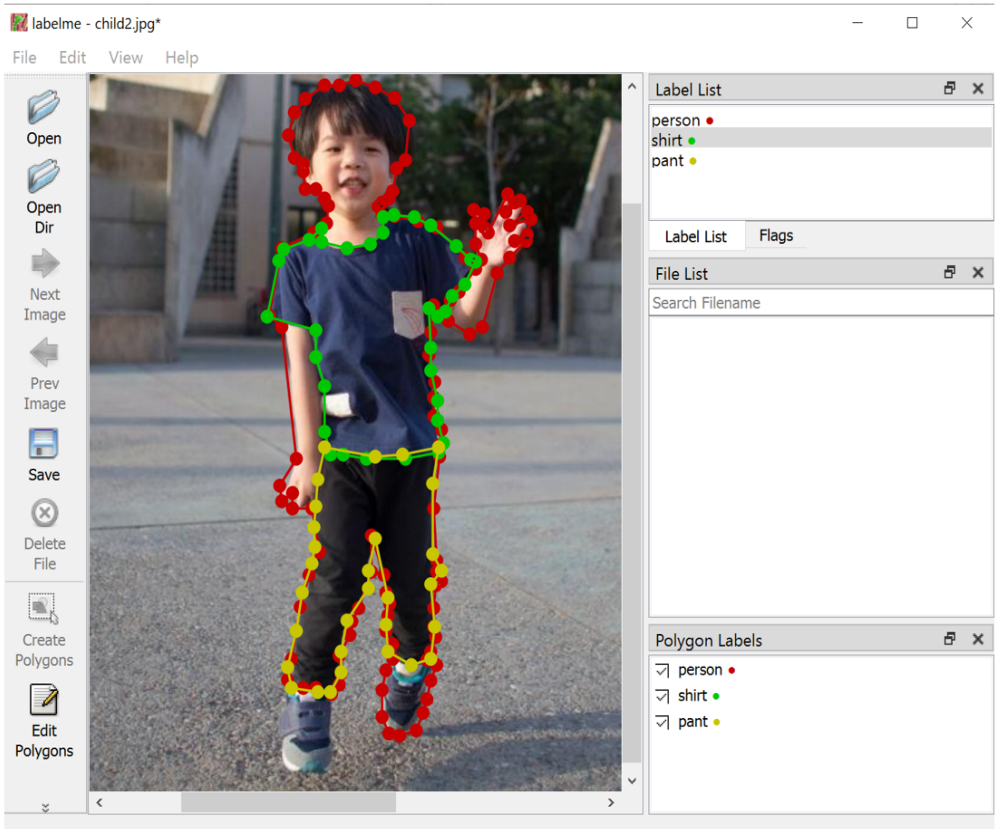


Figure 2. 3: Generating polygon points on the image

5) We will save the file

6) This data is not in appropriate format; to apply deep learning models we need to convert it into coco dataset format.

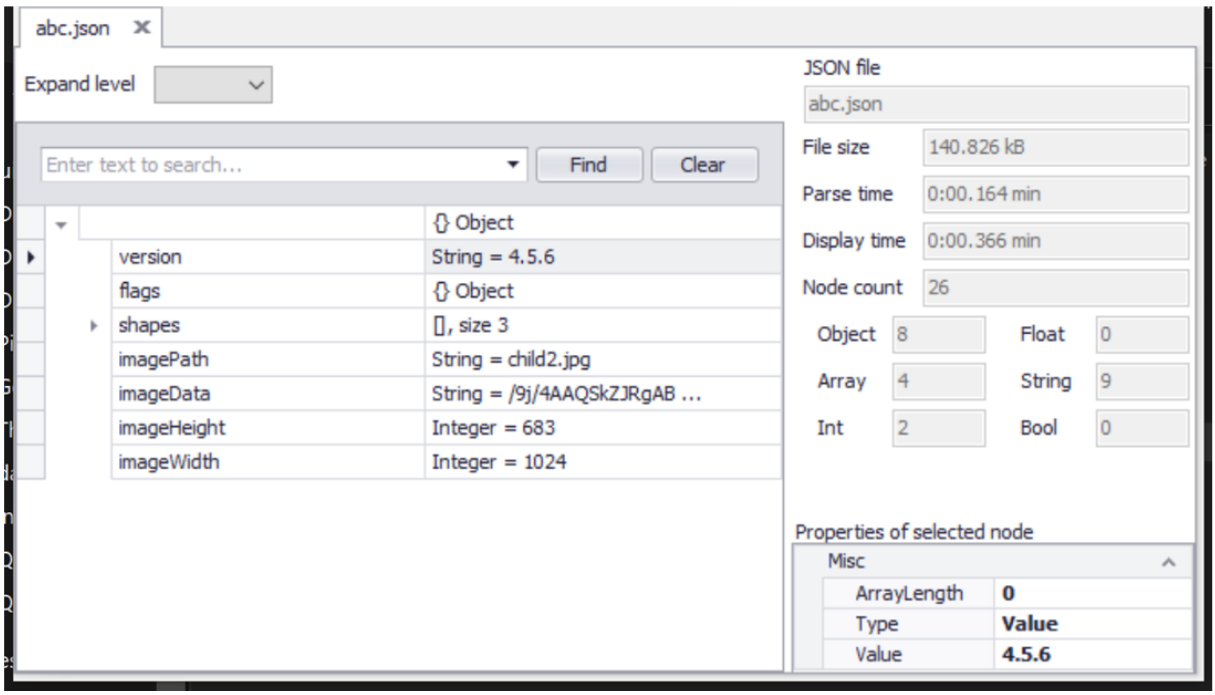


Figure 2. 4: Viewing the file in coco format

7) To convert to coco_dataset format we'll type the following command

- **python labelme2coco.py data_annotated data_dataset_coco --labels labels.txt**

data_annotated is the path of a working directory having polygon data json files and images and data_dataset_coco is the destination directory.

Following is image output

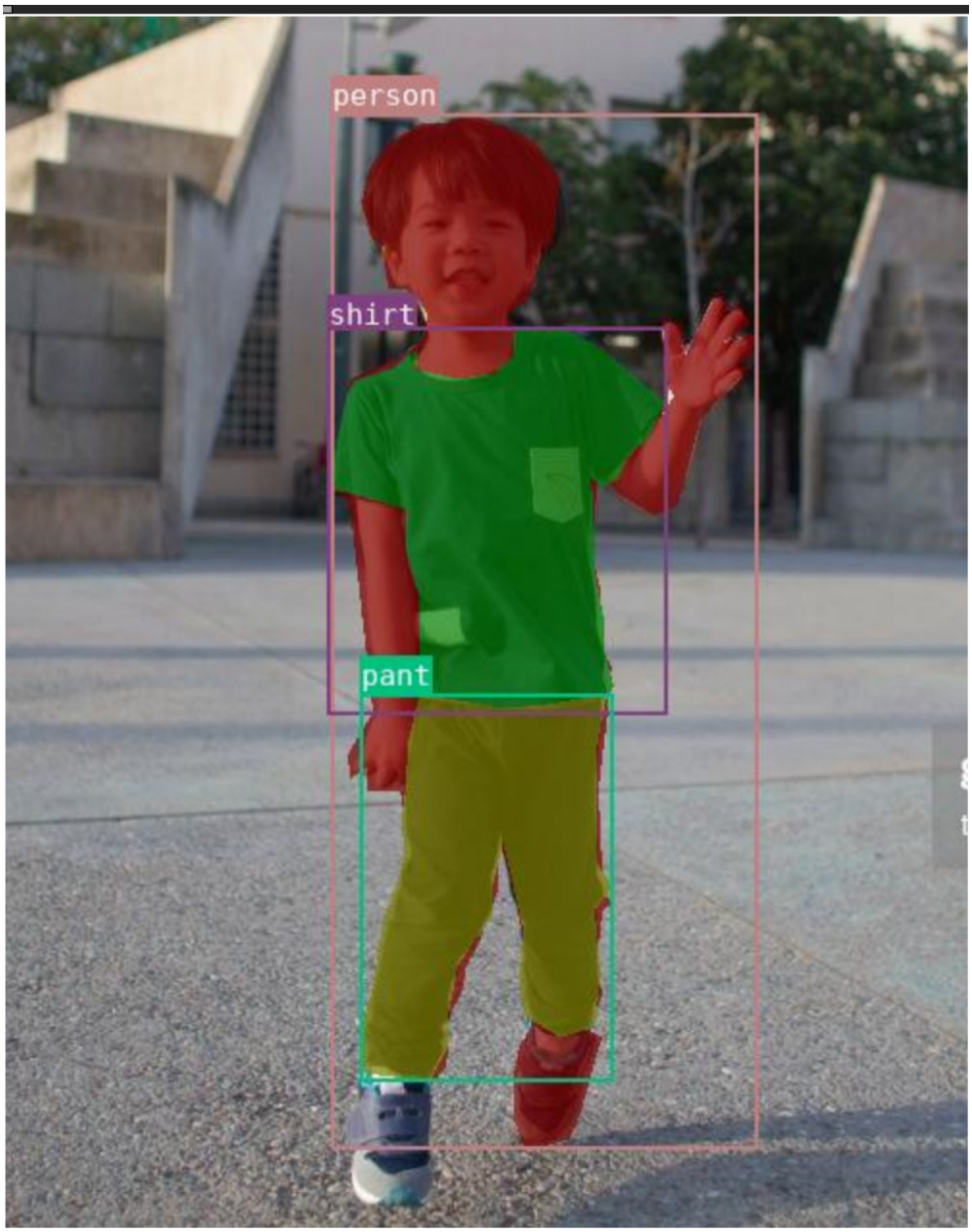


Figure 2. 5: Drawing bounding boxes on the figure defining each class

Coco has created annotations.json file for annotations

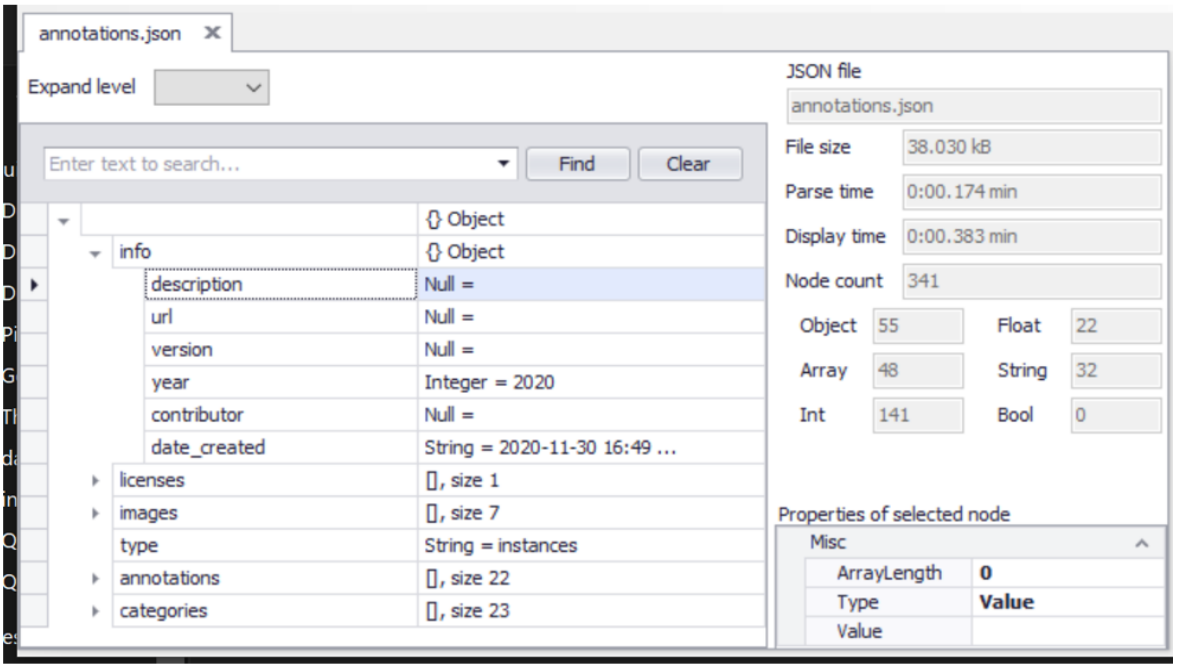


Figure 2. 6: Showing the whole representation of the dataset

8) Now this is the representation of whole dataset and it can be used for training of deep learning models

Training on Customized dataset

Model Used: DeepLab

Link to library: https://github.com/jfzhang95/pytorch-deeplab-xception.git

Model training Intro:

We have used deep learning techniques for semantic segmentation of clothing objects. There are many models which could be trained to get better results. We have referred to this Master's thesis before selecting a model and found out DeepLab V3 suits best for our requirement.

DeepLabV3 Architecture:

DeepLabv3 is an extension of DeepLabv2 that adds an encoder-decoder structure to improve segmentation results in general and along the boundaries of the segmented objects in particular, and uses the improved backbone model ResNet101. DeepLabv3 is used as an encoder, and uses atrous convolution and defines output stride as the ratio of the input image size to the final encoded feature. DeepLabv3 employs a low output stride of 16 to be able to extract denser feature maps, which is preferable in the task of semantic segmentation. The purpose of the decoder is to refine the segmentation along object boundaries. Following figure represents the encoder-decoder structure and shows the concatenation of the up sampled features from the encoder with the convoluted low-level features from the DCNN. After concatenation the result is refined with a 3x3-convolution and then lastly up sampled with a factor of 4. DeepLabv3 set a new State-of-the-Art performance, scoring an MIoU of 82,1% on the Cityscapes dataset.

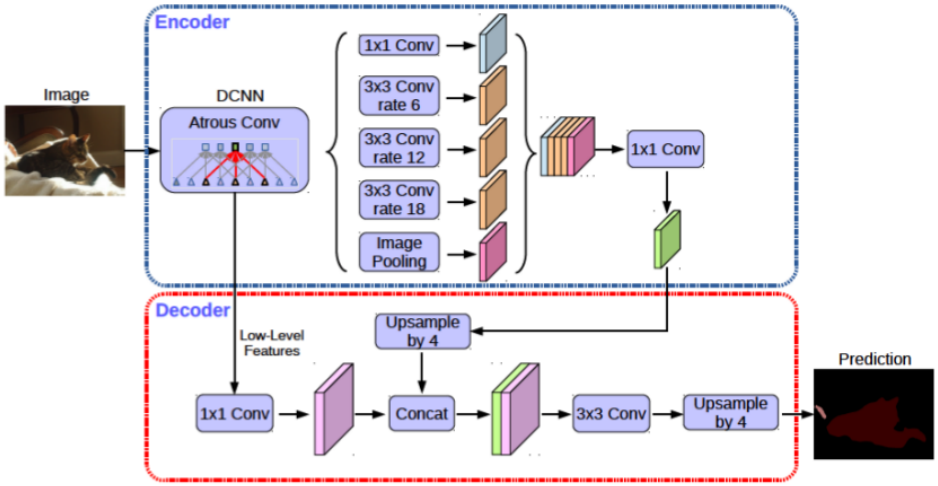


Figure 2. 7: Deep Lab V3 Layered Architecture

Model Training

Our goal is to configure the model and tune the hyper parameter that suits best for better pixel accuracy. We have prepared our own dataset and trained the model on it, in order to perform segmentation on clothing.

Following are the hyper parameters that we are using

```
--backbone resnet

 --lr 0.01
```

```
--workers 2

--epochs 100

--batch-size 2

--checkname deeplab-resnet

--eval-interval 1

--dataset coco
```

After 5 hours of training on 200 customized dataset images we have achieved Validation accuracy

Validation:

[Epoch: 99, numImages: 200]

Acc: 0.8963,

Acc_class: 0.8617,

MIoU: 0.7662,

FWIoU: 0.8196

Loss: 4.846

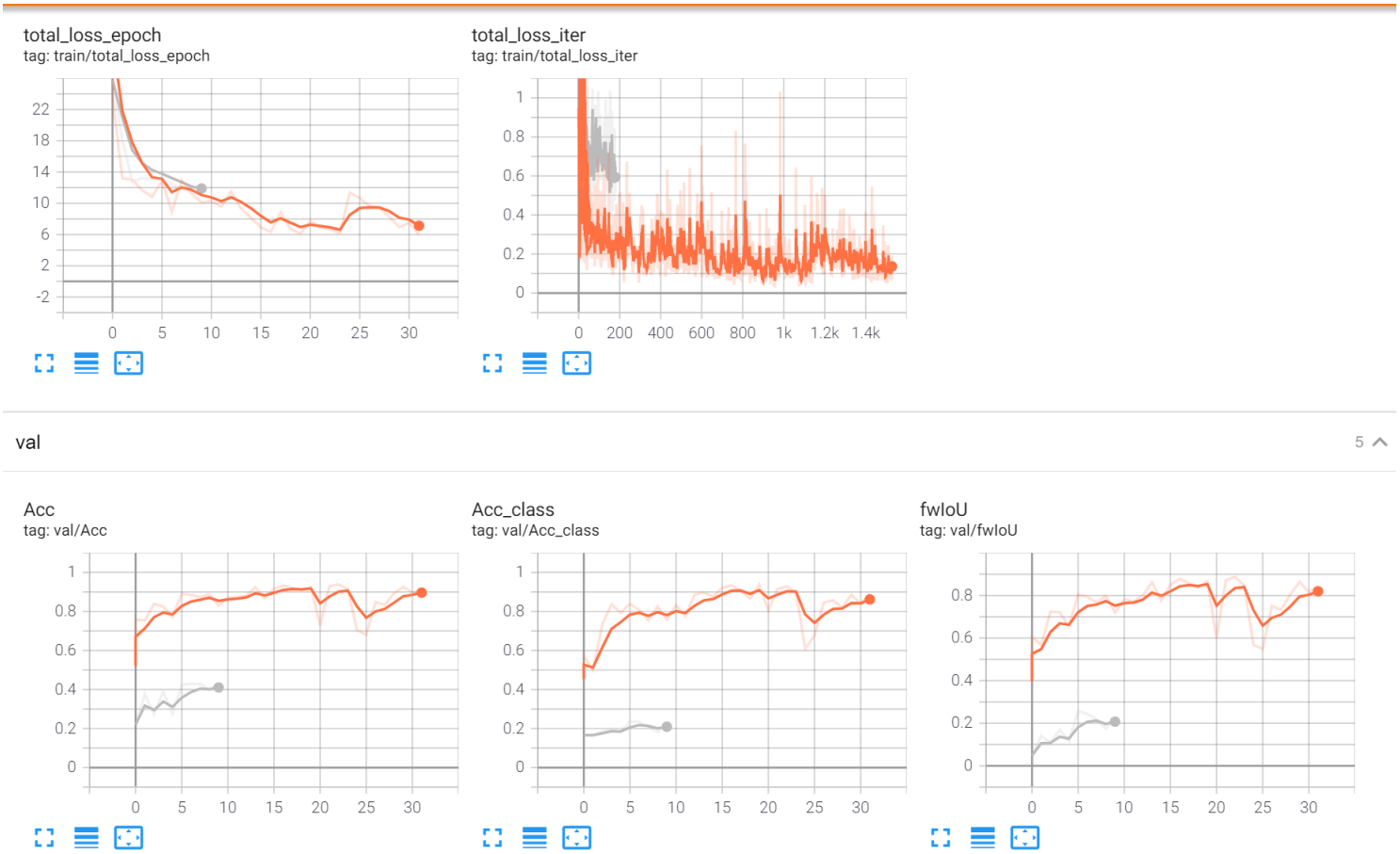


Figure 2. 8: Graph showing Validation accuracy using tensorflow board

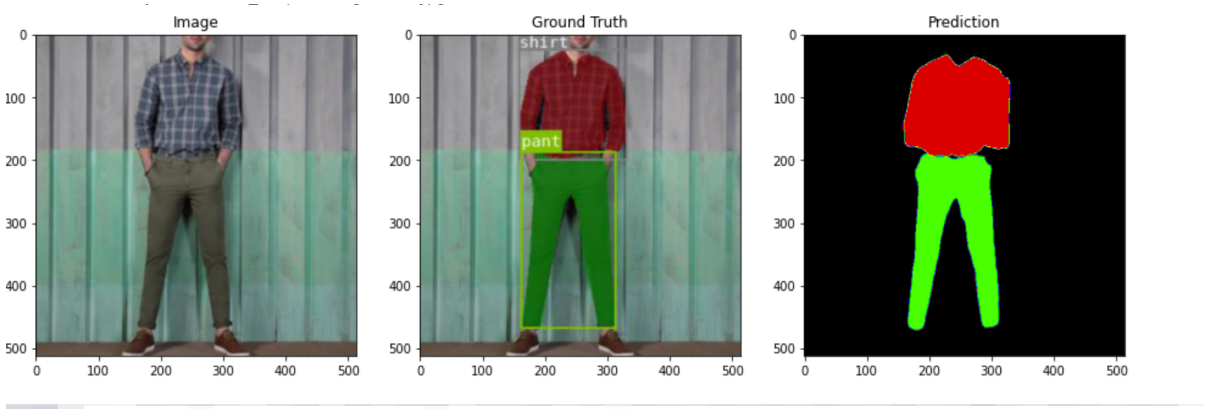
Predicting on Trained Data

Model Used: DeepLab

Link to library: https://github.com/giovanniguidi/deeplabV3-PyTorch.git

We have used different libraries to perform predictions on our trained deep learning model. DeepLab eval() function is called to transform an image into a multiclass image where each number represents a class label. 0 represents the background and 10 represents pants and so on. The list of numbers gets

translated by predictor class and a segmentation mask is generated to show the output to the user. The



output of the segment_image function is three images.

Figure 2. 9 : The figure 1 is the actual image, second is the ground truth and third is the predicted output from the model

Results Achieved:

We have predicted our model on unseen data and achieved following results

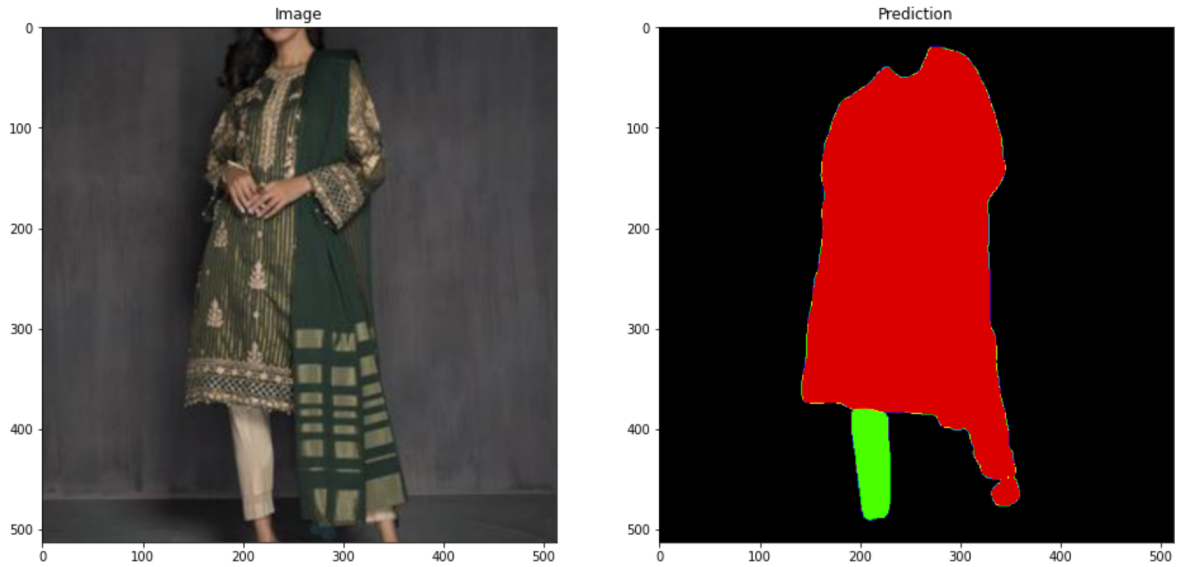


Figure 2. 10 : Actual Vs Predicted

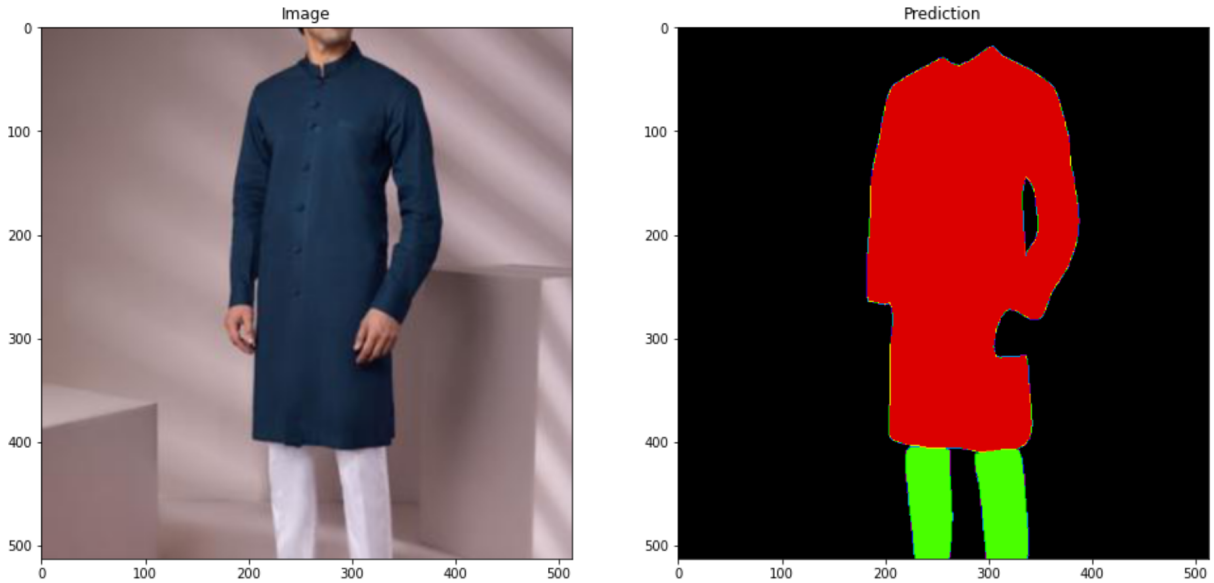


Figure 2. 11: Actual Vs Predicted

Figure 2. 12 : Actual Vs Predicted

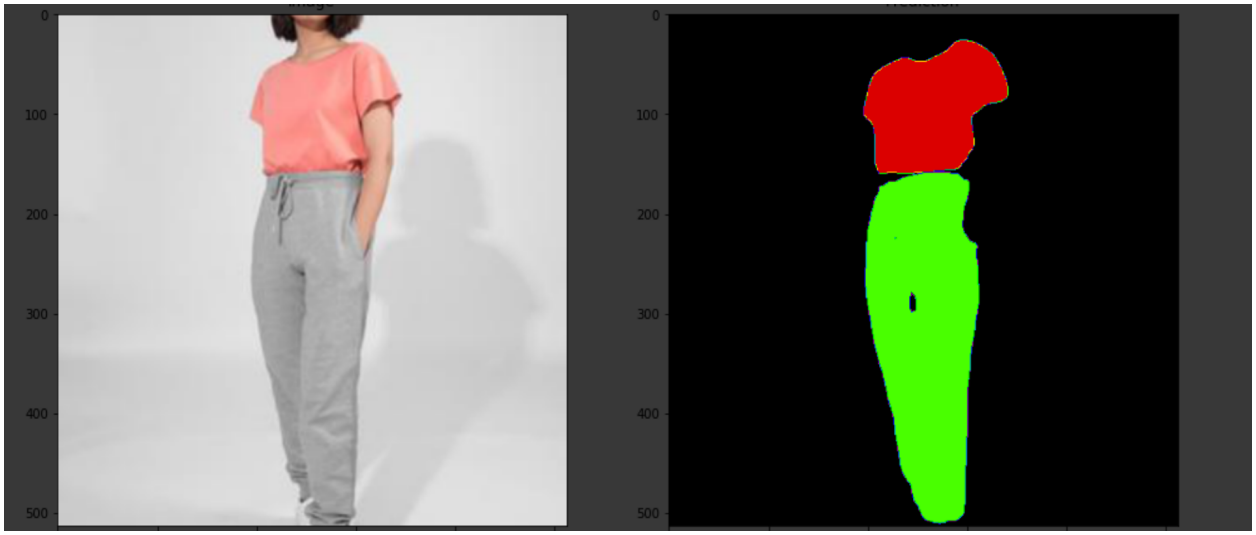


Figure 2. 13: Actual Vs Predicted

Figure 2. 14: Actual Vs Predicted

Website

Website is created using MEAN Stack Development. The user will have access to the website content after authentication. Following are the implementations completed so far.

Login (Authentication & Authorization) and Sign Up

The user should sign up before using the website. He/she should enter his/her first name, last name, email and password. He/she must satisfy all the validation formats e.g. correcting “Email already exists”,

“Password should have at least 8 characters” invalidation errors. Once the user is signed up, he/she is prompted to ‘Login’ Page, where he/she verifies himself/herself using email and password, to proceed. Fig.2.15 and Fig.2.16 shows ‘Login’ and ‘Sign Up’ pages respectively.

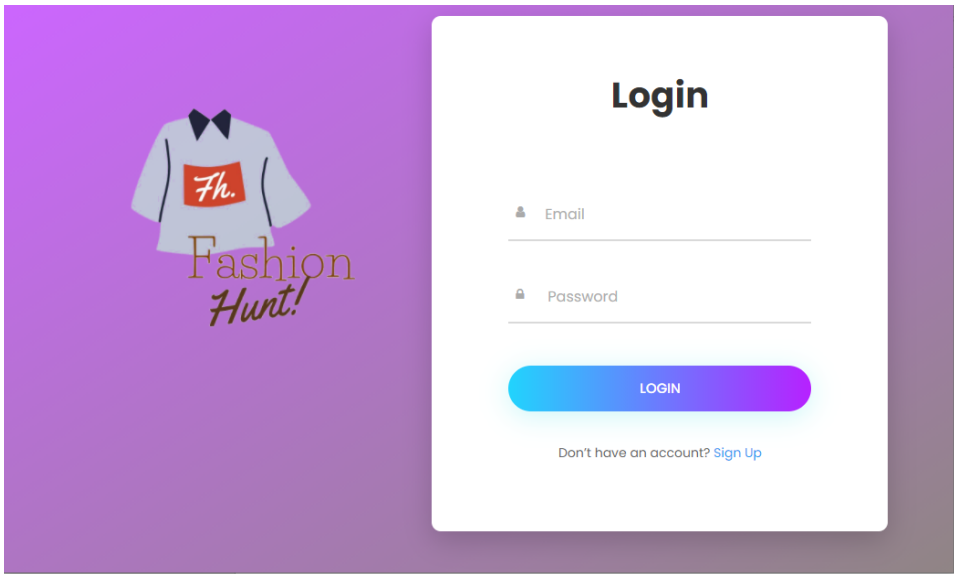


Figure 2. 15: Login Screen to verify user’s account

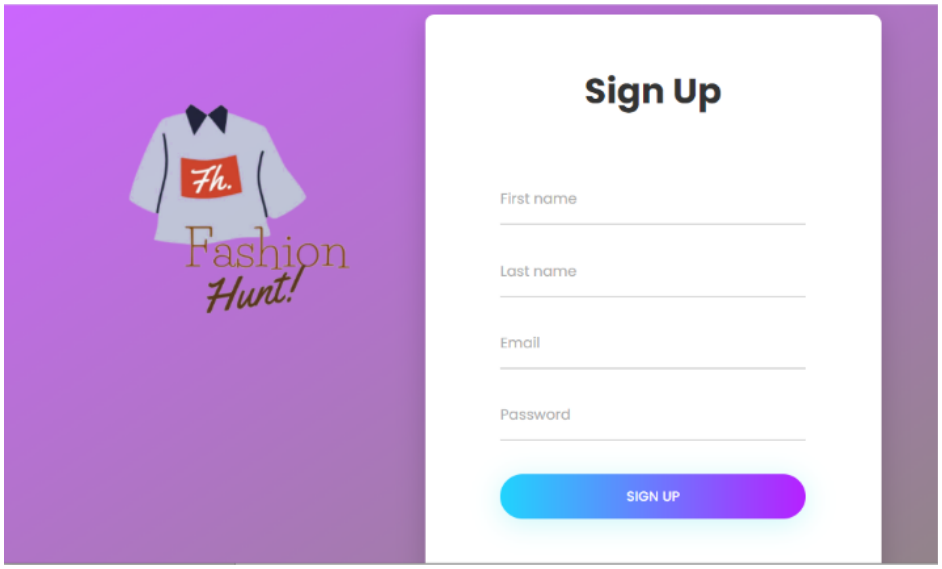


Figure 2. 16 : Sign Up Screen to register user in database

Home Page

The user arrives at the Main Page of the website after the authentication process. Here he/she can view the general details about the website. Fig.2.17, Fig.2.18 and Fig.2.19 shows the Slideshow of the Home Page.

Figure 2. 17 : Slideshow 1 of Fashion Hunt Web App

Figure 2. 18: Slideshow 2 of Fashion Hunt Web App

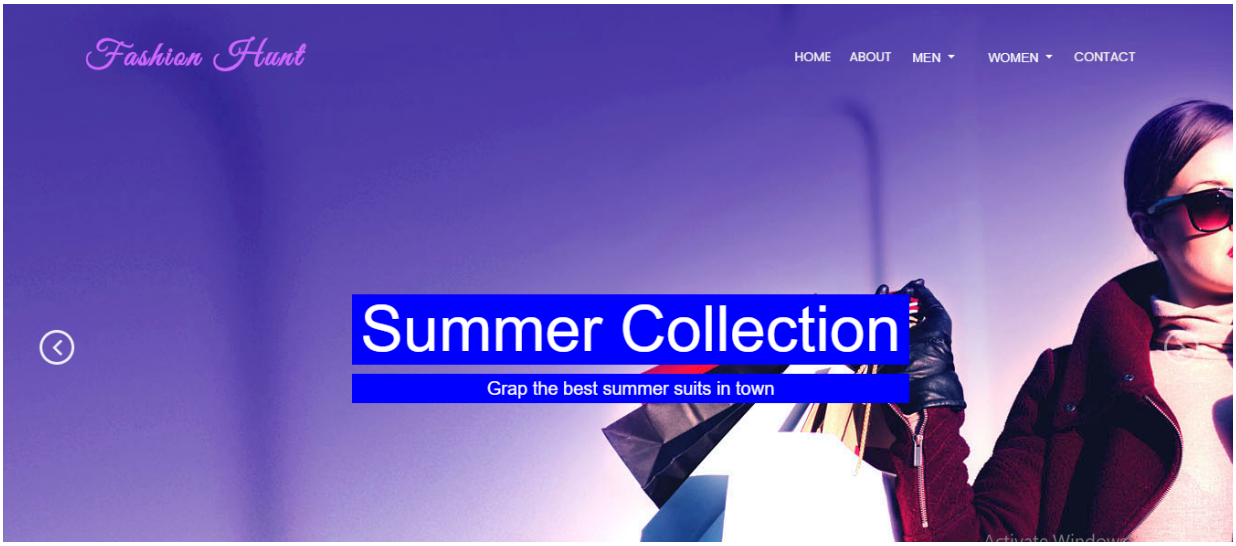


Figure 2. 19 : Slideshow 3 of Fashion Hunt Web App

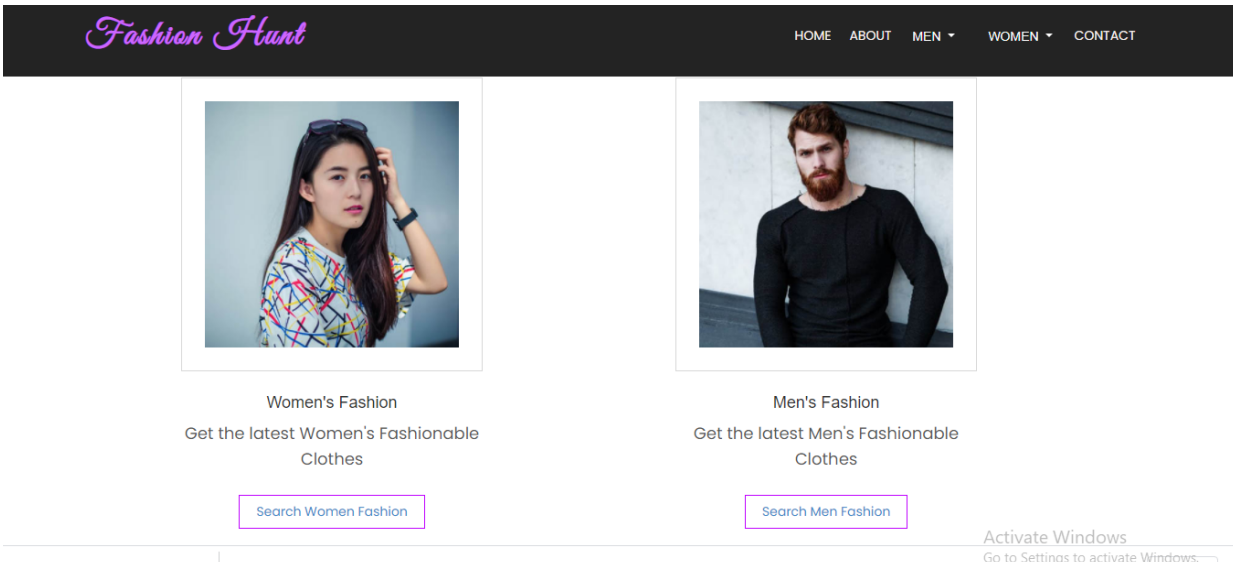


Figure 2. 20 : Optional Selection of Men / Women Collection

Fig.2.20 shows that a user can select ‘Search Women Fashion’ and ‘Search Men Fashion’ in order to search his/her specific desired clothing.

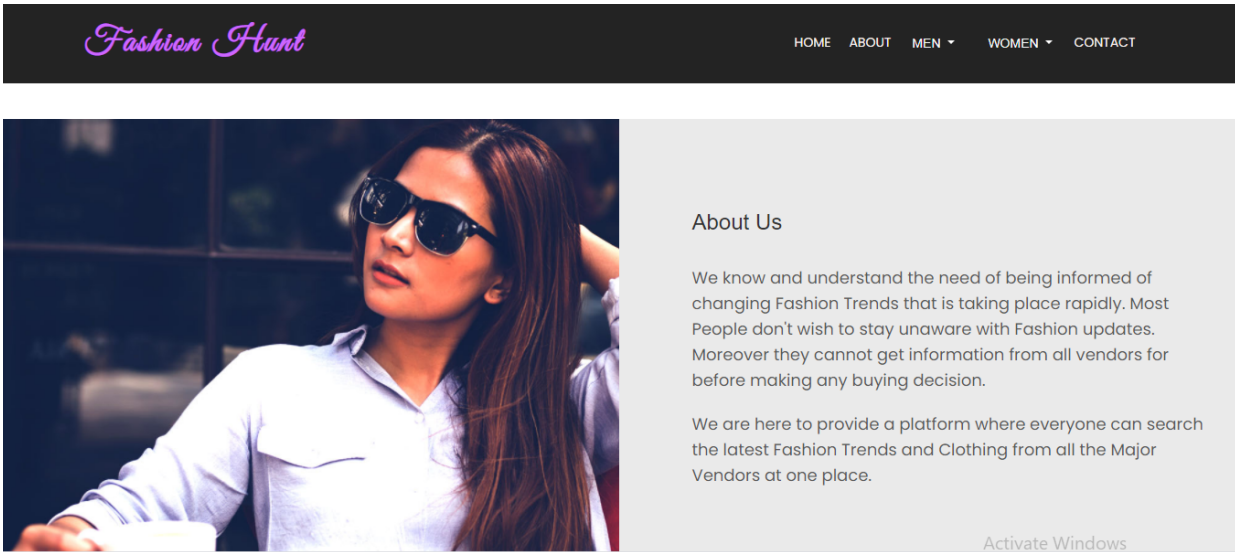


Figure 2. 21 : About Us

Fig.2.21 shows 'About Us' that tells the user the purpose of the website and what the website is offering to do. This information is useful for users to decide whether to proceed exploring the website or not.

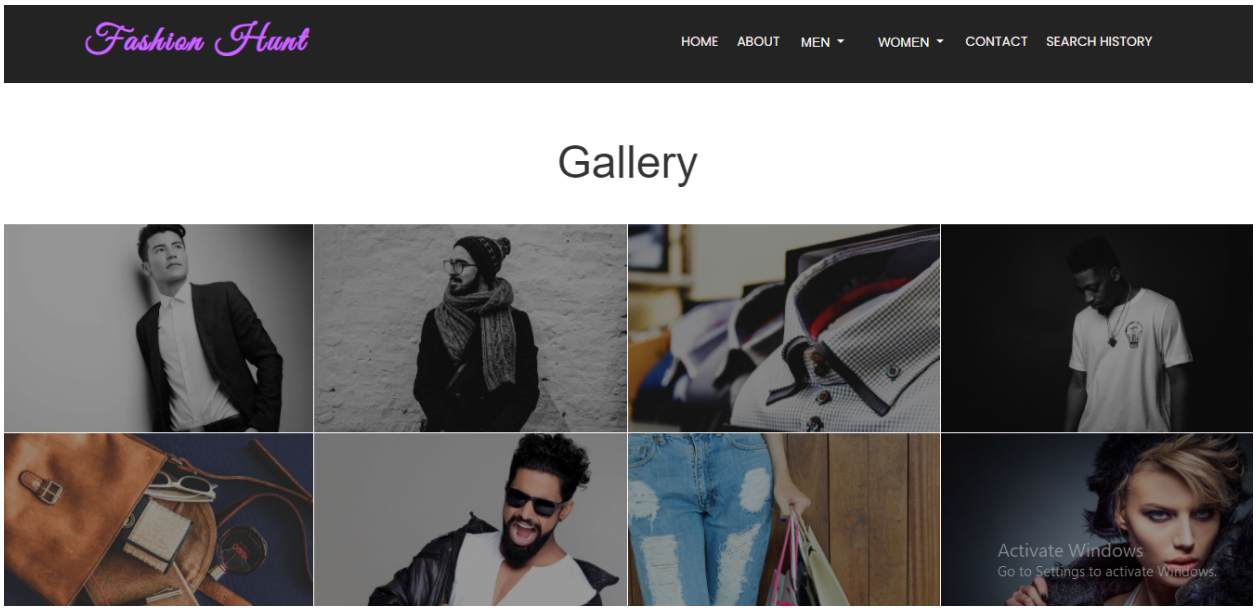


Figure 2. 22 shows the gallery. Gallery is added to attract the customers for using the website.

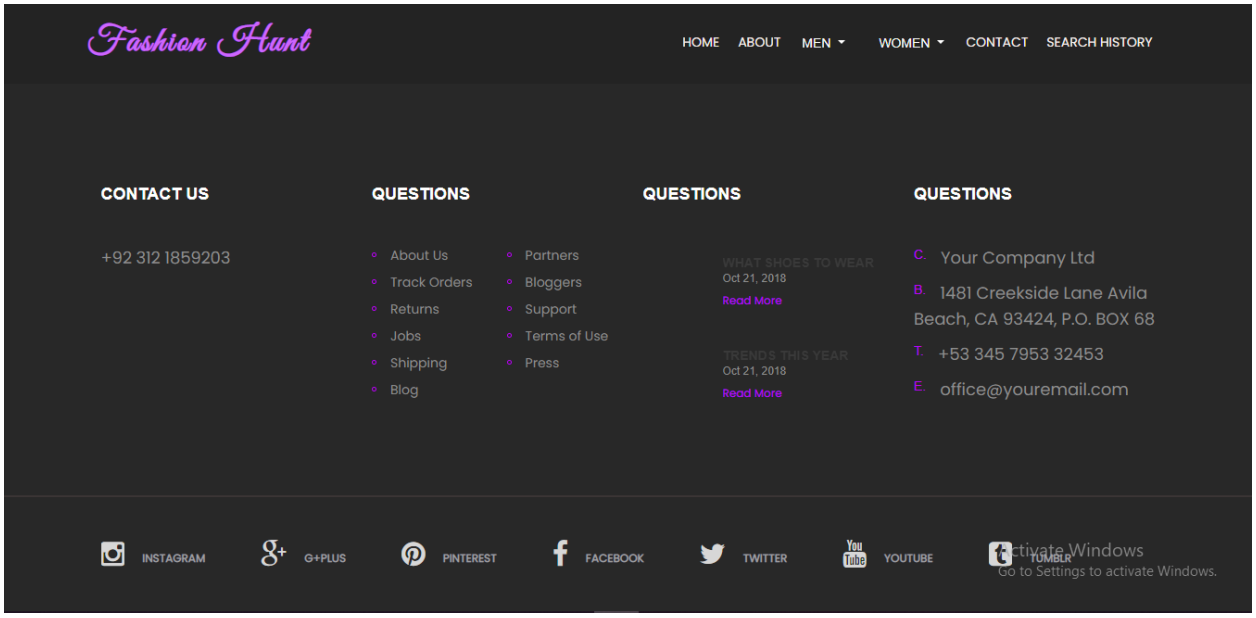


Figure 2. 23: Contacts section

Fig.2.23 shows the contact that can be used in case of any queries.

11.3 Iteration 3

Similar Images retrieval

In this Iteration our task is to get similar clothing images after being segmented, getting similar images is the last interest of the stakeholders as this Fyp is focused on getting similar clothing images that a user can purchase from an e-commerce website. Matrices for similarity are:

1. Color
2. Texture
3. Pattern on clothing item
4. Gender matching
5. Article matching

Not only one but many similar matches are displayed after the model predicts, all results are displayed on the website, each image is having a link, which refers to the e-commerce website where a user can buy that clothing article.

Model Used:

Siamese Model

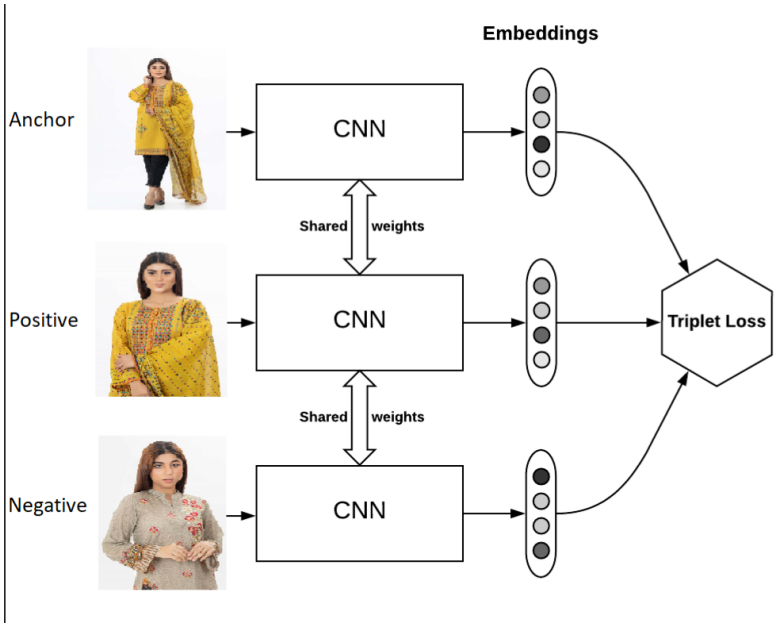


Figure 3.1: Siamese Model

The Siamese Model is very popular for facial recognition but we did this experiment by training it on clothing articles and it performed very well on clothing items as well, we have used github library of torchreid. Torchreid stands for torch re-identification. In our case we are not doing re-identification as we are only displaying the clothing items once. We modified this code and wrote new code that suited best in our scenario.

Model Details:

Siamese model consists of triplet loss function, this model is trained by feeding it triplet of three images per article, first image is positive, second is anchor image and positive and anchor must be of same article but should cover different angle, and third image is negative image which should not be same as article. The model learns the distances between these images and evaluates its loss function. The more similar images will have closer distance and hence low penalty for the loss function.

Library Link: https://github.com/KaiyangZhou/deep-person-reid

Steps to train Siamese model:

The Siamese model is first trained on the data, we have prepared a file called as Siamese_train.py , in our repo of Siamese model, first we need to set path for dataset in this file, We have created a dataset class CvDataset which prepares the triplet, torchreid has a function called as engine, we call engine.run function to train the model on our customized data.

Dataset Preparation:

The dataset for the Siamese model is prepared on the following tree structure.

```
├────siamese_data
│  ├────Train
   │  │  ├────pic_folder1
         │  │  │ ├ positive.jpg
         │  │  │ ├anchor.jpg
   │  │  ├────pic_folder2
         │  │  │ ├ positive.jpg
         │  │  │ ├anchor.jpg
│  ├────Query
   │  │  ├────pic_folder1
         │  │  │ ├ sample1.jpg
   │  │  ├────pic_folder2
         │  │  │ ├ sample2.jpg
```

Here the query is the test folder, which will be utilized for testing and calculating the accuracy.

After training we have saved our model and we can perform predictions over it.

Prediction code:

For similar images retrieval we are running the file, siamese_run.py inside the Siamese github repository, This file was fully customized according to our needs. First, we need to set the query path and gallery path, this is the path of model training images used for comparison. We are extracting the features from the Siamese network and saving them into a tensor file. This is a one time step so we do not have to calculate features on each run. By saving them on file we have optimized our code 10 times. Next we provide a query image. Forward pass is performed on a query and its features are extracted, then these features are compared with trained features from the model that we have already saved in file. We are using faster implementation of Euclidean distance and returning n closest matches after comparing the distances of these features.

Website Implementation

The user uploads an image on the web application. This image is passed to the Flask Server and then to the backend where segmentation of the uploaded image is performed. These segmented images are then sent to the Web application via Flask server.

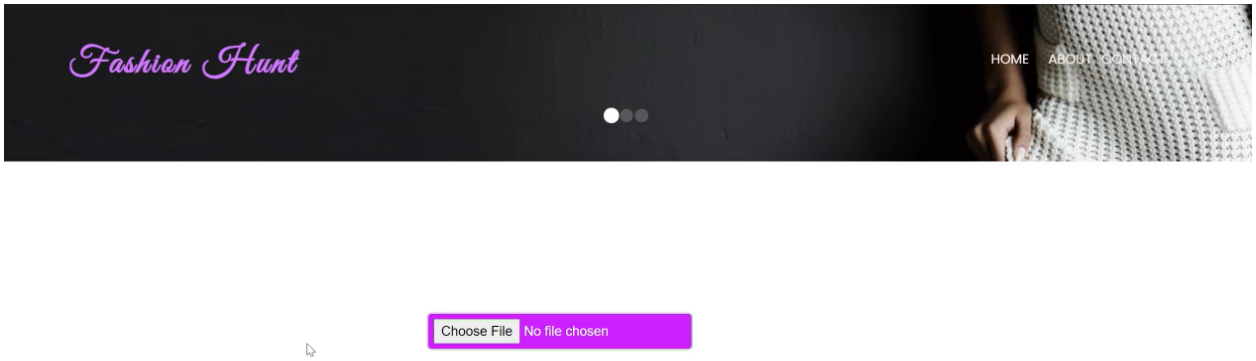


Figure 3.2: Upload Image to the Webpage

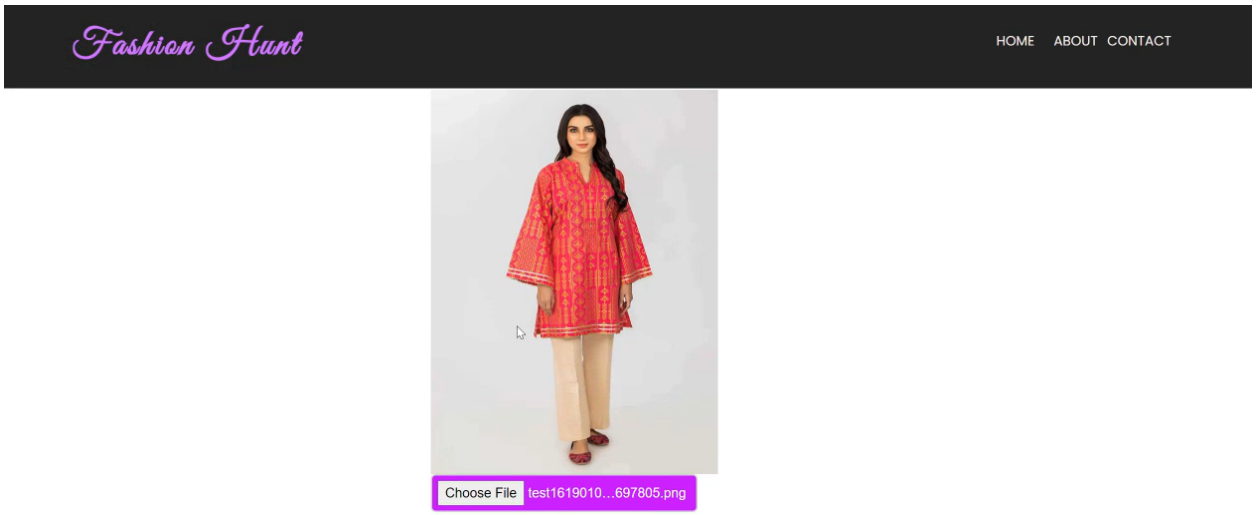


Figure 3.3: Uploaded Image on the Webpage

Figure 3.4: Segmented Images

By clicking on any displayed segmented image, that image is sent again to the backend and the model predicts the matched images and returns them to the Web application which is then displayed with clickable links that prompts the user to the respective clothing shopping site.

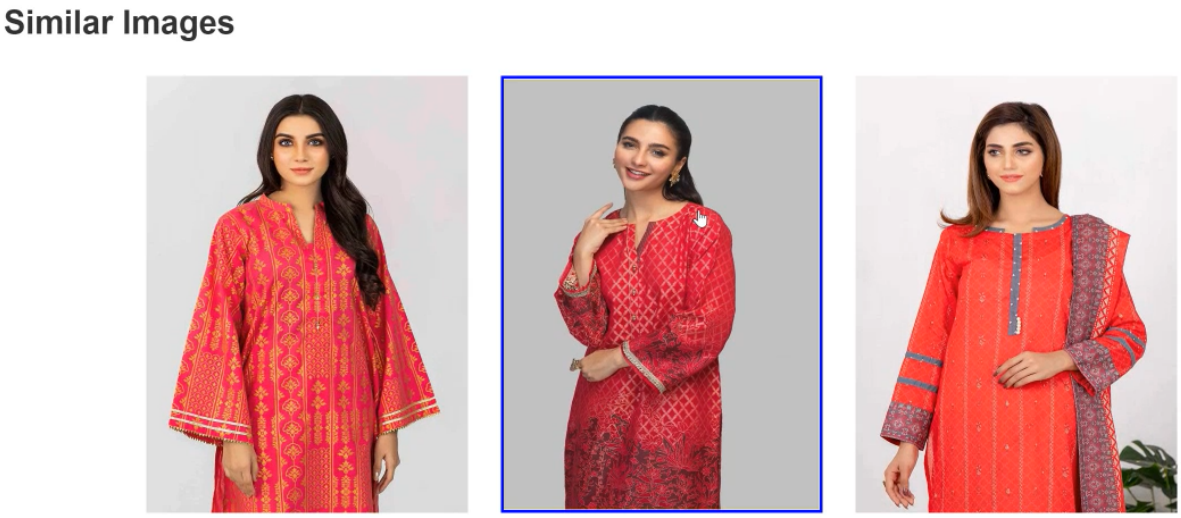


Figure 3.5: Similar Images

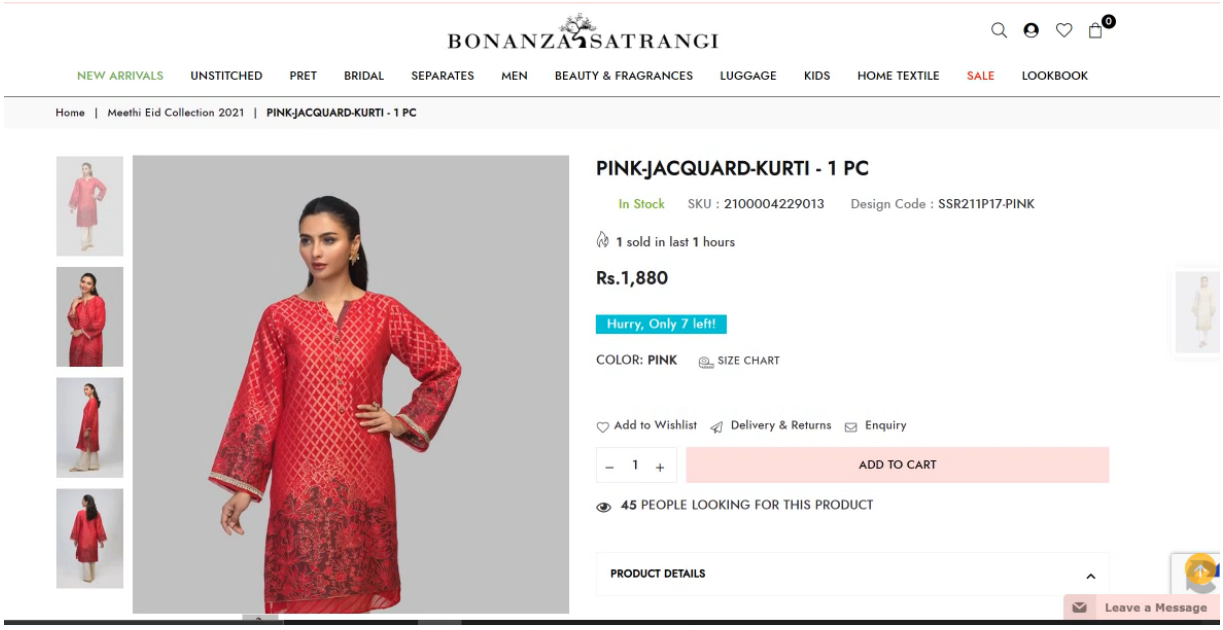


Figure 3.6: Selected Clothing's Official Site

11.4 Iteration 4

Integration of Models with Website

In this iteration we have integrated our two main models with the website, optimized the UI of the website and also improved the performance efficiency and accuracy of the model.

Integration Technique:

We have created a repo called the middleware, the job of middleware is to communicate with both models and work as an interface of the RestApi working on Flask web server. This is a backend server working independently, The flask Api's are called from the Angular frontend and middleware is helping to fetch the data.

Start the flask server, goto middleware folder, run this command:

python flaskie.py

This command will start the server.

API documentation:

```
@app.route("/uploadImage", methods=['POST'])
def image_upload():
'''
    image received is in base64 format and then saved to uploads folder
    Image is segmented after its uploaded

'''
```

```
@app.route("/similarImages", methods=['POST'])
def similar_images():
'''
    query image is recieved as a request parameter and then siamese
    model is called to return n number of similiar images
    returns dictionary of base 64 images back to the website
    '''
```

UI Improvements:

Efficiency Improvements:

In the last Iteration we were using full human sized image for similar image retrieval, now we have optimized our code by making use of segmentation model, we have now trained our model on segmented images, now we are only giving chunks of clothes, this has improved and diverted the focus of model from person faces to clothing items only. The second improvement is that now we do image segmentation only once for one complete clothing pair having pants and a shirt. Before that we were passing each item separately to the uploads API, now we only predict segments once and pass it to a similar image retrieval model that has improved the time twice.

GUI Improvements:

Some significant improvements were made to the UI which is as follows:

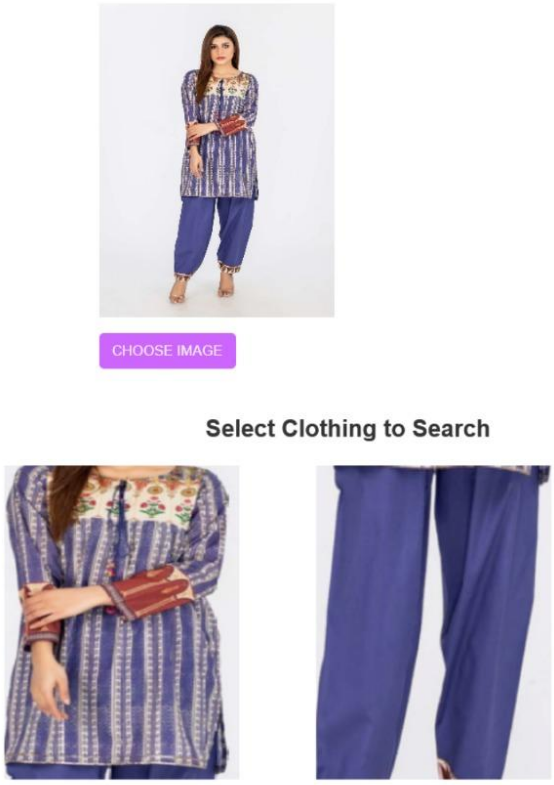


Figure 4.1: Upload and Segmented Images

Figure 4.1 shows the Uploaded image and its segmented images fetched from the backend and displayed on UI.

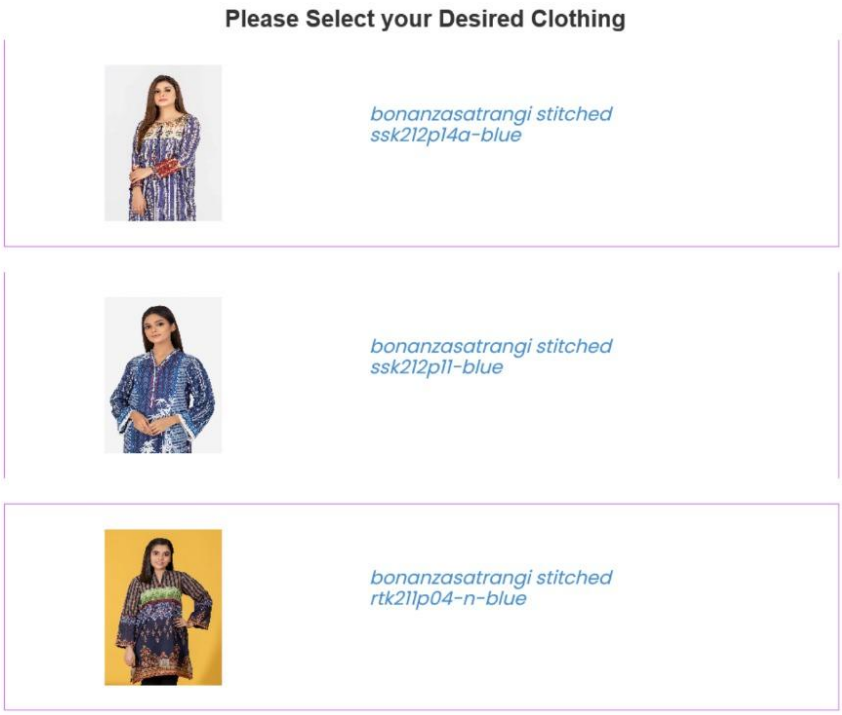


Figure 4.2: Similar Images with Relevant Information

Figure 4.2 shows the Similar Images that are displayed after selecting one of the segmented images in Figure 4.1. These images are shown along with their description and name of the clothing.

Figure 4.3: Selected Image's Clothing Site

Figure 4.3 shows the selected clothing's official site where its information like price, availability, size etc can be seen.

11 Conclusions

Overall our model gives good pixel accuracy but in case of images in which a person is not standing properly, the model can under perform on such images. It misses some details of mask segmentation. This can be improved with increased dataset to train the model even better.

12 References

1. **Paper used in Literature Review:**
   http://cs231n.stanford.edu/reports/2015/pdfs/BLAO_KJAG_CS231N_FinalPaperFashionClassification.pdf

2. **Pytorch Encoding for Person segmentation:**
   https://github.com/zhanghang1989/PyTorch-Encoding/

3. **Dataset for Clothing Segmentation:**
   https://arxiv.org/pdf/1901.07973.pdf

4. **MIoU, FWIoU and PA:**
   https://www.programmersought.com/article/65765418847/

5. **Labelme library used for annotation:**
   https://github.com/wkentaro/labelme

6. **Pytorch deeplab model training code:**
   https://github.com/jfzhang95/pytorch-deeplab-xception.git

7. **Pytorch deeplab model prediction code:**
   https://github.com/giovanniguidi/deeplabV3-PyTorch.git

8. **Paper used for comparison of deep learning models:**
   http://www.diva-portal.se/smash/get/diva2:1466765/FULLTEXT01.pdf

Note: GitHub Repository is kept private. Only the evaluators who have to access the repository will be added as Collaborators (after they share their GitHub username)

13 Glossary

Mean Intersection over Union (MIoU)	This is a standard measurement metric, which calculates the ratio of intersection and union between two sets. In image segmentation, it is the two sets of ground truth and predicted value. It can be converted to the ratio of TP (intersection) to the sum of TP, FN, FP (union). First calculate the intersection ratio within each class, and then calculate the mean.

Weighted Intersection over Union (Frequency Weighted Intersection over Union, FWIoU)	This is a slight improvement on the basis of MioU, and weights are set for each class according to the frequency of occurrence
Pixel accuracy (PA)	This is the simplest indicator used to calculate the ratio between the number of correctly classified pixels and the total number of pixels
MEAN Stack	Using Mongo DB, Express and NodeJS for backend and Angular for frontend development of a webs.