



# **CUSTOM OUTFIT GENERATOR WITH VIRTUAL TRY-ON**



## **A DESIGN PROJECT REPORT**

*Submitted by*

**ARCHANA R S**

**KAVITHA A**

**THAMILARASI B**

**SIVA RANJANI S**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

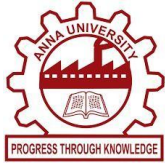
**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

**K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

**SAMAYAPURAM – 621 112**

**JUNE, 2025**



# **CUSTOM OUTFIT GENERATOR WITH VIRTUAL TRY-ON**



## **A DESIGN PROJECT REPORT**

*Submitted by*

**ARCHANA R S (811722001003)**

**KAVITHA A (811722001023)**

**THAMILARASI B (811722001051)**

**SIVA RANJANI S (811722001501)**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

**K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

**SAMAYAPURAM – 621 112**

**JUNE, 2025**

**K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY**  
**(AUTONOMOUS)**  
**SAMAYAPURAM – 621 112**

**BONAFIDE CERTIFICATE**

Certified that this design project report titled “**CUSTOM OUTFIT GENERATOR WITH VIRTUAL TRY-ON**” is the bonafide work of **ARCHANA R S (811722001003), KAVITHA A (811722001023), THAMILARASI B (811722001051), SIVA RANJANI S (811722001501)**, who carried out the design project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other design project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**

Dr. T.Avudaiappan, M.E., Ph.D.,

**HEAD OF THE DEPARTMENT**

Department of Artificial Intelligence  
K.Ramakrishnan College of Technology  
(Autonomous)  
Samayapuram – 621 112

**SIGNATURE**

Mr. A. Joshua Issac, M.E., (Ph.D)

**SUPERVISOR**

ASSISTANT PROFESSOR  
Department of Artificial Intelligence  
K.Ramakrishnan College of Technology  
(Autonomous)  
Samayapuram – 621 112

Submitted for the viva-voce examination held on .....

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## DECLARATION

We jointly declare that the design project report on “**CUSTOM OUTFIT GENERATOR WITH VIRTUAL TRY-ON**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This design project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF TECHNOLOGY**.

**Signature**

---

ARCHANA R S

---

KAVITHA A

---

THAMILARASI B

---

SIVA RANJANI S

Place: Samayapuram

Date:

## ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this design project.

We are glad to credit honourable chairman **Dr. K.RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our design project and offering adequate duration in completing our design project.

We would like to thank **Dr. N. VASUDEVAN, M.E., Ph.D.**, Principal, who gave opportunity to frame the design project the full satisfaction.

We whole heartily thank **Dr. T.AVUDAIAPPAN, M.E., Ph.D.**, Head of the department, **ARTIFICIAL INTELLIGENCE** for providing his encourage pursuing this design project.

We express our deep and sincere gratitude to our design project guide **Mr. A. JOSHUA ISSAC, M.E., (Ph.D)** Department of **ARTIFICIAL INTELLIGENCE**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this design project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

## **ABSTRACT**

The “Custom Outfit Generator with Virtual Try-On” is an innovative AI-powered fashion system that enhances the online shopping experience through prompt-based outfit creation and realistic virtual try-on. Utilizing the capabilities of Generative AI, users can simply enter a text prompt describing their desired outfit style. The system then generates a unique fashion design based on the prompt and allows users to upload their own image to visualize how the outfit would look on them. This solution addresses major challenges in online shopping, particularly the lack of personalization and difficulty in assessing clothing fit. By combining creative design automation with user-specific visualizations, the platform significantly boosts shopping confidence, reduces return rates, and delivers a highly personalized fashion experience.

## TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
	<b>ABSTRACT</b>	<b>v</b>
	<b>LIST OF FIGURES</b>	<b>viii</b>
	<b>LIST OF ABBREVIATIONS</b>	<b>ix</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 OVERVIEW	1
	1.2 OBJECTIVE	2
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>3</b>
	2.1 OUTFIT ANYONE: ULTRA-HIGH QUALITY VIRTUAL TRY-ON FOR ANY CLOTHING AND ANY PERSON	3
	2.2 FASHION SD-X: MULTIMODAL FASHION GARMENT SYNTHESIS USING LATENT DIFFUSION	4
	2.3 MM TRYON: MULTI-MODAL MULTI-REFERENCE CONTROL FOR HIGH-QUALITY FASHION	5
	2.4 FASHION-VDM: VIDEO DIFFUSION MODEL FOR VIRTUAL TRY- ON	6
	2.5 DRESS CODE: AUTOREGRESSIVELY SEWING AND GENERATING GARMENTS FROM TEXT	7
<b>3</b>	<b>SYSTEM ANALYSIS</b>	<b>8</b>
	3.1 EXISTING SYSTEM	8
	3.1.1 Demerits	9

	3.2	PROPOSED SYSTEM	10
	3.2.1	Merits	10
<b>4</b>		<b>SYSTEM SPECIFICATIONS</b>	<b>12</b>
	4.1	HARDWARE SPECIFICATIONS	12
	4.2	SOFTWARE SPECIFICATIONS	12
<b>5</b>		<b>SYSTEM DESIGN</b>	<b>13</b>
	5.1	FLOW DIAGRAM	13
<b>6</b>		<b>MODULES DESCRIPTION</b>	<b>16</b>
	6.1	WEBCAM OR IMAGE CAPTURE MODULE	16
	6.2	USER INTERACTION MODULE	19
	6.3	POSE ESTIMATION MODULE	22
	6.4	OUTFIT GENERATION MODULE	25
<b>7</b>		<b>CONCLUSION AND FUTURE ENHANCEMENT</b>	<b>30</b>
	7.1	CONCLUSION	30
	7.2	FUTURE ENHANCEMENT	31
		<b>APPENDIX A SOURCE CODE</b>	<b>32</b>
		<b>APPENDIX B SCREENSHOTS</b>	<b>37</b>
		<b>REFERENCES</b>	<b>39</b>



## **LIST OF FIGURES**

<b>FIGURE NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
5.1	Flow Diagram	13
6.1	Webcam or Image Capture	18
6.2	User Interaction	21
6.3	Pose Estimation - Skeleton Mapping	25
B.1	Sample Output - 1	37
B.2	Sample Output - 2	37
B.3	Sample Output - 3	38

## LIST OF ABBREVIATIONS

AI	-	Artificial Intelligence
API	-	Application Programming Interface
AR	-	Augmented Reality
BMP	-	Bitmap Image File
CLIP	-	Contrastive Language–Image Pretraining
CPU	-	Central Processing Unit
FID	-	Fréchet Inception Distance
GAN	-	Generative Adversarial Network
GPU	-	Graphics Processing Unit
JPG	-	Joint Photographic Experts Group
KID	-	Kernel Inception Distance
PBR	-	Physically Based Rendering
PNG	-	Portable Network Graphics
UI	-	User Interface
VAE	-	Variational Autoencoder
VTO	-	Virtual Try-On

# CHAPTER 1

## INTRODUCTION

### 1.1 OVERVIEW

Custom Outfit Generator with Virtual Try-On presents an innovative solution in the realm of personalized fashion using artificial intelligence. It integrates advanced deep learning models and computer vision to allow users to virtually try on AI-generated outfits. By simply uploading a photo or using a webcam, users can see themselves dressed in custom outfits generated in real time, offering a highly interactive and engaging experience.

Traditional virtual try-on systems are limited by their dependence on pre-scanned clothing models and lack of real-time interactivity. This system overcomes those limitations by using pose estimation through MediaPipe and generative models like Stable Diffusion enhanced with ControlNet. The result is a seamless and flexible virtual try-on platform that adapts to any user pose and style preference, providing an immersive visualization of how different clothing styles look on the individual.

The overall design includes a user-friendly interface built with Gradio, supporting inputs such as gender and descriptive outfit prompts. This makes the system not only technologically advanced but also practical and accessible to a wide range of users. It finds applications in online retail, fashion design, AR previews, and even social media styling—making it a powerful tool for the future of personalized fashion technology. This project redefines the way users interact with fashion through real-time AI-generated try-ons. It lays the groundwork for smarter, more immersive, and accessible fashion technology.

## 1.2 OBJECTIVE

The main objective is to build an AI-driven platform that enables users to generate and visualize custom outfits on their own images in real-time. The system empowers users to input descriptive prompts such as clothing type, color, or style, which are then used by generative AI to create outfit visuals tailored to their appearance. This brings a new level of personalization and creativity to virtual fashion experiences.

Another key goal is to ensure that the generated outfits align accurately with the user's body posture and orientation. This is achieved through the use of MediaPipe for extracting skeletal keypoints from the uploaded or captured image, which are then used as structural guidance in the generation process via ControlNet. This pose-aware approach significantly improves realism and enhances the user's confidence in the virtual try-on results.

Finally, the project aims to offer a modular, scalable, and easy-to-use system that can be expanded or integrated into existing platforms such as online shopping apps or virtual wardrobe assistants. By relying entirely on free, open-source tools like OpenCV, Gradio, and Stable Diffusion, the system is both cost-effective and adaptable, encouraging further research and development in AI-aided fashion technology. The objective is to deliver a personalized, accurate, and efficient outfit visualization system. It aims to transform how users explore and experience fashion using AI.

## CHAPTER 2

### LITERATURE SURVEY

#### 2.1 OUTFIT ANYONE: ULTRA-HIGH QUALITY VIRTUAL TRY-ON FOR ANY CLOTHING AND ANY PERSON

Ke sun, Jian cao, Qi wang, Linrui Tian, Xindi Zhang, and Lian Zhou

Outfit Anyone introduces a cutting-edge virtual try-on system that generates ultra-high-quality images of individuals wearing any clothing item. This versatility underscores its potential for real-world deployment in the fashion and e-commerce industries.

##### Merits

- **Exceptional Realism:** Delivers photorealistic virtual try-on results, preserving intricate garment details and textures.
- **Robust Versatility:** Supports virtual try-on for any individual, regardless of body shape, pose, or background complexity.
- **Flexible Control Mechanisms:** Incorporates various pose and body shape guidance methods, including OpenPose, SMPL, and DensePose.

##### Demerits

- **User Authentication Concerns:** Currently, users cannot upload their own photos, raising concerns about potential misuse.
- **Accuracy Limitations:** The AI model may not always accurately render clothing items on a person's image.

## 2.2 FASHION SD-X: MULTIMODAL FASHION GARMENT SYNTHESIS USING LATENT DIFFUSION

**Abhishek Kumar Singh, Ioannis pataras**

Fashion SD-X introduces a novel generative pipeline that employs latent diffusion models, enhanced with ControlNet and LoRA fine-tuning, to synthesize high-quality fashion garment images from multimodal inputs like text and sketches. By extending existing virtual try-on datasets with sketch data, the model demonstrates superior performance over traditional stable diffusion models, as evidenced by metrics such as FID, CLIP Score, and KID.

### **Merits**

- **Multimodal Input Integration:** Combines text and sketch inputs to generate detailed and realistic fashion garment images.
- **Enhanced Performance Metrics:** Outperforms traditional stable diffusion models in FID, CLIP Score, and KID evaluations.
- **Dataset Augmentation:** Extends existing virtual try-on datasets by incorporating sketch data, enriching the training material

### **Demerits**

- **Sketch Dependency:** The quality of generated images heavily relies on the clarity and detail of input sketches.
- **Data Diversity Limitations:** Performance may be affected by the variety and quality of training data, potentially limiting generalization.

## 2.3 MM TRYON: MULTI-MODAL MULTI-REFERENCE CONTROL FOR HIGH-QUALITY FASHION

Xujie Zhang, Ente Lin, Xiu Li, Yuxuan Luo, Michael Kampffmeyer

MM Tryon presents a novel approach for high-quality fashion generation integrating multi-modal inputs and multi-reference controls. This method allows for the generation of highly accurate and customizable fashion designs by considering diverse reference sources, such as images, text, and user preferences.

### Merits

- **High Customization:** Offers personalized fashion generation using multi-modal inputs.
- **Visual Fidelity:** Produces high-quality, realistic fashion items with intricate details.
- **Flexible Input Handling:** Supports diverse types of reference data, enhancing adaptability across scenarios.

### Demerits

- **Training Complexity:** Requires large datasets and sophisticated training, making it resource-intensive.
- **Fashion Limitations:** May not generate innovative or rare fashion styles outside of the training data.

## 2.4 FASHION-VDM: VIDEO DIFFUSION MODEL FOR VIRTUAL TRY-ON

**Rajesh Kumar Butteddi, Srija Buttedd**

Fashion-VDM introduces a video diffusion model that enables virtual try-on experiences by synthesizing realistic videos of users trying on various outfits. By leveraging video generation techniques and high-quality image pairings, it improves the traditional still-image-based try-ons. The approach enhances user interaction and personalization by creating dynamic, life like virtual fitting rooms.

### **Merits**

- **Realistic Video Output:** Generates lifelike video try-ons, offering a dynamic and immersive experience.
- **Personalized Experience:** Adapts to users' body shape and movements, providing a customized virtual fitting room.
- **Enhanced User Engagement:** Video-based interaction increases user interest and time spent in virtual try-on.

### **Demerits**

- **High Computational Cost:** Video generation requires significant computational resources, limiting accessibility on low-end devices.
- **Limited Garment Diversity:** Struggles with less common garment types or complex materials.



## 2.5 DRESS CODE: AUTOREGRESSIVELY SEWING AND GENERATING GARMENTS FROM TEXT

**Kai He, Kaixin Yao, Qixuan Zhang, Jingyi Yu, Lingjie Liu**

This paper presents Dress Code, a novel system that uses natural language descriptions to generate 3D garments with realistic sewing patterns and textures. It combines a GPT-based model such as SewingGPT with a custom Stable Diffusion model for fabric appearance. The approach allows users to generate customizable clothing aligned closely with their design preferences.

### **Merits**

- **High Customization:** Users can create diverse designs using natural language inputs.
- **Realistic Output:** Generates physically plausible sewing patterns and PBR textures.
- **Flexible Framework:** Can generalize to various garment types and styles effectively.

### **Demerits**

- **Requires Technical Resources:** High computational cost for training and rendering.
- **Limited Real-time Use:** The system may not support real-time interaction smoothly.

## **CHAPTER 3**

### **SYSTEM ANALYSIS**

#### **3.1 EXISTING SYSTEM**

Virtual try-on technology has gained popularity in recent years, with several systems aiming to digitally simulate how clothing would look on a user. Prominent among these are VITON, TryOnGAN, and CP-VTON. These systems typically rely on deep learning techniques such as Generative Adversarial Networks and image warping to overlay garments onto images of people.

While these systems mark significant technological progress, they share common architectural designs and limitations-most notably, their reliance on pre-scanned garments and limited clothing datasets, which restrict their flexibility and applicability in real-world or design-centric scenarios.

VITON uses a coarse-to-fine image warping strategy to project clothing onto 2D images of people. While it can generate moderately realistic visuals, its performance significantly degrades when dealing with unusual poses or partial occlusions. TryOnGAN improves visual realism by using GANs and learning person-to-clothing mappings but still depends on fixed garment templates. CP-VTON enhances pose handling using dense pose estimation and geometric warping modules but suffers from limited clothing personalization and lacks support for dynamic garment creation.

CP-VTON further enhances garment alignment through a combination of a clothing warping module and a try-on module. By integrating dense pose estimation, CP-VTON achieves better alignment between the garment and the body, even in varied poses. Despite these advancements, the system remains constrained in terms of garment customization and flexibility, particularly when dealing with new clothing styles not present in the training dataset.

### 3.1.1 Demerits

- **Lack of Real-Time Processing:** Existing systems like VITON and TryOnGAN are computationally intensive, leading to slow image generation unsuitable for live applications.
- **Dependency on Predefined Clothing Templates:** Reliance on pre-scanned garments and fixed clothing datasets limits user ability to upload or create new designs.
- **Difficulty Handling Complex Poses:** Traditional methods struggle with unusual body postures, occlusions, and varied viewpoints, causing unrealistic try-on results.
- **Limited Style Customization:** Minimal control over fabric type, sleeve style, fit, or design changes restricts personalization options.
- **Restricted Clothing Dataset:** Limited variety in training data hampers adaptability to new or unique fashion styles.
- **Lack of Dynamic Garment Creation:** Systems cannot generate or modify garments dynamically, limiting creativity for designers and users.
- **Poor Adaptation to Diverse Body Types:** Models are typically trained on standard body shapes, reducing accuracy and inclusivity for different body sizes and proportions.
- **Low Personalization:** Insufficient support for individual style preferences and personalized fitting reduces user engagement and satisfaction.

## 3.2 PROPOSED SYSTEM

The rapid evolution of artificial intelligence and computer vision has greatly influenced the fashion industry, particularly through the advent of virtual try-on systems. However, despite the remarkable progress made by existing solutions such as VITON, TryOnGAN, and CP-VTON, several critical challenges remain unresolved. These challenges have hindered the practical deployment of VTO systems in consumer-facing applications and limited their potential in creative industries such as fashion design. Therefore, there is a pressing need for a next-generation solution that transcends current limitations and delivers a seamless, intelligent, and highly personalized virtual try-on experience.

### 3.2.1 Merits

- **Real-Time Performance:** Delivers instant virtual try-on results using optimized neural networks, enabling smooth and responsive user interaction suitable for mobile apps, AR mirrors, and live fittings.
- **Dynamic Garment Creation:** Allows users to generate custom outfits on-the-fly using sketches, templates, or modular inputs - eliminating the need for pre-scanned garments and enabling full creative control.
- **Robustness to Pose Variations:** Incorporates advanced 3D pose estimation and body-aware modeling to ensure garments fit realistically across a wide range of human poses and body shapes.
- **Personalization and Style Customization:** Offers deep customization options such as changing fabric, color, patterns, and design features - empowering both designers and everyday users to create unique fashion looks.

- **User-Friendly and Intuitive Interface:** Features an interactive drag-and-drop UI with support for sketch-based design, modular component selection, and real-time feedback—making the system accessible to all skill levels.
- **High-Quality Visual Output:** Enhances visual realism through deep learning techniques like style transfer, semantic segmentation, and pose-guided synthesis for lifelike garment previews.
- **Scalability and Integration Potential:** Built to scale across platforms and designed for integration with E-commerce, virtual fashion events, and social media sharing-broadening its commercial and creative applications.
- **Inclusivity and Body Diversity Support:** The system accommodates various body shapes and sizes, promoting inclusivity by allowing users to see how outfits fit their unique physique-encouraging confidence and realistic expectations.
- **Eco-Friendly and Cost-Efficient Fashion Development:** Reduces the need for physical prototypes and samples, cutting down textile waste and production costs-making it an environmentally sustainable solution for designers and brands.
- **Secure and Private:** Ensures user data and images are processed securely or locally if required. Protects user identity and design privacy.

## **CHAPTER 4**

### **SYSTEM SPECIFICATIONS**

#### **4.1    HARDWARE SPECIFICATIONS**

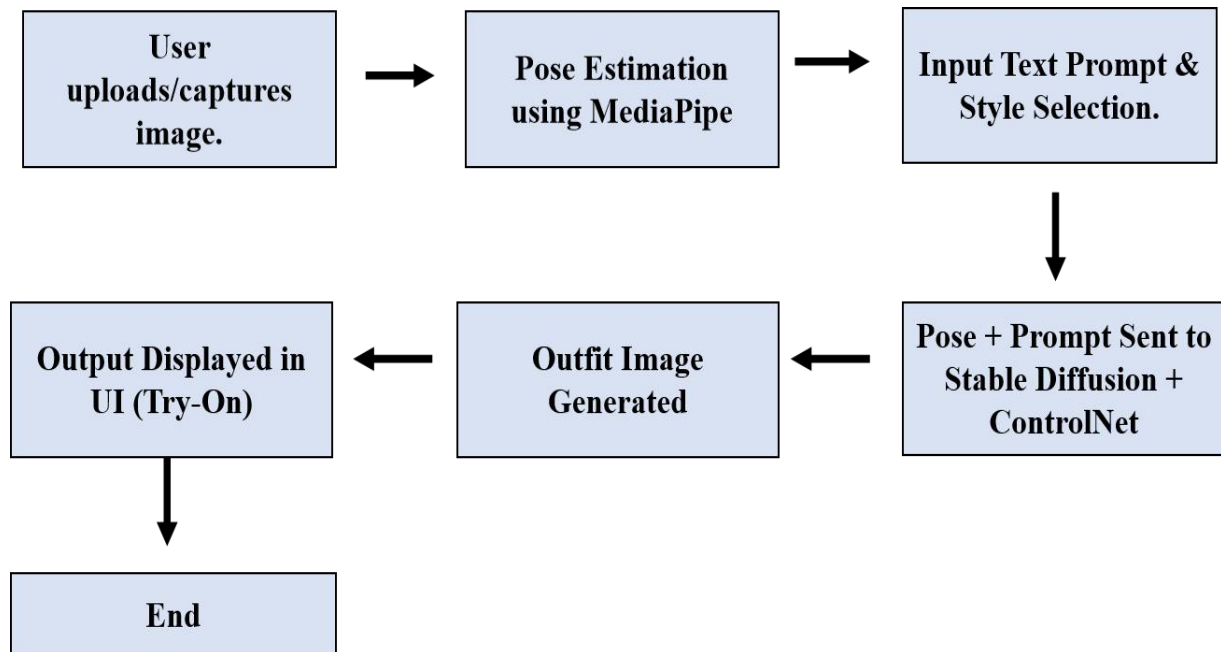
- Intel Core i5 or i7
- Minimum 8GB RAM
- GPU - Optional for Faster Generation

#### **4.2    SOFTWARE SPECIFICATIONS**

- Python 3.9+
- **Libraries:** diffusers, torch, numpy, mediapipe, PIL, gradio
- **Model Checkpoints:** Stable Diffusion v1.5, ControlNet, OpenPose, OpenCV

## CHAPTER 5

### SYSTEM DESIGN



**Fig. 5.1 Flow Diagram**

#### **User Uploads or Captures Image**

The system begins with the user either uploading an existing photo or capturing a new one using a webcam or mobile device. It plays a key role in maintaining high user engagement and satisfaction by simplifying complex back-end processes into easy to use tools.

- The image should contain the user in a clear frontal pose for accurate pose detection.
- This input image serves as the base for the virtual outfit to be overlaid later.
- Image preprocessing may include resizing, cropping, and background simplification to ensure quality input.

## Pose Estimation Using MediaPipe

Once the image is uploaded, it is passed to a pose estimation module using MediaPipe, a Google framework known for its real-time performance.

- MediaPipe extracts key body landmarks like shoulders, elbows, waist, and knees.
- These 2D keypoints define the skeletal structure of the user.
- The extracted pose data is crucial for ensuring garments align correctly with the body shape and orientation.

## Input Text Prompt & Style Selection

The user is then prompted to define their clothing preferences. This can be done in two ways:

- **Text Prompts:** Users describe the outfit they want using natural language. e.g., “red floral dress with puff sleeves”.
- **Style Components:** Alternatively, users can select individual garment components like sleeves, collars, fabrics, etc. from a visual interface.

This input defines the design intent for the outfit generation step.

## Pose + Prompt Sent to Stable Diffusion + ControlNet

The pose data and user prompt are combined and fed into a deep learning pipeline:

- **Stable Diffusion** is used for high-quality image generation from text prompts.
- **ControlNet** acts as a conditional model that guides generation based on pose keypoints, ensuring the outfit conforms to the body’s position and shape.
- The output of this stage is a realistic image of the user wearing the designed outfit.



## Outfit Image Generated

This module finalizes the synthetic outfit image.

- The AI-generated garment is composited onto the user's body with proper alignment and scaling.
- If the garment involves layering like jackets over shirts, semantic segmentation is applied for visual correctness.
- Additional techniques like shading, fabric texture rendering, and boundary smoothing enhance realism.

## Output Displayed in UI

The generated try-on image is presented in the user interface.

- Users can view, rotate if 3D avatars are used, and modify the design.
- Real-time editing is possible - users can change components, colors, or styles and get instant previews.
- Optional features: download, share on social media, or add to a virtual wardrobe.

## End

This is the final stage of the system where the output is shown and user interaction is enabled. It includes several sub-processes:

- **Try-On Output Displayed:** The generated outfit image is displayed in the user interface. Users can view how the custom-designed outfit looks on their own image or avatar.
- **Save & Download Options:** Users can save the try-on image to their local device or cloud storage. Option to download the outfit in different formats e.g., PNG, JPG.

## CHAPTER 6

### MODULES DESCRIPTION

#### 6.1 WEBCAM OR IMAGE CAPTURE MODULE

Webcam or Image capture module serves as the initial data acquisition point for the system, responsible for capturing visual input. It is designed to handle two primary input methods: real-time video streams from a connected webcam and static images uploaded by the user. Furthermore, the module performs essential pre-processing steps on the acquired visual data to prepare it for subsequent tasks, such as pose extraction and generation. By providing a standardized and pre-processed image input, this module ensures the efficiency and consistency of the downstream processing pipeline.

##### Tools, Libraries, and Frameworks

- **OpenCV:** Webcam interfacing, frame grabbing, image ops
- **Pillow:** PIL Image format conversions, saving, loading
- **NumPy:** Array operations, data conversion
- **Threading:** Multithreading for non-blocking frame capture

##### System Requirements and Functionalities

##### Real-Time Webcam Feed Capture

- Access the user's webcam device via OpenCV's cv2.VideoCapture API.
- Continuously grab frames with minimal delay.
- Allow users to pause and capture a specific frame for processing.
- Frame size is standardized to 640x480 or 720p depending on hardware to balance quality and speed.

##### Image Upload Interface

- Accept common image formats: JPEG, PNG, BMP.
- Validate image size and format.

- Convert uploaded images into a format compatible with downstream modules.
- Allow drag-and-drop or file picker for easy uploading.

### **Image Preprocessing Pipeline**

- **Colour space conversion:** OpenCV uses BGR by default; convert to RGB.
- **Resizing:** Uniform scaling to model's required input size e.g., 256x256 or 512x512 pixels.
- **Normalization:** Pixel values scaled to [0,1] or standardized as per pose estimation model needs.
- **Format conversion:** Convert OpenCV NumPy arrays to PIL Image objects for compatibility.
- **Optional histogram equalization:** For lighting correction.

### **User Feedback Loop**

- Display live webcam feed with overlaid messages e.g., "Press 'C' to capture".
- Show thumbnail previews of captured or uploaded images.
- Provide error messages for invalid inputs.

### **Working Mechanism**

#### **Step 1: Real-Time Image Capture**

- OpenCV initializes the webcam stream.
- A live preview is shown to the user.
- The user can freeze a frame using a capture button.

#### **Step 2: Image Upload**

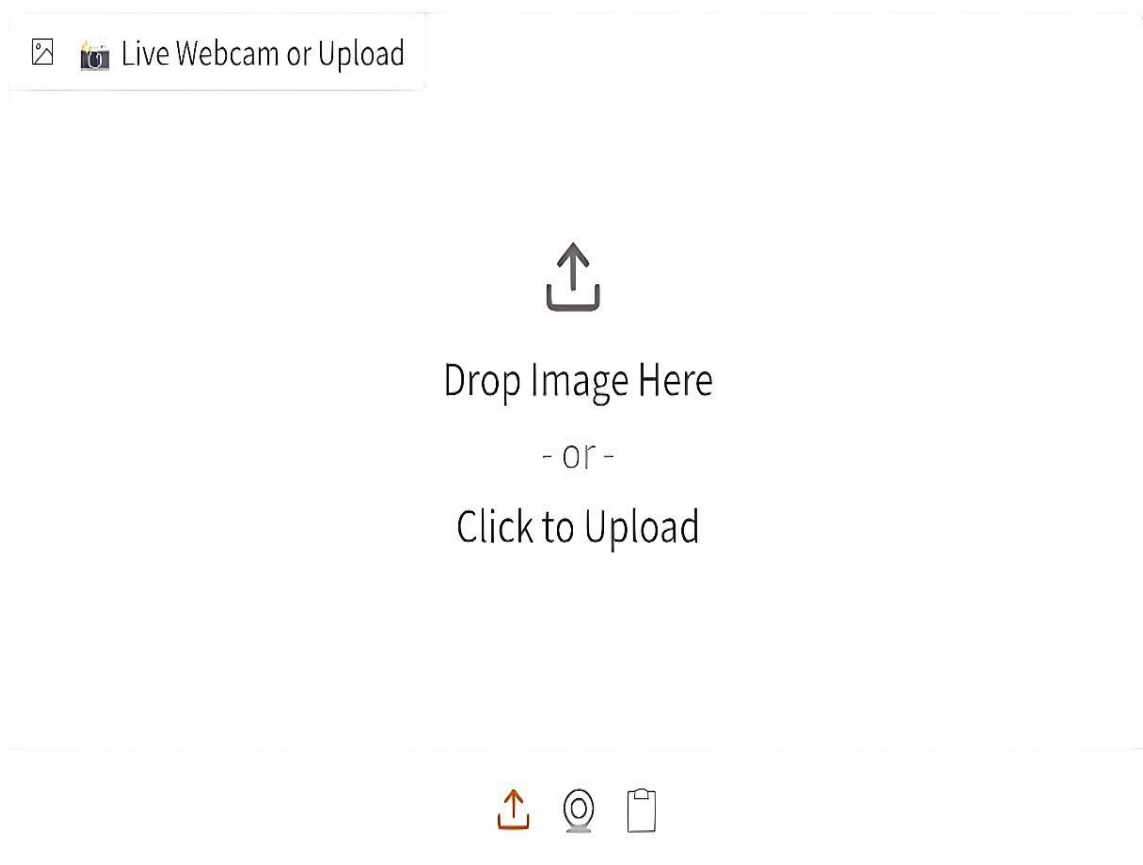
- Gradio or UI layer provides a file upload box.
- Once uploaded, the image is temporarily stored and passed to the preprocessing unit.

### Step 3: Image Preprocessing

- Converts image to RGB from BGR if captured by OpenCV.
- Resizes the image to 512×512 or other dimensions as required.
- Converts image to PIL format.
- Converts image to NumPy array for ML compatibility.

### Step 4: Output

- The cleaned and formatted image is forwarded to the Pose Estimation Module for landmark detection.



**Fig. 6.1 Webcam or Image Capture**

## 6.2 USER INTERACTION

The user interface for the "User Interaction" module is designed using the Gradio framework, which facilitates the creation of interactive web interfaces with minimal coding effort. Gradio offers a collection of pre-built UI components, such as dropdown menus and text input fields, that are utilized to gather necessary information from the user. For the purpose of collecting structured inputs like gender and outfit type, Gradio's dropdown components are employed. These components present users with a clear set of predefined options, ensuring data consistency and ease of selection. Within the Python code, these dropdowns are defined by specifying the available choices e.g., "Male," "Female" for gender, and various outfit categories. Labels are also assigned to these components to provide clear context to the user.

### Tools, Libraries, and Frameworks

- **Gradio UI Framework:** Fast, web-based interface for easy deployment and interaction
- **Python Backend:** Handles input parsing, prompt construction, and function calls to generative models

### System Requirements and Functionalities

#### Style Prompt Input

- Accepts open-ended text descriptions of desired clothing styles e.g., “casual summer dress with floral prints”.
- Utilizes natural language processing considerations to maintain clarity and relevance.
- Input length limited to avoid overly complex prompts that might degrade generation quality.

#### Action Buttons and Controls

- **Generate Outfit:** Starts the generation pipeline using the current user inputs.

- **Save Output:** Allows saving the generated outfit image locally or to cloud storage.
- **Reset:** Clears current inputs for a fresh start.
- **Live Preview Toggle:** Optionally enable or disable real-time outfit generation as inputs change.

## Prompt Construction and Formatting

### Prompt Example

In this system, user inputs are combined to create a descriptive and context-aware prompt for the outfit generation model. For instance, if the user provides the following:

- **Gender:** Female
- **Text Description:** Black business suit with heels

These elements are merged to form the final prompt: “A female model wearing a black business suit with heels in formal style suitable for winter season.”

Each input plays a specific role in the prompt: the gender defines the model’s appearance, the style determines the fashion aesthetic, the description details the clothing, and the season adds contextual background to influence the design and texture of the outfit.

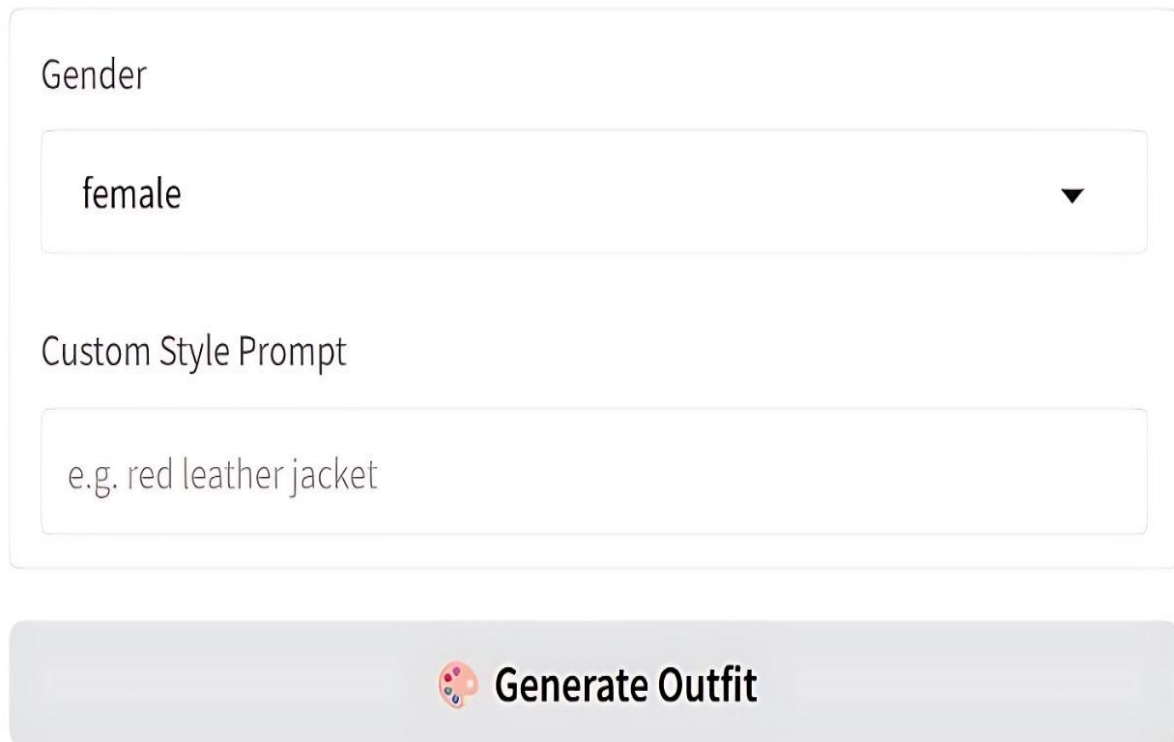
## Input Validation and Error Handling

- Text input length checks max 200 characters.
- Dropdown defaults set to valid options.
- Prevent submission if required fields are empty.
- Provide user feedback for invalid or missing inputs.

## User Experience

- As the user changes options, the prompt updates dynamically.
- Optional live outfit generation preview updates without manual trigger.

## Working Mechanism




The form is titled "Working Mechanism" and contains two input fields and a button. The first field is labeled "Gender" and has a dropdown menu with "female" selected. The second field is labeled "Custom Style Prompt" and contains the text "e.g. red leather jacket". Below these fields is a large grey button with a paint palette icon and the text "Generate Outfit".

Gender

female ▼

Custom Style Prompt

e.g. red leather jacket

 **Generate Outfit**

**Fig. 6.2 User Interaction**

### Step 1: Receiving Input

Users input:

- Text description of clothing e.g., "white kurta with blue jeans".
- Gender selection.
- Style selection like formal, party, etc.

### Step 2: Processing Input

- Inputs are validated e.g., empty string checks.
- Structured into a complete sentence like: "A male model wearing a white kurta with blue jeans in a formal style."

### **Step 3: Execution Handling**

- Final prompt is passed via Python functions to Stable Diffusion for generation.
- The ‘Generate’ button triggers the back-end function.
- The system sends:
  - The prompt.
  - Pose image from Module 3.
- Once generation is complete, the image is displayed back on the UI.

### **Integration with Other Modules**

- Sends fully formatted prompts to the Outfit Generation Module.
- Receives feedback or status from generation for UI updates.
- Works closely with Webcam or Image Capture for user identity context.

## **6.3 POSE ESTIMATION MODULE**

The Pose Estimation module forms a pivotal stage in the virtual try-on outfit generation pipeline, responsible for the precise extraction and representation of human skeletal pose from user-provided imagery. By employing advanced computer vision methodologies, this module analyzes the input image to identify and localize a comprehensive set of anatomical landmarks. These detected keypoints are subsequently transformed into a structured kinematic skeleton, providing essential spatial and articulatory information about the user's posture. This skeletal data acts as a critical conditioning input for the downstream Generative AI module, enabling the synthesis of virtual garments that conform realistically to the user's pose, ensuring accurate draping, alignment, and overall visual coherence within the virtual try-on experience. Pose estimation is the task of detecting and localizing key anatomical landmark such as the joints of the shoulders, elbows, wrists, hips, knees, and ankles-on the human body from 2D or 3D image data. These landmarks are then structured into a coherent skeletal representation, commonly referred to as a kinematic skeleton or pose skeleton.



## **Key System Requirements and Functionalities**

### **Key Pose Extraction**

- Detect key body landmarks including joints such as shoulders, elbows, wrists, hips, knees, ankles.
- Support detection of 33 keypoints following MediaPipe's BlazePose format.
- Provide 2D coordinates normalized relative to the image size.
- Calculate angles and vectors between joints to infer limb orientation.

### **Skeleton Map Generation**

- Convert keypoints into a skeleton or pose map image.
- Create a simplified line drawing or heatmap showing limb connectivity.
- This skeleton image will act as a spatial condition input via ControlNet for Stable Diffusion to generate pose-aware outfits.

### **Real-time Performance**

- Process images quickly to support live webcam feed scenarios.
- Lightweight yet accurate model to balance speed and precision.

## **Pose Detection Algorithms and Processing**

### **MediaPipe BlazePose Model**

- Uses a convolutional neural network to predict 33 2D landmarks.
- Each landmark: x, y, visibility or confidence.
- Landmarks include head, shoulders, elbows, wrists, hips, knees, ankles, and more.
- Provides robustness to occlusions and different body orientations.

## **Key point Post-processing**

- Normalize keypoints to image dimensions range 0-1.
- Filter out low-confidence points below a visibility threshold.
- Calculate vectors such as upper arm direction from shoulder to elbow.
- Calculate angles to understand limb rotation and bending.

## **Tools, Libraries, and Frameworks**

- **MediaPipe BlazePose:** Provides state-of-the-art pose landmark detection
- **OpenCV:** Image handling, preprocessing, visualization
- **NumPy:** Array operations, coordinate transformations
- **ControlNet:** Uses skeleton map as input conditioning

## **Working Mechanism**

### **Step 1: Input Handling**

- The module receives a preprocessed image from Module 1.
- Image is resized to fit MediaPipe's input needs.

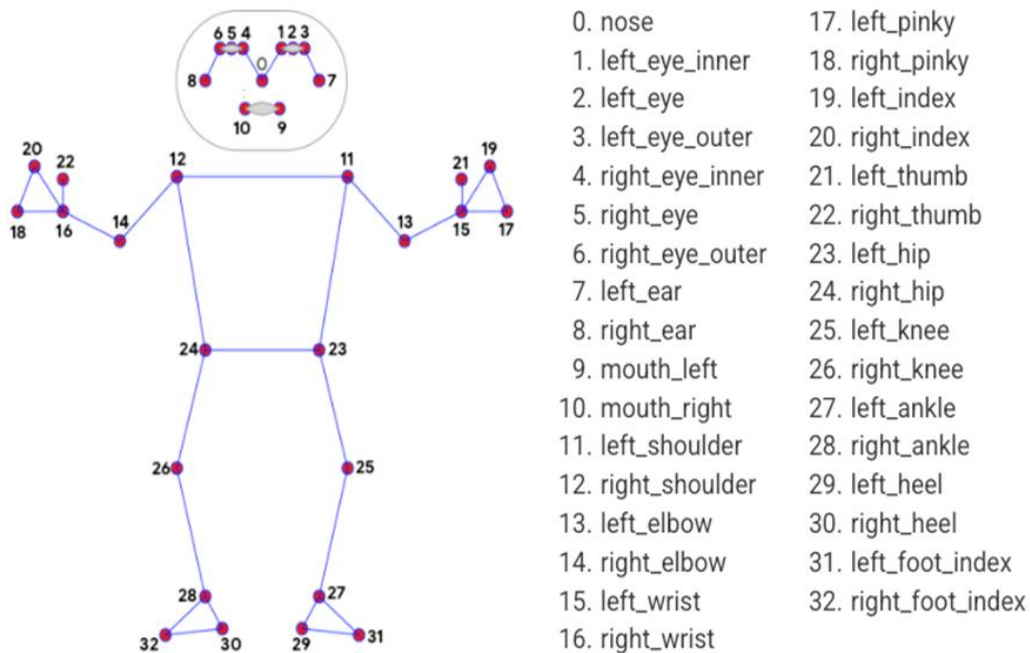
### **Step 2: Landmark Extraction**

- MediaPipe's Pose () class is initialized.
- The system detects joints like Head, Shoulders, Elbows, Knees, Ankles
- Outputs are 33 (x, y, visibility) coordinates.

### **Step 3: Skeleton Mapping**

- Joints are drawn using OpenCV's cv2.line() and cv2.circle().
- Final result is a black image with a white skeleton structure.

- This becomes a ControlNet input to guide outfit synthesis.



**Fig. 6.3 Pose Estimation - Skeleton Mapping**

## 6.4 OUTFIT GENERATION MODULE

The Outfit Generation Module is the core creative engine of the "Custom Outfit Generator with Virtual Try-On" system., responsible for synthesizing photorealistic images of new outfits tailored to the user's specifications and pose. It leverages advanced Generative AI techniques, specifically Stable Diffusion, conditioned on both the user's textual description of the desired clothing and the skeletal pose extracted by the Pose Estimation module. By integrating structural guidance from the pose and semantic information from the text prompt, this module generates novel outfit imagery that is both stylistically coherent with the user's request and anatomically consistent with their body posture, ultimately producing a realistic virtual try-on visualization. The module personalizes outfit generation by adapting to the user's past fashion preferences, body shape, and style history.

## Functionality

This module takes two main inputs: a user-defined clothing description e.g., "a red evening gown with silver heels" and a pose skeleton image derived from the user's webcam or uploaded photo. The description is processed using CLIP to convert the text into a format understandable by the image generation model. Simultaneously, the pose skeleton is fed into ControlNet, which guides the spatial alignment of the clothing.

The Stable Diffusion model uses both inputs to generate a high-resolution outfit image. Optional post-processing steps, including facial merging and segmentation, help overlay the generated outfit onto the user's original image for a realistic virtual try-on experience.

The module personalizes outfit generation by adapting to the user's past fashion preferences, body shape, and style history. Using learned embeddings or fine-tuned models, it enhances the outfit suggestions with tailored details like suitable colors, accessories, and fabric choices—creating more relevant and user-friendly virtual try-on results.

## Key Technologies

- **Stable Diffusion:** Generates images from text prompts.
- **ControlNet:** Ensures generated clothing aligns with the user's pose.
- **CLIP:** Embeds text input into visual semantics.
- **VAE and U-Net:** Support the latent-space operations for efficient generation

## System Requirements and Functionalities

### Text-to-Image Generation

- Generates detailed clothing images conditioned on user text prompts.
- Supports flexible prompt inputs including style, colour, season, and gender.

- Produces diverse outfit designs reflecting user creativity.

### **Pose Conditioning with ControlNet**

- Takes a skeleton map from the Pose Estimation Module as spatial conditioning.
- Ensures generated clothes align precisely with the user's body pose and limb orientation.
- Allows precise control over outfit placement and deformation.

### **Image Merging and Try-On**

- Optionally merges generated clothing onto the original user image.
- Uses semantic segmentation masks from modules like DeepLab or SAM to isolate cloth regions.
- Blends the generated outfit with user face and uncovered body parts for realism.

### **Output Management**

- Displays the generated outfit image on the UI.
- Allows users to save images locally or upload to cloud storage.
- Supports saving multiple outfits for comparison and reuse.

### **Tools, Libraries, and Frameworks**

- **Stable Diffusion:** Text-to-image latent diffusion model for generating images
- **ControlNet:** Provides pose-conditioned control to guide image synthesis
- **VAE: Variational Autoencoder** Encodes images into latent space and decodes back to images
- **CLIP:** Encodes text prompts into embeddings for conditioning
- **PyTorch:** Deep learning framework for model implementation and inference

- **OpenCV or PIL:** Image processing and format conversions

## **Working Mechanism**

### **Step 1: User Input Compilation**

Users provide:

- A text description e.g., "A black leather jacket and blue jeans"
- Gender e.g., Male
- Style e.g., Casual
- Pose image from Module 3

Final Prompt is structured as: "A male model wearing a black leather jacket and blue jeans in casual style". This final prompt ensures clarity and semantic richness for the text-to-image generator. This enriched prompt integrates descriptive, stylistic, and demographic information, enhancing the model's understanding and ensuring accurate, photorealistic outfit generation aligned with user intent.

### **Step 2: Pose Conditioning Input**

- The skeleton map from Module 3: Pose Estimation is formatted as an image.
- This image acts as a ControlNet conditioning input to guide pose accuracy.
- ControlNet ensures clothing aligns with the actual body pose e.g., sleeve falls on the arm.

### **Step 3: Text Embedding Using CLIP**

- The user's descriptive prompt is tokenized and passed through CLIP
- CLIP converts text into semantic embeddings that can be understood by Stable Diffusion.

### **Step 4: Image Generation using Stable Diffusion + ControlNet**

Stable Diffusion generates an image from text in the latent space using:

- Variational Autoencoder for encoding or decoding images.
- U-Net for denoising latent variables into images.
- CLIP embeddings as semantic conditions.
- ControlNet pose map for spatial structure.

Internal Loop:

- Start with a random noise tensor in latent space.
- Apply denoising steps guided by:
  - Text condition - prompt
  - Pose image - ControlNet
- Refine until a realistic clothing image is synthesized.

### **Step 5: Output Decoding and Post-Processing**

The latent image is decoded to a high-resolution output via the VAE decoder. Post-processing steps include:

- Resizing for UI display.
- Optional blending with the original image using segmentation masks DeepLab or SAM.
- Optional color enhancement.

### **Step 6: Virtual Try-On**

If the user enables try-on:

- The face and uncovered body regions from the original image are preserved.
- Generated clothes are merged on top of the body using:
  - Alpha blending
  - Semantic segmentation
  - Pose alignment

## CHAPTER 7

### CONCLUSION AND FUTURE ENHANCEMENT

#### 7.1 CONCLUSION

The development of the **Custom Outfit Generator with Virtual Try-On** system marks a significant step toward redefining the user experience in fashion technology. By leveraging the power of deep learning and computer vision, this system introduces a novel approach to generating and trying on custom outfits in real-time. It eliminates the traditional dependency on predefined garment datasets by integrating AI-driven outfit synthesis and pose-aware virtual try-on capabilities.

Each module within the system plays a critical role in contributing to its overall success. The **Webcam or Image Capture Module** ensures seamless acquisition of user visuals through both real-time webcam input and image upload functionalities. Its preprocessing pipeline guarantees standardized input for downstream tasks, providing the necessary image format and resolution for accurate pose estimation.

The **User Interaction Module** offers an intuitive and dynamic interface, enabling users to describe their desired outfits through natural language prompts and dropdown selections. This not only facilitates ease of use but also enriches the user's design experience by translating input directly into creative fashion representations.

The **Pose Estimation Module**, powered by MediaPipe BlazePose, extracts detailed skeletal keypoints to construct a reliable structural representation of the user's posture. This enables pose-aware alignment during outfit generation, ensuring that garments fit naturally and adapt to diverse body orientations.

At the heart of the system, the **Outfit Generation Module** combines the descriptive power of CLIP, the generative capability of Stable Diffusion, and the structural guidance of ControlNet. This fusion produces high-quality, photorealistic



try-on images that conform to user poses and clothing prompts, delivering an immersive virtual try-on experience.

Collectively, the integration of these modules results in a powerful, modular, and scalable platform that demonstrates the future potential of AI in personalized fashion technology. The system offers creative freedom, personalized garment visualization, and high user engagement through real-time feedback, all while being resource-efficient and deployable on standard hardware.

## **7.2 FUTURE ENHANCEMENT**

Building upon the advancements in virtual try-on technology, the future holds the promise of even more immersive and personalized shopping experiences. Integrating augmented reality with haptic feedback systems could allow users to not only see but also feel the texture and weight of garments, providing a more realistic simulation of how clothes would fit and move in real life. This sensory enhancement would bridge the gap between online and in-store shopping, offering a tactile dimension to virtual try-ons.

Furthermore, the incorporation of artificial intelligence can revolutionize the personalization aspect of virtual try-ons. AI algorithms could analyze user preferences, body types, and even mood to suggest outfits that align with individual styles and occasions. This level of customization would not only enhance user satisfaction but also streamline the shopping process by providing tailored recommendations, reducing decision fatigue, and potentially increasing conversion rates for retailers.

In addition to personalization, the future of virtual try-on technology is poised to contribute significantly to sustainability efforts in the fashion industry. By enabling customers to virtually try on clothes, the need for physical samples and returns is minimized, leading to a reduction in textile waste and carbon emissions associated with shipping and logistics.

## APPENDIX A

### SOURCE CODE

```
import gradio as gr

import numpy as np

import torch

from PIL import Image

import os

import datetime

from diffusers import StableDiffusionControlNetPipeline, ControlNetModel,
UniPCMultistepScheduler

import mediapipe as mp

# Device

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Load ControlNet and Stable Diffusion Pipeline

controlnet = ControlNetModel.from_pretrained(

    "lllyasviel/sd-controlnet-openpose",

    torch_dtype=torch.float32

)

pipeline = StableDiffusionControlNetPipeline.from_pretrained(

    "runwayml/stable-diffusion-v1-5",

    controlnet=controlnet,

    safety_checker=None,
```

```

    torch_dtype=torch.float32
)

pipeline.scheduler = UniPCMultistepScheduler.from_config(pipeline.scheduler.config)

pipeline.to(device)

# Save folder

save_folder = "saved_outfits"

os.makedirs(save_folder, exist_ok=True)

# Options

genders = ["female", "male"]

# Pose Estimation Function

def estimate_pose(image_pil):

    try:

        mp_pose = mp.solutions.pose

        pose = mp_pose.Pose(static_image_mode=True, model_complexity=2)

        image_rgb = np.array(image_pil.convert("RGB"))

        results = pose.process(image_rgb)

        if not results.pose_landmarks:

            print("No landmarks.")

            return None

        annotated = image_rgb.copy()

        mp_drawing = mp.solutions.drawing_utils

        mp_drawing.draw_landmarks(

            image=annotated,

```

```

        landmark_list=results.pose_landmarks,

        connections=mp_pose.POSE_CONNECTIONS

    )

    return Image.fromarray(annotated)

except Exception as e:

    print("Pose error:", e)

    return None

# Generation Function

def generate_outfit(image, gender, prompt):

    if image is None:

        return None, "Please upload or capture an image."

    pose_image = estimate_pose(image)

    if pose_image is None:

        return None, "Pose detection failed. Try a full-body image."

    full_prompt = f"{gender} outfit"

    if prompt:

        full_prompt += f", {prompt}"

    try:

        resized = pose_image.resize((512, 512))

        result = pipeline(prompt=full_prompt, image=resized, num_inference_steps=10)

        if not result or not hasattr(result, "images") or len(result.images) == 0:

            return None, "Generation failed."

        output = result.images[0]

```

```

        return output, "Outfit generated!"

except Exception as e:

    print("Generation error:", e)

    return None, f" {e}"

# Save Output

def save_outfit(image):

    if image is None:

        return None

    filename=os.path.join(save_folder,f"outfit_{datetime.datetime.now().strftime('%Y%m%d_%H%M%S')}.png")

    image.save(filename)

    return filename

# Gradio UI

def ui():

    with gr.Blocks() as demo:

        gr.Markdown("AI Fashion Outfit Generator + Virtual Try-On")

        with gr.Row():

            with gr.Column():

                input_image = gr.Image(label="Live Webcam or Upload", type="pil")

                gender = gr.Dropdown(genders, label="Gender", value="female")

                prompt = gr.Textbox(label="Custom Style Prompt", placeholder="e.g. red leather jacket")

                generate_btn = gr.Button("Generate Outfit")

```

```
with gr.Column():

    output_image = gr.Image(label="Result")

    status = gr.Textbox(label="Status", interactive=False)

    save_btn = gr.Button("Save Outfit")

    saved_path = gr.Textbox(label="Saved Path", interactive=False)

    generate_btn.click(fn=generate_outfit, inputs=[input_image, gender, prompt],
outputs=[output_image, status])

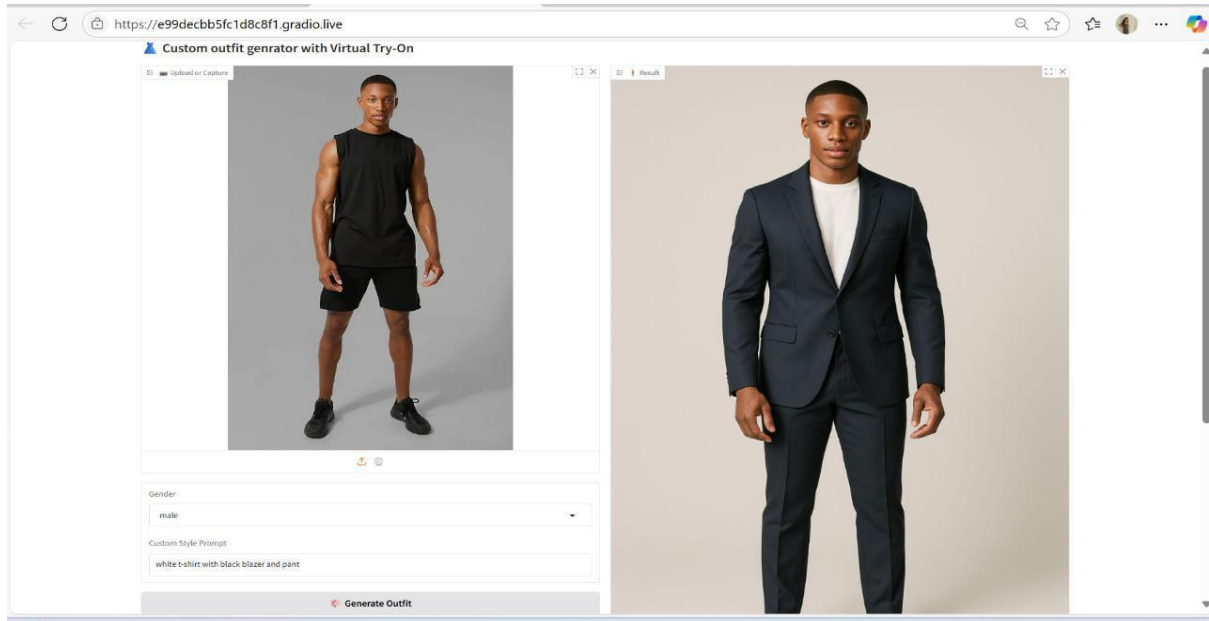
    save_btn.click(fn=save_outfit, inputs=[output_image], outputs=saved_path)

demo.launch(share=True)

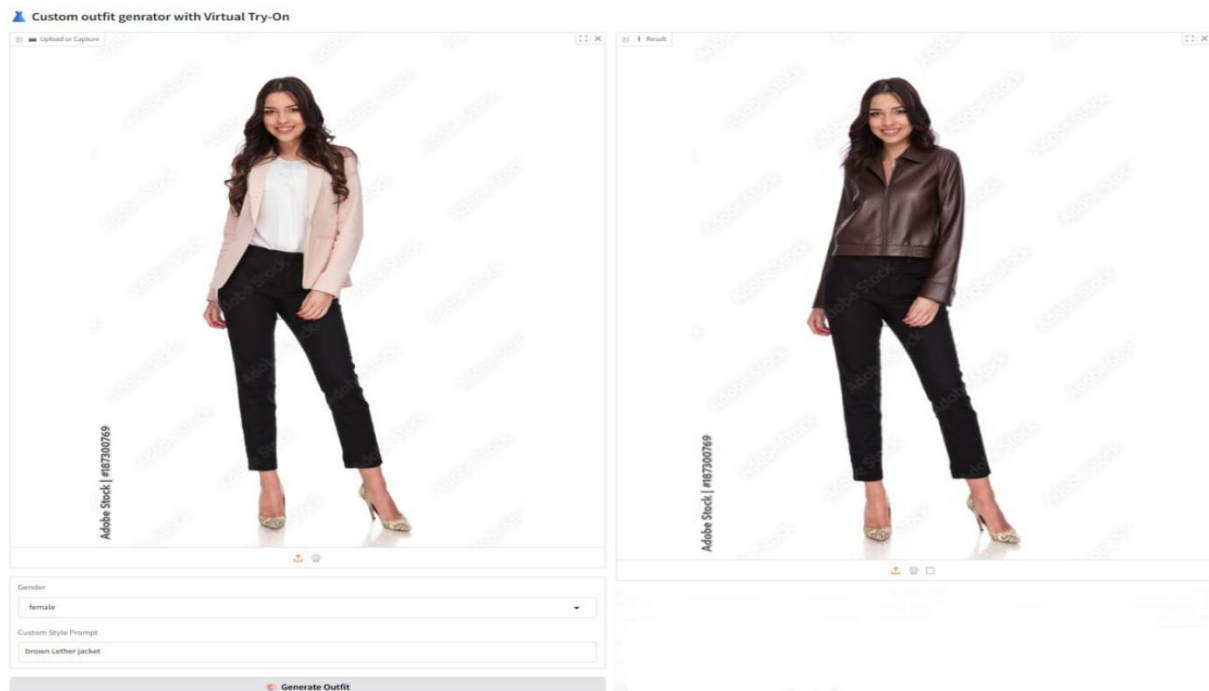
ui()
```

## APPENDIX B

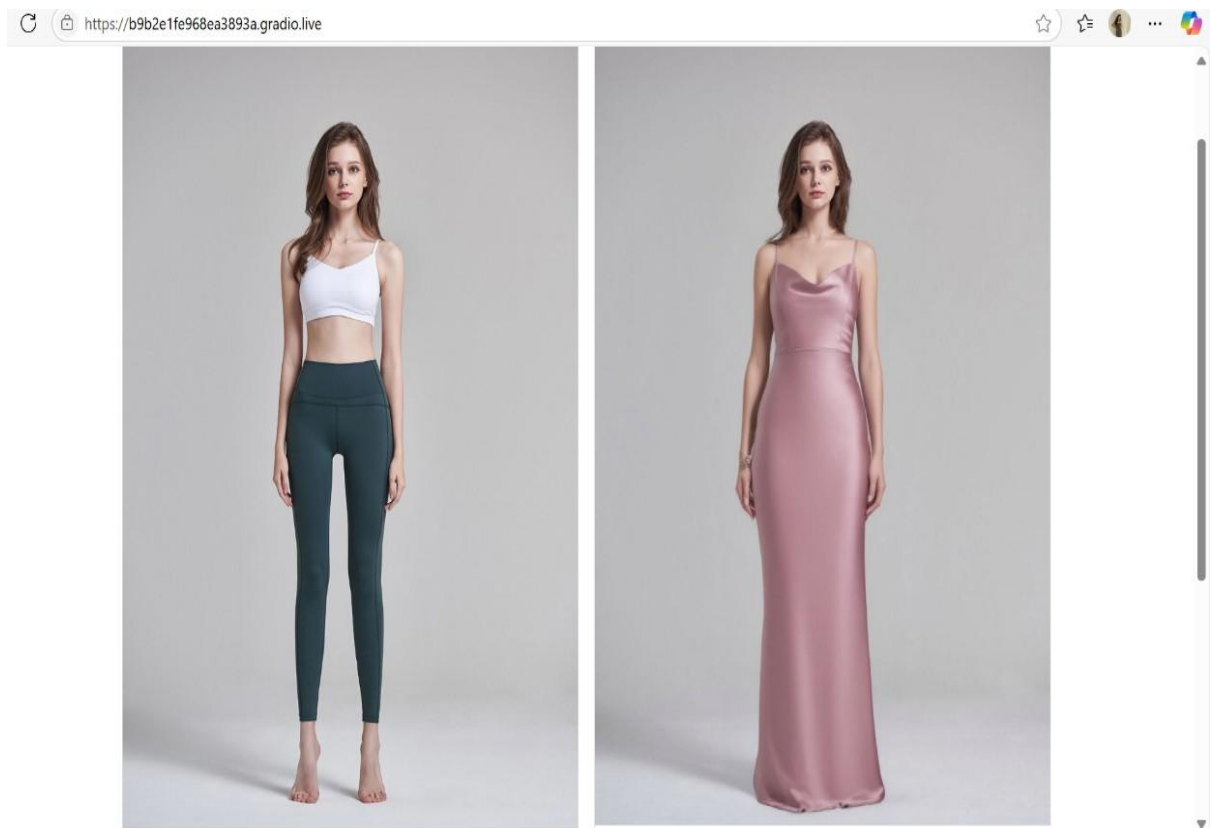
### SCREENSHOTS



**Fig. B.1 Sample Output-1**



**Fig. B.2 Sample Output-2**



**Fig. B.3 Sample Output-3**



## REFERENCES

1. Z. Chen, Y. Liu, Q. Hu, and X. Li, "IDM-VTON: Improving Diffusion Models for Authentic Virtual Try-on in the Wild," in arXiv preprint arXiv:2403.05139, 2024.
2. L. Wang, K. Huang, and Y. Zhang, "CAT-DM: Controllable Accelerated Virtual Try-On with Diffusion Model," in arXiv preprint arXiv:2311.18405, 2023.
3. Y. Li, L. Song, J. Zhu, and C. Ma, "Virtual Try-On with Pose-Garment Keypoints Guided Inpainting," in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 21650–21659, 2023.
4. E. Bochinski, A. Müller, T. Richter, and R. Stiefelhagen, "Pose-Guided Try-On using ControlNet," in CVPR Workshops, pp. 1–6, 2022.
5. P. von Platen, S. Wolf, T. Debus, L. Schmid, and the Hugging Face Team, "Hugging Face Diffusers: State-of-the-Art Diffusion Models in Python", Hugging Face, 2022.
6. R. Dong, T. Yu, Y. Yang, and S. Li, "C-VTON: Context-Driven Image-Based Virtual Try-On Network," in arXiv preprint arXiv:2212.04437, 2022.
7. R. Dong, T. Yu, Y. Yang, and S. Li, "C-VTON: Context-Driven Image-Based Virtual Try-On Network," in arXiv preprint arXiv:2212.04437, 2022.
8. Y. Ge, X. Zhang, Y. Luo, and H. Li, "Disentangled Person Image Generation," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2377–2386, 2020.
9. T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for GANs," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4401–4410, 2019.
10. B. Wang, X. Tang, T. Xu, and X. Liu, "Toward Characteristic-Preserving Image-Based Virtual Try-On Network," in Proceedings of the European Conference on Computer Vision (ECCV), pp. 589–604, 2018.