International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 06 | June – 2025 DOI: 10.55041/ISJEM04606
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

A

Major Project Report On

AUGMENTED REALITY IN FASHION INDUSTRY

Submitted to

Jawaharlal Nehru Technological University, Hyderabad

For the partial fulfillment of requirements for the award of the degree in

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

CHILUKAMARI BHARGAVI	**(21271A05E2)**
SOLLU AKANKSHA	**(21271A05D5)**
PORANDLA ABHISHEK	**(22275A0516)**
AMPATI SRIDUTT	**(21271A05I4)**

Under the Esteemed guidance of

K.VANITHA

Assistant Professor

Dept of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

JYOTHISHMATHI INSTITUTE OF TECHNOLOGY AND SCIENCE

(Autonomous, NBA (CSE, ECE, EEE) and NAAC 'A' Grade)
(Approved by AICTE, New Delhi, Affiliated to JNTUH, Hyderabad)

Nustulapur, Karimnagar 505481, Telangana, India

2024-2025

ABSTRACT

Augmented Reality (AR) is progressively transforming the fashion sector by filling the void between digital technology and consumers' needs. As digital technologies and e-commerce continue to evolve, the necessity for richer, interactive, and engaging shopping experiences has never been more imperative. AR fills this gap by allowing customers to try on clothes and accessories virtually in real time through their smartphones, tablets, or smart mirrors without the need for any physical contact. This ability turns the conventional retail experience into an extremely personal journey. Consumers are able to see how clothes would fit them in their own homes, thus making better buying decisions. Brands thus enjoy lower rates of product returns, enhanced consumer satisfaction, and better consumer interaction. In addition, AR technology promotes eco-friendly fashion behavior by limiting the use of physical samples and overproduction, resulting in lower waste and environmental footprint. Fashion brands are incorporating AR into multiple platforms—like mobile apps, in-store smart mirrors, and online sites—to maximize user experience. These uses not only appeal to technology-savvy consumers but also allow brands to differentiate themselves amidst a highly competitive marketplace. Aside from shopping, AR has profound impacts on marketing, design, and manufacturing in the fashion industry. Interactive AR campaigns and digital fashion shows provide new channels for brand narrative and consumer engagement. Designers can also use AR to prototype and visualize fashion ideas prior to physical production, resulting in streamlined workflows.

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

The fashion industry, traditionally reliant on in-person experiences and physical products, is now experiencing a profound digital transformation fuelled by advancements in immersive technologies—chief among them, Augmented Reality (AR). AR is a cutting-edge technology that overlays digital content onto the real world, allowing users to interact with computer generated images, sounds, and data through devices such as smartphones, tablets, AR glasses, or smart mirrors. In the context of fashion, AR serves as a powerful bridge between the online and offline retail experiences, offering innovative ways for consumers to engage with products without the need for physical contact.

One of the most significant impacts of AR in fashion is the virtual try-on experience. Consumers can now visualize how clothing, accessories, shoes, or even makeup will look on their own body in real-time, simply by using a mobile app or standing in front of an AR-enabled mirror. This not only provides a more personalized and convenient shopping experience but also addresses key challenges in e-commerce, such as high return rates due to poor fit or dissatisfaction with the product's real-world appearance. By enabling customers to make more informed purchase decisions, AR helps reduce product returns, saving costs for retailers and minimizing environmental waste. In physical stores, AR enhances engagement through interactive displays, digital fitting rooms, and gamified brand experiences that attract tech-savvy consumers. Online, AR integration within apps and websites allows customers to test products from the comfort of their homes. Fashion giants like Gucci, Nike, and Zara, as well as emerging tech-driven brands, are already leveraging AR to create immersive shopping environments that reflect their brand identity and improve customer satisfaction.

Beyond retail, AR is also influencing fashion design and marketing. Designers can use AR tools to visualize concepts in 3D, experiment with digital fabrics, and collaborate remotely. In marketing, AR opens the door to virtual fashion shows, digital influencers, and interactive advertisements, which not only captivate audiences but also offer valuable data insights into consumer preferences and behaviour.

Another vital aspect of AR in fashion is its role in promoting sustainability and efficiency. By reducing the need for physical samples, excessive inventory, and unnecessary shipping, AR contributes to a more sustainable supply chain. Consumers are also becoming more conscious of eco-friendly choices, and AR empowers them to explore options in a less wasteful manner.

1. **Augmented Reality (AR)**: A technology that overlays digital information (such as images, sounds, and 3D models) onto the real world through a device like a smartphone or AR glasses.
2. **Virtual Try-On**: A feature enabled by AR that allows customers to digitally test out clothes, accessories, or makeup on themselves in real-time using cameras or AR apps.
3. **Immersive Shopping**: A shopping experience that uses digital technology (like AR/VR) to make the process more interactive, engaging, and lifelike for the consumer.
4. **Interactive Mirror (Smart Mirror)**: A mirror integrated with AR technology that allows users to try on outfits virtually in a retail environment without physically changing clothes.
5. **3D Modelling**: The creation of three-dimensional representations of clothing or accessories, used in AR to simulate how products look and fit on the human body.
6. **Customer Personalization**: The customization of shopping experiences based on individual user data, preferences, and behaviours, often enhanced by AI and AR technologies.
7. **Sustainable Fashion**: Practices in the fashion industry that aim to reduce environmental impact, which AR supports by minimizing waste and reducing returns.
8. **Computer Vision**: A field of artificial intelligence that enables computers to interpret and process visual data—used in AR for face/body detection and virtual fitting.

1.1.1 PROJECT DESCRIPTION

The fashion industry is evolving rapidly, driven by the need for innovation, personalization, and sustainability. Among the most transformative technologies at the forefront of this evolution is Augmented Reality (AR). This project explores the integration of AR in the fashion sector, focusing on how it enhances customer experiences, optimizes business operations, and contributes to sustainable practices.

Augmented Reality blends digital elements into the real-world environment through the use of mobile devices, smart mirrors, or AR glasses. In fashion, this technology enables virtual try-on experiences, where users can see how clothing, accessories, and makeup would appear on their body in real-time without physically wearing them. This functionality not only makes shopping more interactive and enjoyable but also addresses major e-commerce challenges such as high product return rates, customer dissatisfaction, and fit-related issues.

The core objective of this project is to develop a system that allows users to create their personalized 3D avatars and try on virtual garments using AR. By using computer vision and 3D modelling technologies, the platform captures facial and body features to generate accurate digital replicas of users. This allows for a realistic and customized virtual fitting experience, helping users make better purchasing decisions.

On the business side, retailers benefit from reduced inventory needs, fewer returns, and improved customer engagement. Moreover, the use of AR in digital marketing— through interactive ads, virtual fashion shows, and digital influencers—is helping brands attract and retain a tech-savvy customer base.

This project also emphasizes sustainability, a growing concern in the fashion world. By eliminating the need for physical samples and reducing waste associated with product returns and overproduction, AR contributes to eco-friendly practices. The system integrates technologies such as AR SDKs (ARKit, ARCore), AI-driven personalization, and crossplatform development tools like React Native and Flutter to ensure compatibility across devices.

In conclusion, this project demonstrates that AR is not just an add-on technology but a strategic innovation that is reshaping the fashion landscape. By merging physical and digital shopping experiences, AR enhances convenience, personalization, and sustainability. As the fashion industry continues to embrace digital transformation, AR will play a pivotal role in creating smart, immersive, and customer-centric retail ecosystems.

1.1.2 METHODOLOGY

The methodology of this project focuses on the systematic development and implementation of an Augmented Reality (AR) system that enables users to virtually try on fashion items such as clothing and accessories using their personalized 3D avatars. The approach involves multiple phases, including requirement analysis, system design, development, integration, and testing. Each phase is described in detail below:

1. Requirement Analysis

The first phase involves identifying the functional and non-functional requirements of the AR-based virtual try-on system. Inputs were gathered through literature surveys, existing system evaluations, and user feedback. Key requirements included:

- User registration and 3D avatar creation.
- Real-time AR rendering of clothing on avatars.
- Device compatibility for Android, iOS, and web platforms.
- Backend support for storing user data and managing assets.

2. System Design

In this phase, the architecture and workflow of the system were designed using diagrams such as:

- **Use Case Diagram**: To model user interactions with the system.
- **Activity Diagram**: To illustrate the flow of processes like virtual try-on and item selection.
- **Class and Sequence Diagrams**: To define system components, data flow, and interactions between modules.

The system was divided into frontend, backend, and AR processing components to ensure modular and scalable development.

3. 3D Avatar Creation

Users are allowed to create personalized avatars using facial recognition and body measurement inputs. This is achieved using:

- **Computer Vision Libraries** (OpenCV, MediaPipe) for feature detection.
- **3D Modelling Tools** and predefined templates for rendering custom avatars.

4. Augmented Reality Integration

To deliver the virtual try-on experience, AR frameworks were integrated depending on the platform:

- **ARKit** for iOS, **AR Core** for Android, and **Web AR** for browser-based access.

- **Three.js** and **A-Frame** were used for rendering 3D clothing on avatars in realtime.

5. Backend Development

The backend handles user authentication, asset management, and personalization logic. It was built using:

- **Node.js** and **Express.js** for server-side logic.
- **MongoDB** or **Firebase** for data storage and retrieval.
- **AI/ML models** (using TensorFlow or PyTorch) for recommendation and body measurement prediction.

6. Testing and Validation

The system was tested for performance, usability, and accuracy. Key testing steps included: • **Unit Testing**: For individual modules like avatar generation and AR rendering.

- **Integration Testing**: Ensuring smooth interaction between frontend, backend, and AR modules.
- **User Testing**: Feedback collected from sample users to assess interface design and virtual fitting accuracy.

This methodology ensures that the project is implemented in a structured, efficient, and user-centric manner, leveraging the latest technologies in AR and fashion tech.

1.2 EXISTING SYSTEM

The rapid growth of the fashion industry has prompted significant technological innovation, especially in how consumers interact with products. With the rise of ecommerce, one of the most prominent challenges facing both consumers and retailers is the lack of physical interaction with fashion items. As a solution, various virtual tryon systems have been introduced, many of which make use of basic 2D imaging or generic 3D models. While these systems provide some level of interactivity, they fall short in delivering a personalized and realistic shopping experience.

1. 2D-Based Virtual Try-On Platforms

The most common form of virtual try-on systems found in current fashion retail platforms involves **overlaying 2D clothing images** onto static pictures or webcam feeds of the user. These systems typically use basic facial or body recognition to position items such as glasses, hats, or T-shirts on the user's photo.

While this method is relatively easy to implement and widely accessible across mobile devices and web platforms, it comes with several drawbacks:

- **Lack of Depth and Realism**: These systems do not provide an accurate sense of how the garment would fit or move on the user's body.

- **No Customization**: The clothing is rendered in fixed sizes and cannot adapt to different body types or postures.
- **Static Models**: The use of static, flat images limits user interaction and immersion, making the try-on experience feel artificial.

As a result, users are often disappointed with the real-world fit and appearance of products, leading to **high return rates** and **lower customer satisfaction**.

2. Generic 3D Avatars

Some fashion apps and online platforms have begun integrating **basic 3D modelling** to enhance virtual try-on capabilities. These platforms typically allow users to select from a set of **predefined avatars** representing average body types. The selected avatar is then used to simulate the fit of clothing.

However, these existing systems still suffer from important limitations:

- **Non-Personalized Avatars**: Since the avatar is not modeled on the actual user's body measurements or facial features, the try-on experience lacks personal relevance.
- **Limited Real-Time Interactivity**: Many platforms do not support dynamic interaction (e.g., turning, posing, zooming) in real-time, which limits the user's ability to view garments from different angles or in motion.
- **Clothing Rendering Issues**: In many cases, clothing items are rendered in a generic manner, without accounting for fabric behavior, fit dynamics, or motion tracking, which diminishes realism.

3. Technical and Hardware Constraints

The success of virtual try-on systems also depends on device capabilities. Most existing platforms are designed to function on standard smartphones and lack the depth-sensing technology or graphical power required for high-fidelity AR rendering. This results in:

- **Poor Fit Accuracy**: Without depth sensors, the system cannot accurately map the user's body structure, leading to visual misalignments.
- **Low Graphic Quality**: Basic graphics and lag in rendering reduce the sense of immersion and make the system appear outdated or unreliable.

1.3 PROBLEM STATEMENT

Despite the growing integration of technology in the fashion industry, current virtual Try-on solutions are limited in their ability to deliver realistic, personalized, and immersive shopping experiences. Most existing systems rely on 2D overlays or basic, non-customizable

3D avatars that do not accurately represent the user's body shape, facial features, or movement. As a result, these systems often fail to provide a true-to-life visualization of how clothing and accessories would look and fit on an individual.

This lack of personalization leads to several challenges:

- Inaccurate Fit Representation: Users cannot accurately gauge how a product will fit or move with their body, resulting in poor purchase decisions.
- High Return Rates: The mismatch between user expectations and the actual product contributes to frequent product returns, increasing operational costs for retailers.
- Low User Engagement: The static and unrealistic nature of the current virtual try-on experience reduces consumer interest and limits the effectiveness of online fashion platforms.
- Limited Accessibility and Scalability: Small to medium fashion retailers struggle to implement these solutions due to high costs, technical complexity, and the need for large-scale 3D asset creation.

Therefore, there is a clear need for an advanced AR-based system that offers real-time, user specific 3D visualization with dynamic interaction capabilities. Such a system would not only enhance customer satisfaction and reduce return rates but also support sustainable and scalable practices in the fashion industry.

1.4 PROPOSED SYSTEM

To address the limitations of current virtual try-on technologies in the fashion industry, the proposed system aims to develop a personalized, real-time, ARbased virtual tryon platform. Unlike existing solutions that use static 2D overlays or generic 3D models, this system enables users to create highly accurate 3D avatars based on their actual body dimensions and facial features. The integration of advanced computer vision, real-time 3D rendering, and AR frameworks ensures a seamless and immersive user experience that mimics real-world tryons.

System Overview

The proposed system is designed as a cross-platform application accessible via smartphones, tablets, and AR-enabled mirrors. It uses a combination of front-end AR SDKs (like ARKit, ARCore, and WebAR), backend AI-powered personalization tools, and cloud-based data storage to deliver a responsive and intelligent virtual fashion experience.

Key Features and Functionalities

1. Personalized 3D Avatar Creation ○ Users can capture their body measurements and facial features using the device's camera. ○ Computer vision algorithms (using OpenCV and MediaPipe) analyse the captured data to generate a realistic 3D avatar that closely represents the user's body shape, size, and appearance. ○ The avatar serves as a digital twin for trying on garments, accessories, and makeup products.
2. Real-Time AR Virtual Try-On ○ The user can select items from a catalogue of 3D-modeled fashion products. ○ Using AR rendering engines (like Three.js and A-Frame), the selected clothing items are displayed on the user's 3D avatar in real time. ○ Users can view the outfit from multiple angles, zoom in for fabric texture, and even see how the clothing responds to movement (e.g., folds, fit, length).
3. Gesture and Body Tracking ○ The system supports gesture recognition to enable interactive controls such as rotating the avatar, changing outfits, or zooming in on details. ○ Advanced body tracking allows garments to move naturally with the user's body, improving realism and fit accuracy.

4. AI-Powered Recommendations ○ Machine learning models (using TensorFlow or PyTorch) analyse user preferences, past interactions, and body data to suggest styles, sizes, and outfit combinations. ○ This increases personalization and customer satisfaction, while also improving conversion rates for retailers.
5. Platform Compatibility ○ The system is developed using cross-platform tools such as React Native and Flutter to ensure compatibility across Android, iOS, and web environments. ○ ARKit is used for iOS devices, ARCore for Android, and WebAR for browserbased experiences.
6. Cloud Integration and Backend Infrastructure ○ User profiles, 3D assets, and clothing libraries are stored securely in the cloud (using services like Firebase or MongoDB). ○ Backend logic and APIs are managed using Node.js and Express.js to ensure fast data access and scalability.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 06 | June – 2025 DOI: 10.55041/ISJEM04606
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

CHAPTER 2 LITERATURE REVIEW

2.1 LITERATURE REVIEW

The integration of **Augmented Reality (AR)** in the fashion industry has gained significant attention from researchers, developers, and fashion retailers due to its potential to enhance consumer experience, reduce operational costs, and promote sustainable practices. This literature review presents an overview of recent developments, studies, and technological advancements related to AR in fashion, focusing on virtual try-on systems, user experience, technological challenges, and industry applications.

1. Augmented Reality and Virtual Try-On Systems

Early research in the field primarily focused on the concept of **virtual try-on** technology, where users can visualize fashion items on their bodies using digital interfaces. According to Scholz and Smith (2016), virtual try-ons using AR enhance customer confidence in purchasing decisions by offering realistic product previews. They found that AR-based applications significantly increase customer engagement, especially in online shopping environments where tactile feedback is absent. Kim and Forsythe (2008) explored consumer responses to virtual product experience and concluded that **interactivity and visual realism** were critical to increasing purchase intentions. More recent systems have started leveraging **3D body scanning** and **avatar generation** to personalize the experience, but these implementations still face limitations in scalability and realism.

2. User Experience and Engagement

A study by Poushneh and Vasquez-Parraga (2017) highlighted that AR technologies in fashion retail positively impact customer satisfaction by making shopping more **entertaining and interactive**. The ability to visualize clothing in real-time builds emotional connections and reduces uncertainty about size, style, and colour. Moreover, researchers have examined how **gesture-based interaction** and **real-time feedback** in AR apps increase immersion. For instance, Huang and Liao (2015) demonstrated that integrating AR with **social sharing features** allowed users to seek peer feedback, further enhancing decision-making and enjoyment.

3. Technological Advancements

The development of AR in fashion relies on a combination of **computer vision**, **3D rendering**, and **machine learning**. Advances in mobile AR frameworks such as **Apple's ARKit** and **Google's ARCore** have made it possible to deploy AR applications on consumer-grade devices with improved performance and accuracy. Kang et al. (2020) emphasized the importance of accurate **body tracking and motion capture** for creating realistic try-on experiences. They suggested that future AR fashion systems would need to incorporate **AI-powered personalization** to dynamically adapt clothing visuals to diverse body shapes and preferences.

4. Commercial Applications and Industry Use Cases

Leading fashion brands have begun adopting AR to improve customer interaction and drive sales. For example, **Zara and H&M** have experimented with AR-enabled shop windows and in-store AR displays. **Gucci's AR app** allows users to try on sneakers through smartphone cameras, while **Nike** uses AR to measure foot size and suggest the best-fitting shoe model.

Academic research supports these trends, noting that AR not only increases conversion rates but also reduces return rates—a major concern in online fashion retail (Heller et al., 2019). Furthermore, AR enables **sustainable fashion practices** by minimizing the need for physical samples and lowering environmental impact through reduced shipping and returns.

6. Virtual Avatars and Body Scanning Technologies

Research by **Zeng, X., et al. (2018)** explores the role of **3D body scanning** and virtual avatars in the personalization of fashion retail experiences. The study highlights that realistic avatars built from body measurements and facial recognition significantly enhance the accuracy of virtual try-ons. These avatars allow customers to preview how garments will fit on their own bodies, improving satisfaction and confidence in online purchases. However, the study also notes the challenges in capturing accurate body data using consumer-grade cameras, suggesting that further innovation is required in body scanning techniques and depth sensing technologies. **7. Augmented Reality in Fashion Education and Design**

In addition to retail, AR is being used in **fashion design and education**. According to **Choi and Kim (2020)**, AR can support designers in visualizing fabric drape, pattern alignment, and garment layering without physical prototypes. Their study implemented an AR-based design tool in a fashion curriculum and found that students demonstrated increased creativity, collaboration, and spatial awareness. This aligns with the growing trend of **digital fashion**, where garments are designed, visualized, and sometimes even sold as purely digital products for AR/VR use (e.g., in gaming or social media filters).

CHAPTER 3
REQUIREMENTS AND DOMAIN INFORMATION

3.1 REQUIREMENT SPECIFICATIONS

The proposed AR-based virtual try-on system requires both software and hardware components optimized for real-time performance and cross-platform use. On the software side, frontend development will use technologies like React.js or Flutter, while AR features will be powered by ARKit (iOS), ARCore (Android), and webbased libraries such as AR.js or AFrame. Backend development will utilize Node.js or Django, and AI functionalities will be supported by machine learning libraries like TensorFlow. For hardware, mobile devices must support AR functionality, while development machines should have at least an Intel i5 processor, 8GB RAM, and a dedicated GPU for smooth 3D rendering. Cameras with depth sensing are preferred for better tracking. For data storage, the system will use a cloud-based NoSQL database like Firebase or MongoDB to manage user profiles, 3D avatars, and product data securely and efficiently. This setup ensures high performance, scalability, and compatibility across web and mobile platforms.

3.1.1 HARDWARE REQUIREMENTS

The proposed AR-based virtual try-on system requires hardware that supports smooth 3D rendering and real-time augmented reality experiences. For end-users, modern smartphones or tablets with AR capabilities are essential. Devices should support platforms like ARCore (for Android) or ARKit (for iOS), ensuring compatibility with AR features such as virtual tryons, body tracking, and gesture recognition. A good quality camera is necessary for capturing user movements and facial features, while depth-sensing cameras (like LiDAR) can enhance precision and realism. On the development side, computers should have a capable processor (such as Intel i5 or higher), adequate RAM (8GB minimum), and a dedicated graphics card for handling 3D assets and rendering efficiently. Storage space is also important for managing avatars, clothing models, and user data. Overall, the hardware must ensure seamless performance, low latency, and reliable rendering to deliver a responsive, interactive, and immersive AR experience across different platforms.

3.1.2 SOFTWARE REQUIREMENTS

The AR-based virtual try-on system relies on a combination of frontend, backend, and AR technologies to deliver a seamless and interactive experience. On the frontend, frameworks like React.js, Flutter, or React Native are used for building responsive and cross-platform interfaces. For AR functionality, the system integrates with ARKit (iOS), ARCore (Android), and webbased AR libraries such as AR.js or A-Frame for browser compatibility. The backend is developed using frameworks like Node.js or Django, which handle user authentication, data processing, and API management. Cloud databases such as Firebase or MongoDB are used to store user profiles, 3D avatar data, and clothing models securely. Additionally, machine learning libraries like TensorFlow or PyTorch may be used to provide AI-driven recommendations and personalization. Together, these software components ensure the system runs efficiently, supports real-time AR rendering, and delivers a personalized, scalable, and user-friendly experience across devices.

3.2 SYSTEM SPECIFICATIONS

The proposed AR-based virtual try-on system is designed to offer a personalized and immersive fashion experience through a combination of modern software and hardware components. On the software side, the system uses React.js or Flutter for the frontend to ensure responsive and cross-platform compatibility across mobile and web devices. AR features are enabled through ARKit (iOS), ARCore (Android), and AR.js/A-Frame for web-based interaction. The backend is developed using Node.js or Django, supporting user authentication, product management, and real-time data processing. Firebase or MongoDB is used as the primary database to store user profiles, avatar data, and digital clothing assets. To enhance user experience and personalization, machine learning libraries like TensorFlow or PyTorch are integrated for AI-based size and style recommendations. The entire system is cloudenabled to support scalability, real-time performance, and multi-device access. On the hardware side, the system requires AR-supported smartphones or tablets for end users— devices must be compatible with ARKit or ARCore and have decent processing power and camera quality. Depth-sensing cameras, such as LiDAR, are recommended for higher accuracy in avatar creation and gesture tracking. For development purposes, machines should be equipped with at least an Intel Core i5 processor, 8GB RAM, a dedicated GPU (e.g., NVIDIA GTX 1060 or higher), and 512GB SSD storage for smooth handling of 3D rendering and asset management. An HD webcam is necessary for capturing facial and body data during avatar generation. Together, these software and hardware specifications ensure the system is capable of delivering a seamless, real-time, and interactive AR shopping experience that meets both user expectations and industry requirements.

3.3 FLOW CHART

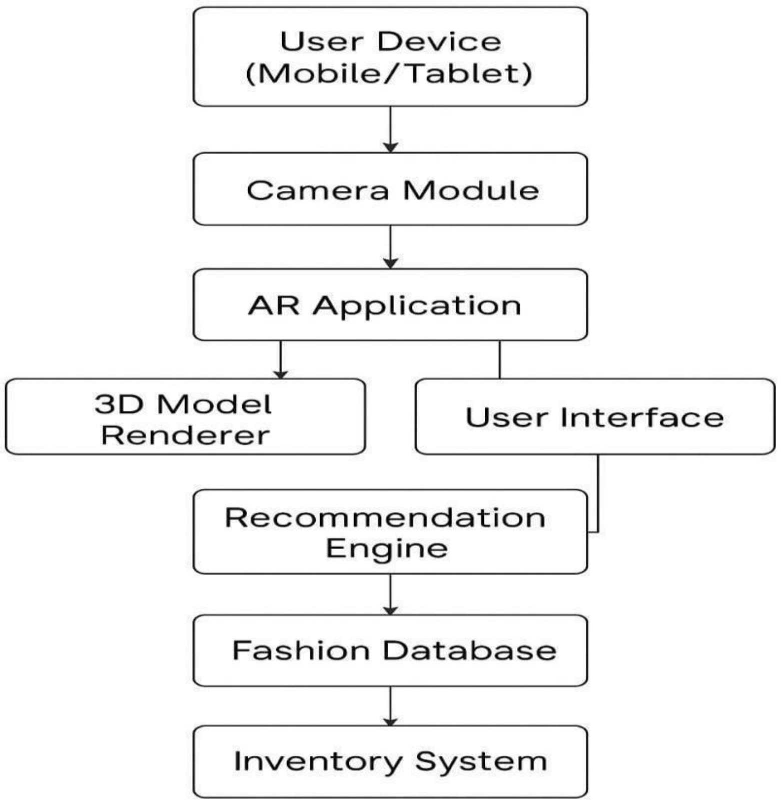

Fig.no. 3.3 Flowchart

The rapid growth of e-commerce in the fashion industry has brought both convenience and challenges. One of the most significant issues is the **lack of physical interaction with products**, which prevents customers from trying on clothing before making a purchase. This limitation often leads to poor size selection, misjudged styles, and dissatisfaction with the final product. Although size charts and product photos help to some extent, they cannot fully replicate the in-store experience of seeing, feeling, and trying on clothing in real time.

To address this gap, many fashion brands have implemented **virtual try-on technologies**, but most existing systems rely on basic 2D image overlays or standardized 3D avatars. These solutions fail to account for individual body dimensions, facial features, or realtime motion, which results in inaccurate and unrealistic visualizations. As a result, users are unable to make confident purchase decisions, and this contributes to a **high volume of product returns**, affecting both customer satisfaction and retailer profitability.

Moreover, the existing systems often lack dynamic interactivity and real-time rendering capabilities. Users cannot view garments from different angles, see how fabrics behave in motion, or make detailed adjustments based on their preferences. The absence of such features leads to **low user engagement**, making the virtual try-on experience feel artificial and unconvincing. Additionally, the graphics quality and fit simulation are often limited due to hardware constraints and inadequate AR frameworks, especially on lower-end consumer devices.

Retailers also face challenges in implementing these systems, particularly smaller brands with limited budgets and technical resources. **Creating and maintaining 3D models** for every clothing item is a resource-intensive task, and integrating AR features into ecommerce platforms can be complex. These limitations make existing solutions difficult to scale or personalize, ultimately restricting their widespread adoption and effectiveness in solving real-world problems in fashion retail. Therefore, there is a pressing need for an **advanced AR-based system** that overcomes these challenges by offering personalized 3D avatars, real-time virtual tryon experiences, and dynamic interaction features. Such a system would not only reduce return rates and boost customer confidence but also support sustainable fashion practices and open new opportunities for brands to engage users more effectively.

3.4 SOFTWARE SPECIFICATIONS

The proposed Augmented Reality (AR) system for the fashion industry is built using a combination of modern web, mobile, AR, and AI technologies to ensure seamless performance, cross-platform compatibility, and an immersive user experience. The frontend is developed using frameworks like React.js, Flutter, or React Native to support both web and mobile platforms. AR functionalities are integrated using ARKit, ARCore, and web-based libraries like AR.js and A-Frame. The backend is powered by Node.js or Django, handling data management, user operations, and API processing. For data storage and real-time sync, Firebase or MongoDB is used. The system also incorporates machine learning through libraries like TensorFlow or PyTorch for smart recommendations, and computer vision tools like OpenCV and MediaPipe for accurate body and face tracking. Together, these software components form a robust and scalable AR platform tailored for the fashion retail environment.

1. Frontend Development:
    a. React.js / Vue.js: Used for building dynamic, user-friendly web interfaces. Enables fast rendering and component-based design for better performance and maintainability.
    b. Flutter / React Native: Frameworks for building high-performance, crossplatform mobile apps. They allow deployment to both Android and iOS from a single codebase.
2. Augmented Reality SDKs:
    a. ARKit (iOS) and ARCore (Android): These SDKs enable native AR features such as motion tracking, surface detection, and 3D object placement.
    b. AR.js / A-Frame / Three.js: These libraries support web-based AR experiences, allowing users to try on clothing directly from a browser without needing a mobile app.
3. Backend Development:
    a. Node.js / Express.js: JavaScript-based environment for handling serverside operations, managing APIs, and ensuring real-time performance.
    b. Django / Flask: Python-based frameworks suitable for managing user data and integrating AI features with strong security and modularity.
4. Database:
    a. Firebase: Offers real-time database services, user authentication, and cloud storage, ideal for responsive AR applications.
    b. MongoDB: A NoSQL database designed to store flexible, JSON-like documents—perfect for handling user profiles and fashion assets.
5. Machine Learning Libraries:
    a. TensorFlow / PyTorch: Provide AI-powered insights by learning user preferences and suggesting appropriate styles, colors, or sizes.

6. Computer Vision Libraries:
    a. OpenCV / MediaPipe: Enable facial recognition, gesture detection, and body tracking to create accurate avatars and ensure garments align with the user's posture and movement.

3.5 STATISTICAL METHODS

1. **High Personalization**: Users get realistic 3D avatars built from their actual measurements and facial features, ensuring a truly individualized experience.
2. **Improved Fit Accuracy**: Real-time AR rendering simulates fabric drape and movement, reducing sizing errors and mismatches.
3. **Enhanced Engagement**: Interactive gesture and body tracking allow users to rotate, zoom, and pose, making the shopping experience more immersive and fun.
4. **Cross-Platform Compatibility**: Built on ARKit, ARCore, and WebAR, the system runs smoothly on iOS, Android, tablets, browsers, and AR-enabled mirrors.
5. **Reduced Return Rates**: More accurate visualizations lead to better purchase decisions, cutting down on costly product returns and logistics.
6. **AI-Driven Recommendations**: Machine learning analyses user data to suggest optimal styles, sizes, and outfit combinations, boosting conversion rates.
7. **Scalability for Retailers**: Cloud-based asset management and modular architecture make it easy for brands of any size to integrate and expand.
8. **Sustainable Practices**: Virtual try-ons eliminate the need for physical samples and lower waste associated with over-production and returns.
9. **Data Insights & Analytics**: Captured usage patterns and preferences provide valuable marketing and inventory-planning insights

CHAPTER 4 SYSTEM METHODOLOGY

4.1 ARCHITECTURE

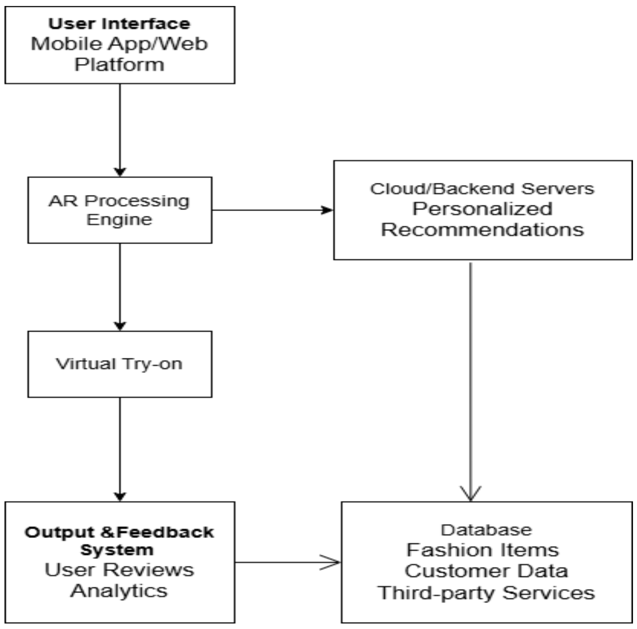

Fig.no 4.1 Architecture Diagram

The system architecture illustrated in the diagram outlines an integrated ARbased virtual try-on platform designed to enhance the online shopping experience. It starts with the user interface, which can be accessed via a mobile app or web platform. This interface allows users to interact with the system and initiates the AR experience. The AR processing engine acts as the core computational component, enabling real-time augmented reality visualization. It powers the virtual try-on feature, allowing users to view fashion items on themselves digitally. Simultaneously, the AR engine communicates with cloud or backend servers that provide personalized recommendations based on user data and interactions. These servers utilize a comprehensive database that contains fashion items, customer data, and information from third-party services. The cloud system ensures dynamic delivery of content and scalability. After users experience the virtual try-on, their feedback and behavior are captured through an output and feedback system. This system collects user reviews and performs analytics to evaluate performance and satisfaction. The insights gathered are sent back to the database, enabling a continuous feedback loop to improve

recommendations and overall experience. This architecture ensures a seamless, intelligent, and user-centric virtual shopping journey, integrating AR, data analytics, and personalized service.

1. User Interface: The user interface is the primary access point for users, available on both mobile applications and web platforms. It facilitates user interaction with the system, allowing them to browse items, select fashion products, and initiate the virtual try-on feature. The design is user-friendly, ensuring ease of navigation and seamless user experience. It also acts as a bridge between the user and the AR engine. 2. AR Processing Engine: This is the core computational module responsible for handling all augmented reality functionalities. It processes real-time data from the user's device (camera, sensors, etc.) to overlay virtual fashion items onto the user's image. The engine ensures high performance and visual accuracy to enhance the realism of the try- on. It also handles alignment, lighting, and movement synchronization.

3. Virtual Try-on: The virtual try-on component allows users to see how clothing or accessories will look on them without physically wearing them. Using AR, the selected fashion items are displayed on the user in real time through their device screen. It offers a highly interactive and engaging experience, helping users make more informed purchase decisions. This feature significantly reduces return rates in online shopping.

4. Cloud/Backend Servers: These servers manage data-intensive operations such as user authentication, session management, and delivering personalized content. They process user data, preferences, and past interactions to generate fashion recommendations tailored to each user. By offloading heavy computations to the cloud, the system remains fast and responsive on user devices. The servers also communicate with the database for up-to-date product and user information.

5. Database: The database is a centralized repository that stores all essential data for the system's functionality. It includes product catalogs, user profiles, interaction histories, and data from third-party fashion providers. This data supports personalization algorithms and ensures the AR experience reflects the latest inventory. The structure of the database is optimized for fast retrieval and secure storage.

6. Output & Feedback System: This system collects and analyzes user feedback, including reviews, ratings, and engagement patterns. It helps in evaluating the effectiveness of the AR try-on feature and the quality of product recommendations. By analyzing behavioral data, the system gains insights into user preferences and

satisfaction levels. The feedback is vital for refining user experience and system performance.

7. Feedback Loop and System Learning: The insights gathered from the output and feedback system are routed back into the database and processing engines. This creates a feedback loop that continuously improves recommendation accuracy and AR realism. It enables adaptive learning, allowing the system to evolve based on user behavior over time. This loop ensures the platform remains relevant and user-centric.

4.2 ALGORITHM

The algorithm powering the AR-based virtual try-on system works through a Series of integrated steps that blend computer vision, 3D modelling, and real-time AR rendering. It begins by capturing live video input through the user's device camera. Using computer vision libraries like OpenCV and MediaPipe, the system detects facial landmarks, body measurements, and posture. These inputs are then used to generate a realistic 3D avatar that reflects the user's shape and appearance. This avatar is rendered using 3D graphics tools like Three.js or A-Frame, allowing it to serve as a digital twin of the user.

Next, the user selects clothing items from a digital catalogue, which contains pre designed 3D garment models. These models are dynamically mapped onto the avatar, scaled, and positioned to fit accurately based on the detected measurements. The system accounts for user movement and gesture input to adjust the clothing in real time, simulating natural fabric behaviour. Using AR platforms like ARKit or ARCore, the final dressed avatar is then overlaid onto the user's real-world environment, allowing them to see a live try-on experience from multiple angles.

To further personalize the experience, an AI module built with TensorFlow or PyTorch analyses user preferences, style history, and body data to recommend suitable clothing options. This adds a layer of intelligence to the system, making it more interactive and usercentric. Overall, the algorithm ensures that users get a responsive, visually accurate, and immersive virtual try-on experience that closely mimics in-store fitting, accessible via mobile or web devices.

4.3 SYSTEM DESIGN

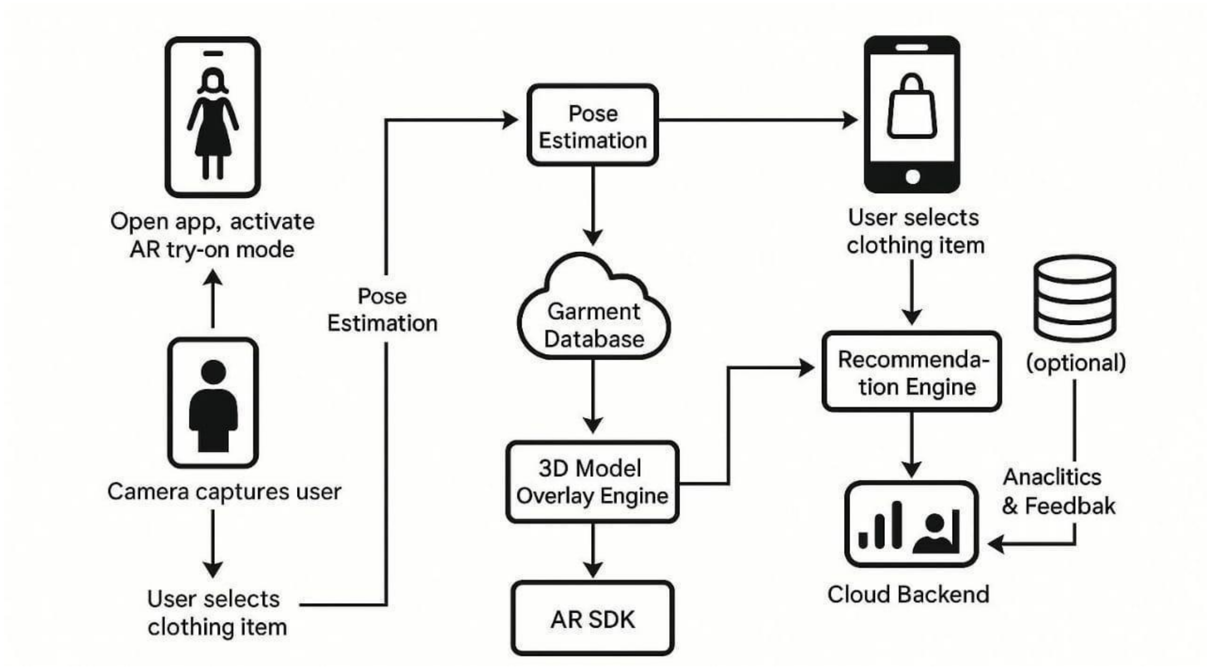

Fig no 4.3 System Design

The system design of the AR-based virtual try-on platform integrates advanced technologies to offer users a seamless and interactive fashion experience. At the front end, users access the platform through a mobile app or web interface, where they can browse products and initiate the virtual try-on. The interface is connected to the AR processing engine, which uses the device camera and motion tracking to project fashion items onto the user in real time. The AR engine incorporates a virtual try-on module that overlays garments accurately based on body posture and movement. This component is connected to cloud-based backend servers responsible for handling user data, executing recommendation algorithms, and managing application logic. These servers communicate with a central database that stores fashion items, customer profiles, and third-party service data. A recommendation engine leverages this data to provide personalized clothing suggestions. After each session, the output and feedback system collects user behaviour data, such as preferences, reviews, and session metrics. This data feeds back into the system to refine recommendations and improve performance. The overall architecture forms a continuous feedback loop, ensuring the platform becomes smarter, more efficient, and more user-centric over time, resulting in a highly engaging and personalized shopping journey.

1. User Interface (Mobile/Web App): Users interact with the system through an intuitive app or web interface. It enables browsing, try-on initiation, and feedback submission. The UI ensures seamless interaction with the AR engine and backend services.
2. AR Processing Engine: This engine processes real-time visual data from the camera. It renders 3D clothing overlays onto the user using motion tracking and body detection. Ensures responsive and realistic AR visualization.
3. Virtual Try-on Module: Embedded in the AR engine, it simulates clothes on the user's live image. It adapts garments based on body movements and angles. Enhances engagement by allowing real-time item switching and fit checking.
4. Cloud/Backend Servers: Handles heavy computations like recommendation processing and user session handling. Ensures fast response time and scalability. Connects the AR engine with the database and other core services.
5. Database: Stores fashion products, user profiles, and third-party information. Supports recommendation logic and keeps AR overlays in sync with the inventory. Regularly updated with real-time interaction data.
6. Personalized Recommendation Engine: Uses AI to suggest products based on user history, preferences, and feedback. Improves shopping relevance and satisfaction. Integrated with the UI and backend.
7. Output & Feedback System: Captures metrics like try-on duration, item preference, and user ratings. Analyse this data for system performance and user experience insights. Feeds result into the database and recommendation engine.
8. Feedback Loop: Creates a cycle of continuous learning and optimization. Enhances AR accuracy, content personalization, and recommendation precision. Makes the system adaptive and user-centric.

4.3.1 DATAFLOW DIAGRAM

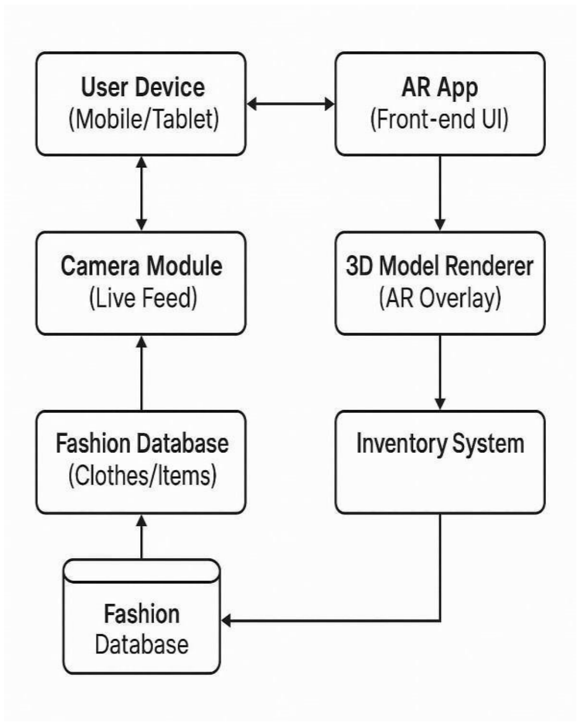
User Device (Mobile/Tablet)
AR App (Front-end UI)
Camera Module (Live Feed)
3D Model Renderer (AR Overlay)
Fashion Database (Clothes/Items)
Inventory System
Fashion Database

Fig no 4.3.1 Dataflow Diagram

The data flow diagram models how data moves through the AR-based virtual try-on platform. It includes four main processes, three data stores, and the external entity the user. Each element works in tandem to deliver a seamless and intelligent shopping experience through augmented reality.

1. External Entity: User

The user is the primary actor and external entity who interacts with the system. The user initiates actions such as browsing items, selecting garments for try-on, viewing personalized recommendations, and providing feedback. The user sends inputs into the system and receives visual AR output and suggestions.

2. Process 1.0: Browse and Select Items

This process starts when the user accesses the mobile or web application. Users browse the available fashion inventory and select items they are interested in. The selected item, along with the live camera feed, is passed to the next process for augmented reality processing. This process queries the Fashion Item Database (D1) to fetch realtime product images, 3D models, and metadata.

3. Process 2.0: AR Try-On Processing

The selected item and camera input are received by the AR Try-On Processing module. This process overlays the chosen clothing item onto the user's live image using AR techniques such as motion tracking and 3D rendering. It fetches garment data from D1 as needed. The visual output is sent back to the user interface, allowing the user to view the virtual try-on in real time. Simultaneously, this process generates session logs, such as how long the item was tried on, movement tracking accuracy, and user engagement level. These logs are passed to Process 4.0 for analysis.

4. Process 4.0: Capture Feedback and Analytics

This process is responsible for collecting feedback and behavioural data. It gathers information from the AR session, including session duration, user actions, likes/dislikes, and post-try-on behaviour. Additionally, users may provide explicit feedback such as ratings or written reviews. All of this data is stored in the Analytics & Feedback Repository (D3). This repository supports continual system improvement.

5. Process 3.0: Personalized Recommendations

Based on user profile information stored in User Profile Database (D2) and the insights from Analytics & Feedback Repository (D3), this process uses machine learning or rule-based algorithms to generate tailored product recommendations. These recommendations are delivered back to the user via the interface, improving shopping relevance and satisfaction.

6. Data Stores:

- D1: Fashion Item Database: Contains product images, AR models, sizing data, and metadata used for try-on.
- D2: User Profile Database: Includes demographic info, preferences, shopping history, and style behaviour.
- D3: Analytics & Feedback Repository: Houses data from try-on sessions and user feedback, fuelling system optimization and personalization.

The DFD effectively captures the core data flows and relationships in the system. It illustrates how user inputs are transformed into rich AR experiences and how backend intelligence personalizes shopping. Each component works in unison, powered by feedback loops and realtime data processing, to offer an engaging and data-driven virtual try-on platform.

4.3.2 UML DIAGRAMS

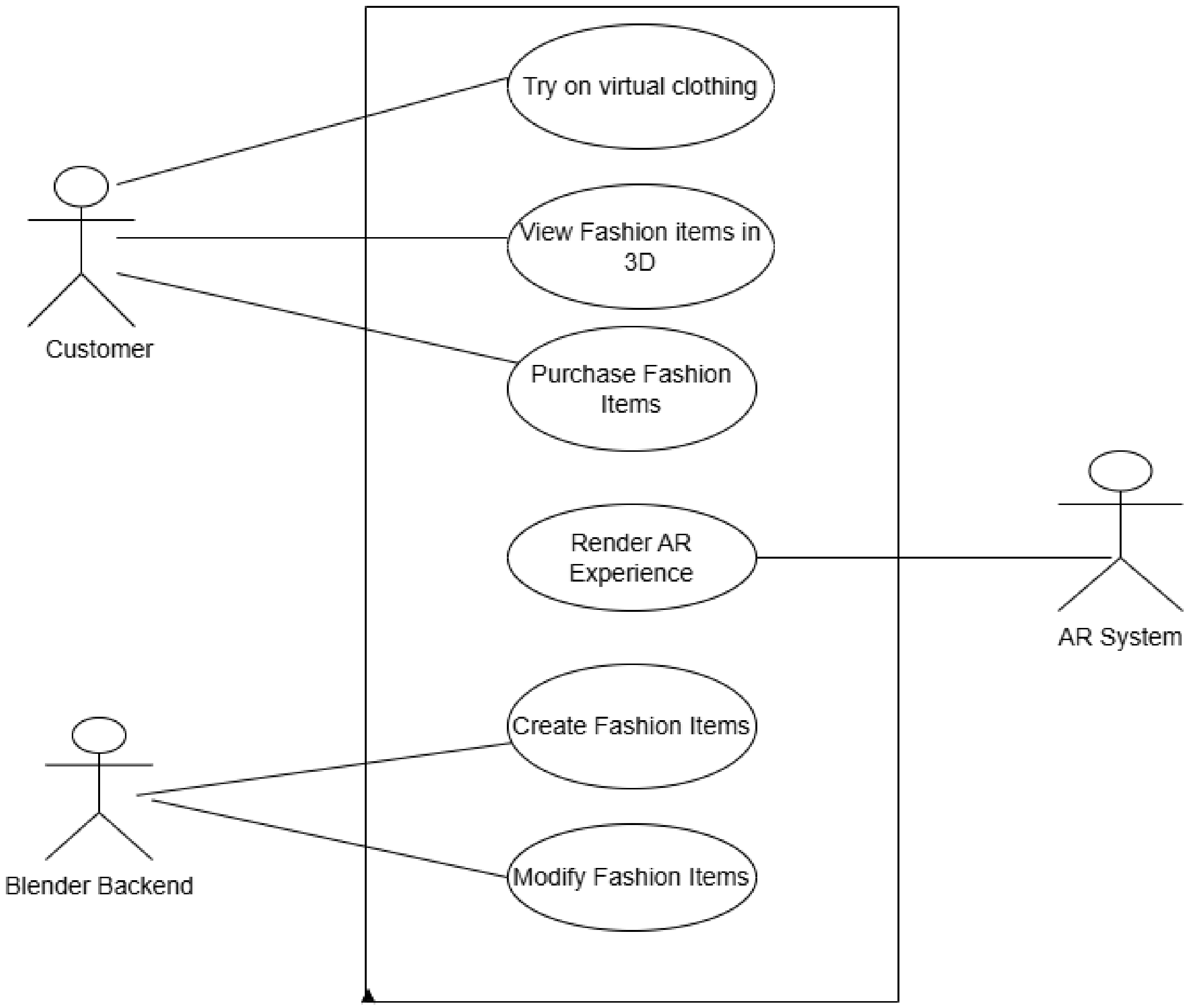

Fig no 4.3.2.1 Use case diagram

The use case diagram for the project *"Augmented Reality in the Fashion Industry"* visually represents the interaction between users and the system's core functionalities. It identifies two primary actors: the User and the Admin. The User, typically a customer, interacts with the system to explore, try on, and purchase fashion items using augmented reality features. The Admin, usually a designer or system manager, is responsible for uploading, modifying, and managing the digital fashion inventory.

The most prominent use case from the user's perspective is the ability to try on virtual clothing. This allows the user to select items from the catalogue and view them on a live camera feed or personalized 3D avatar in real-time. The AR system overlays the clothing accurately using body tracking and spatial alignment, creating an immersive try-on experience. Another key functionality is the ability to view fashion items in 3D, where users can interact with a detailed model of garments by rotating, zooming, and inspecting textures and fit before trying them on. Once satisfied, users can proceed to purchase fashion items, integrating traditional e-commerce operations like cart management, checkout, and payment gateway access.

The render AR experience use case is a technical functionality that handles real-time rendering of the garments over the user’s image or avatar. It utilizes AR frameworks like ARCore, ARKit, or AR.js, depending on the platform, to support smooth performance and accurate display. On the administrative side, the create fashion items use case allows the admin to upload new 3D garments designed using tools like Blender. These digital assets are tagged with necessary metadata and stored in the system’s fashion catalogue. The modify fashion items use case provides the admin with the ability to update or edit existing items—changing the design, texture, or descriptive data based on user feedback or seasonal trends.

All use cases are interconnected through simple relationships with the actors, indicating direct interaction. The User accesses frontend features like AR try-ons and purchasing, while the Admin operates backend systems to manage the digital inventory. This structure supports the overall goal of the system: to offer a highly engaging, personalized, and efficient shopping experience using AR technology. It also enables easy content management for designers and administrators, ensuring that the platform remains up to date with the latest fashion trends and user preferences.

In summary, this use case diagram outlines the core functional scope of the application. It bridges user interaction with technical backend processes, integrating AR, 3D modelling, and e-commerce into a unified system. It reflects the project’s objective to enhance customer satisfaction, promote sustainable shopping, and deliver a futureready retail experience through the power of augmented reality.

CLASS DIAGRAM

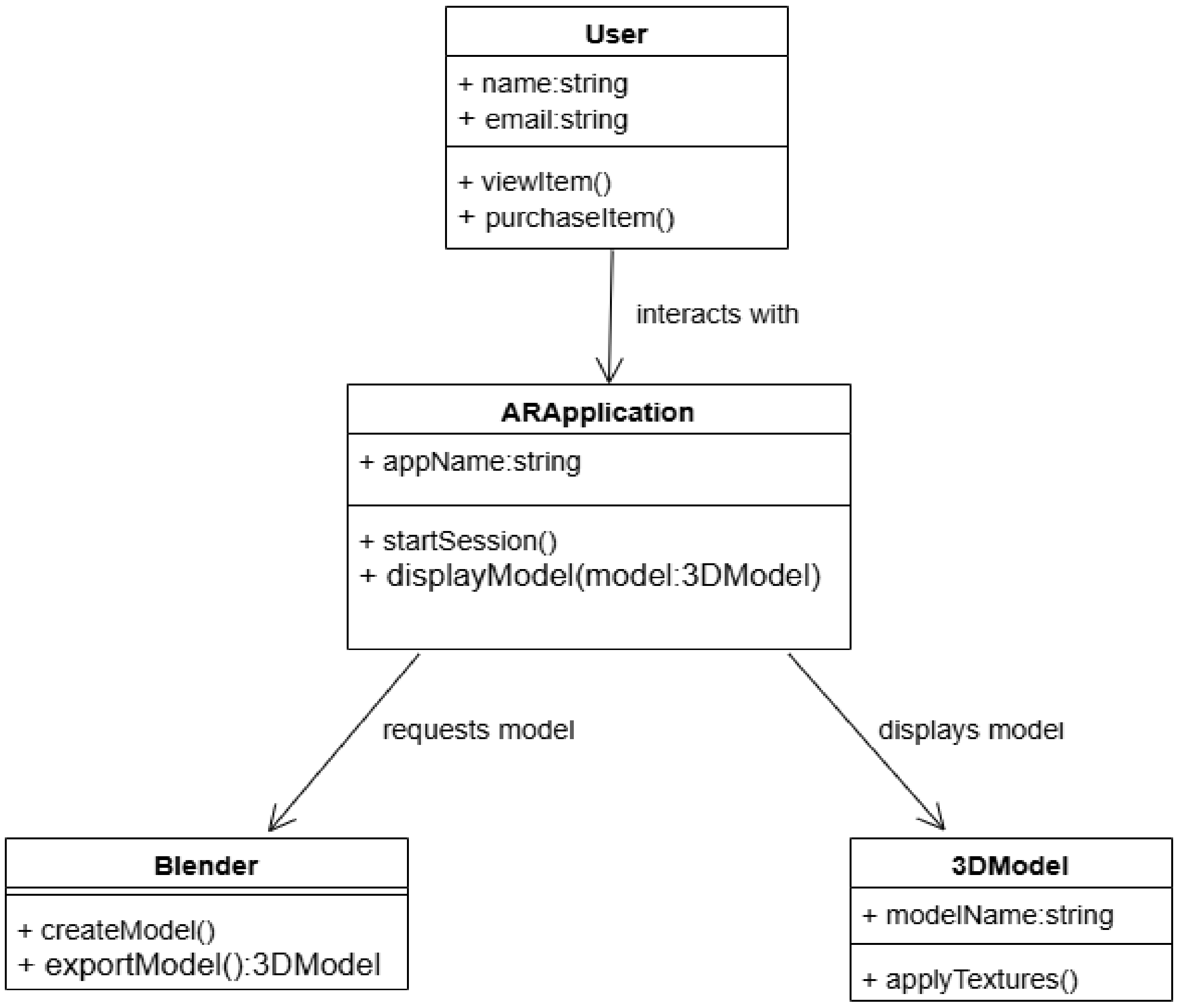

Fig no 4.3.2.2 Class Diagram

The class diagram in the project *"Augmented Reality in the Fashion Industry"* represents the structural blueprint of the AR-based virtual try-on system. It outlines the main classes, their attributes, methods, and the relationships among them, helping developers visualize how different components interact within the application. Key classes include the User class, which stores customer details like user ID, name, and 3D avatar; the Garment class, which holds information about clothing items such as garment ID, type, size, texture, and 3D model; and the Virtual Try On Engine, which is responsible for rendering garments onto the user's avatar in real time. Additional components like the Camera Module capture the user's body or facial data for accurate AR overlay, while the Recommendation Engine suggests outfits based on user preferences and past behavior. These classes are connected through logical associations, such as a user trying on multiple garments or the engine interacting with the camera for real-time feedback. This class diagram is crucial for maintaining a clear and scalable software architecture, enabling an efficient and immersive virtual shopping experience through augmented reality.

SEQUENCE DIAGRAM

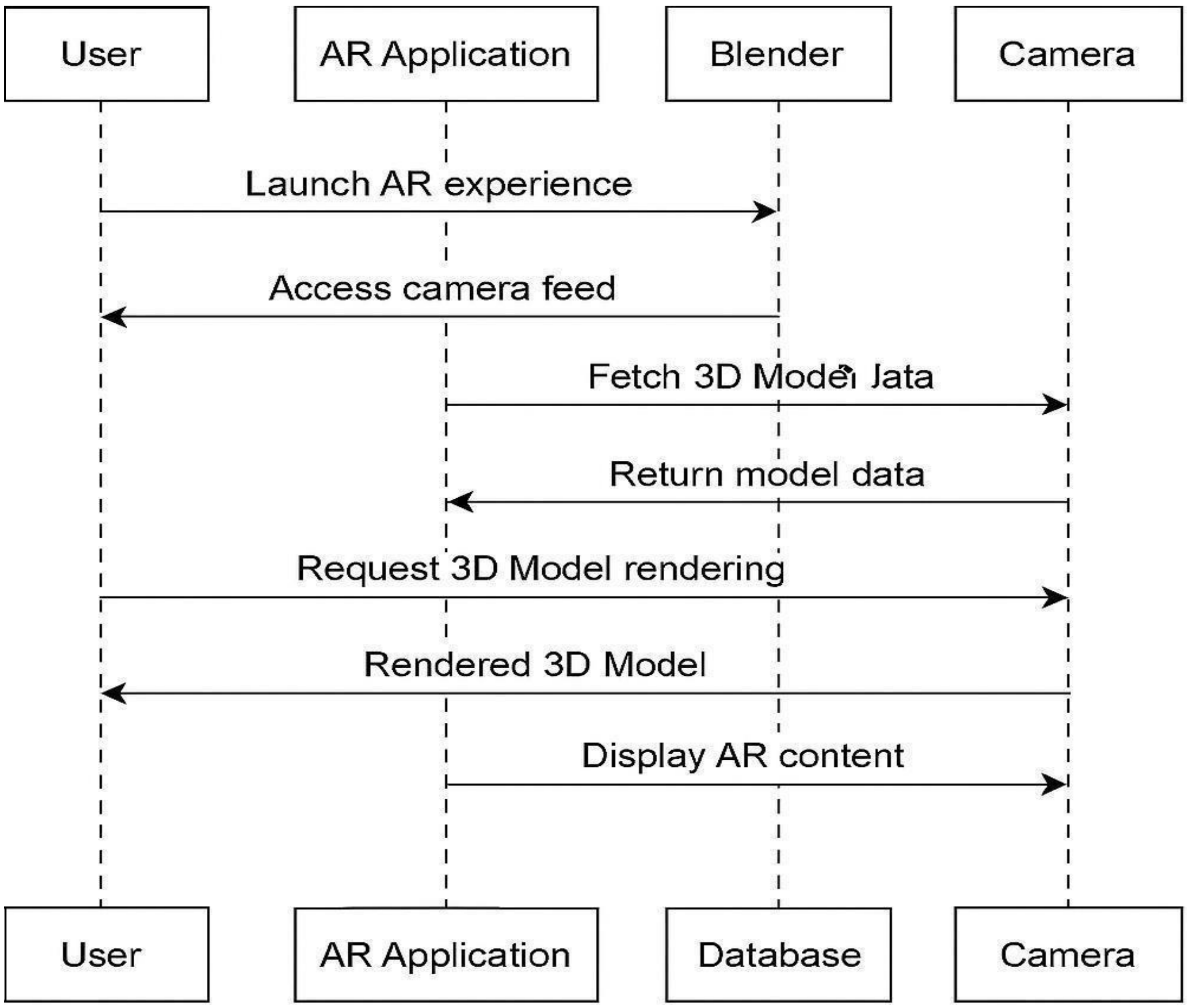

Fig no 4.3.2.3 Sequence Diagram

The sequence diagram in the project *"Augmented Reality in the Fashion Industry"* illustrates the flow of interactions between system components during a virtual tryon session. It shows how objects communicate in a time-ordered sequence to accomplish a specific function. The process typically begins with the User initiating a virtual try-on request through the application interface. This request is sent to the Application Controller, which then calls the Camera Module to capture real-time body or facial data. The captured input is forwarded to the Virtual Try On Engine, which processes the data and overlays selected garments from the Garment Database onto the user's 3D avatar. If personalization is required, the Recommendation Engine may be triggered to suggest outfits based on the user's preferences or browsing history. Finally, the rendered output is displayed to the user in real-time on the screen. Each interaction is shown in the form of messages passed between objects along a timeline, making it easy to understand the flow of control and data. This diagram plays a vital role in ensuring all components interact smoothly, supporting real-time AR rendering and enhancing the user experience by providing seamless, interactive virtual try-ons.

4.3.3 CONTEXT DIAGRAM

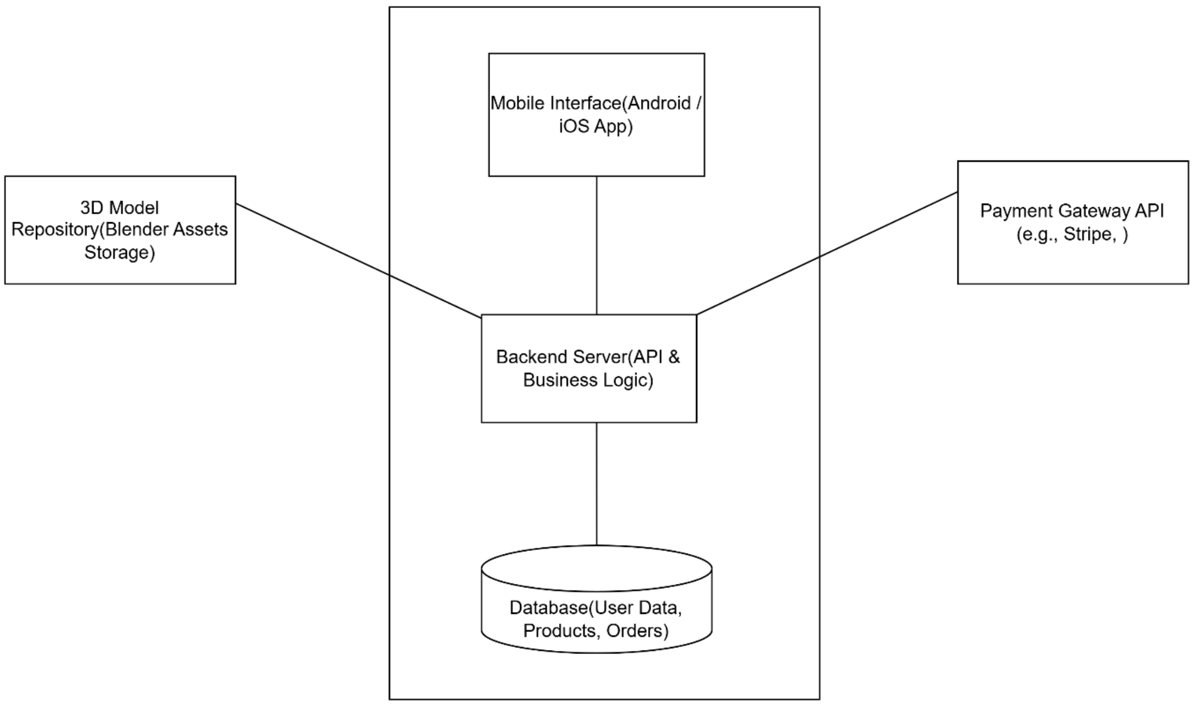
Mobile Interface(Android / iOS App)
3D Model Repository(Blender Assets Storage)
Payment Gateway API (e.g., Stripe,)
Backend Server(API & Business Logic)
Database(User Data, Products, Orders)

Fig.no 4.3.3 Context Diagram

The above context diagram illustrates the high-level architecture of the AR Fashion Application. At the centre is the main system, which integrates multiple components that support the augmented reality-based virtual try-on experience for users. Within this system, the

Mobile Interface serves as the user-facing component, allowing individuals to browse, try on clothes virtually, and make purchases. It communicates directly with the Backend Server, which handles business logic, manages user sessions, and processes requests.

The Backend is responsible for interacting with both the Database and external services. The Database stores essential data such as user profiles, product information, 3D model references, and purchase history. To render virtual clothing and avatars, the backend fetches assets from the 3D Model Repository, where garments created using tools like Blender are stored and maintained. Additionally, when a user initiates a purchase, the backend communicates with a Payment Gateway API to securely handle transactions and return confirmation details to the mobile app.

This architecture ensures smooth communication between internal components while efficiently managing external dependencies. The modular design enables scalability, ease of maintenance, and a seamless augmented reality shopping experience.

CHAPTER 5
EXPERIMENTATION AND ANALYSIS

5.1 EXPERIMENTATION

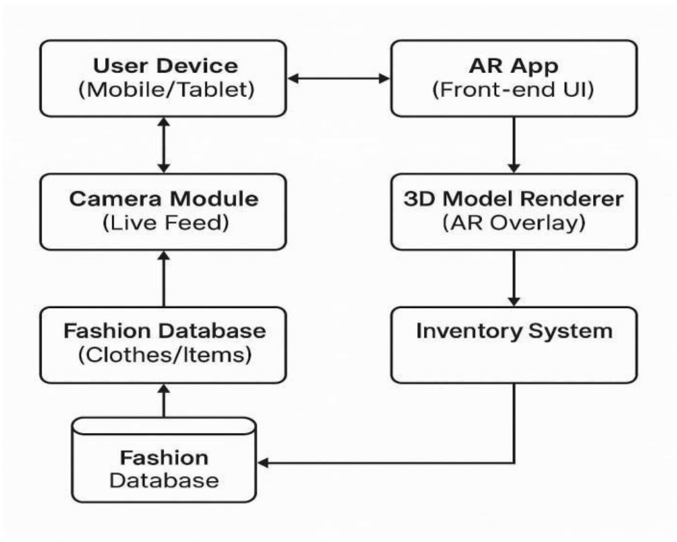

Fig.no 5.1 Experimentation

The diagram represents the system architecture of an Augmented Reality (AR) application designed for the fashion industry, specifically focusing on virtual try-ons. At the core of this system is the User Device, typically a mobile phone or tablet, which serves as the interaction point for the customer. The AR App, installed on the device, provides the frontend interface that allows users to browse clothing items, initiate virtual try-ons, and view ARrendered outfits. Once the user selects an item to try on, the Camera Module activates to capture a live video feed of the user. This feed is then processed by the 3D Model Renderer, which overlays the selected garments onto the user's image using augmented reality technology. The 3D Model Renderer works in conjunction with the Inventory System, which manages the availability and details of each clothing item. This system ensures that only in-stock items are available for virtual try-on and helps maintain accurate product data. Meanwhile, the Fashion Database acts as the central repository for all clothing models, textures, and item details. It supplies the Camera Module with appropriate digital assets needed for rendering. When new items are added or existing ones are updated, the changes are reflected in the Fashion Database through synchronization with the Inventory System.

Overall, the flow of data moves from the user’s selection on the device, through the AR App, and into backend systems that supply visual and inventory information. The camera captures the live input, which is processed and enhanced with AR overlays to create a real-time virtual try-on experience. This system enhances user engagement by allowing them to see how garments would look on them without physical trials, improves shopping convenience, and reduces return rates—making it a powerful tool in modern fashion retail.

SOURCE CODE

```
const    ViewerBG    =
'#eee'; const ViewerUI
= {     canvasWrapper:
document.getElementById('viewerCanvasWrapper'),
cubeWrapper:
document.getElementById('orientCubeWrapper'),     toggleZoom:
document.getElementById('toggleZoom'),     togglePan:
document.getElementById('togglePan'),     toggleOrbit:
document.getElementById('toggleOrbit'),     resetBtn:
document.getElementById('resetBtn'),
   toggleModelBrowser:          document.getElementById('toggleModelBrowser'),
modelBrowser: document.getElementById('modelBrowser'),
   modelBrowserContent:          document.getElementById('modelBrowserContent'),
fileInput: document.getElementById('fileInput'),
   explodeSliderWrapper: document.getElementById('explodeSliderWrapper'),
explodeSlider:
document.getElementById('explodeSlider'),     toggleExplode:
document.getElementById('toggleExplode'),     toggleShare:
document.getElementById('toggleShare'),     shareSidebar:
document.getElementById('shareSidebar'),     loader:
document.getElementById('loader'),
  toggleMeasure: document.getElementById('toggleMeasure'),
                                                 loaderInfo:
document.getElementById('loaderInfo'),    backToHome:
document.getElementById('backToHome'),    webglContainer:
document.getElementById('webglContainer'),    downloadScreen:
document.getElementById('downloadScreen'),    explodeFace:
document.getElementById('explodeFace')
};
function setItemSelected(ele, bool) { if
(bool) {
ele.classList.add('item-selected');
  } else {            ele.classList.remove('item-
selected');
  }
}

function toggle(ele) { if
(ele.getBoundingClientRect().height >
0) {       ele.style.display = 'none';
return false;    } else { ele.style.display
        =
'block';       return true;
  }
}

function    toggleThrough(ele,    through,    cb,
selected=true)  {  through.onclick  =  ()  =>  {  let
bool = toggle(ele);             selected     &&
setItemSelected(through,    bool);    cb    &&
cb(bool);
  }
}
```

```
function          show(ele)        {
ele.style.display = 'block';
}

function          hide(ele)        {
ele.style.display = 'none';
}               function
Viewer() {

   ViewerUI.downloadScreen.onclick =
function() {        const canvas =
renderer.domElement;         renderAll();
        const                        image                         =
canvas.toDataURL("image/png");            const    a     =
document.createElement("a");
        a.href = image.replace(/^data:image\/[^;]/, 'data:application/octet-stream'); a.download
        = "image.png"
        a.click();
    }

   ViewerUI.explodeFace.onclick =
function() {        explodeFace =
this.checked;        resetExplode();
explode();
    }

    let cubeCameraDistance = 1.75;

    let                 cubeWrapper                 =
ViewerUI.cubeWrapper;     let cubeScene
= new THREE.Scene();
    let cubeCamera = new THREE.PerspectiveCamera(70, cubeWrapper.offsetWidth /
cubeWrapper.offsetHeight, 0.1, 100); let
    cubeRenderer = new
THREE.WebGLRenderer({        alpha: true,        antialias:
true,
        preserveDrawingBuffer: true
    });

    cubeRenderer.setSize(cubeWrapper.offsetWidth, cubeWrapper.offsetHeight);
    cubeRenderer.setPixelRatio(window.deivicePixelRatio);
    cubeWrapper.appendChild(cubeRenderer.domElement);


    let materials = [];
            let texts = ['RIGHT', 'LEFT', 'TOP', 'BOTTOM', 'FRONT',
                                                          'BACK'];

   let textureLoader = new THREE.TextureLoader();
    let    canvas = document.createElement('canvas');      let ctx
= canvas.getContext('2d');

    let size = 64;
canvas.width = size;
canvas.height = size;

    ctx.font = 'bolder 12px "Open
sans", Arial';     ctx.textBaseline = 'middle';
ctx.textAlign = 'center';

    let mainColor = '#fff'; let
otherColor  =
'#ccc';

    let bg = ctx.createLinearGradient(0, 0,
0,   size);              bg.addColorStop(0,
mainColor);              bg.addColorStop(1,
otherColor);

   for (let i = 0; i < 6; i++) {        if
(texts[i] == 'TOP') {
ctx.fillStyle = mainColor;
} else if (texts[i] == 'BOTTOM')
{          ctx.fillStyle =
otherColor;
      }          else          {
ctx.fillStyle = bg;
      }              ctx.fillRect(0,   0,   size,   size);
ctx.strokeStyle = '#aaa';  ctx.setLineDash([8, 8]);
ctx.lineWidth = 4;       ctx.strokeRect(0,
0,  size,  size);             ctx.fillStyle  =  '#999';
ctx.fillText(texts[i],  size  /  2,  size  /  2);
materials[i] = new THREE.MeshBasicMaterial({
         map: textureLoader.load(canvas.toDataURL())
      });
   }

   let planes = [];

   let planeMaterial = new THREE.MeshBasicMaterial({ side:
THREE.DoubleSide,        color: 0x00c0ff,
transparent: true,       opacity: 0, depthTest:
false
   });     let planeSize = 0.7;      let planeGeometry = new
THREE.PlaneGeometry(planeSize, planeSize);
```

```
   let a = 0.51;

   let plane1 = new THREE.Mesh(planeGeometry, planeMaterial.clone());
plane1.position.z = a;     cubeScene.add(plane1);      planes.push(plane1);

   let plane2 = new THREE.Mesh(planeGeometry, planeMaterial.clone()); plane2.position.z
   = -a; cubeScene.add(plane2);
   planes.push(plane2);  let plane3 = new
   THREE.Mesh(planeGeometry, planeMaterial.clone());
   plane3.rotation.y = Math.PI / 2; plane3.position.x = a;
   cubeScene.add(plane3); planes.push(plane3);

   let plane4 = new THREE.Mesh(planeGeometry, planeMaterial.clone());
                              plane4.rotation.y = Math.PI / 2; plane4.position.x =
-a;                           cubeScene.add(plane4);      planes.push(plane4);

   let plane5 = new THREE.Mesh(planeGeometry, planeMaterial.clone());
                             plane5.rotation.x = Math.PI / 2; plane5.position.y =
a;                           cubeScene.add(plane5);      planes.push(plane5);

   let plane6 = new THREE.Mesh(planeGeometry,
planeMaterial.clone());     plane6.rotation.x = Math.PI / 2;     plane6.position.y = -a;
cubeScene.add(plane6);      planes.push(plane6);

   let            groundMaterial            =            new
THREE.MeshBasicMaterial({        color: 0xaaaaaa
   });
   let groundGeometry = new THREE.PlaneGeometry(1, 1);     let
groundPlane = new THREE.Mesh(groundGeometry,
groundMaterial);     groundPlane.rotation.x = -Math.PI / 2;  groundPlane.position.y
   = -0.6;

  cubeScene.add(groundPlane);
```

```
        let cube = new THREE.Mesh(new THREE.BoxGeometry(1, 1, 1), materials);
  cubeScene.add(cube);

  function      updateCubeCamera()  {     cubeCamera.rotation.copy(camera.rotation); let
    dir = camera.position.clone().sub(controller.target).normalize();
    cubeCamera.position.copy(dir.multiplyScalar(cubeCameraDistance));
  }

  let activePlane = null; cubeRenderer.domElement.onmousemove =

  function(evt) {

    if                    (activePlane)                    {
activePlane.material.opacity = 0;
activePlane.material.needsUpdate = true;          activePlane
= null;
    }

    let          x
= evt.offsetX;
let y = evt.offsetY; let
size =
cubeRenderer.getSi
ze(new
THREE.Vector2());
let mouse = new
THREE.Vector2(x /
size.width * 2 - 1, -y
/ size.height * 2 + 1);

    let raycaster = new THREE.Raycaster();          raycaster.setFromCamera(mouse,
cubeCamera);
```

```
        let intersects = raycaster.intersectObjects(planes.concat(cube));

        if (intersects.length > 0 && intersects[0].object != cube)
{           activePlane = intersects[0].object;
activePlane.material.opacity = 0.2;
activePlane.material.needsUpdate = true;
      }
    }
    let startTime =
    0;    let duration =
    500;
    let        oldPosition        =        new
    THREE.Vector3(); let newPosition
    = new THREE.Vector3(); let
   play = false;

    cubeRenderer.domElement.onclick = function(evt) {

        cubeRenderer.domElement.onmousemove(evt);

        if (!activePlane || hasMoved) { return
false;
        }

        oldPosition.copy(camera.position);

        let       distance       =       camera.position.clone().sub(controller.target).length();
newPosition.copy(controller.target);

        if (activePlane.position.x !== 0) {                    newPosition.x +=
activePlane.position.x < 0 ? -distance : distance;
        } else if (activePlane.position.y !== 0) {                newPosition.y +=
activePlane.position.y < 0 ? -distance : distance;
```

```
        } else if (activePlane.position.z !== 0) {                newPosition.z +=
activePlane.position.z < 0 ? -distance : distance;
        }

        //play = true;
        //startTime          =          Date.now();     camera.position.copy(newPosition);
    }

    cubeRenderer.domElement.ontouchmove = function(e)
        {  let        rect       = e.target.getBoundingClientRect(); let
        x              =          e.targetTouches[0].pageX       - rect.left;
      let y = e.targetTouches[0].pageY - rect.top;
    cubeRenderer.domElement.onmouse    move({
                        offsetX: x,         offsetY:
    y
        });
    }

    cubeRenderer.domElement.ontouchstart = function(e)
{         let rect = e.target.getBoundingClientRect();
          let x = e.targetTouches[0].pageX - rect.left;
          let y =
e.targetTouches[0].pageY - rect.top;         cubeRenderer.domElement.onclick({
            offsetX: x,             offsetY:
y
        });
    }

    ViewerUI.fileInput.addEventListener('input',
function(evt) {         let file = evt.target.files[0];         if
(file) {
            show(ViewerUI.loader);
            ViewerUI.loaderInfo.innerHTML =
'Reading file...';            let reader = new
FileReader();          reader.onload = function(e) {             loadModel(e.target.result);
            }
            reader.onerror = function(err) {
                ViewerUI.loaderInfo.innerHTML = 'Error reading file! See console for more
    info.';            console.error(err);
            }
            reader.readAsDataURL(file);
                }
            });

    hide(ViewerUI.loader); let

    hasMoved = false;

    function antiMoveOnDown(e) {
hasMoved = false;
    }
    function antiMoveOnMove(e) {
hasMoved = true;
    }

    window.addEventListener('mousedown', antiMoveOnDown, false);
window.addEventListener('mousemove', antiMoveOnMove, false);
window.addEventListener('touchstart', antiMoveOnDown, false);
window.addEventListener('touchmove', antiMoveOnMove, true);

let showExploded =
false;     let
explodeFactor = 0;     let
explodeFace =
      !true;

    toggleThrough(ViewerUI.explodeSliderWrapper, ViewerUI.toggleExplode,
(bool) => {         if (!bool) {            resetExplode();
      }   else   {                                   explodeFactor   =
ViewerUI.explodeSlider.value; explode();
            }
         });

    function
        resetExplode() { let
        temp =
        explodeFactor; let
        temp2 =
        explodeFace;
        explodeFace = true;
        explodeFactor = 0;
    explode();
    explodeFace =
    false; explode();
    explodeFactor =
    temp;
        explodeFace = temp2;
    }

    toggleThrough(ViewerUI.shareSidebar,                          ViewerUI.toggleShare);
toggleThrough(ViewerUI.modelBrowser, ViewerUI.toggleModelBrowser);
```

```
   ViewerUI.explodeSlider.oninput =
function() {        explodeFactor =
this.value;        explode();
   }

   function    explode()    { for
(let i = 0; i <
loadedMeshes.length; i++) {            let node
= loadedMeshes[i];            if
(explodeFace) {                        let
defaultPositionArray =
node.defaultPositionArray;                 let positionArray =
node.geometry.attributes.position.array;                 let
normalArray = node.geometry.attributes.normal.array;
let indexArray = node.geometry.index.array;                for (let j =
0; j < indexArray.length; j++) {                      let index
=   indexArray[j]                                              let   position   =   new
THREE.Vector3(defaultPositionArray[index * 3], defaultPositionArray[index * 3 + 1],
defaultPositionArray[index * 3 + 2]);
                let    normal    =    new    THREE.Vector3(normalArray[index    *    3],
normalArray[index *
3 + 1], normalArray[index * 3 + 2]) position.add(normal.multiplyScalar(explodeFactor));
                                                                  positionArray[index
      * 3] = position.x;
      positionArray[index * 3 + 1] = position.y;          positionArray[index
      * 3 + 2] = position.z;

            }
           node.geometry.attributes.position.needsUpdate = true;
   node.geometry.computeBoundingBox();
   node.geometry.computeBoundingSphere();
         } else {
node.position.copy(node.defaultPosition).add(node.defaultPosition.clone().normali
ze().multiplyScalar(explodeFactor));
```

```
            }
        }
    }
    ViewerUI.toggleZoom.onclick     =     function()     {
setZoomMode();
        setItemSelected(selectedModeElement,
false);     selectedModeElement = this;     setItemSelected(this,
true);
    }
    ViewerUI.togglePan.onclick     =     function()     {
setPanMode();
        setItemSelected(selectedModeElement,
false);     selectedModeElement = this;     setItemSelected(this,
true);
    }
    ViewerUI.toggleOrbit.onclick     =     function()     {
setOrbitMode();
        setItemSelected(selectedModeElement,
false);     selectedModeElement = this;     setItemSelected(this,
true);
    }
    ViewerUI.toggleMeasure.onclick =
    function() {    isInMeasureMode =
    !isInMeasureMode; if
    (!isInMeasureMode) {
            lineScene.remove.apply(lineScene,                    lineScene.children);
            spriteScene.remove.apply(spriteScene, spriteScene.children);
        }                setItemSelected(this,
        isInMeasureMode);
    }
    ViewerUI.resetBtn.onclick   =   ViewerUI.backToHome.onclick   =function()   {
    resetAll();
    }
```

```
function
resetAll() {        controller.rese t();
        lineScene.remove.apply(lineScene, lineScene.children);
spriteScene.remove.apply(spriteScene, spriteScene.children);        isInMeasureMode
= false;
        setItemSelected(ViewerUI.toggleMeasure, false);
        ViewerUI.explodeSliderWrapper.style.display
= 'none';        ViewerUI.explodeFace.checked =
false;        explodeFace = false;

setItemSelected(ViewerUI.toggleExplode, false);
                    resetExplode();        resetSelect();
    }
function  updateSelectDom(child) {
                                                    if
(child.itemWrapper) {              if
(child.isSelected) {
                child.itemWrapper.querySelector('.graph-name').style.color = '#03a9f4';
            } else {                child.itemWrapper.querySelector('.graph-
name').style.color = 'inherit';              }

        }
    }
    function setOrbitMode() {
controller.enableZoom = true;
controller.enablePan = true;
controller.enableRotate = true;
controller.mouseButtons = {
            LEFT: THREE.MOUSE.ROTATE,
            MIDDLE: THREE.MOUSE.DOLLY,
            RIGHT: THREE.MOUSE.PAN


    }
```

```
   ; }
   function      setPanMode()  {     controller.enableZoom = false;
      controller.enablePan = true;
controller.enableRotate = false;
controller.mouseButtons = { LEFT:
THREE.MOUSE.PAN,
         MIDDLE: THREE.MOUSE.PAN, RIGHT:
         THREE.MOUSE.PAN
      };
   }
   function setZoomMode() {
controller.enableZoom = true;
controller.enablePan = false;
controller.enableRotate = false;
controller.mouseButtons = { LEFT:
THREE.MOUSE.DOLLY,
         MIDDLE: THREE.MOUSE.DOLLY,
         RIGHT: THREE.MOUSE.DOLLY
      };
   }
   let               wrapper              =
ViewerUI.canvasWrapper;     let scene
= new THREE.Scene();    let camera          =        new
THREE.PerspectiveCamera(70,        wrapper.offsetWidth   / wrapper.offsetHeight, 0.1,
1000);
   let renderer = new
THREE.WebGLRenderer({
antialias: true,     alpha: false
   });
   renderer.setClearColor(new THREE.Color(ViewerBG));
   renderer.autoClear = false;
   renderer.setPixelRatio(window.deivicePixelRatio); let
```

```
    isInMeasureMode = false; let lineScene = new THREE.Scene(); let
    spriteScene = new THREE.Scene(); function makeCircleImage() {
        let  canvas  =  document.createElement('canvas'); let
ctx =
canvas.getContext('2d'); let
size = 32; canvas.width =
size;
canvas.height = size;         let r =
size * 0.8 / 2;               let blur
= size - r; ctx.shadowBlur
= 5;
ctx.shadowColor = '#555'; ctx.fillStyle
= '#fff'; ctx.beginPath();
        ctx.arc(size / 2, size / 2, r, 0, Math.PI
* 2);        ctx.closePath();        ctx.fill();
        ctx.shadowBlur = 0;
ctx.fillStyle = '#009bff';        ctx.beginPath();

        ctx.arc(size / 2, size / 2, r * 0.5, 0,
Math.PI * 2);        ctx.closePath();
ctx.fill();        return canvas;
    }
    let circleTexture = new
THREE.CanvasTexture(makeCircleImage())      let circleMaterial
= new THREE.SpriteMaterial({          map:
circleTexture,
    sizeAttenuation: false
    });
    let    circleSprite     =        new     THREE.Sprite(circleMaterial);
    circleSprite.scale.setScalar(0.08);  let lineMaterial = new
    THREE.LineBasicMaterial({     color:
    0x009bff,     linewidth: 10
    }); let activeLine =
    null;

    renderer.domElement.onclick =
function(evt) {      if (hasMoved) {         return
false;
      }       evt
= evt ||
window.event;       let x =
evt.offsetX;
let y = evt.offsetY;
      let size = renderer.getSize(new THREE.Vector2());         let mouse = new
THREE.Vector2(x / size.width * 2 - 1, -y / size.height
*   2   +   1);                     let   raycaster   =   new   THREE.Raycaster();
raycaster.setFromCamera(mouse, camera);
      let intersects =
raycaster.intersectObjects(loadedMeshes);       if (!isInMeasureMode)
{          resetSelect();
      }       if
(intersects.length > 0) { if
(isInMeasureMode) {
let point = intersects[0].point; if
(!activeLine) {
                let sprite1 = circleSprite.clone();
                                    let sprite2
= circleSprite.clone();
sprite1.position.copy(point.clone());
sprite2.position.copy(point.clone());
spriteScene.add(sprite1);
spriteScene.add(sprite2);
                let         lineGeometry         =         new         THREE.Geometry();
    lineGeometry.vertices.push(sprite1.position,
    sprite2.position);            let line = new
    THREE.Line(lineGeometry, lineMaterial);                line.sprite1
    = sprite1;             line.sprite2 = sprite2;             lineScene.add(line);
    activeLine = line;
            }                       else                        {
    activeLine.geometry.vertices[1].copy(point);

activeLine.geometry.verticesNeedUpdate = true;
makeDistanceSprite();                  activeLine = null;
            }
} else {
            let mesh = intersects[0].object;
mesh.isSelected = true;               updateMeshInteractionMaterial(mesh);
          }
       } else {           if (isInMeasureMode) { if
(activeLine) { lineScene.remove(activeLine);
spriteScene.remove(activeLine.sprite1);
spriteScene.remove(activeLine.sprite2);
activeLine = null;
            }
          }
       }    }     function resetSelect()
{ scene.traverse(child => {
child.isSelected = false;          if
(child.isMesh && child.material) {
updateMeshInteractionMaterial(child);
          }
          updateSelectDom(child);
       });
    }
    renderer.domElement.onmousemove       =    function(evt)    { evt
    = evt || window.event;
```

```
        if (!isInMeasureMode) { return;
        }
        let             x
= evt.offsetX;
let y = evt.offsetY;
        let size = renderer.getSize(new THREE.Vector2());           let mouse = new
THREE.Vector2(x / size.width * 2 - 1, -y / size.height
*    2    +    1); let raycaster = new THREE.Raycaster(); raycaster.setFromCamera(mouse,
camera);
        let intersects =
raycaster.intersectObjects(loadedMeshes);           if
(isInMeasureMode && activeLine) {             if
(intersects.length > 0) {                 let point = intersects[0].point;
activeLine.geometry.vertices[1].copy(point);

activeLine.geometry.verticesNeedUpdate = true;
            }                                         else                                         {
activeLine.geometry.vertices[1].copy(activeLine.geometry.vertices[0]);
activeLine.geometry.verticesNeedUpdate = true;
            }
        }
    }
    /*renderer.domElement.ontouchmove = function(e)
{        let rect = e.target.getBoundingClientRect();
           let x = e.targetTouches[0].pageX - rect.left;
           let y = e.targetTouches[0].pageY - rect.top;
renderer.domElement.onmousemove({
offsetX: x,        offsetY: y
        });
    }*/ renderer.domElement.ontouchstart =
    function(e) {     let rect =
    e.target.getBoundingClientRect();      let x
    = e.targetTouches[0].pageX - rect.left; let y
    = e.targetTouches[0].pageY - rect.top;
    renderer.domElement.onclick({         offsetX: x,
            offsetY: y
        });
    }
    function    makeDistanceSprite()    { let
canvas =
document.createElement('canvas');          let ctx
= canvas.getContext('2d');              let fontsize
= 32;
        ctx.font  =  'bolder  '  +  fontsize  +  'px  "Open  Sans",
Arial';   let v = activeLine.geometry.vertices;   let length =
v[0].clone().sub(v[1]).length().toFixed(1);          let text
= '~ ' + length;        let size = ctx.measureText(text); let
paddingLeft = 20;   let paddingTop = 10;
let margin = 10;
        canvas.width = size.width + paddingLeft * 2 + margin * 2; canvas.height
= fontsize + paddingTop * 2 + margin * 2;

        ctx.shadowBlur = 10; ctx.shadowColor =
'#555';                                  ctx.fillStyle
= '#009bff';
        roundRect(ctx, margin, margin, canvas.width - margin * 2, canvas.height - margin
*  2,  10);                ctx.shadowBlur  =  0;  ctx.fillStyle  =  '#fff';  ctx.textAlign  =  'left';
ctx.textBaseline = 'top';
        ctx.font = 'bolder ' + fontsize + 'px "Open Sans", Arial'; ctx.fillText(text,
        paddingLeft + margin, paddingTop +
    margin);     let texture = new
    THREE.CanvasTexture(canvas);      let sprite = new
    THREE.Sprite(new THREE.SpriteMaterial({         map:
    texture,         sizeAttenuation: false
       }));
let h =
0.7;              sprite.scale.set(0.002 * canvas.width,
0.0025 *
canvas.height).multiplyScalar(h); sprite.position.copy(v[0].clone().add(v[1]).multiplyScalar(0.5));
                                                              spriteScene.add(sprite);
    }
    function roundRect(ctx, x, y, w,
h, r) {              ctx.beginPath();
ctx.moveTo(x + r, y);          ctx.lineTo(x
+ w - r, y);
       ctx.quadraticCurveTo(x + w, y, x + w, y + r); ctx.lineTo(x
+ w, y + h - r);
       ctx.quadraticCurveTo(x + w, y + h, x + w - r, y + h);
ctx.lineTo(x + r, y + h);
       ctx.quadraticCurveTo(x, y + h, x, y + h - r); ctx.lineTo(x, y
+ r);
       ctx.quadraticCurveTo(x, y,
x + r, y);          ctx.closePath();          ctx.fill();
    }
    function
updateMeshInteractionMaterial(mesh)          {       if
(mesh.isHidden) {
           mesh.interactionMaterial.color          =         hiddenColor;
mesh.interactionMaterial.opacity = hiddenAlpha;
       }                        else                        {
    mesh.interactionMaterial.opacity = 1;
      }     if (mesh.isSelected) {
    mesh.interactionMaterial.color =
    selectColor;
```

```
        mesh.itemWrapper.querySelector('.grap                    h-
        name').style.color = '#03a9f4';
       } else {      mesh.itemWrapper.querySelector('.graph-name').style.color
    = 'inherit';
       }
       mesh.interactionMaterial.needsUpdate
    = true;    if (!mesh.isSelected &&
    !mesh.isHidden) {          mesh.material =
mesh.defaultMaterial;
       }  else  {                    mesh.material  =
mesh.interactionMaterial;
       }
    }    function onResize() {
let width = wrapper.offsetWidth; let
height = wrapper.offsetHeight;
renderer.setSize(width, height, false);
             camera.aspect =  width /
height;
camera.updateProjectionMatrix();
    }
    onResize();

wrapper.appendChild(renderer.domElement)
;             window.addEventListener('resize',
onResize,    false);    let    gltfLoader    =    new
THREE.GLTFLoader();  let  loadedScene  =  null;
let loadedMeshes = [];     let d = 5;  let
selectColor=                      new
THREE.Color('#42006b');     let hiddenColor
=  new  THREE.Color('#555');              let
hiddenAlpha = 0.3;

    let interactionMaterial = new
```

```
        THREE.MeshPhongMaterial({ transparent: true, color:
        selectColor, side: THREE.DoubleSide, precision:
        'mediump'
    });
    function loadModel(url)
    {     resetAll();     if
    (loadedScene) {
    scene.remove(loadedSce ne);
                loadedScene =
    null;
          loadedMeshes.length = 0;
}
        show(ViewerUI.loader); ViewerUI.modelBrowserContent.innerHTML =
        '';
ViewerUI.loaderInfo.innerHTML = 'Loading model...';
gltfLoader.load(
          url,                function
onLoad(gltf) {
loadedScene = gltf.scene;              scene.add(gltf.scene);
// After loading the model and adding to scene  const
objectListContainer = document.getElementById('objectList');
const meshObjects
= []; gltf.scene.traverse((child) => {     if (child.isMesh) {
meshObjects.push(child);          const label =
document.createElement('label');          const
checkbox = document.createElement('input');
checkbox.type = 'checkbox';        checkbox.checked
= true;
        checkbox.addEventListener('change', () => { child.visible
= checkbox.checked;
        });
        label.appendChild(checkbox);
label.appendChild(document.createTextNode(child.name || 'Unnamed Mesh'));
```

```
objectListContainer.appendChild(label);
   }
});
   gltf.scene    =    gltf.scene    ||
gltf.scenes[0];          let object =
gltf.scene;
            const box = new THREE.Box3().setFromObject(object);
                                                 const size =
box.getSize(new THREE.Vector3()).length();          const center
= box.getCenter(new THREE.Vector3());
controller.reset();
            object.position.x  +=  (object.position.x  -
center.x);                 object.position.y  +=
(object.position.y          -          center.y);
object.position.z  +=  (object.position.z  -  center.z);
controller.maxDistance  =  size  *  10;  camera.near  =
size / 100;                          camera.far =
size                   *                    100;
camera.updateProjectionMatrix();
camera.position.copy(center);
camera.position.x     +=     size     /     2.0;
camera.position.y     +=     size     /     5.0;
camera.position.z     +=     size     /     2.0;
directionalLight.position.setScalar(size);
camera.lookAt(center); controller.saveState();
gltf.scene.traverse((node) => {                  if
(node.isMesh      &&      node.material)      {
node.geometry.computeBoundingBox();
node.material.side     =     THREE.DoubleSide;
node.material.precision        =        'mediump';
node.material.needsUpdate = true;
                  node.interactionMaterial        =        interactionMaterial.clone();
node.defaultMaterial = node.material;
                    node.defaultPositionArray                                          =
Array.from(node.geometry.attributes.position.array);                   node.defaultPosition
= node.position.clone();
                      loadedMeshes.push(node);
                  } }); let
               content =
               ViewerUI.modelBrowserContent; let counter
               = 0; let parentLevel = 0;   function

               makeSceneGraph(obj) {

                  if (obj.children.length === 0 && !obj.isMesh)
{                        return;
                  }

                         let        itemWrapper        =        document.createElement('div');
                      itemWrapper.classList.add('graph-item-wrapper');

                  let             item             =             document.createElement('div');
item.classList.add('graph-item');

                  itemWrapper.appendChild(item);

                  content.appendChild(itemWrapper); let n
                  = 0;
let obj2 = obj;
while (obj2 != gltf.scene) {
obj2 = obj2.parent;                        n++;
                  }
                  item.style.paddingLeft = n * 1.5 + 'em';
obj.itemWrapper = itemWrapper;

let left = document.createElement('div'); left.classList.add('graph-left'); let right =
document.createElement('div'); right.classList.add('graph-right'); item.appendChild(left);
item.appendChild(right);
```

```
    if (obj.children.length > 0) { parentLevel++;        let
              folder =
document.createElement('div');

                   folder.style.marginRight                    =                    '10px';
 folder.classList.add('graph-folder');
                   folder.innerHTML       =      '<i      class="fa      fa-folder-open"></i>';
 left.appendChild(folder);

                   obj.isFolderOpen       =       true;
 obj.openFolder = function() {
                      folder.innerHTML = obj.isFolderOpen ? '<i class="fa fa-folder-
 open"></i>'
 :      '<i      class="fa      fa-folder"></i>';
 obj.traverse(child => {
 if (obj === child) { return;
      }                                            if (child.itemWrapper) {
 if        (child.parent.isFolderOpen      &&      obj.isFolderOpen)       {
 child.itemWrapper.style.display = 'block';
                           }                             if
(!obj.isFolderOpen) {
child.itemWrapper.style.display = 'none';
                           }
```

```
                            }
                        });
        }
            folder.onclick = () => {            obj.isFolderOpen
            =         !obj.isFolderOpen;
          obj.openFolder();
            }

            for (let i = 0; i < obj.children.length; i++) {
              makeSceneGraph(obj.children[i]);
            }

          }

let name = document.createElement('div');
name.classList.add('graph-name');
name.innerHTML = obj.name || 'None';
left.appendChild(name);

          name.onclick = function()
{                   resetSelect();
obj.traverse(child => {                         child.isSelected
= true;
                  if          (child.isMesh           &&          child.material)           {
updateMeshInteractionMaterial(child);
                  }
                  updateSelectDom(child)
                });
          }

          let visible = document.createElement('div');
```

```
visible.classList.add('graph-visible');                 visible.innerHTML
= '<i class="fa fa-eye"></i>';

                obj.showMesh = function() {                          visible.innerHTML =
obj.isMeshVisible ? '<i class="fa fa-eye"></i>' :
'<i             class="fa              fa-eye-slash"></i>';
                obj.traverse(child
                => {        if
                (child.itemWrapper)
                {          let eye = child.itemWrapper.querySelector('.graph-visible');
                    eye.innerHTML = obj.isMeshVisible ? '<i class="fa fa-eye"></i>' : '<i
class="fa fa-eye-slash"></i>';                          eye.style.color = obj.isMeshVisible
? 'inherit' : 'rgba(0, 0, 0, 0.3)';
                      }                          if (child.isMesh
&& child.material) {
child.isHidden                              =                              !obj.isMeshVisible;
updateMeshInteractionMaterial(child);
                            }
                         });
                }
obj.isHidden = false; obj.isSelected
= false; obj.isMeshVisible = true;
visible.onclick = function() {
                   obj.isMeshVisible       =       !obj.isMeshVisible;
obj.showMesh();
                }

                right.appendChild(visible)

             }

             makeSceneGraph(gltf.scene)
```

```
            hide(ViewerUI.loader);


         },
         function onProgress(xhr) {
            ViewerUI.loaderInfo.innerHTML = Math.round(xhr.loaded / xhr.total * 100) +
'% loaded'; }, function
         onError(err) {
             ViewerUI.loaderInfo.innerHTML = 'Error loading model! See console for more
                                          info.'; console.error('Error loading model!', err);
         }
);
   }
   let controller = new THREE.OrbitControls(camera,
renderer.domElement);    controller.enabled = true;    controller.enableDamping
= true;    controller.dampingFactor = 0.5;    controller.screenSpacePanning =
true;
   let cubeController = new THREE.OrbitControls(camera,
cubeRenderer.domElement);    cubeController.enablePan = false;
cubeController.enableZoom = false;    cubeController.rotateSpeed = 0.125;    let
selectedModeElement = ViewerUI.toggleOrbit;    setOrbitMode(); camera.position.z
= d;                    camera.lookAt(scene.position);    controller.update();
controller.saveState();
   let ambientLight = new
THREE.AmbientLight();
ambientLight.intensity = 1;
scene.add(ambientLight);
   let directionalLight = new THREE.DirectionalLight();
directionalLight.position.set(200, 200, 200)
directionalLight.intensity = 0.5;
scene.add(directionalLight);
   /*let light1 = new
THREE.PointLight(0xffffff);
```

```
light1.position.set(100, 100, 100);
scene.add(light1);
   let light2 = new
   THREE.PointLight(0xffffff);
   light2.position.set(100, 100, -100);
   scene.add(light2);
   let light3 = new
THREE.PointLight(0xffffff);
light3.position.set(-100, 100, 100);
scene.add(light3);      let light4 = new
THREE.PointLight(0xffffff);
light4.position.set(-100, 100, -100);
scene.add(light4);
   light1.intensity = light2.intensity = light3.intensity = light4.intensity
= 0.3;*/     let stop = false;     function renderAll() { renderer.clear();
renderer.render(scene, camera);         updateCubeCamera();
        cubeRenderer.render(cubeScene,
cubeCamera);         renderer.clearDepth(); if
(isInMeasureMode) { renderer.clearDepth();
renderer.render(lineScene, camera);
renderer.clearDepth();
            renderer.render(spriteScene, camera);
        }
   }
function  animate(time) {
                    if (stop) {
                    return;
}         if (play) { let
now = Date.now();
            let x = Math.min(1, (now - startTime) / duration);
camera.position.copy(oldPosition).lerp(newPosition, x)
          if (x === 1) {            play = false;
}
     }

  requestAnimationFrame(ani mate);
  controller.update();
  renderAll()
}
  requestAnimationFrame(animate);

return {       loadModel:
loadModel
  };  }  function  draggable(ele,
toggleEle)  {   let  startX  =  0;  let
startY      =      0;      function
onMouseDown(evt) {      evt
=    evt    ||    window.event;
startDrag(evt.clientX, evt.clientY);
     window.addEventListener('mousemove', onMouseMove, true);
  }    function
onMouseMove(evt) { evt
= evt || window.event;
       let newX =
evt.clientX; let newY =
evt.clientY;

moveDrag(newX, newY);
  }
  function                   onMouseUp()                 {
window.removeEventListener('mousemove', onMouseMove, true);
```

```
   }
function startDrag(x,
y) { startX      =
            x; startY =
y;
   }
   function moveDrag(newX,
newY) {         let deltaX = newX
- startX;   let deltaY = newY -
startY;     startX =  newX;
            startY = newY;
let x = ele.offsetLeft + deltaX; let y
= ele.offsetTop +
deltaY;        x < 0
&& (x = 0);        y < 0 && (y = 0); let w
= ele.parentNode.offsetWidth -
ele.offsetWidth;         let h =
ele.parentNode.offsetHeight - ele.offsetHeight; x >
w && (x = w);              y > h && (y = h);
ele.style.left = x + 'px';         ele.style.top = y + 'px';
   }
   toggleEle.addEventListener('mousedown',              onMouseDown,              true);
window.addEventListener('mouseup', onMouseUp, true);
}
```

5.2 RESULTS

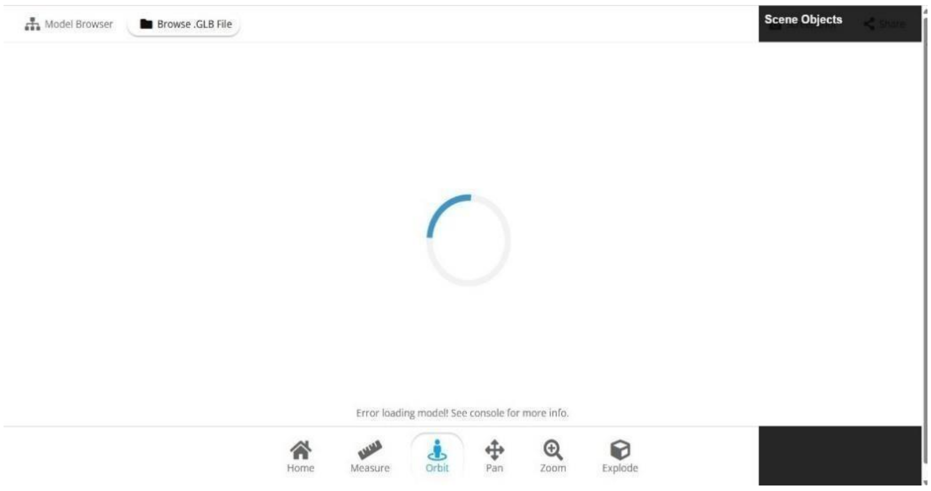

Fig no 5.2.1 Home page

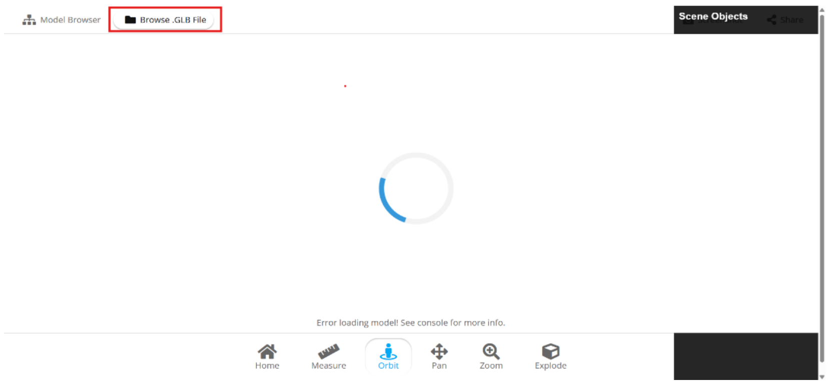

Fig no 5.2.2 Browse GLB File

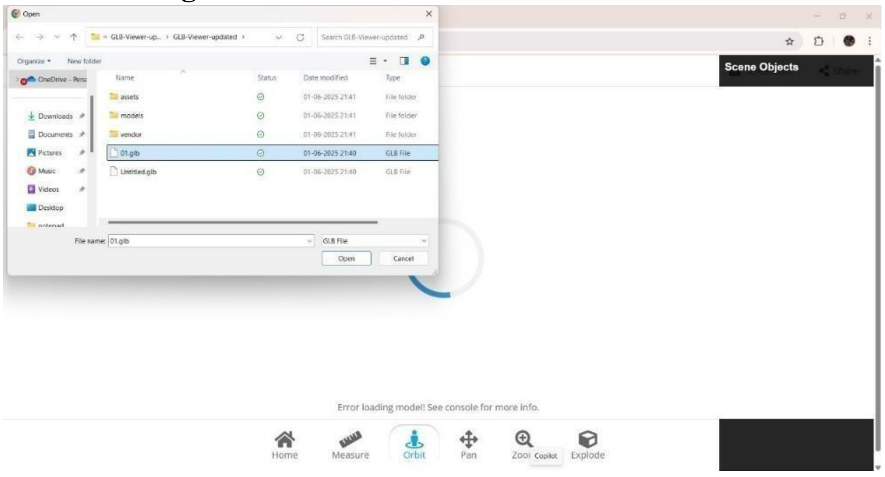

Fig no 5.2.3 Extracting the file

A .glb (GL Transmission Format Binary) file named "01.glb" is being selected for opening in a 3D model viewer of user.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 06 | June – 2025 DOI: 10.55041/ISJEM04606
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

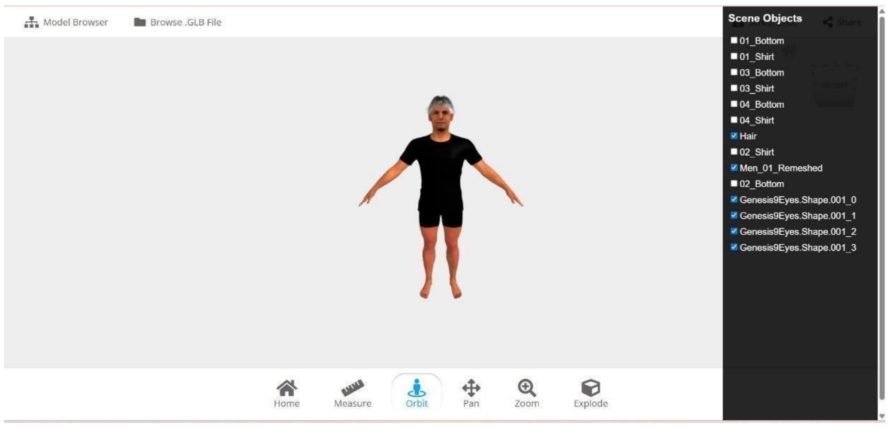

Fig no 5.2.4 Virtual Try- on

Here the different outfits can see and user can try on.

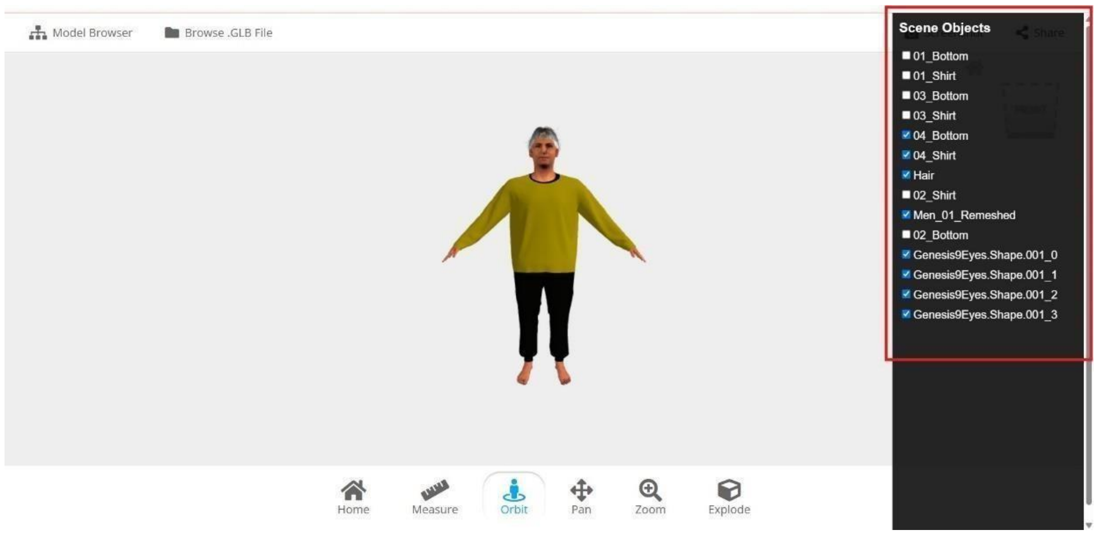

Fig no 5.2.5 Trying different Outfits

This is the final result were user can wearing virtual clothing and user can make decisions by using tools like Orbit, Pan, Zoom, and Measure for interacting with the 3D model

5.3 TESTING

5.3.1 TYPES OF TESTING

In the context of augmented reality (AR) applied to the fashion industry, testing is critical to ensure that the application delivers a seamless, realistic, and engaging experience. AR allows users to virtually try on clothes, accessories, or makeup using smartphones, tablets, or AR glasses, bridging the gap between physical and digital shopping. However, this cuttingedge technology depends on many technical and experiential factors working in harmony. Testing in this domain includes both technical validation (such as tracking accuracy, latency, compatibility with devices) and user-centered testing (such as how intuitive and realistic the virtual tryon experience feels). It also involves assessing the system under different environmental conditions, verifying 3D model fit and behavior, and measuring performance across user demographics and device types.

Moreover, since the fashion industry is driven by aesthetics and user satisfaction, visual quality, personalization, and social sharing features must be thoroughly tested. Without adequate testing, an AR app can fail to provide value to users, lead to poor reviews, and ultimately harm the brand's reputation.

Below are six critical types of testing for your AR-in-fashion project, each of which I will explain in detail:

Usability Testing

Usability testing is crucial in any application involving human interaction, and even more so in AR for fashion, where user experience (UX) drives adoption and retention. This type of testing assesses how easily and effectively real users can interact with your AR solution to try on clothing or accessories, navigate through menus, and complete tasks like purchasing or sharing.

Usability testing begins with creating user scenarios—such as a shopper trying on a jacket, customizing colours, or sharing their look on social media. Participants from your target audience are asked to perform these tasks while being observed. The tester monitors how intuitively users interact with the app, whether they get confused, how long it takes them to perform actions, and whether errors occur.

A key focus in AR usability testing is ease of onboarding. For example, can users easily scan their body or face to initiate the AR experience? Is the system guiding them clearly on what to do next? Complicated setup or unclear instructions can frustrate users and cause drop-offs. Other factors such as gesture recognition, menu placement, and interaction feedback (haptic or visual cues) are tested for intuitiveness. Testers also collect qualitative data through interviews or surveys to understand subjective impressions: Was the experience enjoyable? Did users trust the AR try-on to be accurate? Did they feel immersed or distracted by glitches?

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 06 | June – 2025 DOI: 10.55041/ISJEM04606
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Since AR is relatively new for many users, accessibility testing is also important—ensuring that users of different ages, abilities, or digital familiarity levels can still interact effectively.

Usability testing can be done using moderated sessions, remote recordings, eyetracking, or session replay tools. Post-session analysis highlights pain points, drop- off moments, and usability bottlenecks. Iterative testing (test-feedback-retest) helps designers refine the UI and interaction flows continuously.

In AR fashion, good usability leads to better customer satisfaction, higher engagement, and more completed purchases. It also builds brand credibility, as users associate a seamless virtual try-on with professionalism and innovation. Conversely, poor usability can lead to frustration, bad reviews, and app abandonment—even if the underlying AR technology is sound.

Therefore, usability testing is not just about testing a feature, but validating the entire user journey from opening the app to confidently choosing an outfit. It helps bridge the gap between technological possibilities and human expectations.

Performance Testing

Performance testing ensures that your AR fashion application operates smoothly across different environments and usage scenarios. Given the real-time nature of AR—where 3D clothing must be rendered and superimposed on the user's image—speed, responsiveness, and stability are vital to success.

One of the key components of performance testing is assessing frame rate. A high frame rate (typically 30-60 frames per second) is essential for fluid AR rendering. Dropped frames or lag during rendering can break immersion and negatively affect the user experience. If a user rotates their body and the AR garment lags or distorts, it damages trust in the product. Another area of concern is latency—the delay between user action (like moving the phone or changing posture) and the corresponding AR response. Low latency is necessary to maintain real-time interactivity. This is especially crucial in try-on applications where the garment must adjust dynamically to movement.

You'll also want to test resource usage, particularly CPU and GPU consumption. AR apps are resource-intensive, and on mobile devices, excessive use can lead to overheating, battery drain, or crashes. Performance testing should include stress testing, which evaluates how the app behaves under peak usage (e.g., multiple users, long sessions, switching between products quickly).

Another crucial aspect is load testing, especially if your app includes cloud-based features like saving virtual outfits, accessing user profiles, or integrating with e- commerce platforms. You'll want to ensure that servers can handle multiple requests without slowing down.

Device fragmentation poses another performance challenge. Android devices in particular have varying camera qualities, processors, and sensors. Performance testing needs to include a range of devices (flagships to mid-tier) to ensure consistent behavior across the board.

Performance tools such as Firebase Performance Monitoring, Unity Profiler, or Apple's Instruments tool (for iOS) can be used to gather performance metrics. Regular updates and new features must also go through regression performance testing to ensure that enhancements don't degrade existing performance.

Good performance testing ensures your AR fashion app feels real-time, seamless, and immersive. In a competitive industry where user patience is limited, optimizing performance is crucial for adoption, engagement, and positive reviews.

Compatibility Testing

Compatibility testing ensures that your AR solution functions correctly across a diverse ecosystem of hardware and software configurations. In the fashion industry, where your target audience could be using various smartphones, tablets, or AR glasses, it's essential to ensure that the experience is consistent and glitch-free on all supported platforms.

AR in fashion typically relies on advanced features like motion tracking, depth sensing, and camera access. These features vary significantly across devices. For instance, a flagship iPhone might support LiDAR-based depth mapping, while a mid-range Android phone might rely on basic RGB cameras. Compatibility testing verifies that your app scales down features gracefully and still offers usable performance and accuracy across all devices.

There are four main areas of compatibility to test:

1. Device Compatibility: Test across a range of smartphones and tablets (e.g., iOS, Android, different brands). Also test with different AR frameworks like ARKit, ARCore, or WebAR.
2. OS Compatibility: Ensure the app performs well on different versions of iOS and Android. New OS updates can sometimes break AR features.
3. Browser Compatibility (for WebAR): If your AR experience is browser-based, it must be tested on Chrome, Safari, Firefox, and Edge.
4. Network Compatibility: Users may be on 5G, 4G, or Wi-Fi. The app should load efficiently across varying network speeds, especially if it fetches 3D models or user profiles from the cloud.

Compatibility testing also includes localization testing (for different languages and regions), accessibility (for users with disabilities), and UX responsiveness across screen sizes.

Ultimately, compatibility testing ensures maximum user reach and consistent quality, preventing device-based failures that can damage brand reputation or lead to poor reviews in app stores.

Accuracy and Fit Testing

Accuracy and fit testing is one of the most critical evaluations in augmented reality fashion applications. It focuses on ensuring that virtual garments, accessories, or makeup align precisely with the user's body, face, or environment, and that they behave realistically as the user moves. This type of testing validates how well the digital overlay matches real-world physical proportions and movements, which is fundamental for creating trust in a virtual tryon experience.

The main objective here is to assess how accurately the AR system detects body landmarks (like shoulders, waist, hips) and calibrates clothing to fit various body types. Poor calibration can result in virtual clothes that “float,” shift unnaturally, or clip through the body. These issues can be off-putting for users and damage the credibility of the fashion brand.

Accuracy and fit testing typically includes the following components:

1. Body Tracking Precision: AR applications use skeleton tracking, pose estimation, and body segmentation algorithms. Testing ensures these systems

recognize body shapes and sizes correctly and remain stable even as users turn, bend, or gesture.

2. Size Matching Algorithms: The digital clothing must adapt to users' real dimensions. Testing validates whether size adjustments based on scanned measurements or visual estimations actually reflect real-life fit.
3. Occlusion Testing: Ensures virtual items correctly interact with the physical body—
e.g., a jacket sleeve should not float above the arm or disappear when the user raises it. Occlusion testing ensures garments behave naturally.
4. Environmental Factors: Testing under different lighting conditions, camera angles, or user clothing (e.g., tight vs. loose clothes) ensures the system maintains accuracy across varied scenarios.

This type of testing often involves both automated assessments (using test environments, avatars, and datasets) and real-user testing where participants of different body types interact with the system and provide feedback on realism and comfort of virtual garments.
Accurate fit testing is crucial not just for aesthetics but for business outcomes. If users feel confident that the virtual garment accurately reflects how it would fit in reality, they're more likely to complete a purchase. Conversely, poor fitting could lead to returns, negative reviews, and loss of trust.
In sum, accuracy and fit testing ensures that AR fashion solutions are not only visually convincing but also functionally reliable, providing users with the confidence to make informed buying decisions.

Visual Quality Testing

Visual quality testing focuses on evaluating the realism, rendering fidelity, lighting, and aesthetics of the virtual garments and accessories in the AR environment. In fashion, where visual appeal is everything, this testing is critical to delivering an immersive and stylish experience that matches or exceeds physical retail.
Key components of visual quality testing include:

1. Texture and Fabric Simulation: AR garments must realistically replicate realworld fabrics—whether it's the shine of silk, the softness of cotton, or the stiffness of leather. Visual quality testing examines whether these materials look authentic under various lighting and viewing angles.

2. Lighting and Shadows: AR apps often use real-time lighting estimation to match the virtual content to the physical environment. Testing ensures that garments adapt to different lighting conditions (indoor, outdoor, daylight, evening) and cast realistic shadows and highlights.
3. Animation Smoothness: When users move, the virtual garments should deform naturally—wrinkling, stretching, or flowing just like physical clothes. Testing confirms that animations are smooth, transitions are seamless, and the clothing behaves according to physics principles.
4. Colour Fidelity: Accurate colour rendering is crucial, especially for fashion items where exact shades matter. Testing verifies that colours appear consistent across devices and under different lighting scenarios.
5. Resolution and Pixelation: Testing ensures high-resolution rendering without blurring or jagged edges, especially on zoom or close-up views.

This testing often involves both technical tools (like Unity or Unreal Engine profilers, shader debugging, and lighting simulators) and user-based evaluations, where fashion professionals or end users provide feedback on the garment's visual appeal. Poor visual quality can break immersion and harm brand perception. Users expect virtual tryons to match studio-quality product visuals. High-fidelity visuals help users appreciate design details, increasing the chances of engagement and purchase.

Ultimately, visual quality testing ensures that your AR fashion app delivers photorealistic, stylish, and professional-quality experiences, critical for both branding and customer satisfaction.

A/B Testing for User Engagement

A/B testing involves comparing two or more versions of a feature or interface to determine which one performs better based on user behaviour. In AR for fashion, A/B testing can help optimize everything from the virtual try-on interface to product presentation, camera modes, or call-to-action placements.

For example, you might test:

- Version A: A clean interface with a minimalist design
- Version B: A colourful interface with icons and animated guidance Or:
- Version A: Virtual try-on begins with body scan
- Version B: Try-on starts instantly with camera overlay You then track key performance indicators (KPIs) such as:

- Time spent in the app
- Number of try-ons per session
- Add-to-cart or purchase rate
- Share rate (e.g., sharing look on social media)
- Bounce rate (users who drop off early)

This data helps determine which version drives higher engagement, better user retention, or more conversions. A/B testing must be conducted carefully to avoid biased results. Randomized user groups, equal distribution, and consistent conditions are necessary to ensure fairness.

Additionally, multivariate testing (testing combinations of variables) can be used to fine-tune user experience. For instance, combining camera angle presets, UI layouts, and visual feedback can reveal optimal combinations that enhance user satisfaction. Tools like Google Optimize, Firebase A/B Testing, or in-app analytics platforms are commonly used to run and monitor these tests.

The goal is to make data-driven design decisions that improve user experience, conversion, and retention. In a competitive digital fashion market, small improvements in interaction flow or visuals can lead to significant increases in user engagement and sales.

A/B testing allows you to refine your product based on real-world behavior rather than assumptions—ensuring that your AR solution evolves in line with user preferences.

5.3.2 TEST CASES

Serial No.	Test Case	Expected Result	Result	Remarks
1	Launch the AR app on Android and iOS devices	App launches successfully with no crashes	Pass/Fail	Ensure multiplatform compatibility

2	Use AR try-on feature with default lighting indoors	Virtual garment aligns accurately with body and reacts naturally to movement	Pass/Fail	Test under natural indoor light
3	Rotate body to view 360° of garment	Garment renders correctly from all angles and doesn't distort	Pass/Fail	Full-body pose detection
4	Test app on low-end Android device	App loads with slightly lower fidelity but without crashing	Pass/Fail	Performance scaling verification
5	Wear a jacket and try virtual dress (layering scenario)	App detects physical clothing and adjusts the virtual one accordingly	Pass/Fail	Occlusion handling test
6	Attempt try-on with plus-size model	Virtual garment resizes accurately to model's body size	Pass/Fail	Body type adaptability
7	Switch between different virtual outfits quickly	App responds instantly and loads new garments without crashing	Pass/Fail	Speed and memory handling
8	Use AR app outdoors in sunlight	Clothing overlays remain visible and stable without glare or misalignment	Pass/Fail	Bright-light condition testing
9	Select "Buy Now" after virtual try-on	Redirects to product page or cart correctly with selected variant	Pass/Fail	E-commerce integration check
10	Load app using mobile data (4G/5G)	App loads quickly and core features are usable with minimal latency	Pass/Fail	Network responsiveness

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The integration of Augmented Reality (AR) into the fashion industry represents a transformative shift in how consumers interact with products and brands. Through virtual tryons, immersive product experiences, and real-time personalization, AR bridges the gap between physical and digital fashion. However, the success of such technology-driven experiences depends not only on innovative design but also on rigorous testing and experimentation. A well-developed AR solution must provide users with seamless interaction, high accuracy in garment fitting, and visually compelling outputs. Without systematic testing, even the most promising AR applications can suffer from performance glitches, usability issues, or mismatched expectations—leading to customer dissatisfaction and reduced brand credibility.

This project highlights the significance of six essential testing types: usability testing, performance testing, compatibility testing, accuracy and fit testing, visual quality testing, and A/B testing for user engagement. Each of these plays a vital role in refining the user experience and ensuring technical robustness. Usability testing confirms that users can intuitively navigate the AR system, while performance testing ensures the app operates smoothly across various scenarios. Compatibility testing widens accessibility by confirming the AR app works on different devices, operating systems, and screen sizes. Equally important is the accuracy and fit testing, which determines how realistically virtual garments adapt to various body types and movements. Visual quality testing guarantees high fidelity and realism, which is essential in fashion, a domain driven largely by aesthetics. Finally, A/B testing enables developers to optimize user interaction and business metrics based on real-world behavior.

In conclusion, the implementation of AR in fashion is not just about deploying a visually impressive tool but about delivering a reliable, responsive, and engaging user journey. Robust experimentation and methodical testing form the foundation for trust in the technology, paving the way for greater user adoption and commercial success. As consumer expectations grow and digital fashion experiences become the norm, brands that invest in comprehensive testing will be better positioned to lead innovation and customer satisfaction. This project reinforces the idea that testing is not an afterthought, but a strategic necessity—critical to realizing the full potential of AR in revolutionizing the fashion industry.

6.2 FUTURE SCOPE

The future scope of Augmented Reality in the fashion industry is vast and promising. As AR technology advances, fashion brands can offer hyperpersonalized experiences using AI-driven body scanning and real-time garment adaptation. Integration with virtual reality (VR) and the metaverse will create fully immersive fashion shows, virtual stores, and avatar-based shopping. Additionally, AR can support sustainable practices through virtual sampling, reducing waste and overproduction. The technology also holds potential for smart mirrors and ARpowered fitting rooms in physical stores, blending offline and online retail.

With 5G connectivity and improved device compatibility, AR will become faster, more accessible, and widely adopted. As user expectations grow, brands that innovate in AR will lead in customer engagement, retention, and satisfaction. In the coming years, AR will not just enhance shopping—it will redefine the entire fashion retail experience from design to delivery.

REFERENCES

[1] https://www.unrealengine.com/en-US/.

[2] Alex M Andrew. Multiple view geometry in computer vision. Kybernetes, 2001.

[3] Slawomir Bak, Peter Carr, and Jean-Francois Lalonde. Domain adaptation through synthesis for unsupervised person re-identification. In Proceedings of the European Conference on Computer Vision (ECCV), pages 189–205, 2018.

[4] Igor Barros Barbosa, Marco Cristani, Barbara Caputo, Aleksander Rognhaugen, and Theoharis Theoharis. Looking beyond appearances: Synthetic training data for deep cnns in reidentification. Computer Vision and Image Understanding, 167:50–62, 2018.

[5] Christian Chang. Modeling, UV Mapping, and Texturing 3D Game Weapons. Wordware Publishing, Inc., 2006.

[6] MakeHuman Community. MakeHuman: Open Source Tool for Making 3D Characters, 2020. http://www.makehumancommunity.org.

[7] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xi-¨ aowei Xu. Densitybased spatial clustering of applications with noise (dbscan). In Proc. of the Second International Conference on Knowledge Discovery and Data Mining, pages 226–231, 1996.

[8] Michela Farenzena, Loris Bazzani, Alessandro Perina, Vittorio Murino, and Marco Cristani. Person re-identification by symmetry-driven accumulation of local features. In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 2360–2367. IEEE, 2010.

[9] Yuying Ge, Ruimao Zhang, Lingyun Wu, Xiaogang Wang, Xiaoou Tang, and Ping Luo. A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. CVPR, 2019.

[10] Yixiao Ge, Feng Zhu, Dapeng Chen, Rui Zhao, and Hongsheng Li. Self-paced contrastive learning with hybrid memory for domain adaptive object re-id. In Advances in Neural Information Processing Systems, 2020.

[11] Xintong Han, Zuxuan Wu, Zhe Wu, Ruichi Yu, and Larry S Davis. Viton: An imagebased virtual try-on network. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7543–7552, 2018.

[12] Yang Hu, Dong Yi, Shengcai Liao, Zhen Lei, and Stan Z Li. Cross dataset person reidentification. In Asian Conference on Computer Vision, pages 650–

664. Springer, 2014. [13] Verica Lazova, Eldar Insafutdinov, and Gerard PonsMoll. 360-degree textures of people in clothing from a single image. In 2019 International Conference on 3D Vision (3DV), pages 643–653. IEEE, 2019. 1

[14] KENNETH LEVENBERG. A method for the solution of certain non-linear problems in least squares. Quarterly of Applied Mathematics, 2(2):164–168, 1944. [15] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person reidentification. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 152–159, 2014.