VIRTUAL DRESSING ROOM

(PROJECT PHASE- I)

submitted in partial fulfillment of the requirements

for the award of the degree in

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

DHANUSH G (201191101014)

SAKTHIVEL K (201191101050)

SHUVAM PANDAY J (201191101053)

Dr. M.G.R.
EDUCATIONAL AND RESEARCH INSTITUTE
DEEMED TO BE UNIVERSITY
University with Graded Autonomy Status
(An ISO 21001 : 2018 Certified Institution)
Periyar E.V.R. High Road, Maduravoyal, Chennai-95. Tamilnadu, India.

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

DECEMBER 2023

Dr. M.G.R.
EDUCATIONAL AND RESEARCH INSTITUTE
DEEMED TO BE UNIVERSITY
University with Graded Autonomy Status
(An ISO 21001 : 2018 Certified Institution)
Periyar E.V.R. High Road, Maduravoyal, Chennai-95. Tamilnadu, India.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report (Project Phase-I) is the bonafide work of

Mr. DHANUSH G Reg. No 201191101014

Mr. SAKTHIVEL K Reg. No 201191101050

Mr. SHUVAM PANDAY J Reg. No 201191101053, who carried out the project entitled "VIRTUAL DRESSING ROOM" under our supervision from June 2023 to Dec 2023.

Internal Guide	**Project Coordinator**	**Department Head**
(Name and Date)	(Name and Date)	(Name and Date)

Submitted for Viva Voce Examination held on _____

Internal Examiner	**External Examiner**
(Name in Capital letters with Signature)	(Name in Capital letters with Signature)

DECLARATION FORMAT

We DHANUSH G (201191101014), SAKTHIVEL K (201191101050), SHUVAM PANDAY J (201191101053.) hereby declare that the Project Report (Project Phase-I) entitled "VIRTUAL DRESSING ROOM"

is done by us under the guidance of Mrs RASIDHA BANU B is submitted in partial fulfillment of the requirements for the award of the degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering.

.

1.

2.

DATE:

3.

PLACE: **SIGNATURE OF THE CANDIDATE(S)**

ACKNOWLEDGEMENT

We would first like to thank our beloved Chancellor **Thiru. Dr.A.C. Shanmugam, B.A., B.L.**, President **Er. A.C.S. Arunkumar, B.Tech.,** and Secretary **Thiru A.Ravikumar** for all the encouragement and support extended to us during the tenure of this project and also our years of studies in this wonderful University.

We express my heartfelt thanks to our Vice Chancellor **Dr. S.Geethalakshmi** in providing all the support of my Project.

We express my heartfelt thanks to our Head of the Department, **Prof. Dr. S. Geetha**, who has been actively involved and very influential from the start till the completion of our project.

Our sincere thanks to our Project Coordinators **Dr.P.V ANANTHAN** & **FAWAZ ABDULKADER** and Project guide **RASIDHA BANU.B** for their continuous guidance and encouragement throughout this work, which has made the project a success.

We would also like to thank all the teaching and nonteaching staffs of Computer Science and Engineering department, for their constant support and the encouragement given to us while we went about to achieving my project goals.

INDEX

ABBREVIATIONS

UI - User Interface

CV - Computer Vision

ML - Machine Learning

HTML - HyperText Markup Language (for web-based interfaces)

CG - Computer Graphics

UV Mapping - Texture coordinate mapping

CNN - Convolutional Neural Network

GUI – Graphical User Interface

ABSTRACT

A virtual dressing room could revolutionize the online shopping experience by providing customers with a realistic preview of how different clothing items look on their own bodies. This innovative technology could mitigate concerns about sizing and fit, ultimately boosting consumer confidence in making online apparel purchases. By implementing virtual dressing rooms, online sellers could not only improve customer satisfaction but also reduce the likelihood of returns, streamlining the overall shopping process. This transformative tool has the potential to redefine the landscape of e-commerce and set a new standard for personalized and convenient online shopping.

Keywords: Virtual dressing room, contour algorithm, double exposure, fashion retail, virtual try-on, user experience, clothing fit, realism, e-commerce, online shopping, computer vision, privacy, security.

CHAPTER 1 – INTRODUCTION

The growth of e-commerce has transformed the retail landscape, offering unparalleled convenience and accessibility to consumers worldwide. However, one persistent challenge that continues to impede the seamless adoption of online apparel shopping is the absence of a physical fitting experience. The inability to try on clothing before making a purchase often leads to uncertainty about fit, style, and overall satisfaction.

In response to this challenge, we propose a groundbreaking solution: a Virtual Fitting Room (VFR) empowered by deep learning neural networks. The Virtual Fitting Room aims to bridge the gap between the digital and physical shopping experiences by providing users with a realistic and personalized virtual try-on environment.

This project delves into the intricate integration of computer vision and machine learning techniques to create an immersive and accurate virtual fitting experience. The core of our solution lies in the utilization of deep learning neural networks, trained on a diverse dataset encompassing various body shapes and garment styles. By doing so, our system endeavors to emulate the interaction between clothing and individual body types, enabling users to visualize how different garments would look and fit on their unique physiques

CHAPTER 2-LITERATURE REVIEW

[1] Title: An Enhanced Virtual Fitting Room using Deep Neural Networks

Authors: A.I. Gamage

Description: As the customer's experience in present fit-on rooms is considered as an essential part of the textile industry, these fit-on rooms play a huge role in the textile shops. It is quite an arduous method and generates problems like long queues, having to change clothes individually, privacy problems and wasting time. The proposed convolutional neural network-based Virtual Fit-on Room helps to prevent the above mentioned problems. This product contains a TV screen, two web cameras, and a PC. It captures the customer's body by using two web cameras and displays the customer's dressed body. The combination of CNN in Deep learning and AR processes the body detection and generates the customer's dressed object. The application uses the stereo vision concept to get body measurements. The system detects customer age, gender, face type, and skin tones which are used to recommend cloth styles to customers. Another requirement of this system is customizing styles according to the customer requirements and suggests different styles of clothes.

[2] Title: Image-based Virtual Fitting Room

Authors: Jie Chen

Description:

Virtual fitting room is a challenging task yet useful feature for e-commerce platforms and fashion designers. Existing works can only detect very few types of fashion items. Besides they did poorly in changing the texture and style of the selected fashion items. In this project, we propose a novel approach to address this problem. We firstly used Mask R-CNN to find the regions of different fashion items, and secondly used Neural Style Transfer to change the style of the selected fashion items. The dataset we used is composed of images from PaperDoll dataset and annotations provided by eBay's ModaNet. We trained 8 models and our best model massively outperformed baseline models both quantitatively and qualitatively, with 68.72% mAP, 0.2% ASDR.

[3] Title: Application Fitting Room using Virtual Technology: A Systematic Literature Review

Authors: Edbert Junus

Description:

Due to the pandemic of 2020, the e-commerce industry has grown significantly as people have become more open to online shopping. According to eMarketer, retail e-commerce sales increased by 25.7%. One of the industries with the most growth is Fashion. Although according to many reports, the e-commerce fashion industry is expected to grow, there are problems in this online fashion industry where it's hard for customers to decide on buying the apparel that may fit them. One of the solutions for this problem is through fitting room application using virtual technology. A virtual fitting room can give novelty to customers a good experience where customers can virtually try clothes without actually wearing them. This research aims to review the procedure and algorithm involved in a virtual fitting room application. The research method uses a systematic literature review using the PRISMA framework. Based on our study, we can conclude that Virtual Fitting Room applications involve finding the lean body of the target image, then inserting the input image

[4] Title: Generative Models for Fashion Industry using Deep Neural Networks

Authors: Ildar Lomov

Description:

The progress of deep learning models in image and video processing leads to new artificial intelligence applications in Fashion industry. We consider the application of Generative Adversarial Networks and Neural Style Transfer for Digital Fashion presented as Virtual fashion for trying new clothes. Our model generate humans in clothes with respect to different fashion preferences, color layouts and fashion style. We propose that the virtual fashion industry will be highly impacted by accuracy of generating personalized human model taking into account different aspects of product and human preferences. We compare our model with state-of-art VITON model and show that using new perceptual loss in deep neural network architecture lead to better qualitative results in generating humans in clothes.

[5] Title: Facial expression-enhanced recommendation for virtual fitting rooms

Authors: Ying Xue

Description:

With the development of Augmented Reality (AR) technology in the retail industry, virtual fitting room (VFR) are considered promising enhancement of e-commerce by providing users with an immersive environment to try on new products, especially fashion products. While allowing users having more vivid impression of products, virtual fitting rooms also offer sellers more channels to collect information on user preferences, which can be used to enhance recommender systems. This study proposes to leverage facial expression recognition technology together with fine-grained human-computer interactions in virtual fitting rooms to personalize product recommendations. This paper proposes a recommendation algorithm based on confidence setting, negative feedback sampling, and matrix factorization to model user behaviors in virtual fitting rooms

CHAPTER 3- AIM AND SCOPE OF PRESENT INVESTIGATION

Our proposed Virtual Fitting Room (VFR) system leverages the power of Convolutional Neural Networks (CNNs), Haarcascade and OpenCV to create a sophisticated and accurate virtual try-on experience. Convolutional Neural Networks are employed to handle the complex task of extracting features from images, while OpenCV provides robust computer vision functionalities to enhance various aspects of the virtual fitting process.

The core of our system lies in the utilization of CNNs for both pose estimation and semantic segmentation. For accurate body positioning, the CNN-based pose estimation module identifies key landmarks on the user's body, ensuring precise alignment of virtual garments. Simultaneously, the semantic segmentation module, powered by CNNs, distinguishes between the user's body and the clothing item, allowing for a detailed separation that is essential for a realistic virtual try-on.

OpenCV plays a pivotal role in augmenting the capabilities of our proposed system. Image processing techniques provided by OpenCV are employed for tasks such as color correction, edge enhancement, and contour smoothing, contributing to a visually appealing and realistic representation of virtual garments on the user. Additionally, OpenCV's robust features aid in real-time tracking and adjustment of virtual garments as the user moves, enhancing the dynamic nature of the virtual fitting experience.

CHAPTER 4-EXPERIMENTAL OR MATERIAL AND METHOD

4.1. REQUIREMENT ANALYSIS

Requirements are a feature of a system or description of something that the system is capable of doing in order to fulfil the system's purpose. It provides the appropriate mechanism for understanding what the customer wants, analysing the needs assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification and managing the requirements as they are translated into an operational system.

4.1.1. PYTHON:

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, we don't need to declare the type of variable because it is a dynamically typed language.

For example, x=10 .Here, x can be anything such as String, int, etc.

Python is an interpreted, object-oriented programming language similar to PERL, that has gained popularity because of its clear syntaxand readability. Python is said to be relatively easy to learn and portable, meaning its statements can be interpreted in a number of operating systems, including UNIX-based systems, Mac OS, MS-DOS, OS/2, and various versions of Microsoft Windows 98. Python was created by Guido van Rossum, a former resident of the Netherlands, whose favourite comedy group at the time was Monty Python's Flying Circus. The source code is freely available and open for modification and reuse. Python has a significant number of users.

Features in Python

There are many features in Python, some of which are discussed below

- Easy to code
- Free and Open Source
- Object-Oriented Language
- GUI Programming Support
- High-Level Language
- Extensible feature
- Python is Portable language
- Python is Integrated language
- Interpreted Language

4.2. ANACONDA

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the anaconda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI).

The big difference between anaconda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason anaconda exists.

When pip installs a package, it automatically installs any dependent Python packages without checking if these conflict with previously installed packages. It will install a package and any of its dependencies regardless of the state of the existing installation. Because of this, a user with a working installation of, for example, Google Tensorflow, can find that it stops working having used pip to install a different package that requires a different version of the dependent numpy library than the one used by Tensorflow. In some cases, the package may appear to work but produce different results in detail.

In contrast, anaconda analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g. the user may wish to have Tensorflow version 2,0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

Opensource packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the anaconda install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. Anything available on PyPI may be installed into a anaconda environment using pip, and anaconda will keep track of what it has installed itself and what pip has installed.

Custom packages can be made using the anaconda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of Python packaged with anaconda.

4.3. Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) provided as part of the Anaconda distribution, a popular platform for data science and scientific computing tasks. Its primary purpose is to offer users a convenient means of managing Anaconda packages, environments, and channels without requiring extensive familiarity with command-line interfaces. Navigator provides a user-friendly environment where users can efficiently launch applications, explore available packages, install them into specific environments, and execute the installed packages. It supports searching for packages both on Anaconda Cloud, a repository of over 7,500 data science packages, and within local Anaconda Repositories.

One of the key features of Anaconda Navigator is its ability to manage environments. Environments in Anaconda allow users to create isolated Python environments with specific sets of packages installed. This enables users to work on different projects with different package dependencies without encountering conflicts. Navigator facilitates the creation, deletion, and management of these environments, providing a straightforward interface for users to control their development and analysis environments.

Anaconda Navigator comes with a set of default applications that cater to various aspects of data science and development. These applications are pre-configured within the Navigator interface for easy access. Some of the default applications included in Navigator are:

- JupyterLab: A web-based interactive development environment for Jupyter notebooks, offering support for code editing, data visualization, and collaborative work.
- Jupyter Notebook: The classic interface for creating and sharing documents containing live code, equations, visualizations, and narrative text.
- QtConsole: A console application that provides an enhanced interactive Python shell with features like syntax highlighting and tab completion.
- Spyder: An integrated development environment (IDE) specifically designed for scientific computing, data analysis, and machine learning.
- Glue: A tool for linking and visualizing data across multiple files and formats, commonly used in scientific research and data analysis workflows.
- Orange: A data visualization and analysis tool that offers a visual programming interface for building and executing data analysis workflows.
- RStudio: An integrated development environment for the R programming language, widely used for statistical computing and data analysis.
- Visual Studio Code: A powerful and extensible code editor developed by Microsoft, featuring support for various programming languages and extensive customization options.

These default applications cover a broad range of functionalities required for data exploration, analysis, visualization, and development in fields such as data science, machine learning, statistics, and scientific computing. Additionally, users can extend Navigator's capabilities by installing additional packages and integrating them into their workflows, thereby enhancing their productivity and efficiency in data-related tasks.4.3.1. JUPYTER NOTEBOOK

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

Jupyter Notebook can connect to many kernels to allow programming in different languages. By default, Jupyter Notebook ships with the IPython kernel. As of the 2.3 release[11][12]

(October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.

The Notebook interface was added to IPython in the 0.12 release[14] (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s. According to The Atlantic, Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

4.4. ARTIFICIAL INTELLINGENCE:

Artificial intelligence (AI) represents the capacity of computer programs or machines to simulate human-like thinking and learning processes. It has evolved into a vast interdisciplinary field that intersects computer science, cognitive psychology, philosophy, neuroscience, and engineering. At its core, AI aims to equip machines with capabilities that resemble human intelligence, enabling them to perceive, reason, learn, and adapt to various tasks and environments.

Traditionally, the concept of AI was often associated with replicating human cognitive abilities such as reasoning, problem-solving, and decision-making. However, as technology advanced, the definition of AI expanded to encompass a broader range of functionalities, including those that don't necessarily mirror human cognition but still demonstrate intelligent behavior.

One key aspect of AI is its emphasis on creating systems that can process and analyze vast amounts of data to derive meaningful insights and make informed decisions. This involves employing techniques such as machine learning, natural language processing, computer vision, and robotics. Through these methods, AI systems can recognize patterns, classify information, and extract valuable knowledge from diverse datasets.

AI applications span a wide spectrum of domains and industries, with notable examples including:

- Face Recognition: AI-powered systems can analyze images or video streams to identify and recognize human faces, enabling applications such as biometric authentication, surveillance, and social media tagging.
- Learning: AI algorithms can ingest large datasets and discover underlying patterns or relationships, facilitating tasks such as predictive modeling, recommendation systems, and personalized content delivery.
- Planning: AI systems can generate optimal sequences of actions to achieve specific goals or objectives, enabling applications in logistics, resource allocation, and automated scheduling.
- Decision Making: AI models can analyze complex scenarios, weigh various factors, and make decisions based on predefined criteria, aiding fields such as finance, healthcare, and autonomous vehicles.

Furthermore, AI is not limited to executing predefined tasks but also encompasses the ability to adapt and learn from experience. This aspect, known as machine learning, enables AI systems to improve their performance over time through exposure to new data and feedback mechanisms.

In essence, artificial intelligence leverages computer science principles and methodologies to simulate intelligent behavior in machines. By harnessing the power of data analysis, problem-solving algorithms, and adaptive learning mechanisms, AI holds the promise of revolutionizing industries, transforming societal processes, and augmenting human capabilities in unprecedented ways. As research and development in AI continue to progress, the potential for innovative applications and transformative impacts on society remains boundless.

4.5. DEEP LEARNING

Deep learning, a subset of machine learning, is a powerful approach to artificial intelligence (AI) that has gained significant traction in recent years. It revolves around the concept of representation learning within artificial neural networks, which are computational models inspired by the structure and function of the human brain. What distinguishes deep learning from traditional machine learning methods is the utilization of deep neural networks, characterized by multiple layers of interconnected nodes, or neurons. The term "deep" signifies the depth of these networks, indicating the presence of numerous layers through which data passes during processing.

Deep learning techniques can be categorized into various paradigms, including unsupervised learning, semi-supervised learning, and supervised learning, each suited to different types of tasks and data availability. Unsupervised learning aims to extract patterns and structure from unlabeled data, semi-supervised learning combines labeled and unlabeled data to improve learning performance, while supervised learning relies on labeled data to train models to make predictions or classifications.

In diverse fields such as computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, climate science, material inspection, and board game programming, deep learning architectures have demonstrated remarkable capabilities. Some of the notable deep learning architectures include:

1. Deep Neural Networks (DNNs): Multilayer perceptron models consisting of interconnected layers of neurons, capable of learning complex hierarchical representations of data.

2. Deep Belief Networks (DBNs): Probabilistic graphical models composed of multiple layers of stochastic, latent variables, and used primarily for unsupervised learning tasks.

3. Deep Reinforcement Learning: A branch of machine learning where agents learn to interact with environments through trial and error, guided by rewards, and employing deep neural networks to approximate value functions or policies.

4. Recurrent Neural Networks (RNNs): Neural network architectures designed to process sequential data by maintaining internal memory states, making them well-suited for tasks involving temporal dependencies, such as speech recognition and language modeling.

5. Convolutional Neural Networks (CNNs): Specialized neural network architectures designed for processing structured grid data, such as images, by leveraging shared weights and hierarchical feature extraction.

6. Transformers: A class of neural network architectures based on self-attention mechanisms, widely used for natural language processing tasks, including language translation and text generation.

While artificial neural networks draw inspiration from biological systems, they differ significantly from the dynamic and analog nature of biological brains. Most artificial neural networks exhibit static and symbolic behavior, with fixed weights and discrete activations, unlike the plasticity and analog processing observed in biological neurons.

In summary, deep learning algorithms leverage deep neural networks to learn intricate representations of data, enabling them to achieve state-of-the-art performance across various domains. Their remarkable success has propelled advancements in AI and continues to drive innovations in fields ranging from healthcare and finance to autonomous vehicles and robotics.

4.5.1. Features of Deep Learning:

Deep learning systems represent a groundbreaking advancement in artificial intelligence, offering capabilities that revolutionize the way data is analyzed and patterns are extracted. Here's an expanded discussion on the key points you've mentioned:

4.5.2. Automatic Feature Extraction:

One of the distinguishing features of deep learning systems is their ability to automatically extract relevant features from raw data. Traditional machine learning methods often require manual feature engineering, where domain experts identify and encode relevant features for the model. However, deep learning systems can autonomously learn hierarchical representations of features directly from the data. This capability significantly reduces the need for human intervention in the feature engineering process, leading to more efficient and scalable solutions.

4.5.3. Processing Structured and Unstructured Data:

Deep learning systems excel in processing both structured and unstructured data types. Structured data refers to data organized in a tabular format with predefined columns and rows, such as databases or spreadsheets. Unstructured data, on the other hand, lacks a predefined data model and includes formats like text, images, and audio. Deep learning algorithms, including convolutional neural networks (CNNs) for images, recurrent neural networks (RNNs) for sequential data, and transformers for natural language processing, designed to handle various data modalities effectively. This versatility enables deep learning systems to tackle a wide range of tasks across different domains.

4.5.4. Accuracy:

Deep learning systems are renowned for their high accuracy in modeling complex relationships within data. By leveraging deep neural networks with multiple layers, these systems can capture intricate patterns and dependencies that may be difficult to discern using traditional methods. The hierarchical representation learning facilitated by deep architectures enables the model to extract increasingly abstract features at each layer, leading to enhanced predictive performance. As a result, deep learning algorithms frequently achieve state-of-the-art accuracy in tasks such as image classification, speech recognition, and natural language understanding.

4.5.5. Analyzing Large Amounts of Data:

Another strength of deep learning systems is their capacity to analyze vast quantities of data efficiently. With the exponential growth of data in today's digital age, traditional analytical techniques may struggle to extract meaningful insights from large datasets. Deep learning algorithms, however, are well-suited to handle big data challenges. They can scale effectively across distributed computing environments and leverage parallel processing capabilities of modern hardware architectures, enabling them to process massive datasets with speed and accuracy. Moreover, deep learning models have demonstrated the ability to uncover complex patterns and correlations in data, even when these insights were not explicitly included in the training data. This capability makes deep learning systems invaluable for tasks such as image recognition, text analysis, and voice processing, where large-scale data analysis is essential for deriving actionable insights.

In summary, deep learning systems represent a paradigm shift in artificial intelligence, offering unparalleled capabilities for automatic feature extraction, processing diverse data types, achieving high accuracy, and analyzing large datasets. These characteristics make them indispensable tools for a wide range of applications across industries, driving innovations and unlocking new possibilities in data-driven decision-making and problem-solving

4.5.6. Classification of Deep Learning

At a broad level, Deep learning can be classified into three types:

1. Supervised learning
2. Unsupervised learning
3. Partially Supervised (semi-supervised)

1) Supervised Learning

Supervised learning is a type of machine learning where the algorithm is trained on a labeled dataset, which means that the input data is paired with the corresponding correct output. In other words, the algorithm is provided with input-output pairs, and the goal is to learn a mapping function from the input to the output.

In the context of deep learning, which is a subfield of machine learning, supervised learning involves using neural networks to learn complex mappings from inputs to outputs. These neural networks are composed of layers of interconnected nodes (neurons) that process the input data and produce an output. During the training process, the network adjusts its internal

parameters (weights and biases) based on the difference between its predictions and the true outputs in the labeled training data.

The training process typically involves an optimization algorithm (e.g., gradient descent) that minimizes a loss function, which measures the difference between the predicted outputs and the true outputs. The goal is to find the optimal set of parameters that minimizes this loss, allowing the model to generalize well to new, unseen data.

Supervised learning in deep learning is widely used in various applications, such as image recognition, natural language processing, speech recognition, and many others. It is called "supervised" because the process involves a "teacher" (the labeled data) guiding the learning algorithm to make accurate predictions.

2) Unsupervised Learning

Unsupervised learning is a type of machine learning where the algorithm is given input data without explicit instructions on what to do with it. Unlike supervised learning, there are no labeled outputs provided during training. The goal of unsupervised learning is to find patterns, relationships, or structures in the data without explicit guidance. In the context of deep learning, unsupervised learning encompasses various approaches, and common types are:

- Clustering
- Dimensionality Reduction
- Generative Models

4.6. GUI (Graphical User Interface)

A Graphical User Interface (GUI) in Python refers to a visual way of interacting with a computer program. Instead of relying on text-based commands, GUIs utilize graphical elements such as windows, buttons, menus, and other visual components to enable user interaction. Python provides several libraries for creating GUI applications, with Tkinter being the default and widely used option. GUIs enhance user experience by offering an intuitive and visually appealing environment for interacting with software. Developers can design interfaces that make complex functionalities accessible to users through mouse clicks, keyboard input, or touch interactions, catering to a broad range of applications from desktop software to mobile apps. The design and implementation of GUIs in Python involve the use of specific libraries that simplify the creation and management of graphical elements, allowing developers to build interactive and user-friendly applications.

4.7. CNN

Convolutional Neural Networks (CNNs) represent a pivotal advancement in computer vision, revolutionizing various tasks such as image recognition, object detection, and classification.

Below is a detailed expansion on the components and workings of CNNs, followed by a comprehensive elaboration on the Haar Cascade algorithm:

1. Convolutional Layers:

CNNs leverage convolutional layers to autonomously learn hierarchical representations of features from input images. These layers employ filters, also known as kernels, which convolve over the input image, extracting local patterns and features. Through feature mapping, CNNs can progressively capture increasingly abstract features as data traverses through successive layers.

2. Pooling Layers:

Pooling layers, often following convolutional layers, serve to downsample the spatial dimensions of the data. Popular pooling techniques include max pooling, which retains the maximum value within a specified region, thereby reducing computational complexity and focusing the network's attention on the most significant features.

3. Activation Functions:

Non-linear activation functions, such as Rectified Linear Units (ReLU), are applied to the output of convolutional layers. These functions introduce non-linearity into the network, allowing it to learn complex patterns and relationships within the data, thereby enhancing its representational capacity.

4. Fully Connected Layers:

Towards the end of the CNN architecture, fully connected layers are typically employed for classification tasks. These layers establish connections between every neuron in the preceding layer and every neuron in the subsequent layer, enabling the network to learn intricate relationships between high-level features and make accurate predictions.

5. Training and Backpropagation:

CNNs undergo training via a process known as backpropagation, where the network adjusts its parameters (weights and biases) based on the disparity between predicted and actual outputs. Optimization algorithms, such as stochastic gradient descent, are commonly employed to iteratively minimize this error and enhance the network's predictive performance.

6. Pre-trained Models:

Due to the computational intensity of training deep neural networks, pre-trained models on extensive datasets like ImageNet are often utilized as starting points. These pre-trained models can be further fine-tuned on smaller, task-specific datasets, thereby expediting the training process and enhancing the model's performance on specific tasks.

4.8 Haar Cascade Algorithm:

The Haar Cascade algorithm, pioneered by Viola and Jones in 2001, stands as a prominent machine learning technique renowned for its real-time face detection capabilities. Below are detailed insights into its key components:

Haar-like Features:

The algorithm employs a collection of simple rectangular filters known as Haar-like features to capture information about intensity differences across various regions of an image. These features serve as foundational elements for detecting objects of interest within the image.

Integral Image:

To expedite the computation of Haar-like features, the integral image technique is harnessed. This technique enables rapid calculation of the sum of pixel values within any rectangular region of the image, facilitating efficient feature extraction and subsequent analysis.

Training the Classifier:

At the heart of the Haar Cascade algorithm lies the utilization of AdaBoost (Adaptive Boosting), a machine learning technique that amalgamates weak classifiers based on Haar-like features into a robust classifier. AdaBoost iteratively emphasizes misclassified examples, enabling the classifier to focus on challenging-to-detect objects and thereby improving overall detection accuracy.

In summary, CNNs and the Haar Cascade algorithm serve as prime exemplars of the versatility and efficacy of machine learning techniques in tackling complex computer vision tasks. These methodologies, through continuous refinement and innovation, continue to propel advancements in AI applications across a broad spectrum of domains, revolutionizing how we perceive, interact with, and interpret visual data.

4.9. Cascade of classifiers:

The trained classifier is organized into a cascade of stages, where each stage consists of several weak classifiers. The cascade structure allows for quick rejection of non-object regions, improving efficiency by not evaluating the entire image for every stage.

Sliding window approach:

The Haar Cascade algorithm applies a sliding window technique over the image at different scales to detect objects of various sizes.

Thresholding:

The final detection decision is made based on a threshold, and if an image region passes all stages of the cascade, it is considered a positive detection

CHAPTER 5 –RESULTS AND DISCUSSION

5.1.PROPOSED SYSTEM

Our proposed Virtual Fitting Room (VFR) system represents a cutting-edge fusion of Convolutional Neural Networks (CNNs), Haarcascade, and OpenCV, aimed at delivering a sophisticated and accurate virtual try-on experience. By leveraging these powerful technologies, we aim to revolutionize the way consumers interact with clothing online, providing a seamless and immersive shopping experience from the comfort of their homes.

At the heart of our VFR system lies the utilization of Convolutional Neural Networks (CNNs), renowned for their prowess in handling complex image processing tasks. In our implementation, CNNs serve a dual purpose: pose estimation and semantic segmentation. The CNN-based pose estimation module is tasked with identifying key landmarks on the user's body, facilitating precise alignment of virtual garments. By accurately determining the user's body positioning, this module ensures that virtual clothing items appear seamlessly integrated and realistically draped over the user's physique.

Simultaneously, the semantic segmentation module, also powered by CNNs, plays a crucial role in distinguishing between the user's body and the clothing item being tried on. This segmentation process enables a detailed separation between the user and the garment, ensuring that the virtual try-on experience is as authentic and true-to-life as possible. By precisely delineating the boundaries between the user and the clothing item, our system can simulate the appearance of garments with remarkable fidelity, capturing intricate details and nuances that contribute to a convincing virtual representation.

In addition to CNNs, our VFR system leverages the robust computer vision functionalities provided by OpenCV to augment its capabilities further. OpenCV enables a plethora of image processing techniques, which are instrumental in enhancing various aspects of the virtual fitting process. Tasks such as color correction, edge enhancement, and contour smoothing are seamlessly integrated into our system, resulting in a visually appealing and realistic representation of virtual garments on the user. By fine-tuning the appearance of virtual clothing items, we ensure that users can accurately assess how different garments will look on them, fostering confidence and satisfaction in their purchasing decisions.

Moreover, OpenCV's robust features play a pivotal role in real-time tracking and adjustment of virtual garments as the user moves. By continuously monitoring the user's movements and adjusting the position and orientation of virtual garments accordingly, our system delivers a dynamic and interactive virtual fitting experience. Users can freely explore different styles and fits, confident that the virtual garments will adapt seamlessly to their body movements and gestures.

In conclusion, our proposed Virtual Fitting Room (VFR) system represents a convergence of state-of-the-art technologies aimed at redefining the online shopping experience. By harnessing the power of Convolutional Neural Networks, Haarcascade, and OpenCV, we aim

to provide users with a compelling and immersive virtual try-on experience, empowering them to make informed and confident purchasing decisions from the comfort of their homes..

Virtual Fitting Rooms (VFRs) represent a transformative innovation in the realm of online shopping, offering a multitude of advantages for both consumers and retailers alike. By harnessing the power of technologies such as Convolutional Neural Networks (CNNs), Haarcascade, and OpenCV, VFRs provide a sophisticated and immersive virtual try-on experience that revolutionizes the way people shop for clothing online.

Advantages of Virtual Fitting Rooms:

1. Convenience and Time Savings:

Virtual try-ons eliminate the need for consumers to travel to physical stores, saving them valuable time and effort. With VFRs, users can conveniently explore a vast array of clothing options from the comfort of their own homes, without the hassle of commuting or navigating crowded malls. This convenience not only enhances the overall shopping experience but also leads to significant cost savings related to transportation and other associated expenses.

2. Improved Confidence in Purchasing Decisions:

One of the primary concerns when shopping for clothing online is the uncertainty surrounding correct sizing and fit. VFRs address this issue by allowing users to virtually try on garments before making a purchase. By seeing how clothing items look and fit on their own bodies in a virtual environment, users can make more informed decisions and mitigate the risk of dissatisfaction with their purchases. This increased confidence leads to higher conversion rates and reduced rates of returns, benefiting both consumers and retailers.

3. Personalization and Immersive Experience:

VFRs offer a personalized and immersive shopping experience that caters to individual preferences and style preferences. Users can experiment with different styles, colors, and combinations to create their ideal outfits, all within the virtual environment. This level of customization enhances user engagement and satisfaction, fostering a deeper connection with the brand and increasing the likelihood of repeat purchases.

4. Accessibility and Inclusivity:

Virtual try-ons democratize the shopping experience by making it more accessible to individuals with disabilities or mobility limitations. By removing physical barriers associated with traditional brick-and-mortar stores, VFRs ensure that everyone has equal access to a wide range of clothing options and can participate in the joy of shopping. Additionally, VFRs can cater to diverse body types and sizes, promoting inclusivity and representation within the fashion industry.

5. Cost-Effectiveness for Retailers:

Implementing VFR technology can yield significant cost savings for retailers by reducing the need for physical store infrastructure and minimizing inventory management expenses. By offering virtual try-ons, retailers can streamline their operations, optimize their inventory

levels, and reduce overhead costs associated with maintaining brick-and-mortar stores. Additionally, VFRs enable retailers to gather valuable data and insights into consumer preferences and purchasing behavior, allowing them to tailor their offerings and marketing strategies more effectively.

6. Sustainability:

Virtual try-ons contribute to sustainability efforts by reducing the environmental impact associated with traditional retail practices. By minimizing the need for consumers to travel to physical stores, VFRs help reduce carbon emissions and energy consumption associated with transportation. Additionally, VFRs can help reduce waste by facilitating more informed purchasing decisions, thereby reducing the likelihood of returns and exchanges.

Virtual try-ons play a crucial role in sustainability efforts by reducing the environmental impact associated with traditional retail practices. By minimizing the need for consumers to travel to physical stores, VFRs help mitigate carbon emissions and decrease energy consumption associated with transportation. Additionally, VFRs contribute to waste reduction by facilitating more informed purchasing decisions, thereby reducing the likelihood of returns and exchanges.

environment, VFRs empower users to experiment with different styles, colors, and combinations, enabling them to find the perfect outfit from the comfort of their homes. This level of customization fosters engagement and satisfaction, leading to increased brand loyalty and repeat purchases.

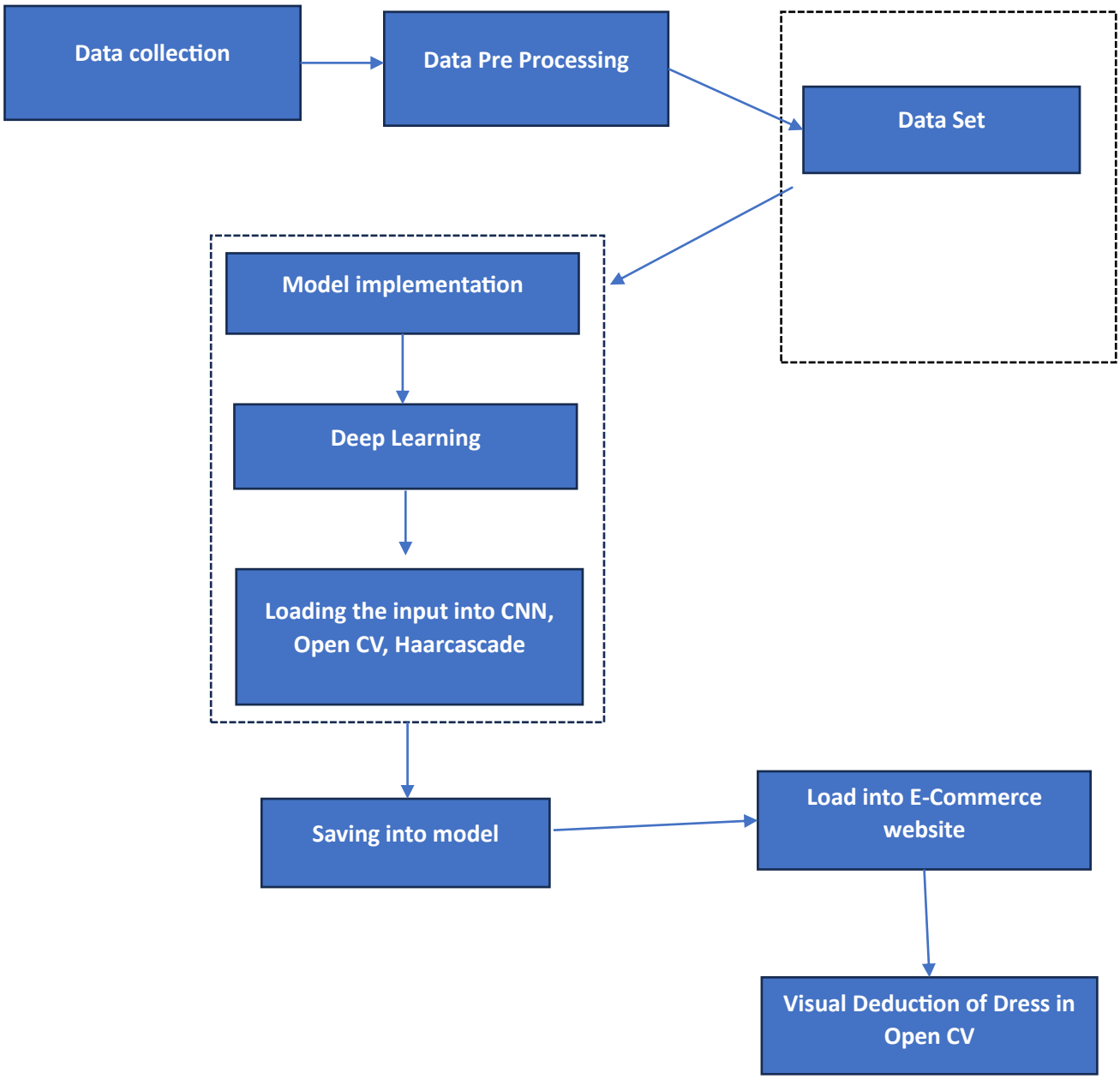
Implementing VFR technology can yield significant cost savings for retailers by streamlining operations and optimizing inventory management. By reducing the need for physical store infrastructure and minimizing overhead costs, VFRs provide a cost-effective solution for retailers looking to enhance their online presence. Additionally, VFRs enable retailers to gather valuable data and insights into consumer preferences, allowing them to tailor their offerings and marketing strategies more effectively.

Virtual try-ons play a crucial role in sustainability efforts by reducing the environmental impact associated with traditional retail practices. By minimizing the need for consumers to travel to physical stores, VFRs help mitigate carbon emissions and decrease energy consumption associated with transportation. Additionally, VFRs contribute to waste reduction by facilitating more informed purchasing decisions, thereby reducing the likelihood of returns and exchanges.

environment, VFRs empower users to experiment with different styles, colors, and combinations, enabling them to find the perfect outfit from the comfort of their homes. This level of customization fosters engagement and satisfaction, leading to increased brand loyalty and repeat purchases.

Implementing VFR technology can yield significant cost savings for retailers by streamlining operations and optimizing inventory management. By reducing the need for physical store infrastructure and minimizing overhead costs, VFRs provide a cost-effective solution for

5.2.SYSTEM ARCHITECTURE



5.3.SYSTEM MODULES:

- Module 1 : Data collection
- Module 2 : Data Pre processing
- Module 3 : Model implementation
- Module 4 : Loading the trained model
- Model 5: Prediction

Module 1 : Data Collection:

- Acquiring a diverse dataset of clothing items, including images from different angles and under various lighting conditions.
- Annotation of the dataset to label clothing categories, colors, and other relevant attributes.

Module 2: Data Preprocessing

- Image resizing and normalization to ensure uniformity in the dataset.
- Augmentation techniques to increase the dataset size and improve model generalization.
- Data cleaning to remove any irrelevant or corrupted images.

Module 3: Model Implementation

- Introduction to Convolutional Neural Networks (CNN) for image processing.
- Implementation of a CNN model for recognizing and segmenting clothing items.
- Utilizing Haarcascade for face detection in images.

Module 4 : Loading the trained model

- Saving and loading the trained CNN model for later use.
- Integration of the trained model into the virtual dressing room application.

Module 5 : Prediction:

- Using the loaded model to make predictions on new images.
- Overlaying segmented clothing items onto the user's image.
- Real-time rendering and updating as the user interacts with the virtual dressing room.

5.4.SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth N with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are

i. Economical Feasibility

ii. Technical Feasibility

iii. Social Feasibility

5.4.1. Economic Feasibility

The economic impact analysis conducted for the development of the system aimed to assess the financial implications of the project on the organization. Given the finite resources allocated for research and development, it was imperative to justify expenditures and ensure that the project remained within budgetary constraints. As a result, the decision to utilize freely available technologies played a crucial role in managing costs and achieving financial viability.

The study recognized the importance of optimizing resource allocation and maximizing the value derived from each investment. By leveraging freely available technologies, the project was able to minimize upfront costs associated with software acquisition and licensing fees. This approach allowed the organization to allocate financial resources more efficiently, directing funds towards critical areas such as custom development, implementation, and testing.

The utilization of freely available technologies also contributed to reducing the total cost of ownership over the system's lifecycle. With no recurring licensing fees or subscription costs, the organization could maintain the system at a lower operational cost, thereby enhancing its long-term economic sustainability. Moreover, by avoiding dependence on proprietary solutions, the organization retained greater flexibility and control over the system's development roadmap, enabling it to adapt to evolving business requirements without incurring significant additional expenses.

However, it is important to note that while many technologies used in the project were freely available, there may have been costs associated with customization, integration, and support. These expenses were carefully evaluated to ensure alignment with the organization's budgetary constraints and strategic objectives. Any investments made in custom development or specialized solutions were justified based on their potential to deliver tangible value and competitive advantage to the organization.

In summary, the decision to leverage freely available technologies played a pivotal role in ensuring the economic viability of the project. By minimizing upfront costs, reducing total cost of ownership, and maintaining flexibility and control over the system's development, the organization was able to maximize the return on investment and achieve its strategic goals within budgetary constraints. Moving forward, continued diligence in managing expenditures and optimizing resource allocation will remain critical to sustaining the economic impact of the developed system on the organization..

5.4.2. Technical Feasibility

The technical feasibility analysis conducted for this study aimed to evaluate the system's requirements to ensure that it does not impose excessive demands on available technical resources. It is essential for any system developed to operate within reasonable limits to

prevent overburdening the client with high technical requirements. The goal is to design a system with modest requirements, minimizing the need for significant changes or upgrades during implementation.

Key considerations in assessing technical feasibility include hardware, software, network infrastructure, and system scalability. By carefully evaluating these factors, the study aimed to identify potential challenges and ensure that the system's technical requirements align with the client's capabilities and constraints.

5.4.3. Hardware Requirements:

The system's hardware requirements were thoroughly examined to ensure compatibility with the client's existing infrastructure. This included assessing the processing power, memory, and storage capacity needed to support the system's operations. By specifying modest hardware requirements, the study aimed to minimize the need for costly hardware upgrades or investments.

5.4.4. Software Requirements:

Evaluation of software requirements focused on identifying compatible software platforms and tools that align with the client's existing technology stack. By leveraging widely used and standardized software solutions, the study aimed to reduce the complexity of system implementation and minimize the need for specialized software licenses or custom development.

5.4.5. Network Infrastructure:

Assessment of network infrastructure requirements aimed to ensure that the system's communication protocols and bandwidth usage are optimized for efficient data transmission. By designing the system to operate within the client's network constraints, the study aimed to prevent congestion and latency issues that could impact system performance.

5.4.6. Scalability:

Scalability considerations were essential to accommodate future growth and expansion of the system. By designing a scalable architecture that can adapt to changing demands, the study aimed to future-proof the system and minimize the need for costly upgrades or redevelopments as the client's needs evolve over time.

In conclusion, the technical feasibility analysis underscored the importance of designing a system with modest requirements to minimize the impact on available technical resources. By carefully assessing hardware, software, network infrastructure, and scalability considerations, the study aimed to ensure that the developed system can be implemented seamlessly with minimal changes or disruptions to the client's operations. This approach not only reduces implementation costs but also enhances the system's long-term sustainability and adaptability to future changes..

5.4.7. Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the

users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

5.5. TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub – assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.5.1 TYPES OF TESTS

UNIT TESTING

Unit testing constitutes a fundamental aspect of software development, encompassing the design and execution of test cases aimed at validating the internal logic of individual program units. The primary objective of unit testing is to ascertain that each component of the software functions correctly and produces valid outputs in response to various inputs. This process involves meticulously examining all decision branches and internal code flows within the unit, ensuring that every aspect of its functionality operates as intended.

Unit testing is conducted at the granular level, focusing on verifying the behavior of isolated software units independent of the broader system context. It typically occurs following the completion of the implementation of a single unit, before integration with other units or components. As a form of structural testing, unit testing relies on a deep understanding of the construction of the software unit under examination and involves invasive techniques to scrutinize its internal workings.

The essence of unit testing lies in its ability to perform basic tests at the component level, thoroughly evaluating specific business processes, applications, or system configurations. Each unit test is meticulously crafted to target a particular aspect of the software's functionality, ensuring that it adheres to documented specifications and exhibits clearly defined inputs and expected outputs. By systematically testing each unique path of a business process, unit tests provide assurance that the software accurately performs its intended operations under various conditions.

In essence, unit testing serves as a critical quality assurance mechanism, enabling developers to detect and rectify defects early in the development lifecycle. By validating the correctness of individual software units in isolation, unit testing contributes to the overall reliability, robustness, and maintainability of the software system. Additionally, unit tests serve as an essential component of a comprehensive testing strategy, complementing other testing methodologies such as integration testing, system testing, and acceptance testing to ensure the overall quality of the software product.

INTEGRATION TESTING

Integration testing represents a critical phase in the software development lifecycle, focusing on evaluating the interoperability and functionality of integrated software components within the context of a unified system. Unlike unit testing, which verifies the behavior of individual program units in isolation, integration testing assesses how these units function together as a cohesive whole. This process aims to determine if the integrated components effectively collaborate and operate as a unified program, thereby validating that the system functions as intended.

Integration testing is characterized by an event-driven approach, where tests are designed to simulate various events or interactions that occur within the software system. These tests typically evaluate the basic outcomes of screens, user interfaces, or data exchanges between different modules or subsystems. By examining the interactions between integrated components, integration tests seek to identify any inconsistencies, dependencies, or communication issues that may arise during runtime.

The primary objective of integration testing is to demonstrate that although individual components may have passed unit testing satisfactorily, their combination within the integrated system is correct and consistent. This validation is essential to ensure that the software behaves as expected when all components are interconnected and functioning together. Integration tests serve as a crucial validation mechanism, providing assurance that the integrated system meets the specified requirements and behaves predictably under various scenarios.

Integration testing is specifically aimed at exposing the problems that may arise from the combination of components, such as interface mismatches, data inconsistencies, or functionality conflicts. By identifying and addressing these issues early in the development process, integration testing helps mitigate risks associated with system integration and ensures the overall reliability, stability, and performance of the software system.

In summary, integration testing plays a pivotal role in validating the interoperability and functionality of integrated software components within a unified system. Through event-driven testing and evaluation of integrated behaviors, integration testing provides essential insights into the reliability and consistency of the software system, ultimately contributing to the delivery of high-quality software products that meet user expectations and business requirements.

FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised

Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

SYSTEM TEST

System testing represents a crucial phase in the software development lifecycle, aimed at verifying that the entire integrated software system meets specified requirements and functions as intended. Unlike unit testing and integration testing, which focus on validating individual components or the interaction between components, system testing evaluates the software system as a whole to ensure that it delivers the desired functionality and performance.

At its core, system testing is designed to test a configured instance of the software system to ensure that it produces known and predictable results. This involves executing various test scenarios and user interactions to validate that the system behaves in accordance with the defined requirements and specifications. System testing encompasses a broad range of testing techniques and methodologies, including functional testing, performance testing, usability testing, and security testing, among others.

One example of system testing is configuration-oriented system integration testing, where the focus is on validating the system's behavior under different configurations or deployment environments. This type of testing ensures that the software system can adapt to various settings, configurations, or external dependencies without compromising its functionality or performance. By systematically testing different configuration scenarios, system testing helps identify and address compatibility issues, configuration errors, and other deployment-related challenges.

System testing is inherently comprehensive and encompasses all aspects of the software system, including its user interfaces, business logic, data processing capabilities, and integration with external systems or services. It verifies that the system meets functional requirements by validating its behavior against expected outcomes and predefined acceptance criteria. Additionally, system testing evaluates non-functional aspects of the system, such as performance, scalability, reliability, and security, to ensure that it meets the desired quality standards and user expectations.

The success of system testing relies on thorough planning, meticulous test case design, and systematic execution of test scenarios across different system configurations and environments. By rigorously testing the entire software system in a controlled environment, system testing provides stakeholders with confidence in the system's reliability, robustness, and suitability for deployment in production environments.

In summary, system testing serves as a critical validation mechanism to ensure that the entire integrated software system meets requirements and delivers the intended functionality and performance. Through comprehensive testing of functional and non-functional aspects, system testing helps identify defects, mitigate risks, and validate the readiness of the software system for deployment in real-world scenarios. WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

BLACK BOX TESTING

Black Box Testing is a fundamental software testing technique that focuses on evaluating the functionality of a software application without any knowledge of its internal workings, structure, or programming language. Unlike other testing methods that require insights into the underlying code or system architecture, Black Box Testing treats the software as an opaque entity, akin to a "black box," where the internal mechanisms are not visible or accessible to the tester.

The primary principle of Black Box Testing is to assess the software solely based on its externally observable behavior, without delving into the implementation details. Testers interact with the software through its user interface or defined input channels, providing inputs and evaluating outputs to verify whether the system behaves as expected. This approach is analogous to testing a physical device or appliance without knowledge of its internal circuitry or mechanisms.

One of the key aspects of Black Box Testing is that test cases are derived from definitive source documents, such as specifications or requirements documents. These documents outline the expected behavior and functionality of the software system, serving as the basis for designing test scenarios and expected outcomes. Testers craft test cases based on these specifications, ensuring that the software meets the specified requirements and behaves in accordance with user expectations.

During Black Box Testing, testers are not concerned with how the software accomplishes its tasks internally. Instead, they focus solely on the inputs provided to the software and the corresponding outputs generated in response. This abstraction allows testers to assess the software's functionality from a user's perspective, mimicking real-world usage scenarios and interactions without requiring knowledge of the underlying codebase or implementation details.

Black Box Testing encompasses various testing techniques, including functional testing, usability testing, acceptance testing, and regression testing. Each technique focuses on different aspects of the software's behavior, such as its functionality, user interface, performance, and compatibility with external systems. By employing a combination of these techniques, testers can thoroughly evaluate the software's capabilities and ensure that it meets quality standards and user requirements.

In summary, Black Box Testing is a critical component of the software testing process, enabling testers to assess the functionality and behavior of a software application without knowledge of its internal workings. By focusing on externally observable behavior and

relying on definitive source documents, Black Box Testing ensures that the software meets specified requirements and behaves as expected from a user's perspective. This approach enhances the reliability, usability, and quality of the software, ultimately contributing to a positive user experience.

FRONTEND:

Index.html

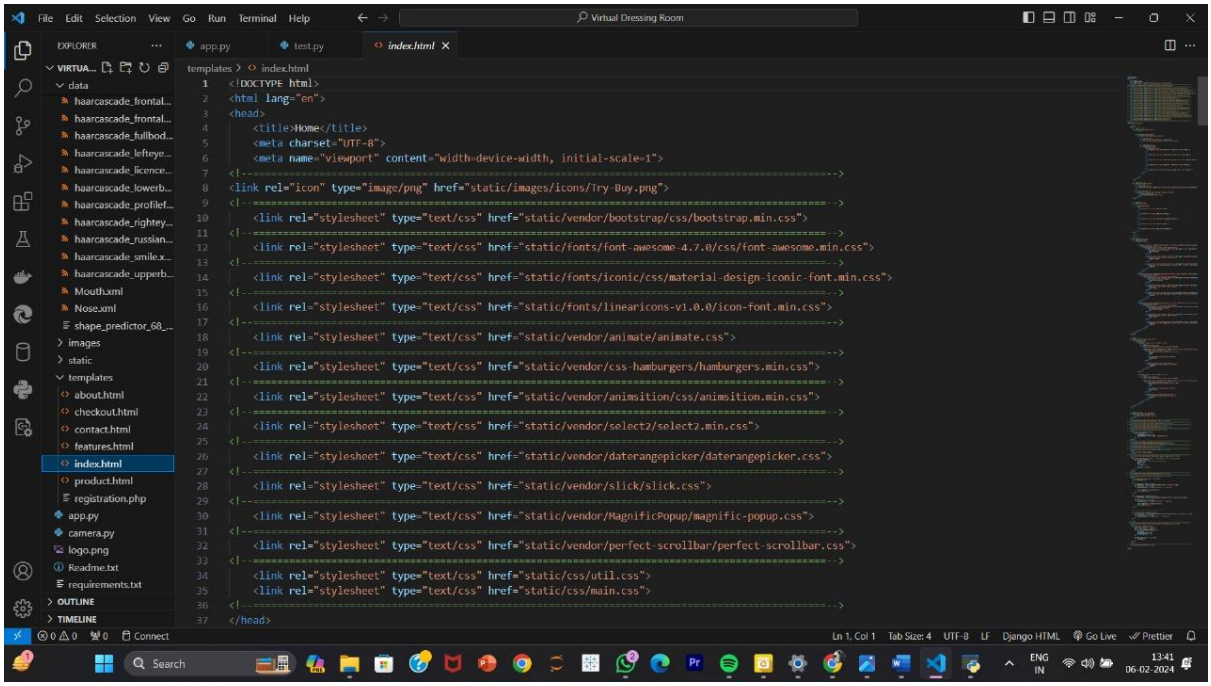


Fig 5.1

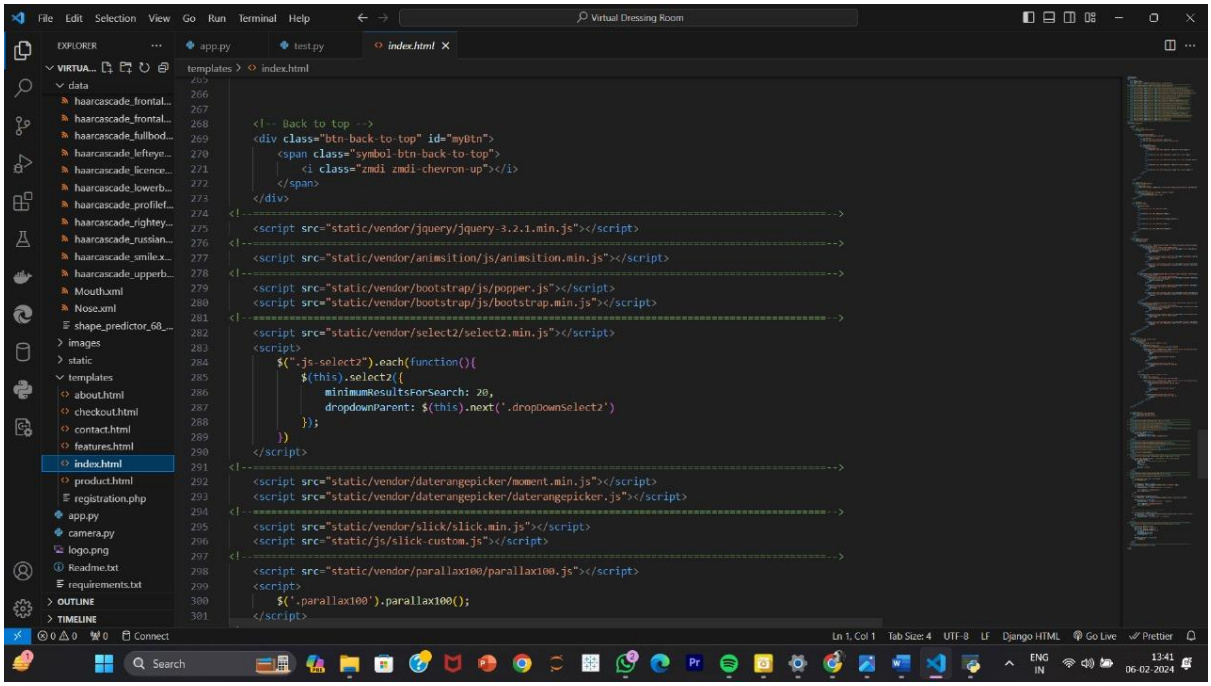


Fig 5.2

Flask:

App.py

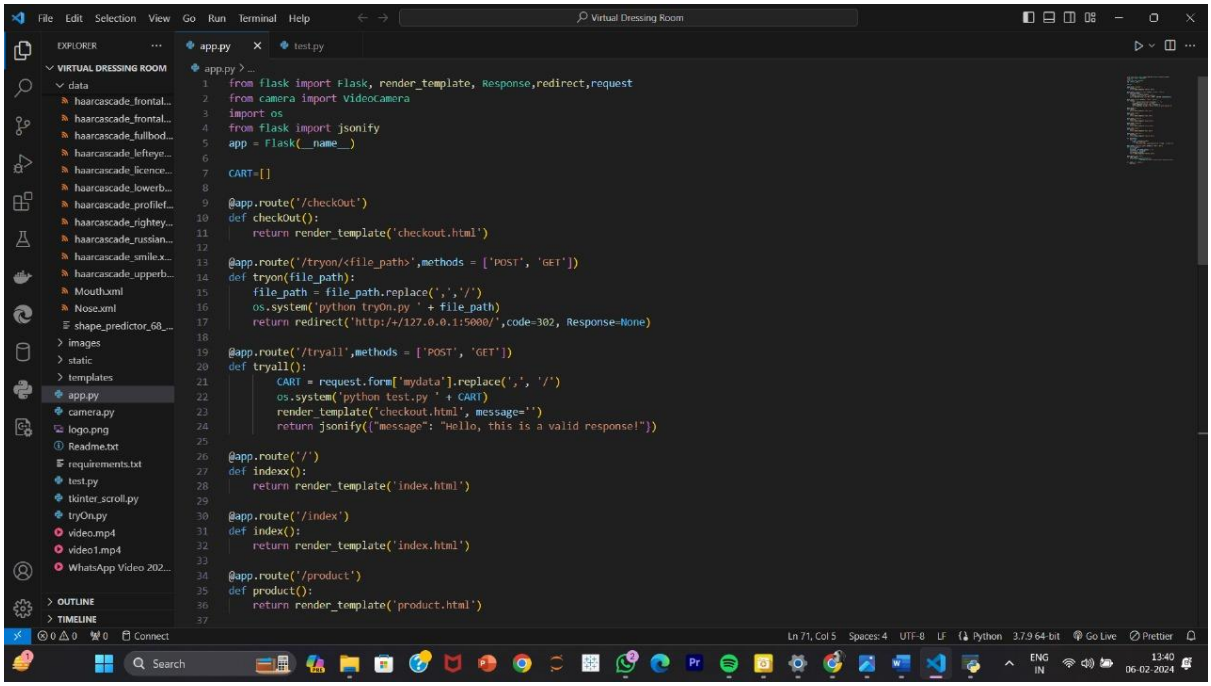


Fig 5.3

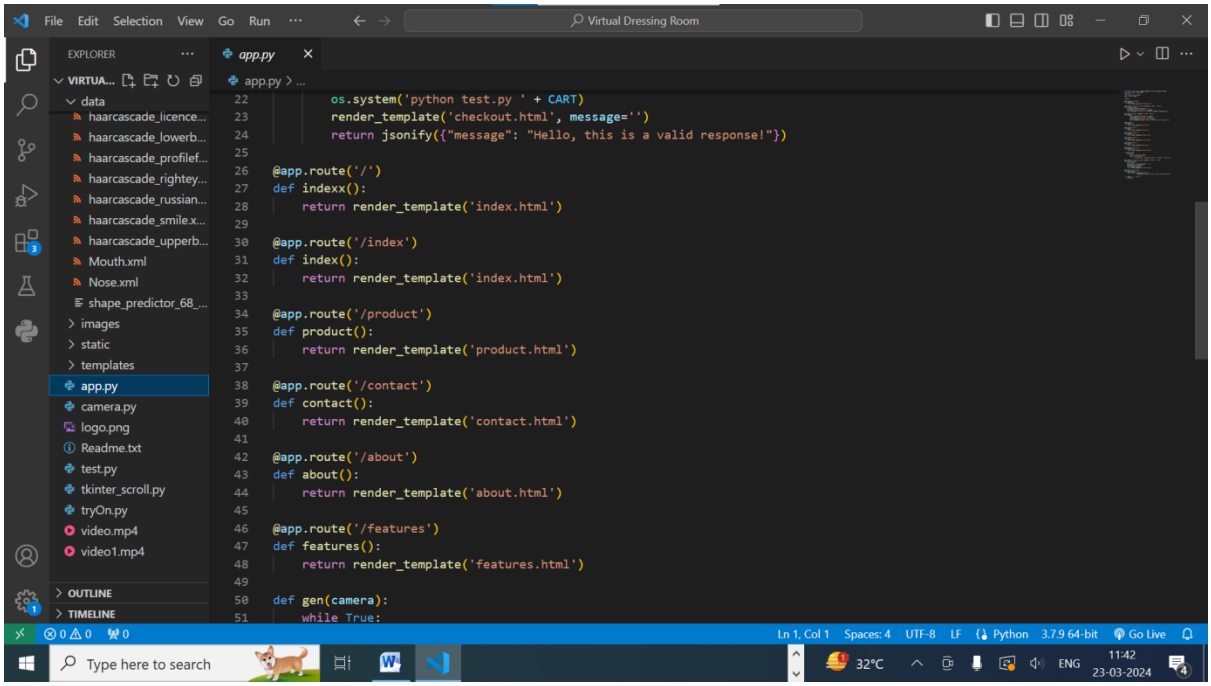

File Edit Selection View Go Run
Virtual Dressing Room
EXPLORER
app.py
os.system('python test.py ' + CART)
render_template('checkout.html', message='')
return jsonify({"message": "Hello, this is a valid response!"})
@app.route('/')
def indexx():
return render_template('index.html')
@app.route('/index')
def index():
return render_template('index.html')
@app.route('/product')
def product():
return render_template('product.html')
@app.route('/contact')
def contact():
return render_template('contact.html')
@app.route('/about')
def about():
return render_template('about.html')
@app.route('/features')
def features():
return render_template('features.html')
def gen(camera):
while True:
Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.7.9 64-bit Go Live
Type here to search
32°C
ENG
11:42
23-03-2024

Fig 5.4

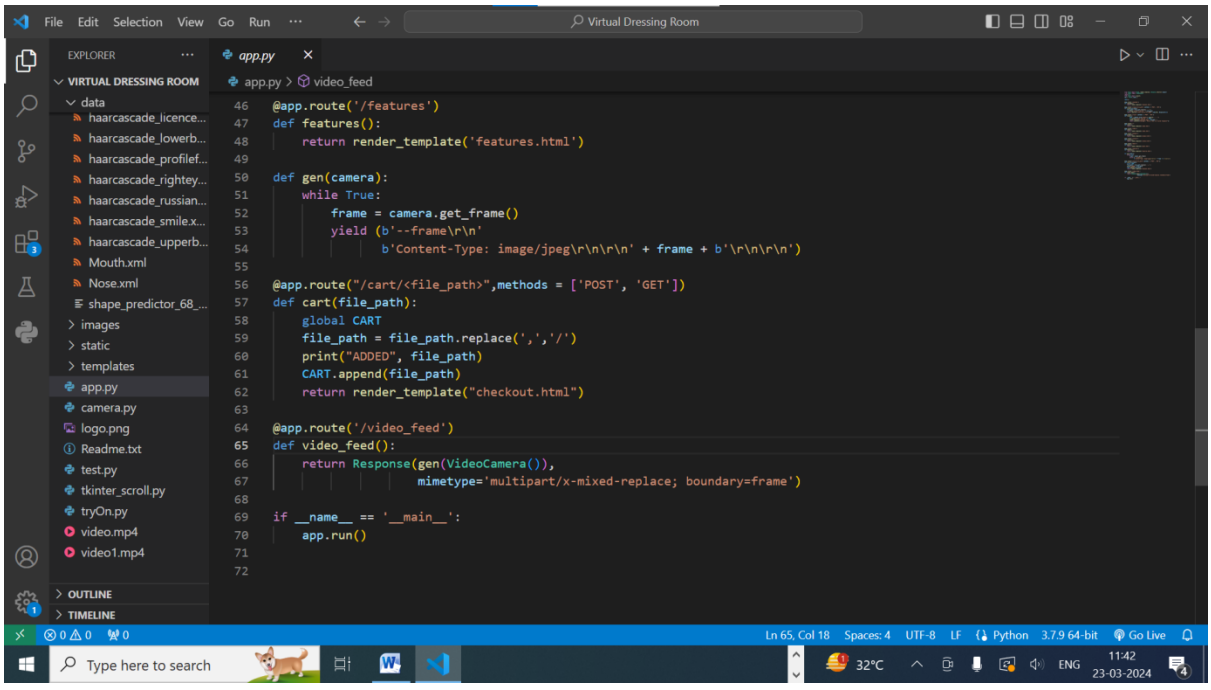

File Edit Selection View Go Run
Virtual Dressing Room
EXPLORER
app.py
VIRTUAL DRESSING ROOM
@app.route('/features')
def features():
return render_template('features.html')
def gen(camera):
while True:
frame = camera.get_frame()
yield (b'--frame\r\n'
b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')
@app.route("/cart/<file_path>",methods = ['POST', 'GET'])
def cart(file_path):
global CART
file_path = file_path.replace(',','/')
print("ADDED", file_path)
CART.append(file_path)
return render_template("checkout.html")
@app.route('/video_feed')
def video_feed():
return Response(gen(VideoCamera()),
mimetype='multipart/x-mixed-replace; boundary=frame')
if __name__ == '__main__':
app.run()
Ln 65, Col 18 Spaces: 4 UTF-8 LF Python 3.7.9 64-bit Go Live
Type here to search
32°C
ENG
11:42
23-03-2024

Fig 5.5

HAAR CASCADE:

Dataset

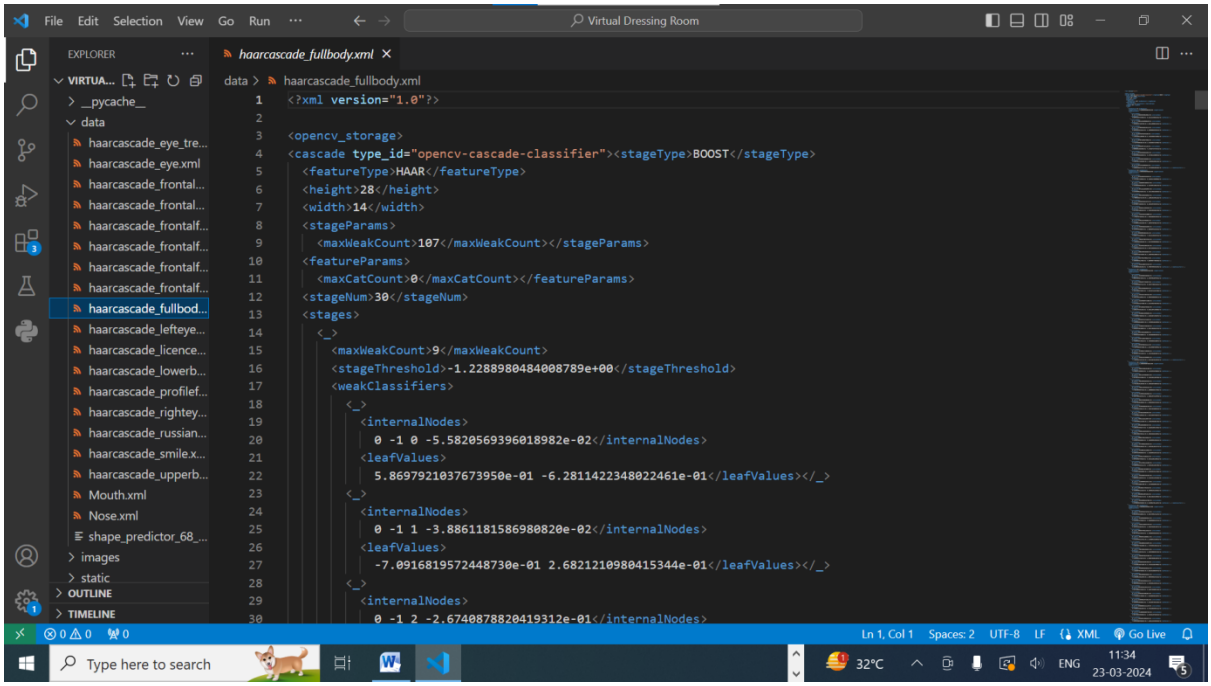


Fig 5.6

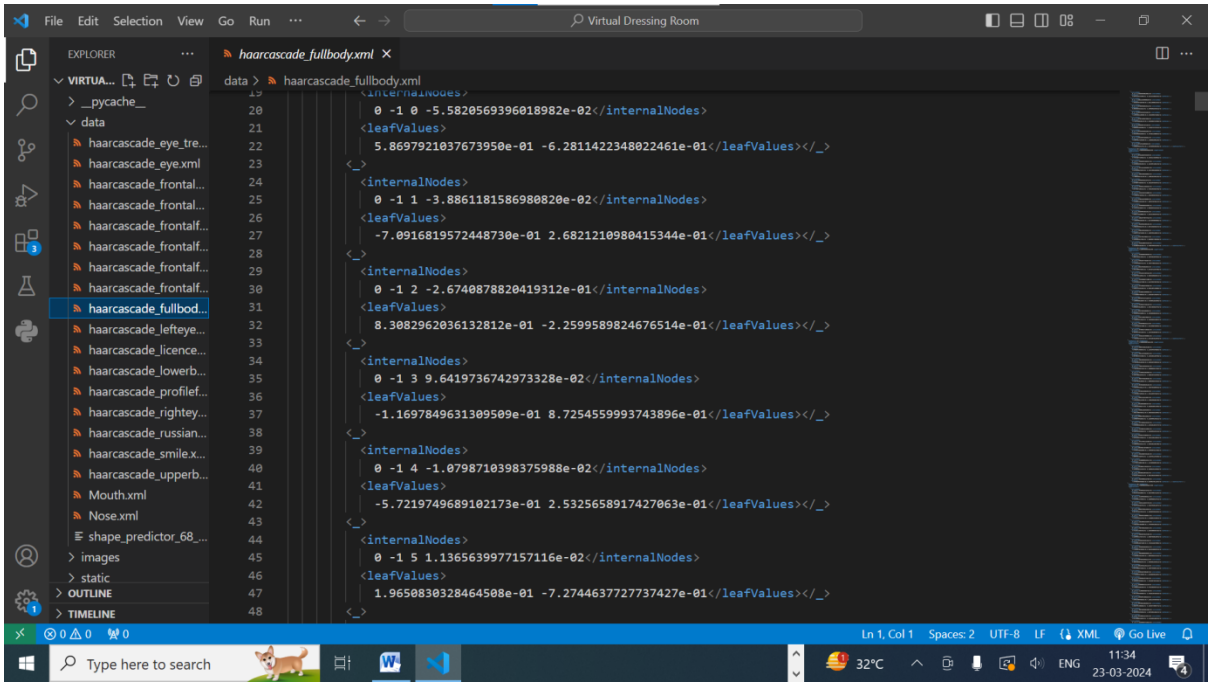


Fig 5.7

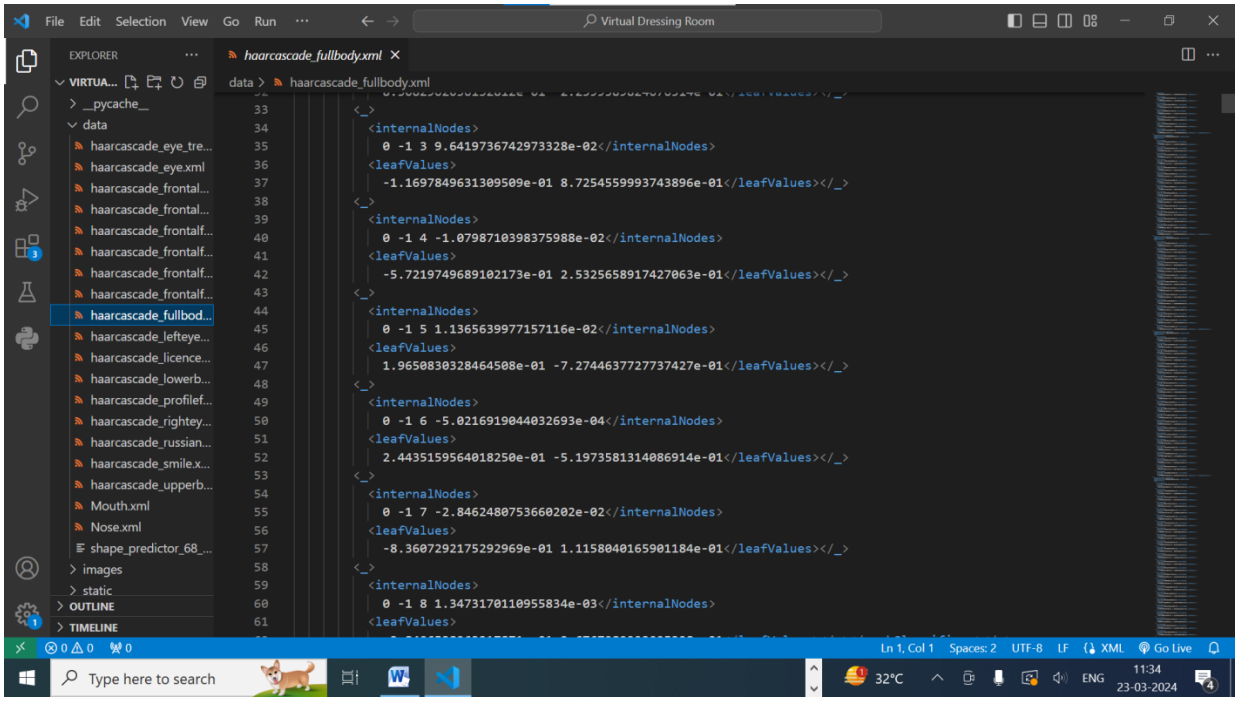

File Edit Selection View Go Run
Virtual Dressing Room
haarcascade_fullbody.xml
data > haarcascade_fullbody.xml
<internalNodes>
0 -1 3 9.6419736742973328e-02</internalNodes>
<leafValues>
-1.1697849631309509e-01 8.7254559993743896e-01</leafValues></_>
<internalNodes>
0 -1 4 -1.0798710398375988e-02</internalNodes>
<leafValues>
-5.7219749689102173e-01 2.5325658917427063e-01</leafValues></_>
<internalNodes>
0 -1 5 1.1365639977157116e-02</internalNodes>
<leafValues>
1.9650830328464508e-01 -7.2744637727737427e-01</leafValues></_>
<internalNodes>
0 -1 6 -5.0216919044032693e-04</internalNodes>
<leafValues>
2.4435159564018250e-01 -5.1973581314086914e-01</leafValues></_>
<internalNodes>
0 -1 7 -2.8462480753660202e-02</internalNodes>
<leafValues>
-8.3607292175292969e-01 1.1158040165901184e-01</leafValues></_>
<internalNodes>
0 -1 8 1.3473170110955834e-03</internalNodes>
<leafValues>
Ln 1, Col 1 Spaces: 2 UTF-8 LF XML Go Live
Type here to search
32°C ENG 11:34 23-03-2024

Fig 5.8

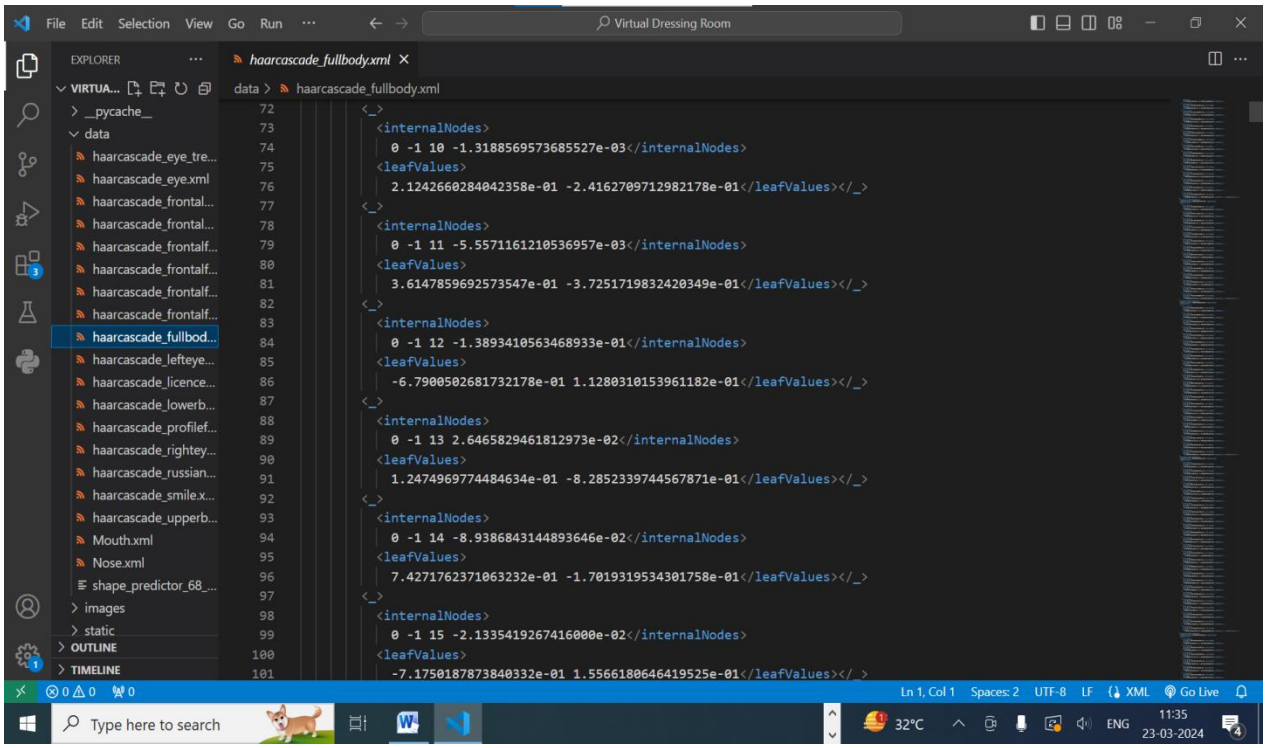

File Edit Selection View Go Run
Virtual Dressing Room
haarcascade_fullbody.xml
data > haarcascade_fullbody.xml
<internalNodes>
0 -1 10 -1.3188569573685527e-03</internalNodes>
<leafValues>
2.1242660284042358e-01 -2.4162709712982178e-01</leafValues></_>
<internalNodes>
0 -1 11 -5.5571161210536957e-03</internalNodes>
<leafValues>
3.6147859692573547e-01 -3.7251719832420349e-01</leafValues></_>
<internalNodes>
0 -1 12 -1.3893410563468933e-01</internalNodes>
<leafValues>
-6.7900502681732178e-01 1.1280310153961182e-01</leafValues></_>
<internalNodes>
0 -1 13 2.6465829461812973e-02</internalNodes>
<leafValues>
1.2474969774484634e-01 -8.2852339744567871e-01</leafValues></_>
<internalNodes>
0 -1 14 -8.9386843144893646e-02</internalNodes>
<leafValues>
7.4271762371063232e-01 -1.7019319534301758e-01</leafValues></_>
<internalNodes>
0 -1 15 -2.1335419267416000e-02</internalNodes>
<leafValues>
-7.1750187873840332e-01 1.5566180646419525e-01</leafValues></_>
Ln 1, Col 1 Spaces: 2 UTF-8 LF XML Go Live
Type here to search
32°C ENG 11:35 23-03-2024

Fig 5.9

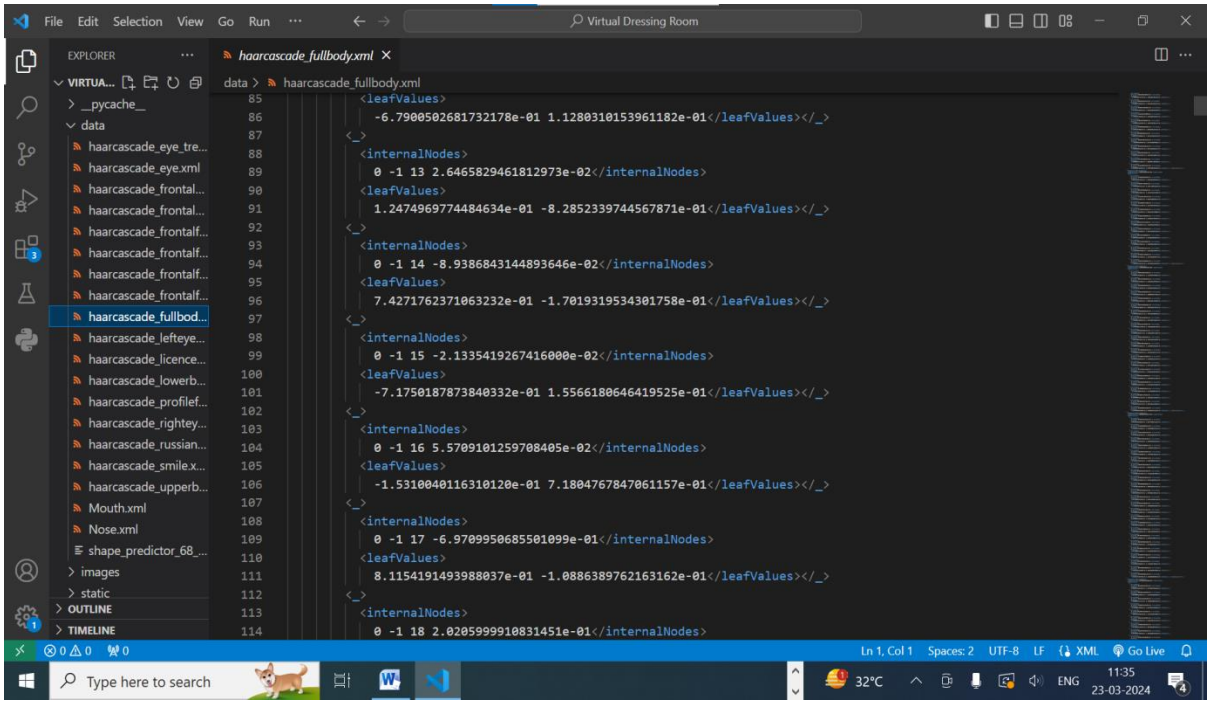


Fig 5.10

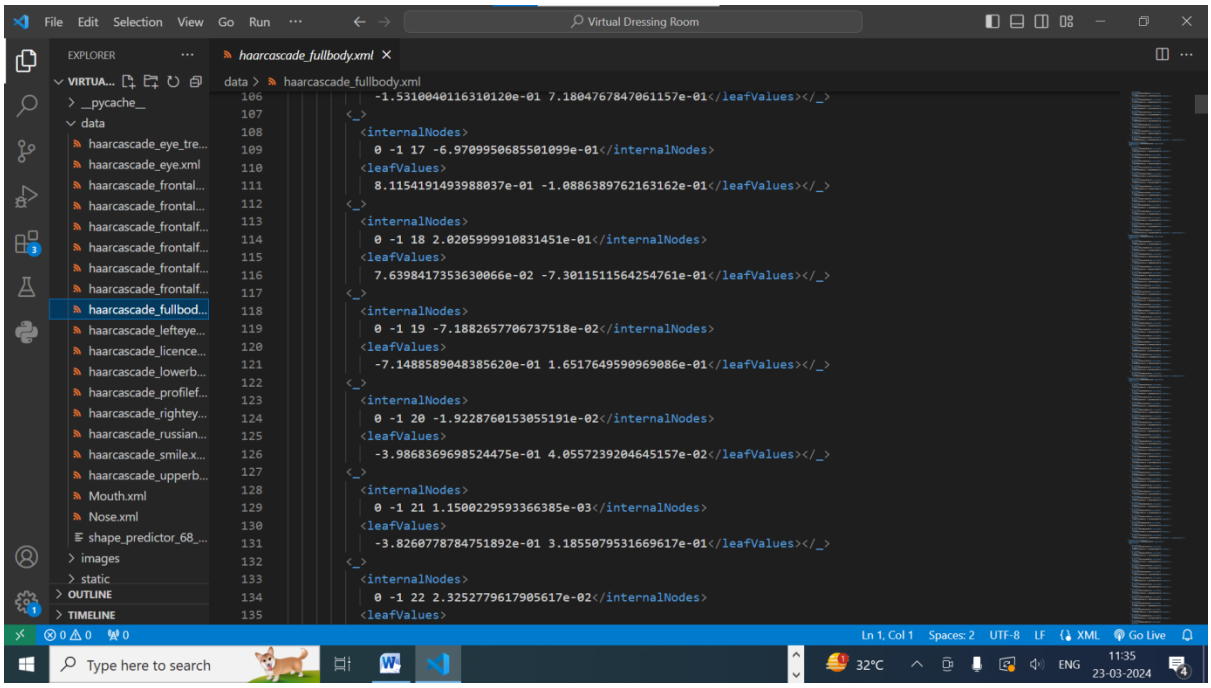


Fig 5.11

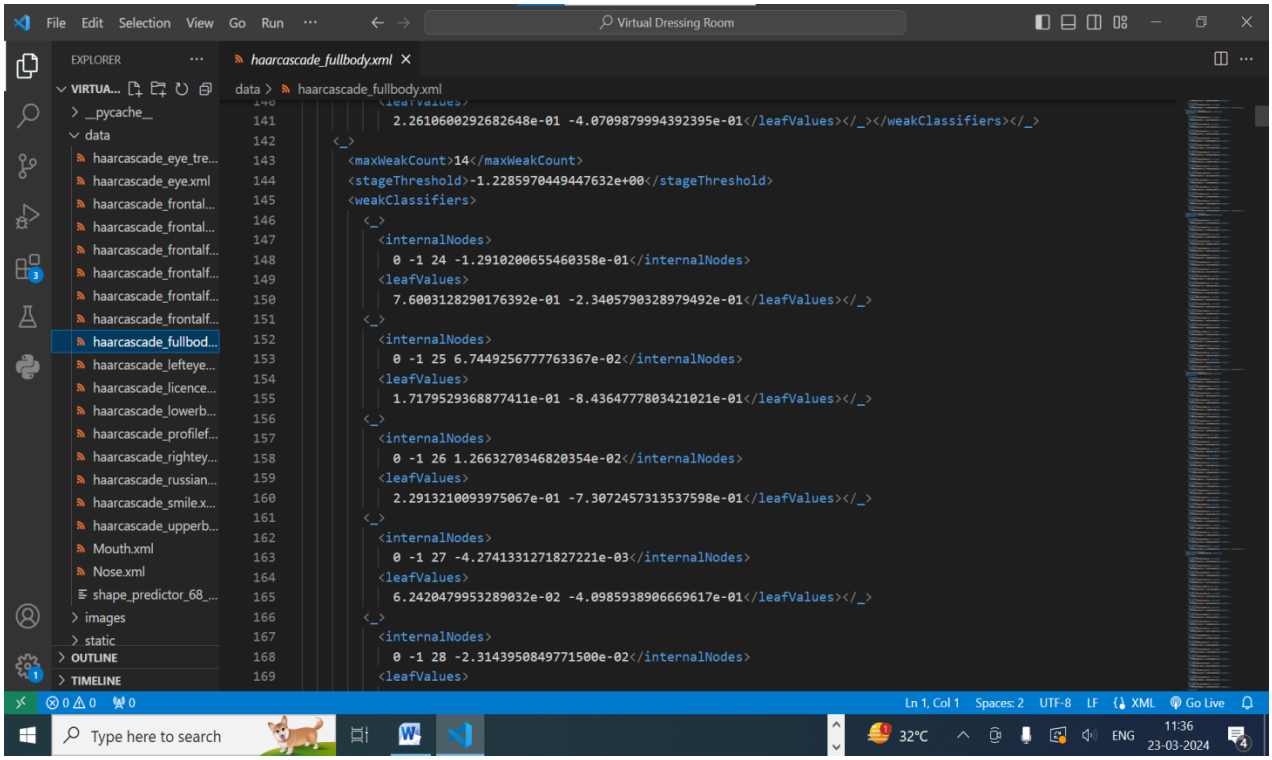


Fig 5.12

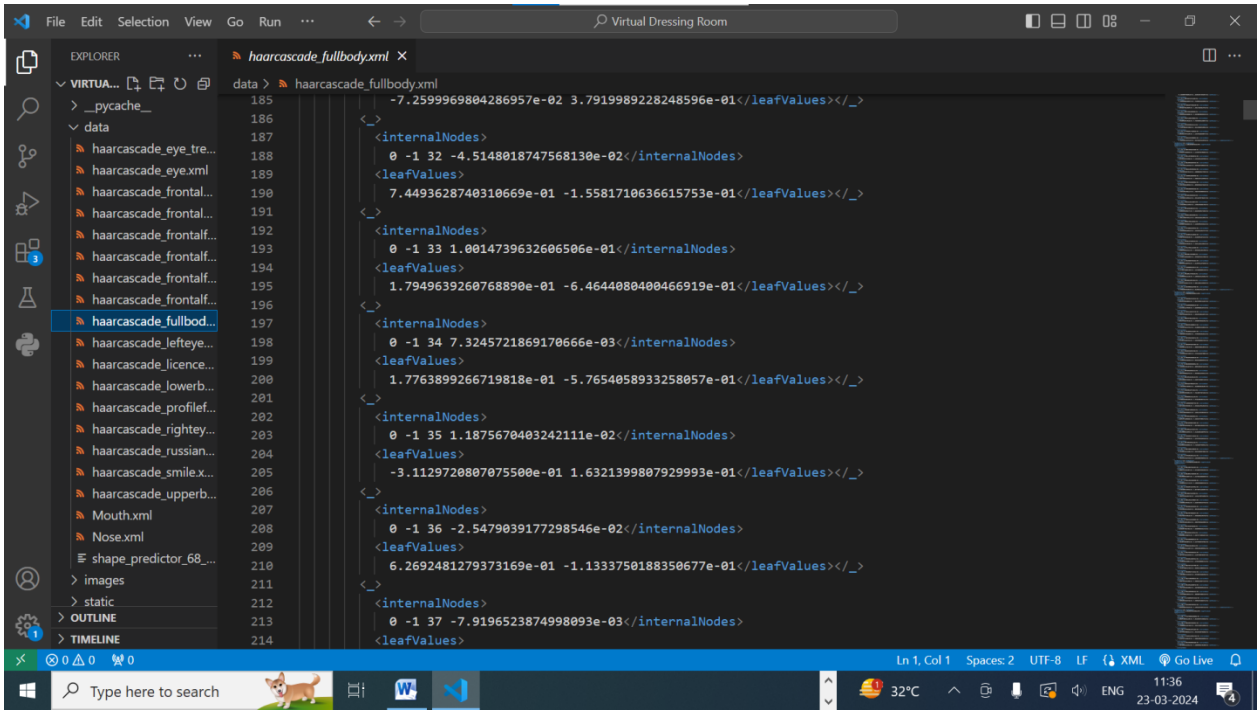


Fig 5.13

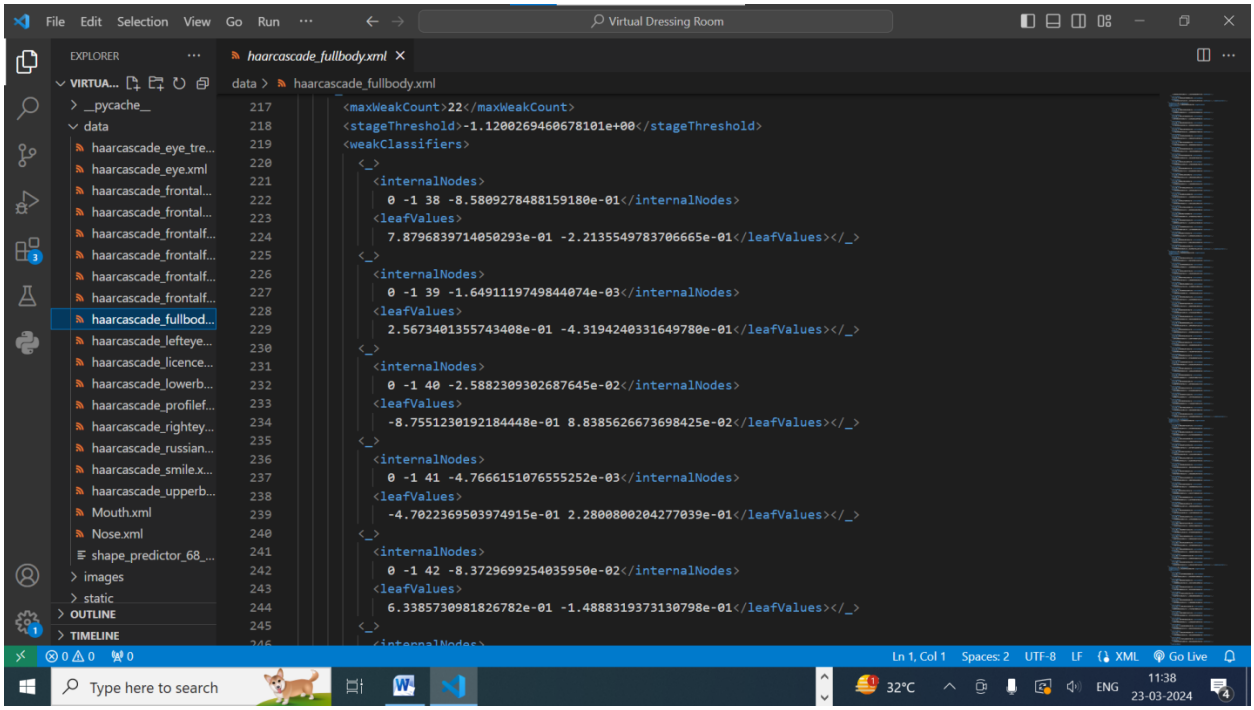


Fig 5.14

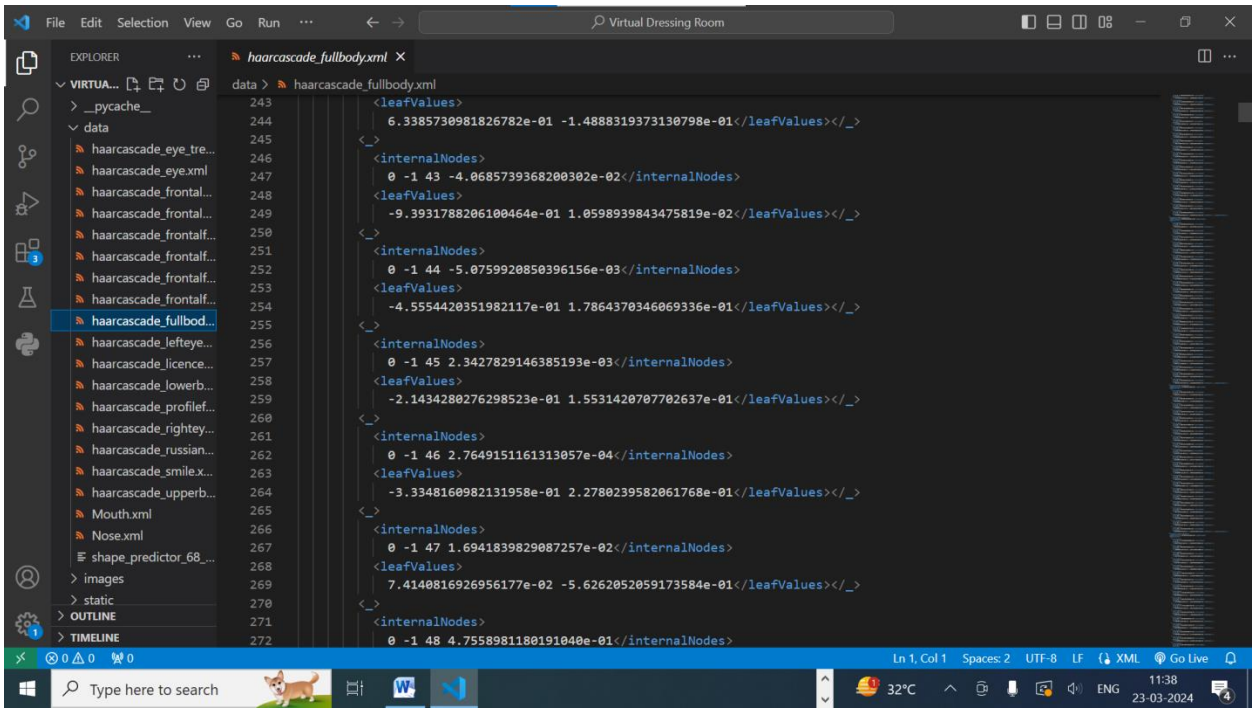


Fig 5.15

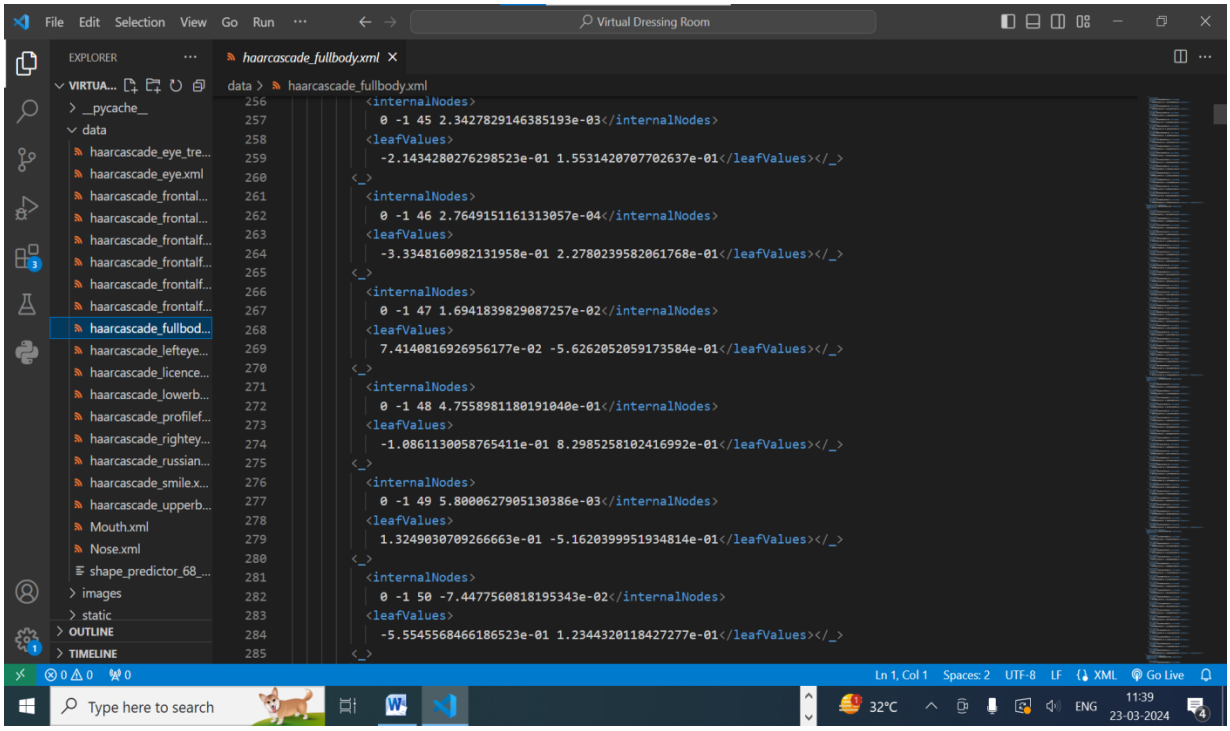


Fig 5.16

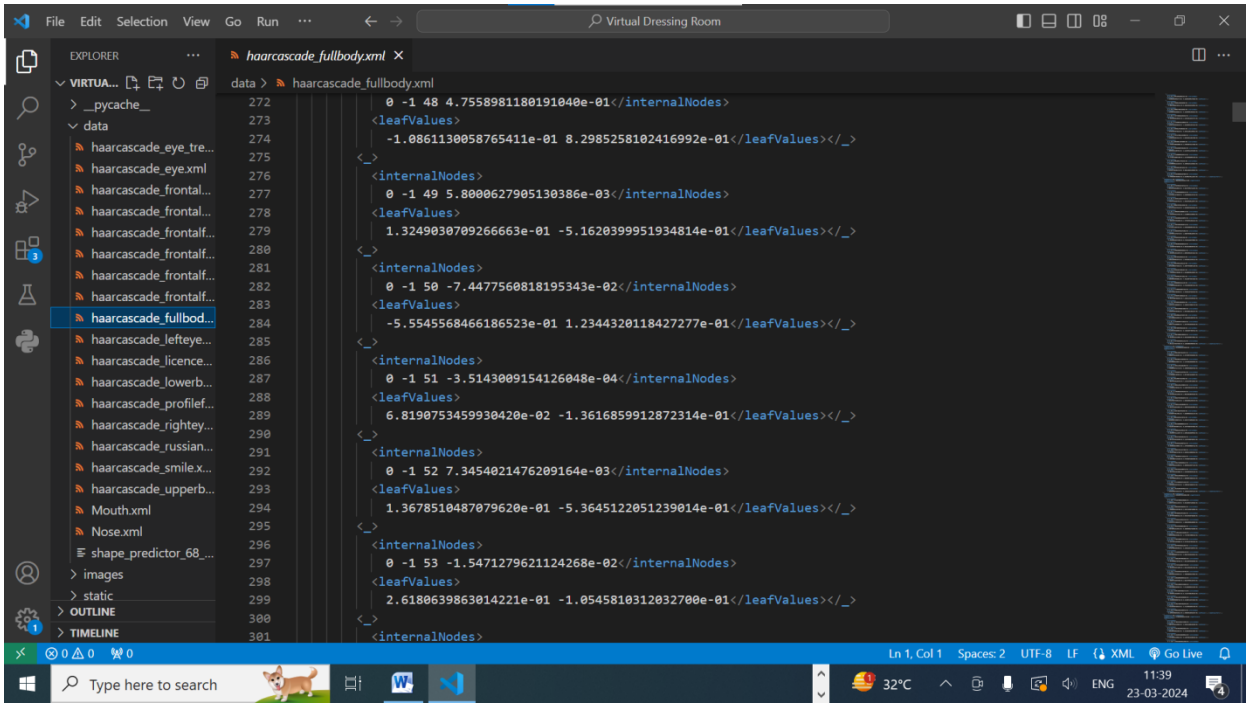


Fig 5.17

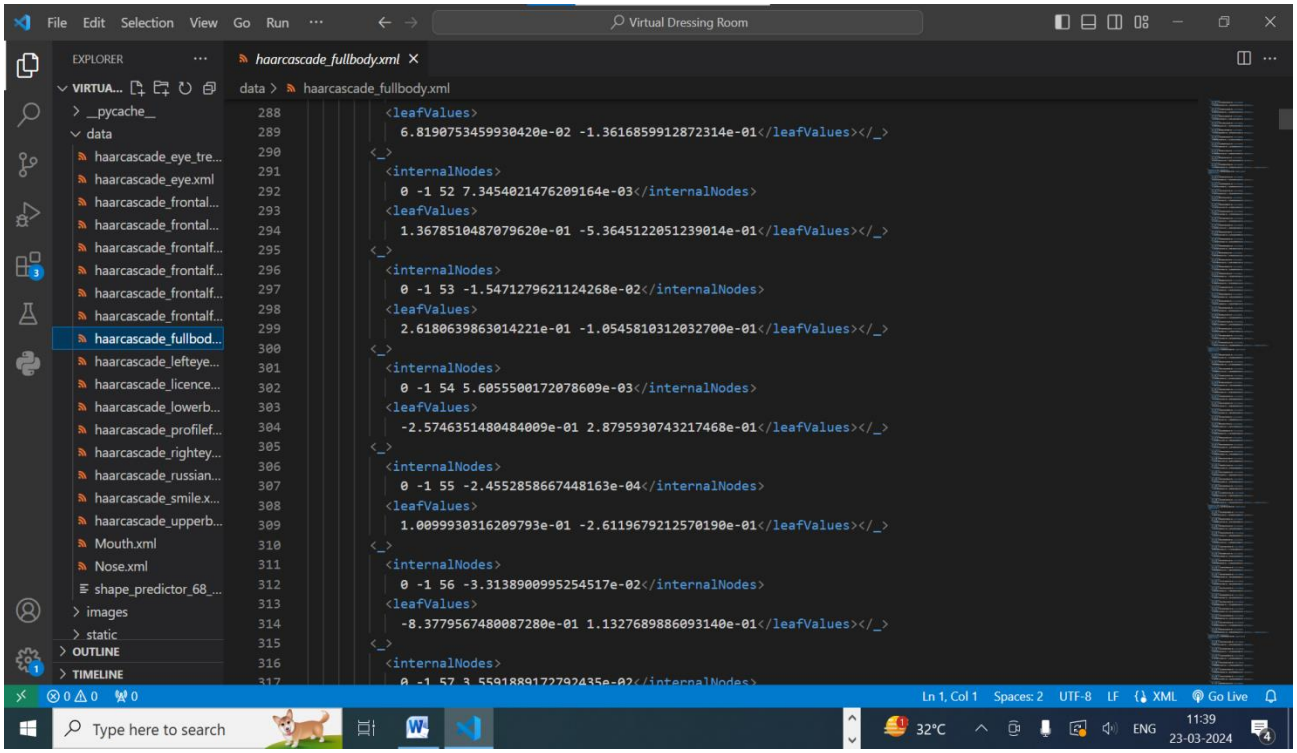


Fig 5.18

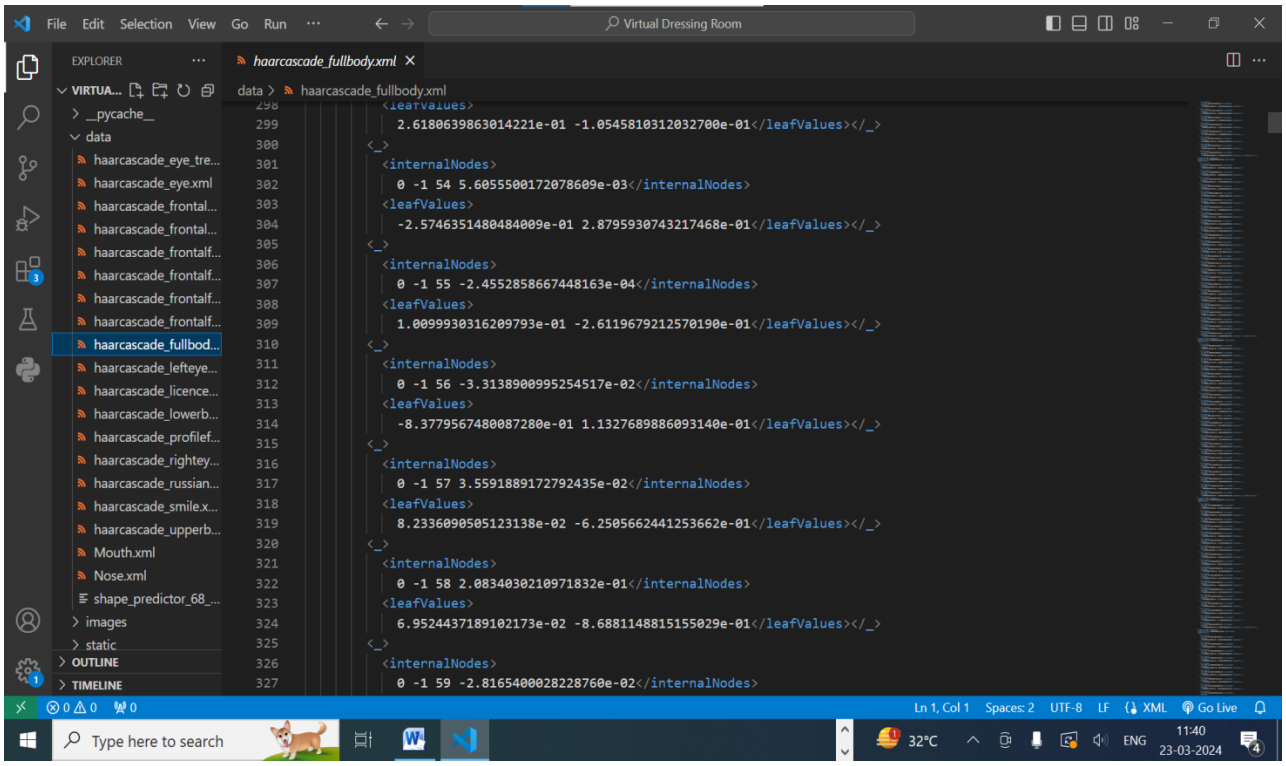


Fig 5.19

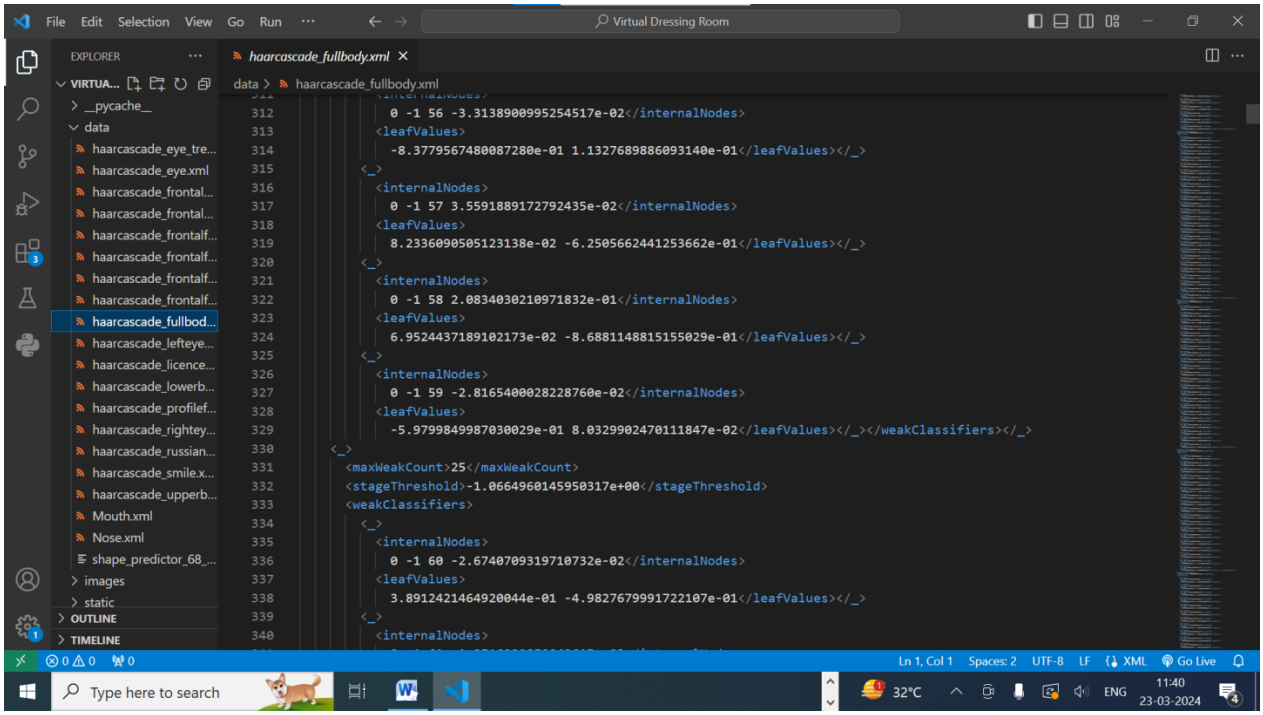


Fig 5.20

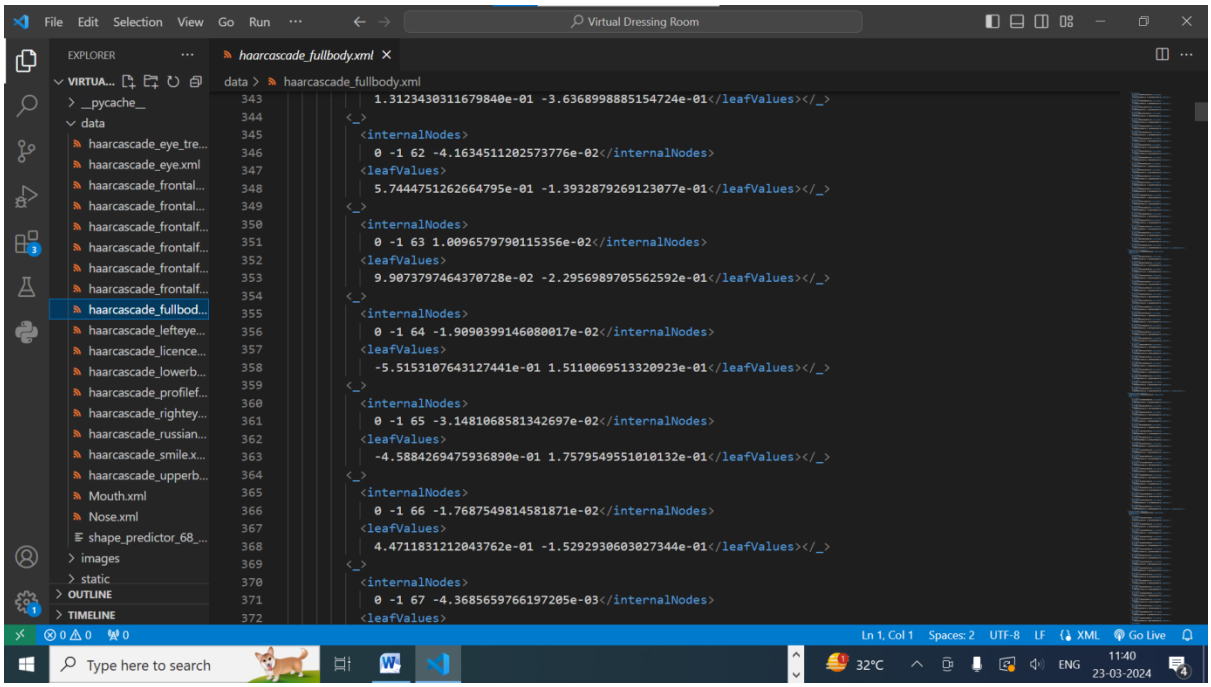


Fig 5.21

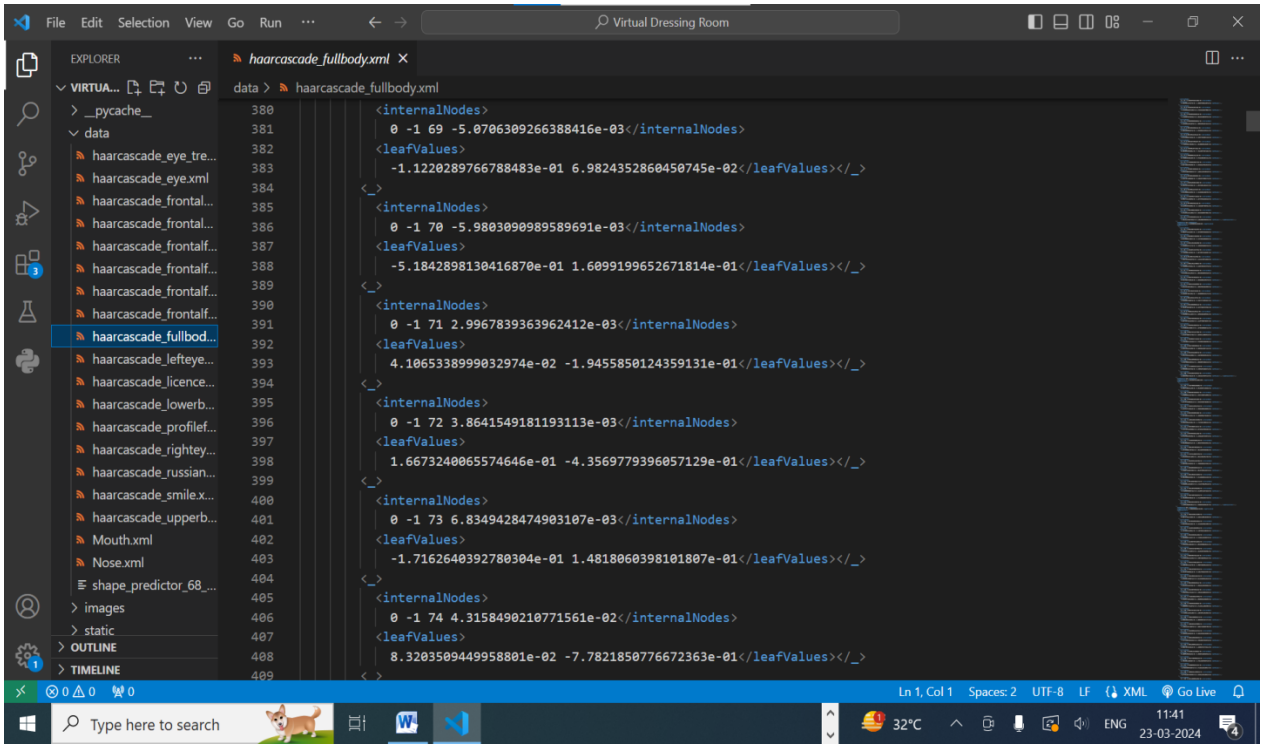


Fig 5.22

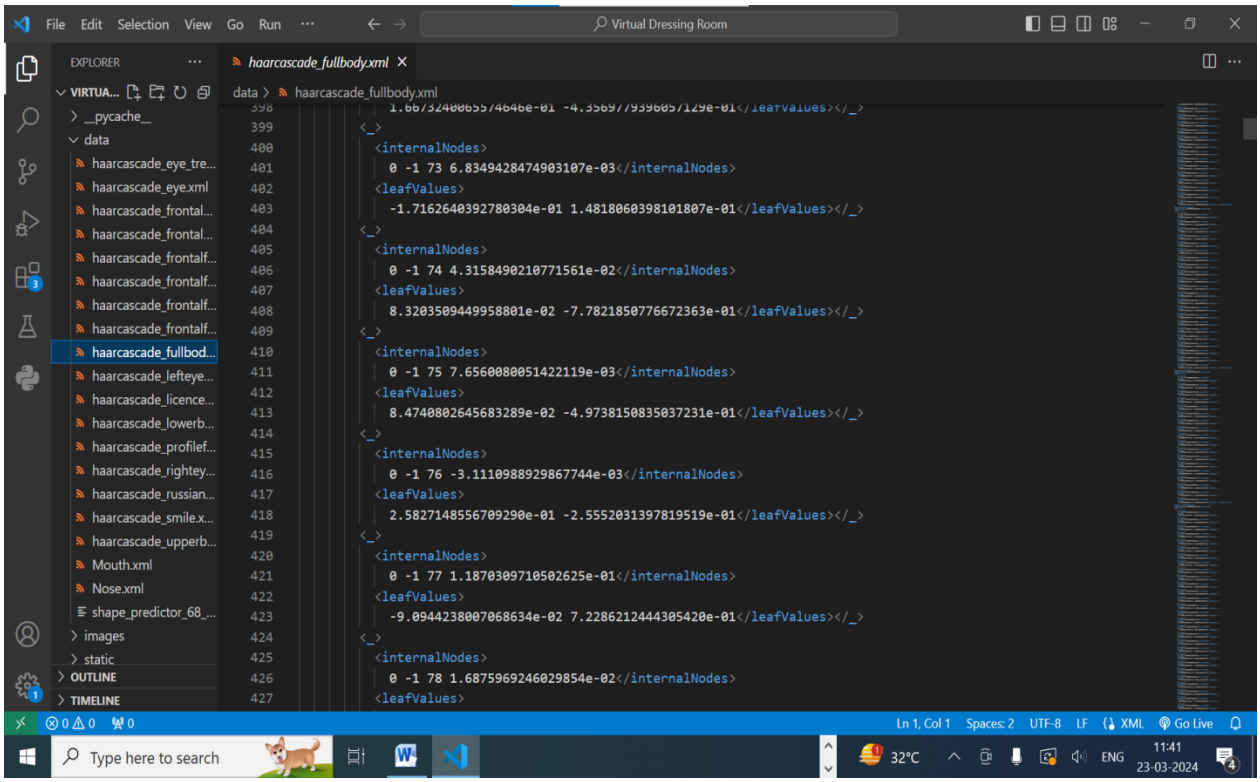


Fig 5.23

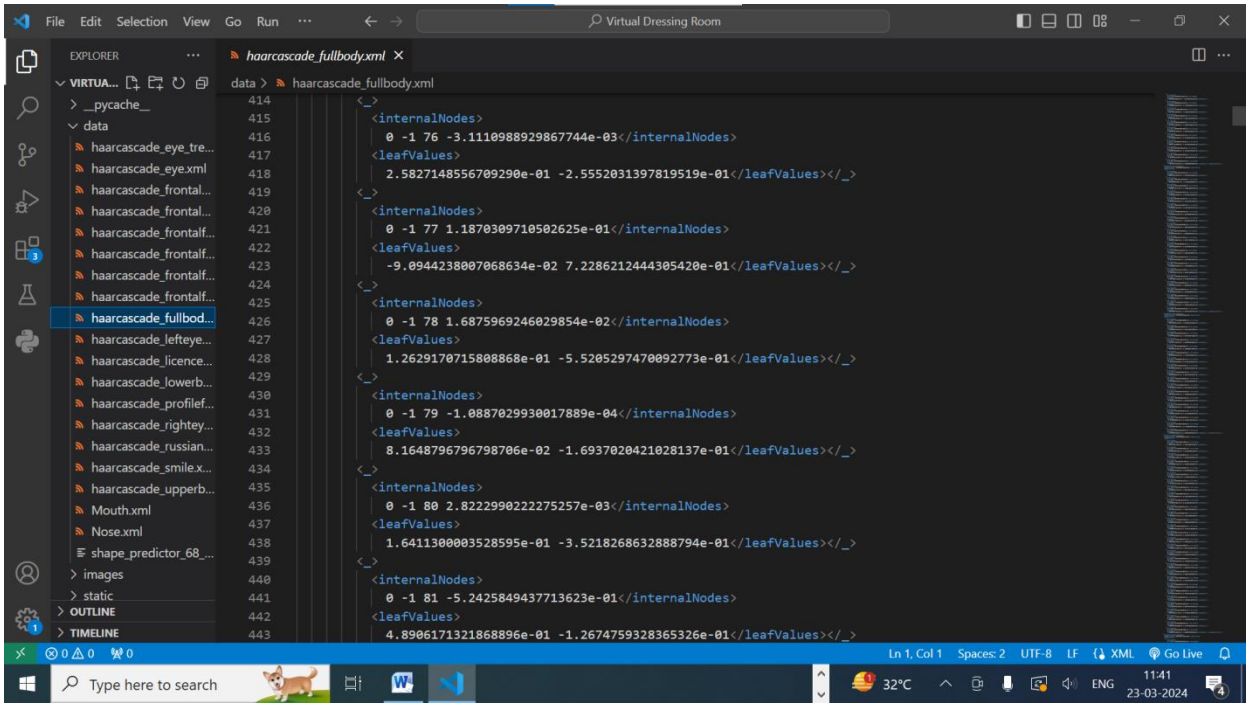


Fig 5.24

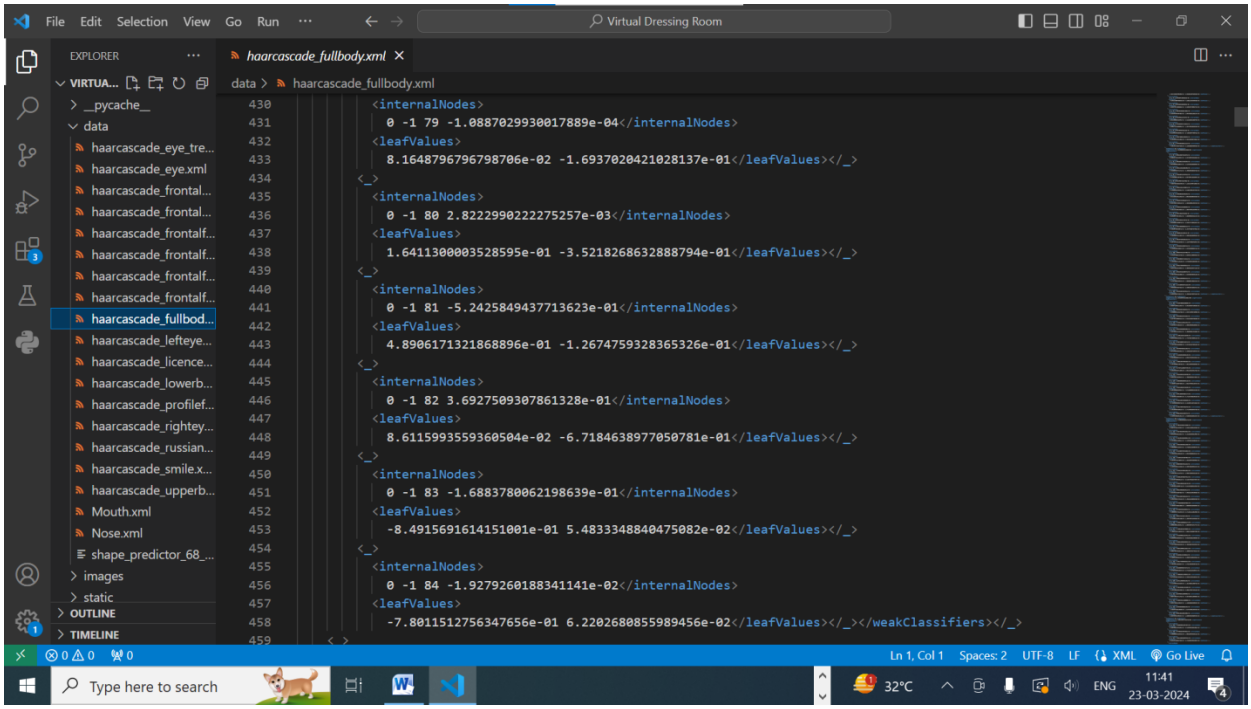
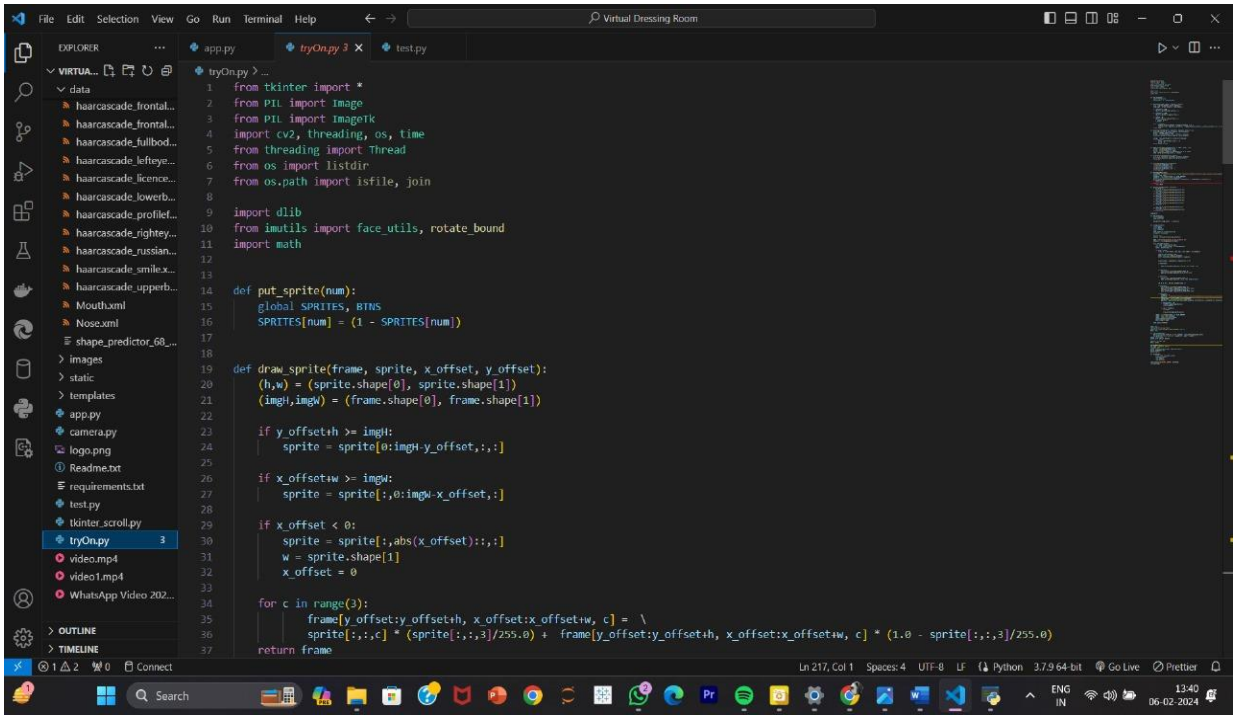


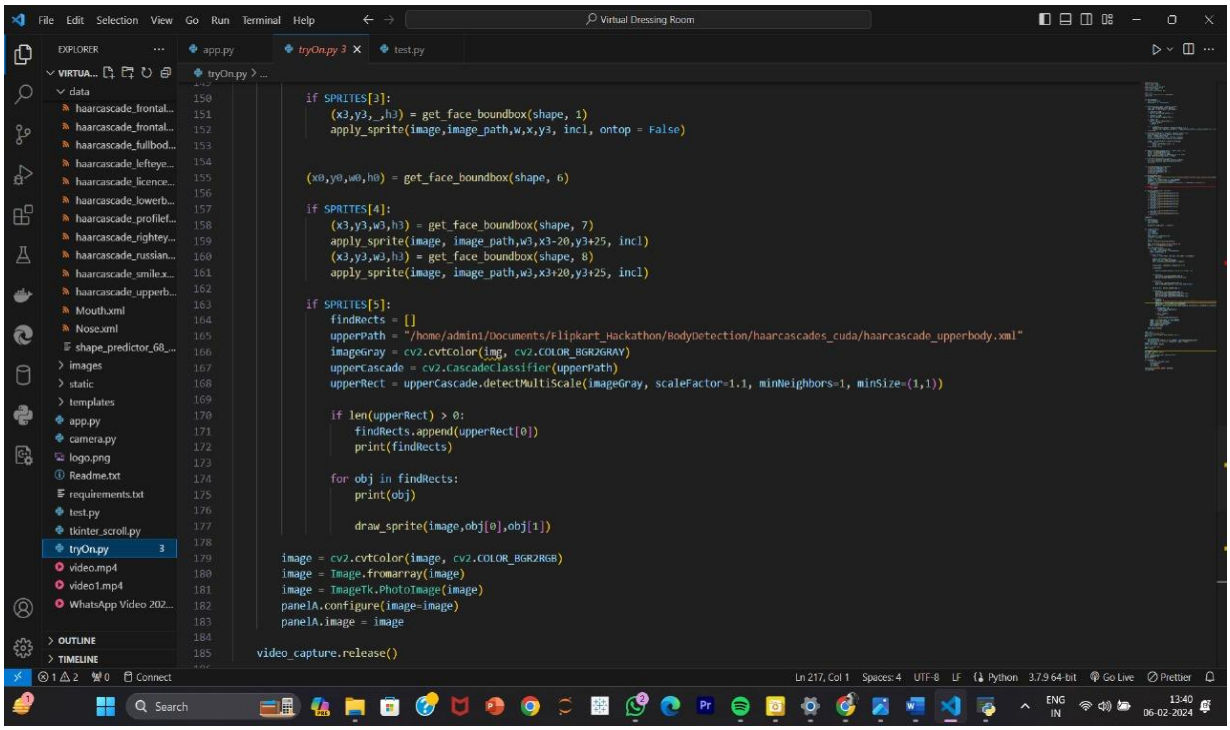
Fig 5.25

testOn.py



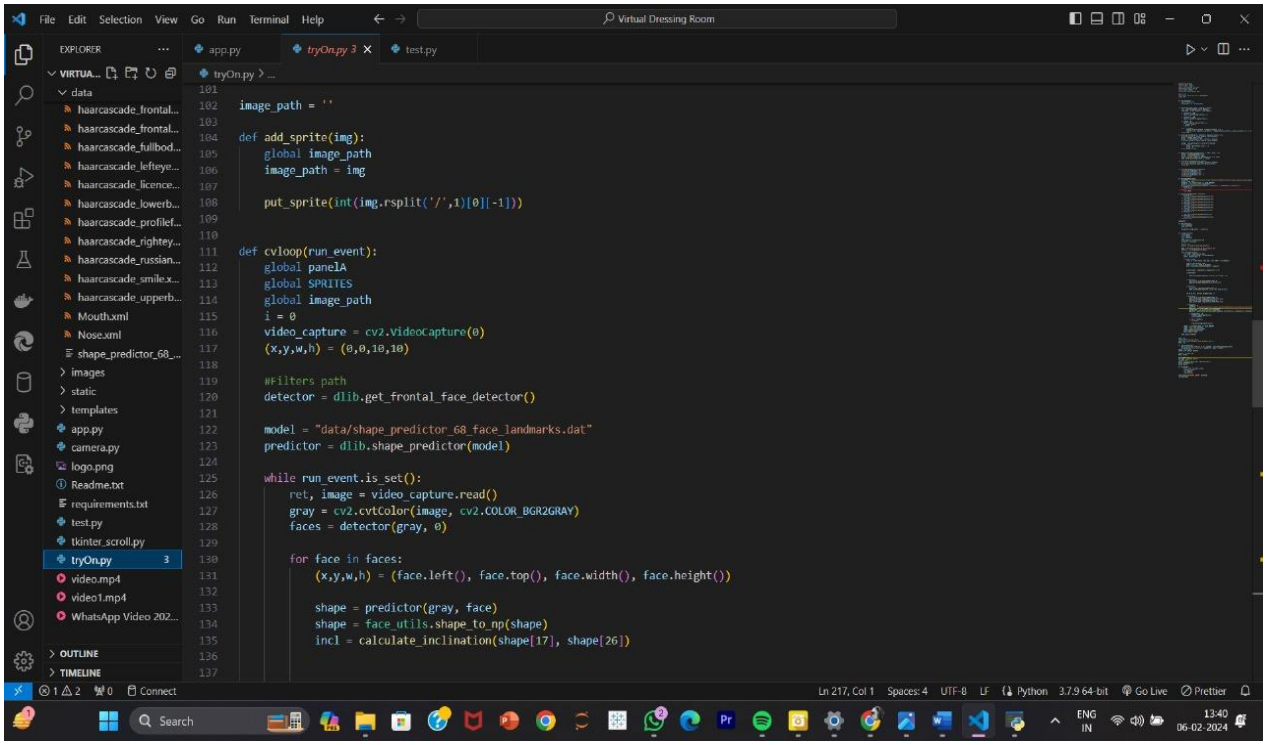
```
from tkinter import *
from PIL import Image
from PIL import ImageTk
import cv2, threading, os, time
from threading import Thread
from os import listdir
from os.path import isfile, join

import dlib
from imutils import face_utils, rotate_bound
import math


def put_sprite(num):
    global SPRITES, BTNS
    SPRITES[num] = (1 - SPRITES[num])


def draw_sprite(frame, sprite, x_offset, y_offset):
    (h,w) = (sprite.shape[0], sprite.shape[1])
    (imgH,imgW) = (frame.shape[0], frame.shape[1])

    if y_offset+h >= imgH:
        sprite = sprite[0:imgH-y_offset,:,:]

    if x_offset+w >= imgW:
        sprite = sprite[:,0:imgW-x_offset,:]

    if x_offset < 0:
        sprite = sprite[:,abs(x_offset)::,:]
        w = sprite.shape[1]
        x_offset = 0

    for c in range(3):
            frame[y_offset:y_offset+h, x_offset:x_offset+w, c] =  \
            sprite[:,:,c] * (sprite[:,:,3]/255.0) +  frame[y_offset:y_offset+h, x_offset:x_offset+w, c] * (1.0 - sprite[:,:,3]/255.0)
    return frame
```

Fig 5.26



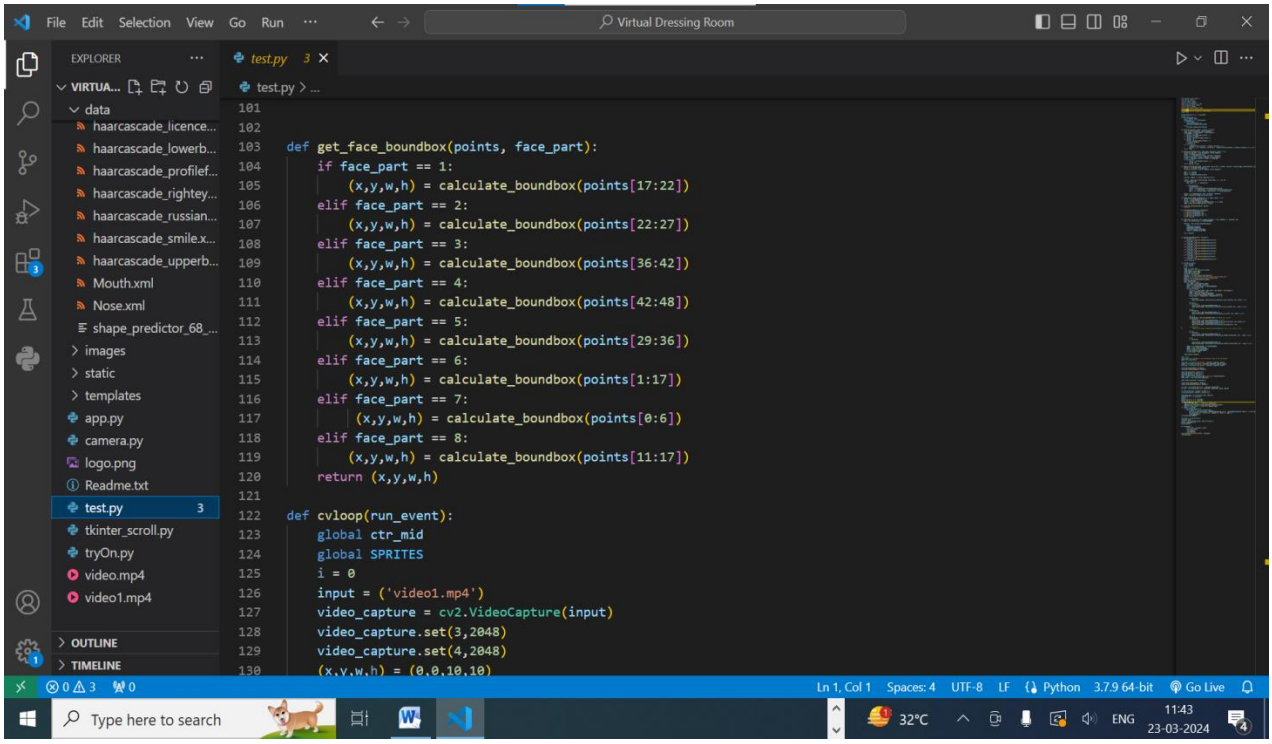
```
            if SPRITES[3]:
                (x3,y3,_,h3) = get_face_boundbox(shape, 1)
                apply_sprite(image,image_path,w,x,y3, incl, ontop = False)


            (x0,y0,w0,h0) = get_face_boundbox(shape, 6)

            if SPRITES[4]:
                (x3,y3,w3,h3) = get_face_boundbox(shape, 7)
                apply_sprite(image, image_path,w3,x3-20,y3+25, incl)
                (x3,y3,w3,h3) = get_face_boundbox(shape, 8)
                apply_sprite(image, image_path,w3,x3+20,y3+25, incl)

            if SPRITES[5]:
                findRects = []
                upperPath = "/home/admin1/Documents/Flipkart_Hackathon/BodyDetection/haarcascades_cuda/haarcascade_upperbody.xml"
                imageGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                upperCascade = cv2.CascadeClassifier(upperPath)
                upperRect = upperCascade.detectMultiScale(imageGray, scaleFactor=1.1, minNeighbors=1, minSize=(1,1))

                if len(upperRect) > 0:
                    findRects.append(upperRect[0])
                    print(findRects)

                for obj in findRects:
                    print(obj)

                    draw_sprite(image,obj[0],obj[1])

        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image = Image.fromarray(image)
        image = ImageTk.PhotoImage(image)
        panelA.configure(image=image)
        panelA.image = image

    video_capture.release()
```

Fig 5.27



```
image_path = ''

def add_sprite(img):
    global image_path
    image_path = img

    put_sprite(int(img.rsplit('/',1)[0][-1]))


def cvloop(run_event):
    global panelA
    global SPRITES
    global image_path
    i = 0
    video_capture = cv2.VideoCapture(0)
    (x,y,w,h) = (0,0,10,10)

    #Filters path
    detector = dlib.get_frontal_face_detector()

    model = "data/shape_predictor_68_face_landmarks.dat"
    predictor = dlib.shape_predictor(model)

    while run_event.is_set():
        ret, image = video_capture.read()
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        faces = detector(gray, 0)

        for face in faces:
            (x,y,w,h) = (face.left(), face.top(), face.width(), face.height())

            shape = predictor(gray, face)
            shape = face_utils.shape_to_np(shape)
            incl = calculate_inclination(shape[17], shape[26])
```

Fig 5.28



```
def get_face_boundbox(points, face_part):
    if face_part == 1:
        (x,y,w,h) = calculate_boundbox(points[17:22])
    elif face_part == 2:
        (x,y,w,h) = calculate_boundbox(points[22:27])
    elif face_part == 3:
        (x,y,w,h) = calculate_boundbox(points[36:42])
    elif face_part == 4:
        (x,y,w,h) = calculate_boundbox(points[42:48])
    elif face_part == 5:
        (x,y,w,h) = calculate_boundbox(points[29:36])
    elif face_part == 6:
        (x,y,w,h) = calculate_boundbox(points[1:17])
    elif face_part == 7:
        (x,y,w,h) = calculate_boundbox(points[0:6])
    elif face_part == 8:
        (x,y,w,h) = calculate_boundbox(points[11:17])
    return (x,y,w,h)

def cvloop(run_event):
    global ctr_mid
    global SPRITES
    i = 0
    input = ('video1.mp4')
    video_capture = cv2.VideoCapture(input)
    video_capture.set(3,2048)
    video_capture.set(4,2048)
    (x,y,w,h) = (0,0,10,10)
```

Fig 5.29

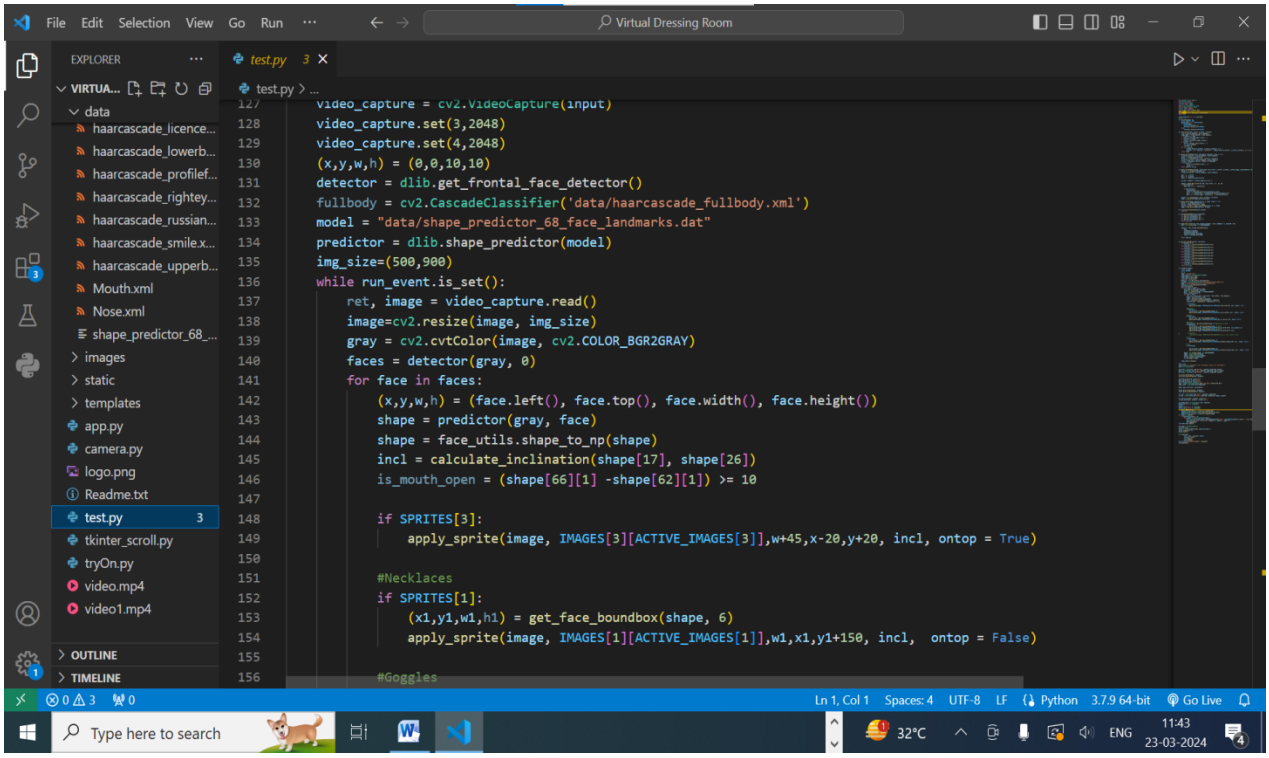


Fig 5.30

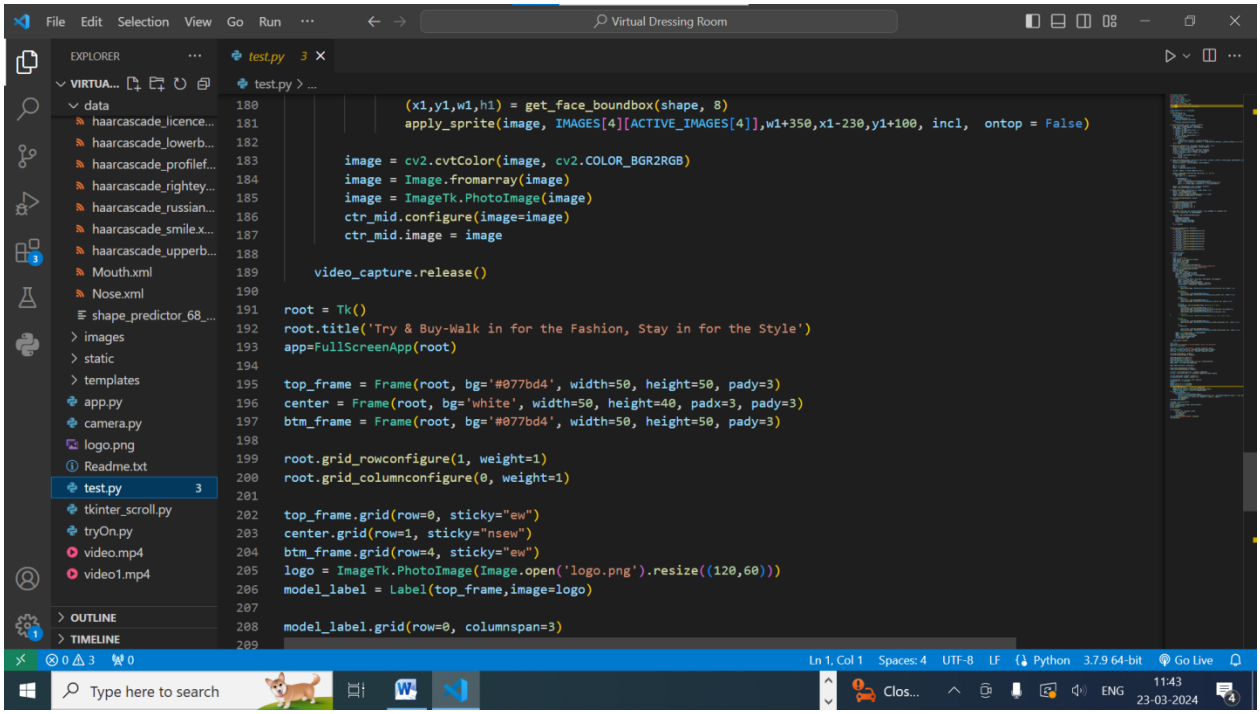


Fig 5.31

RESULT

Output:

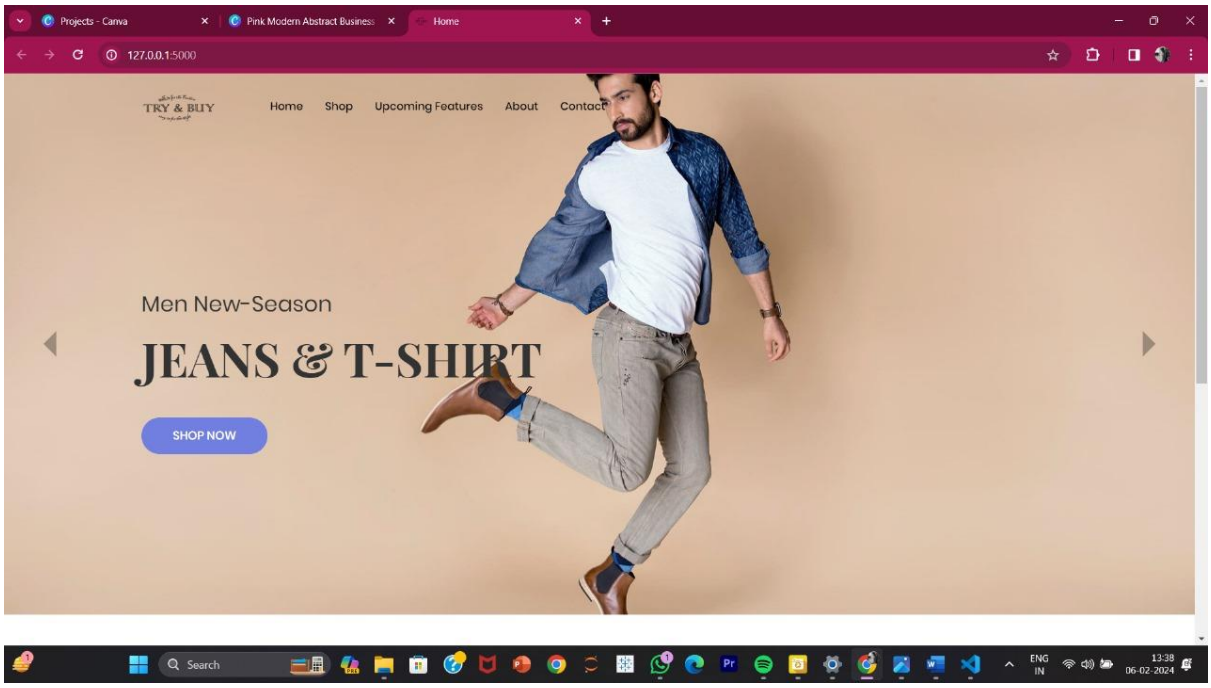


Fig 5.32

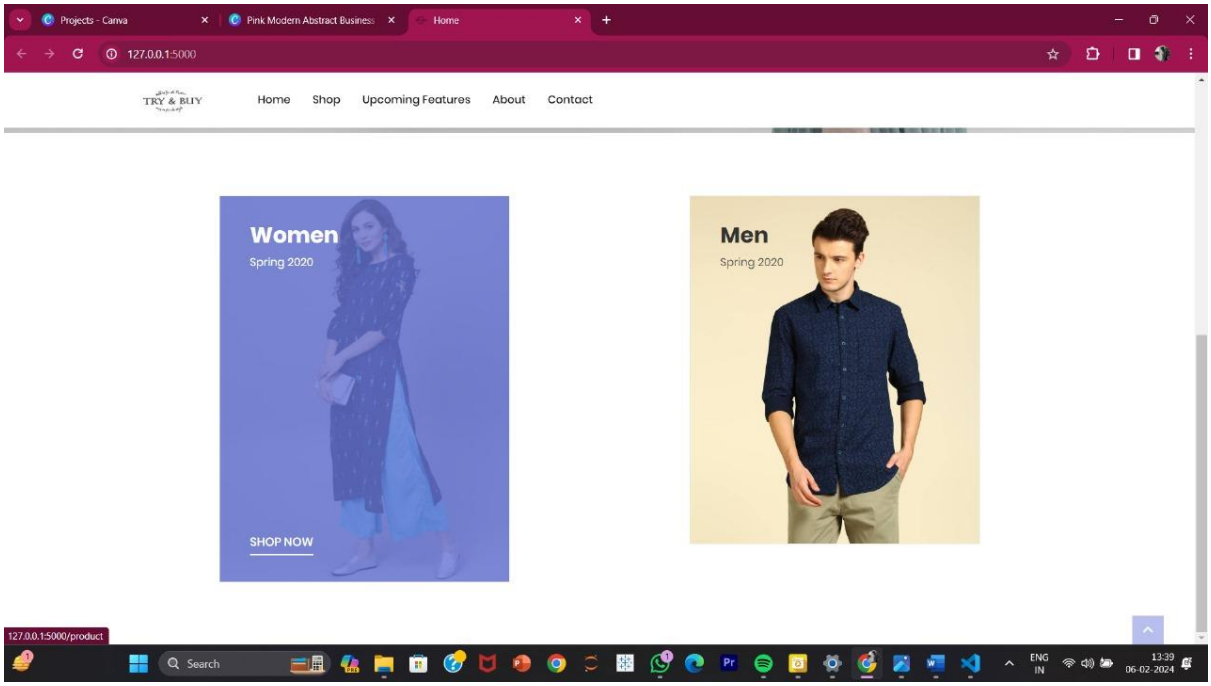


Fig 5.33

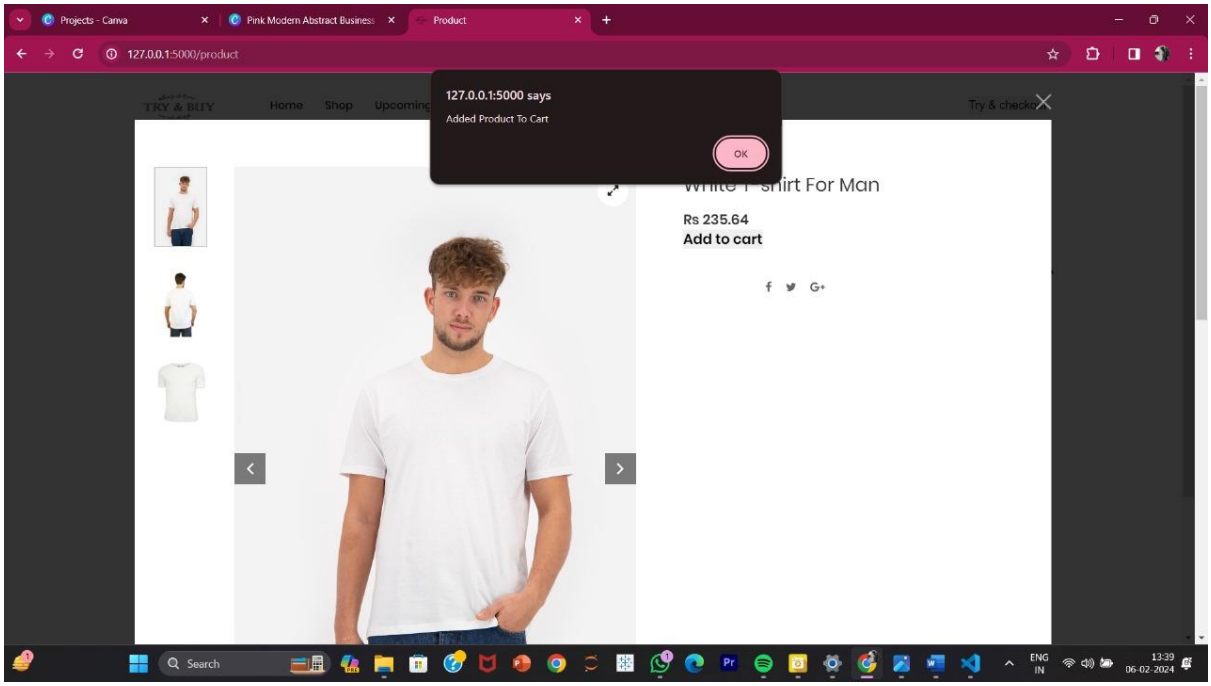
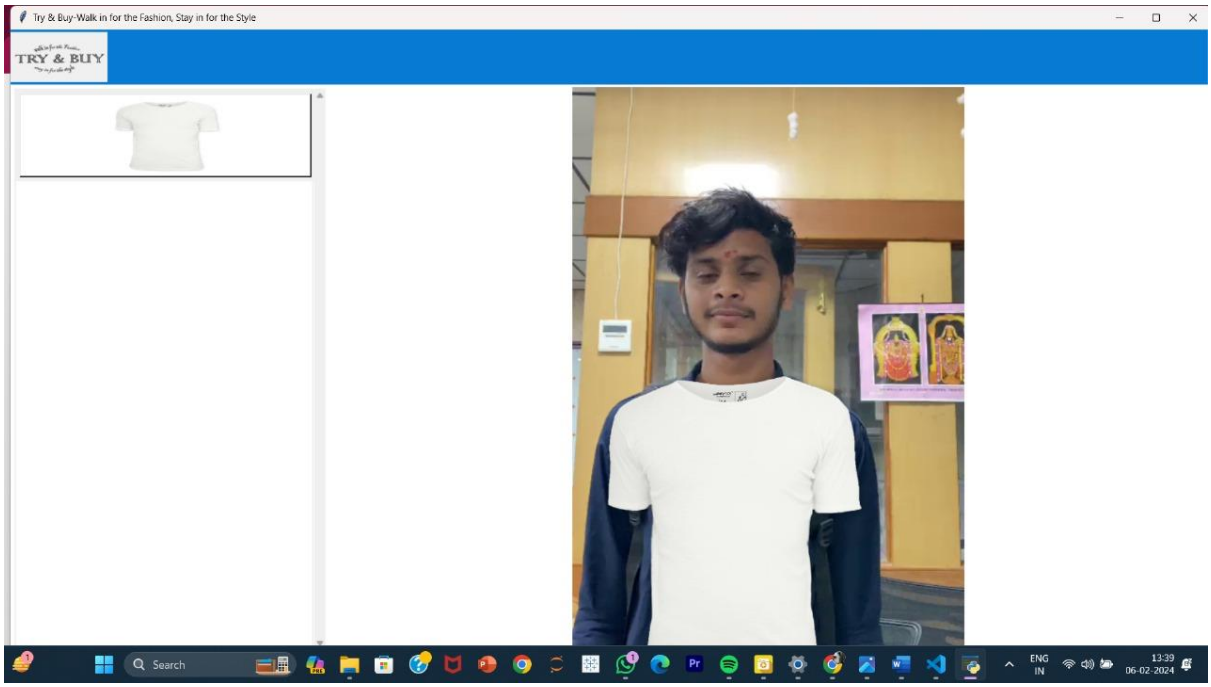


Fig 5.34

FINAL RESULT:



CHAPTER 6-CONCLUSION

In conclusion, the Virtual Fitting Room (VFR) presented in this paper, powered by deep learning neural networks, haarcascade and OpenCV, signifies a significant stride towards revolutionizing the online apparel shopping experience. The synergistic use of Convolutional Neural Networks for precise pose estimation and semantic segmentation, coupled with the versatile computer vision capabilities of OpenCV, creates a system that addresses longstanding challenges in the virtual try-on domain. The advantages of an immersive, accurate, and visually appealing virtual fitting experience contribute to increased user confidence, ultimately fostering a more enjoyable and satisfying online shopping journey. As technology continues to advance, the proposed VFR not only showcases the potential of deep learning and computer vision in the retail sector but also serves as a testament to the transformative impact these innovations can have on shaping the future of digital fashion retail. The ongoing refinement of such systems, coupled with collaborative efforts between researchers, developers, and retailers, holds the promise of further enriching the online shopping landscape and elevating customer satisfaction to new heights.

BIBLIOGRAPHY

[1] K.Srinivasan, K.Porkumaran, G.Sai Narayanan, "Intelligent human body tracking, modelling and activity analysis of video surveillence system:A Survey",Journal of convergence in engineering, technology and science, Vol.1,pp.1-8,2009.

[2] Max Mignotte,"Segmentation by Fusion of Histogram based K-Means Clusters in different color space"IEEE Transactions on Image Processing, Vol.17,pp.780-787,2008.

[3] F. Cordier, W. Lee, H. Seo, and N. Magnenat-Thalmann, "From 2D Photos of Yourself to Virtual Try-on Dress on the Web", Springer, pp. 31–46,2001.

[4] D. Protopsaltou, C. Luible, M. Arevalo-Poizat, and N. MagnenatThalmann, "A body and garment creation method for an internet based virtual fitting room", in Proc. Computer Graphics International 2002 (CGI '02). Springer, pp. 105–122,2002.

[5] F. Cordier, H. Seo, and N. Magnenat-Thalmann, "Made-tomeasure technologies for an online clothing store",IEEE Comput. Graph. Appl., vol. 23, no. 1, pp. 38–48, Jan. 2003.

[6] K.Srinivasan, K.Porkumaran, G.Sai Narayanan,"Skin colour segmentation based 2D and 3D human pose modelling using Discrete

Wavelet Transform",Journal of Pattern recognition and image Analysis,Springer,Vol.21,pp.740-753,2011.

[7] R. Brouet, A. Sheffer, L. Boissieux, and M.-P. Cani, "Design preserving garment transfer",ACM Trans. Graph., vol. 31,No. 4, pp. 36:1–36:11, Jul. 2012.

[8] W. Xu, N. Umentani, Q. Chao, J. Mao, X. Jin, and X. Tong,"Sensitivityoptimized rigging for example-based real-time clothing synthesis",ACM Trans. Graph. (Proc. of SIGGRAPH 2014), Vol. 33, No. 4, Aug. 2014.

[9] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan, "Scanning 3D full human bodies using kinects",IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE Virtual Reality), Vol. 18, No. 4, pp. 643–650, 2012.

[10] J. Ehara and H. Saito, "Texture overlay for virtual clothing based on pca of silhouettes", in Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality, ser. ISMAR '06. IEEE Computer Society, pp.139–142,2006