HASSAN FAROOQI
01-134221-028
M UMER ALI
01-134221-056

When AR meets Style

Bachelor of Science in Computer Science

Supervisor: Dr Saneeha Amir

Department of Computer Science
Bahria University, Islamabad

November 2025

Certificate

We accept the work contained in the report titled “When AR Meets Style”, written by Hassan Farooqi AND M Umer Ali as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by . . . :

Supervisor: Dr Saneeha Amir

Internal Examiner: Name of the Internal Examiner (Title)

External Examiner: Name of the External Examiner (Title)

Project Coordinator: Mr. Asim Ali

Head of the Department: Dr. Muhammad Khurram Ehsan (Professor/HOD)

Novemeber 2025

Abstract

Our project, *"When AR Meets Style"*, introduces a unique approach for e-commerce where we introduce virtual try on with clothing products, allowing customers to try on clothes before buying. We provide both Real-Time and AI trained 2D model where user could upload photo to check the cloth on himself or see it real time using his camera. This capability increaces customer interaction with the site as well as address the issue of high return rates(which is around 20–30% [1]) due to customers not finding the clothing material suitable for their body or how it fits them .

The project is developed using the MERN stack technology (MongoDB, Express.js, React.js, Node.js), making it accessible to both users and sellers through a web browser.It also focuses on scalability and efficiency, using cloud storage for 3D assets and secure payment options (such as Stripe) for seamless transactions. The React-based frontend interacts with the Express.js backend to ensure a responsive and user-friendly experience. Additionally,a retailer dashboard , which provide analytics and insights into user activity and product performance

As a result, our project not only enhances customer satisfaction but also establishes a modern foundation for the future of e-commerce.

Acknowledgments

All praise is to Almighty Allah who bestowed upon us a minute portion of His infinite knowledge, by virtue of which we were able to complete this meaningful and challenging task.

We are deeply thankful to our supervisor, Dr. Saneeha Amir, for his unwavering guidance, support, and insightful feedback throughout the development of our project. Without his continuous supervision and encouragement, the successful completion of this work would not have been possible.

We also extend our heartfelt gratitude to our parents and families, whose constant support, prayers, and belief in us have been a source of motivation and strength during this academic journey.

HASSAN FAROOQI , M UMER ALI
Islamabad, Pakistan

November 2025

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

DSA	Data Structure and Algorithms
OOP	Object Oriented Programming
PF	Programming Fundamentals
SE	Software Engineering
SQL	Structured Query Language
UNESCO	United Nations Educational, Scientific and Cultural Organization
UNICODE	Unique, Universal, and Uniform Character enCoding
XML	Extensible Markup Language

Chapter 1

Introduction

In today modern era where technology is advancing very fast, the advancement in e-commerce is also neeeded as it is one of the most booming industry .Traditional online retail is facing several problems like high return rate , because customers expectations don't match regarding the product they are receiving and customers cannot see how the product suits and fits on them, so virtual try-on technology is giving new ways for making customer confidence and engagement better Our project will provide interactive platform that will help both the customer and the seller in buying and selling of clothing products.The platform will be using a Web-based virtual try-on system that allows real-time visualization of clothes on users bodies through normal web browsers alongside a 2D approach using AI that will perfectly fit the given cloth on the customer image. This is meaning that customers can see how clothes look on them before they make the purchase

Also, the platform will be interactice , elegant and simple to use as we are using React-based frontend with an Express.js backend , Retailers are getting benefit from analytics dashboards which gives insights into user engagement and inventory trends, helping them to understand customer preferences better and manage their stock more efficiently. This integration of virtual try-on and ecommerce is filling the gap between online shopping and physical try-on experiences, it engages customer and helps both customers and sellers by creating a modernized, technology-driven retail system that increases customer satisfaction and will help increacing business profitability for sellers[2].

1.1 Background

When AR Meets Style which is a contemporary web based platform that will enable online shopping to be more interactive. The system enables users to view and have an opportunity of trying on outfits, either in real time or by posting a photo which would enable the user

see how wearing various clothes would appear and fit them. Online shopping usually creates an illusion of size, fit, and appearance that makes them return the products, and results in customer dis-satisfaction. Our platform is a solution to such issues as it provides more precise and captivating experience to users.

We have used Three.js, a JavaScript library that is used to make and render 3D objects on web browsers. In this way, the users have the opportunity to view themselves in a variety of clothing that can be rotated, zoomed, or changed in real time. Every piece of clothing is modeled in 3D to provide a better visualization compared to static image, in case the user does not find the outcome of the latter satisfactory he might revert to 2D Try on which with the assistance of an AI trained model provide much more accurate results as compared to 3D visualization [3]. The site is built with MERN stack (MongoDB, Express.js, React.js, and Node.js) as it offers the user a scalable, responsive, and easy web experience. Three.js will do the interaction and 3D visualization in the browser, where no extra program is needed.

1.2 Problem Description

Since the use of E-commerce is becoming the order of the day in Pakistan, it is also carrying with it some challenges that are likely to frustrate the customers as well as the businesses. Among the key issues, there is the fact that the customers cannot see how clothes would fit them before purchasing them. You are also not seeing the actual appearance of a product and how it will look on your body shape when you are shopping online because you are only seeing flat and stationary pictures of the products. This is causing a problem of guessing whether a dress, shirt or a pair of shoes will suit you thus causing a lot of guesses. In many cases, individuals are finding themselves placing orders of items that appear to look dissimilar in real life .

These are the problems that our project is directly addressing through the use of virtual try-on technology to provide an interactive shopping experience to customers .Our system is letting customers virtually try on clothes, which means that they are able to see how the outfits are fitting before they make a purchase. This is simplifying the process of making the correct selection of style and minimizing the possibility of purchasing something that they may not like on themselves.

1.3 Objectives

The primary objective of our project is to bring a interactive online shopping experience.Our goal is to build a user-friendly e-commerce system using the MERN stack (MongoDB, Express.js, React.js, Node.js) that is allowing users to virtually try on outfits using both 2D

and 3-D approach

- **An fully functional E-commerce site** We will be developing a full fledge e-commerce store with all its functionalities like selling , buying of products , managing them .It will allow users to easily browse items , add them to the cart, and complete purchases safely. It will also include an admin panel for managing inventory and maintaining overall store operations

- **2-D Virtual Try-on** We will be integrating a 2-D virtual try on so user could see how the cloths look on them by simply uploading their photo and get a realstic view of how the product would suit them

- **3-D Virtual Try-on** We will also be developing a real-time 3D try-on feature where the user can open their camera and the selected clothing will adjust to their body dimensions, allowing them to preview it on themselves.

- **Implementing safe payment integration:** We will be integrating trusted payment gateways, such as Stripe to provide a seamless and secure checkout process.This will make payment safe for user as Stripe follows industry-standard security measures

1.4 Project Scope

We will be delivering a fully functional e-commerce website that is built with MERN stack (MongoDB, Express.js, React.js, Node.js) along with the try on capability and making it sure that use of 3D visualization and camera capabilities doesnt impact the website performance. This platform will be complete with all essential features that are needed for online shopping. The website will be responsive and working smoothly on all devices including smartphones and desktop computers, making it accessible for everyone.

The main feature of our platform will be the virtual try-on capability that is using Three.js technology for 3D rendering and real-time body tracking using media pipe.It will also feature a 2D approach using AI trained model which will provide more relastic view to how cloths will look on a person. Customers will be able to use their smartphone or computer camera to see how clothes look and fit on their own bodies or provide a picture of themself and our 2D model will make the garment fit on their given picture.

.

To ensure user security, the platform is deployed on a network whic supports (HTTPS) and we are using Stripe for safe payment integration. This will help us to protect user data during browsing and checkout.

What We Will Not Be Delivering:

We will not be developing a mobile application. Our focus is on creating a web-based platform that works through browsers, so users will not have a dedicated mobile app to download. However, since our website will be responsive, it will work properly on mobile browsers.
We will not be creating the 3D garment models for all clothing items ourselves. The 3D models that are required for the virtual try-on feature will be using pre-made or sample models for demonstration purposes. Creating custom 3D models for every single clothing item is requiring specialized 3D modeling skills and paid softwares which is beyond the scope of this project.

Limitations of Our System:

We were unable to implement full functional AR capability as AR is usually designed for mobile frameworks [4] with mobile sensors (LiDAR, depth cameras) , which cant be accessed on a website . Therefore, our AR feature will be a simplified demonstration rather than a fully realistic try-on system.

Additionally The Ar technology is still developing and improving, so it comes with few limitations. Current virtual try-on systems are providing good visual previews, but they cannot perfectly replicate real-world clothing fit and appearance due to technological constraints, to tackle this issue we also have added the 2D approach using an AI trained model which provides better and realistic result as compared to the 3D AR approach [3][5]. The accuracy of the 3D Ar try-on is also depending on other factors such as user's camera, the lighting of the room. The virtual try-on system is providing a visual preview but cannot replicate the actual feeling of wearing clothes. It is mainly showing how the item looks visually on the body but cannot guarantee perfect fit measurements because the technology for measuring exact body dimensions through camera is not yet fully developed.

Solution Application Areas

Our project can be used in many areas of e-commerce and fashion. It can help both shoppers and businesses. It is good for online fashion stores. The platform helps people who want an interactive shopping experience, so they can choose clothes confidently from home. It also helps businesses by increasing customer engagement, improving sales, and making customers more loyal with a new, technology-based shopping experience.

Chapter 2

Literature Review

AR technology has been regarded as a strong technology within the e-commerce industry, especially in the fashion e-commerce industry. AR is enabling the customers to see the products in real-time by overlapping it with the real world using their device cameras. Recent researches are demonstrating that AR based shopping experience is greatly boosting customer confidence and lowering the returns. [6].

Nevertheless, most of the available AR virtual try-on solutions are mobile apps that are forcing customers to download and install particular applications on their smart phones. The reason behind this is that in most cases AR technology is developed on a platform such as Unity or native mobile development tools that cannot be accessed by a web browser. Such technologies are necessitating specific mobile applications to operate effectively and this is posing a challenge to users who would like to gain fast access by use of web browsers without any form of installation. This missing feature of web-based solutions is creating an opportunity of platforms that can deliver AR experiences directly on browsers with new web technologies.

As mentioned earlier in the project scope, conventional 3D AR virtual try-on systems are experiencing a number of limitations and problems. They are, among others, forcing the users to keep on changing their camera angles, or the present level of the AR technology that is not yet on par to make 100 percent accurate representations of how the clothes will fit and look in real life. In order to deal with these issues we will be also introducing a 2D AR functionality in our application whereby the users can just upload a static picture of themselves and the application will apply the garment that they have selected on the picture. This 2D solution is addressing the issues that normal 3D AR is experiencing.

Normal accessibility without downloading apps on regular browsers is being guaranteed by the use of web-based technology. MERN stack is sustaining a dynamic interface and secure payment systems. Through web-based accessibility, dual virtual try-on, and a seller platform, this project is filling the existing market gaps and benefiting customers and sellers in the growing e-commerce business.

2.1 Overview

'When AR Meets Style' is an web-based e-commerce platform that is designed to improve the online shopping experience in Pakistan. The platform is built using the MERN stack (MongoDB, Express.js, React.js, Node.js) and is integrating virtual try-on technology to help customers see how products will look on them before making purchase decisions. Unlike traditional online shopping that is relying on static product images, our platform is offering both 3D real-time virtual try-on using Three.js and a 2D photo-based try-on feature where users can upload their images. This dual approach will help us limit the limitations of current AR technology while making the shopping experience more interactive and reliable. With a appealing design and user-friendly interfaces , our platform is aiming to reduce the high return rates in online fashion retail, boost customer confidence, and create new opportunities for sellers in the growing Pakistani e-commerce market.

2.2 Existing E-Commerce Technologies

2.2.1 Traditional E-Commerce Methods

Traditional online shopping platforms are primarily using static images, size charts, and product descriptions to showcase prodcuts. Although these approaches have rendered online shopping convenient, they are not usually meeting the various demands of the consumers especially when it comes to visualizing how the garments fit or appear on the various body figures. Research is indicating that this reliance on static visuals is leading to high return rates, with online fashion retail experiencing returns of(which is around 20–30% [1]) due to sizing issues or unmet expectations. The lack of interactive and realistic product previews is limiting customer confidence, resulting in dissatisfaction and increased logistical costs for retailers who have to handle returns, restocking, and refunds. Studies, such as those by industry analysts, are showing us that traditional e-commerce is struggling to replicate the tactile and visual experience of in-store shopping, where customers can try on clothes and see how they fit before buying. This gap between online and physical shopping experiences is underscoring the need for innovative solutions to enhance the online shopping process and make it more reliable and satisfying for customers.

2.2.2 Existing AR Based E-Commerce Platforms

- **Zara AR APP:** The AR application that Zara offers enables customers to see virtual models. dressing up in the actual set-ups in their actual setting with their smartphones. It uses AR to show clothes but does not have body tracking to use in personalized try-on.

- **ASOS Virtual Catwalk:** OSOS applies AR to create virtual models on the customers. environment, displaying the appearance of clothes in motion. It does not however permit users. to experiment with clothes on their respective bodies, which restricts customization.

- **Wanna Kicks:** It is an AR app that is oriented to virtual sneaker try-on, where foottracking technology is employed to superimpose a 3D shoe model on the feet of the user. While effective for footwear, it does not support the full-body clothing try-on and is app-based, which decreases. accessibility

Tool Name	**Focus**	**Technology Used**	**Interactive Features**
ZARA AR App	Clothing Visualization	AR, Mobile App	Virtual models in real-world settings
ASOS Virtual Catwalk	Clothing Display	AR, Mobile App	Virtual runway in user's environment
Wanna Kicks	Footwear Try-On	AR, Foot Tracking	3D shoe try-on via smartphone camera

Table 2.1: Comparison of AR Fashion Apps

Even though improvements have been made in AR-based shopping tools, some of the limitations continue to exist. which are impeding their working efficiency in changing the online shopping experience, specifically in the case of fashion retail.

- **Majority Are Mobile Applications:**Most current AR virtual try-on solutions are applications that are making users download and install certain applications on their handsets. their smartphones. This is due to the fact that in most cases AR technology is constructed on platforms. similar to Unity or native mobile development tools not supported by web browsers. Studies are pointing out that e-commerce customer likes site more as compared. to apps [7], and so restricting the accessibility and popularization of such AR. solutions. Access to web-based AR solutions is limited and they are very few in the market. to users that want to browse quickly using web browsers and no installation take place. requirements.

- **Disadvantages of Current AR Technology:** The virtual try-on technology of AR currently available has several weaknesses. has a number of technical drawbacks which are impinging on user experience. The technology is also not that developed to give 100 percent accurate representations. of the way clothes will be worn and appear in the real world [8]. The users have to make continuous adjustments in camera angles, perfect lighting conditions and cope with it. and with performance problems on lower-end machines. As well, AR applications are eating up a lot of computing power, resulting in delays, bugs, and irritating. issues, particularly on obsolete hardware. This problem might be addressed with the help of offering a 2D strategy that none of the brands mentioned above have.

- **No 2D Alternative Option:** Current AR solutions are only currently concentrating on No 2D photo-based substitutes to 3D real-time try-on experiences. This that it is causing issues to those users who do not feel comfortable with the use of cameras in real time, possess inadequate lighting, or have less powerful processing devices. The lack of a less complicated 2D one, where users can post a fixed picture and view it. clothing that is put on top of it is restricting accessibility and convenience among many prospective. customers..

Table 2.2: Comparison of Existing AR Solutions and Limitations

Limitation	**Description**
Majority Are Mobile Applications	Most AR solutions are app-based requiring downloads, which is limiting accessibility. Web-based AR solutions are very rare.Studies shows that in e-commerce customer prefer website more as compared to apps [7].
Disadvantages of Current AR Technology	AR Technology is not always accurate, it takes constant adjustment of the camera and good lighting and in most technology lacks real-time body tracking. Fast solutions, requires lots of computational time making them lag on. low-end devices.
No 2D Alternative Option	Only 3D real-time try-on is available with no simpler 2D photo-based option. This is limiting accessibility for users with poor cameras, lighting, or lower-end devices.

2.3 Research Gaps and Contribution

According to the literature analysis, the existing research has a number of gap holes. AR virtual try-on environment and e-commerce in which our project is positioned. First, the most of the available AR solutions are in the form of mobile apps that have to be downloaded, and AR based on web is extremely rare. It is this gap that our project is addressing by offering a fully. a web-based virtual try-on solution which can be accessed by regular browsers without any. easy and simple to install apps, which is convenient and accessible to every user. Second, current AR technology is being limited We are dealing with this by using a two-pronged approach. that also covers both 3D live try-on and an option of a 2D photo-based try-on functionality that provide users with. customization depending on their desires and capability of devices.

Chapter 3

Requirement Specifications

3.1 Existing System

The current methods of online fashion retail are majorly based on the old systems of ecommerce that utilize the help of stagnant images of the product, size charts. While some advanced AR-based virtual try-on solutions are being applied to platforms, and they are predominantly. mobile applications created by big brands with some of them requiring subscription. for using virtual try on. Such available AR solutions are facing various constraints such as the need to download an app that makes access less accessible, and high computational with not able to provide simpler substitutes such as 2D photo-based try-ons

3.2 Proposed System

The proposed system When AR Meets Style is an all-round web-based e-commerce. platform which is handling the shortcomings of existing systems . Built using the MERN stack The platform, (MongoDB, Express.js, React.js, Node.js), is offering a full-fledged platform. online shopping experience, which can be accessed by using the conventional web browser and without. requiring any app downloads. The system is adopting a two-way virtual try-on system. that has both 3D real-time try-on with Three.js technology to have interactive experiences. and a photo-based try-on option which is 2D so that users can just upload their photos in order to see. how clothes appear on them, and the issues of camera corrections and illumination are removed. requirements.Also the seller panel on which sellers had the ability to add products, control inventory, manage orders, and monitor sales without the use of technical skills. Our intuitive system is enabling an all-encompassing environment that empowers by integrating webbased access, customizable virtual try-ons,

all-inclusive seller support and interfaces that are user friendly, making our system an inclusive platform that benefits. both customers and seller

3.3 Functional Requirements

3.3.1 User Roles

The platform has three main roles:

- **Customers:** Customers can browse through different categories of clothing and shoes, view product details and images, use the virtual try-on feature to see how products look on them, add items to shopping cart, make secure payments through integrated payment gateways, track their order status, and provide reviews and ratings for products they purchased.
- **Admin/Seller** To be able to manage user accounts , place product listing and delete unacceptable content, product categories, customer complaints and dispute, etc. track performance and activity on the monitoring platform, and guarantee the smooth running of the entire system.

3.3.2 Customer Functional Requirements

- **C1:** Register, email address, password, log-in to account, profile information. (Name, contact, username), and rest password.
- **C2:** Browse products by categories, filter products by size, color, price range.
- **C3:** View detailed product information including images, descriptions, available sizes, prices, and seller information.
- **C4:** 3D real-time virtual try-on capability- Allow camera access to view how. products can view their body in real-time body tracking.
- **C5:** Select 2D photo-based virtual try-on by uploading a static photo and having cloths superimposed on the picture as preview.
- **C6:** Manage shopping cart by adding, editing, and removing items; view cart total and proceed to checkout.
- **C7:** Complete secure payments through integrated payment gateways (Stripe) with HTTPS encryption.
- **C8:** View order history and track order order details including past. purchases

3.3.3 Admin Functional Requirements

- **A1:** Authenticate and log into the dashboard of the administration with valid log-in credentials and role-based access control.
- **A2:** Maintain product inventory by changing stock levels, identifying them as out of stock. adding in the stock, editing the information on the products and deleting products.
- **A3:** Add to product categories: Create new product categories, modify the existing ones, and systematizing the category structure.
- **A4:** See the customer orders with full order information such as customer quantities, information, ordered items and delivery addresses.
- **A5:** Keep track of performance of the system, logs of errors, user activities to maintain smoothness. operation of the platform.
- **A6:** Setup platform settings such as payment gateway settings, notification security configurations, templates and security configurations.

3.4 Non-Functional Requirements

- **Security:** The security of all the user data such as personal information, passwords, and payment needs to be coded and kept safely. The platform is using HTTPS encryption for all data transmission. Payment processing through Stripe is following industry-standard security practices and PCI DSS compliance. User passwords with Mongo DB should is hashed using secure algorithms before storing in the database.
- **Usability:** We have used advanced technologies like tailwind to make web app completely responsive for all screen sizes including laptops, tablets, and mobile phones.There is proper instructions to guide user for example with virtual try on , on how to use it alongside simple and interactive design of website along with the seller panel so seller could operate without requiring much technical knowledge
- **Reliability:** We have used hosting technologies like render that provides upto 95 percent up time for web applications even during high traffic time,
- **Maintainability:** The code is well structured and every module is made seperatly to ensure easy maintenance and future update

3.5 Use case

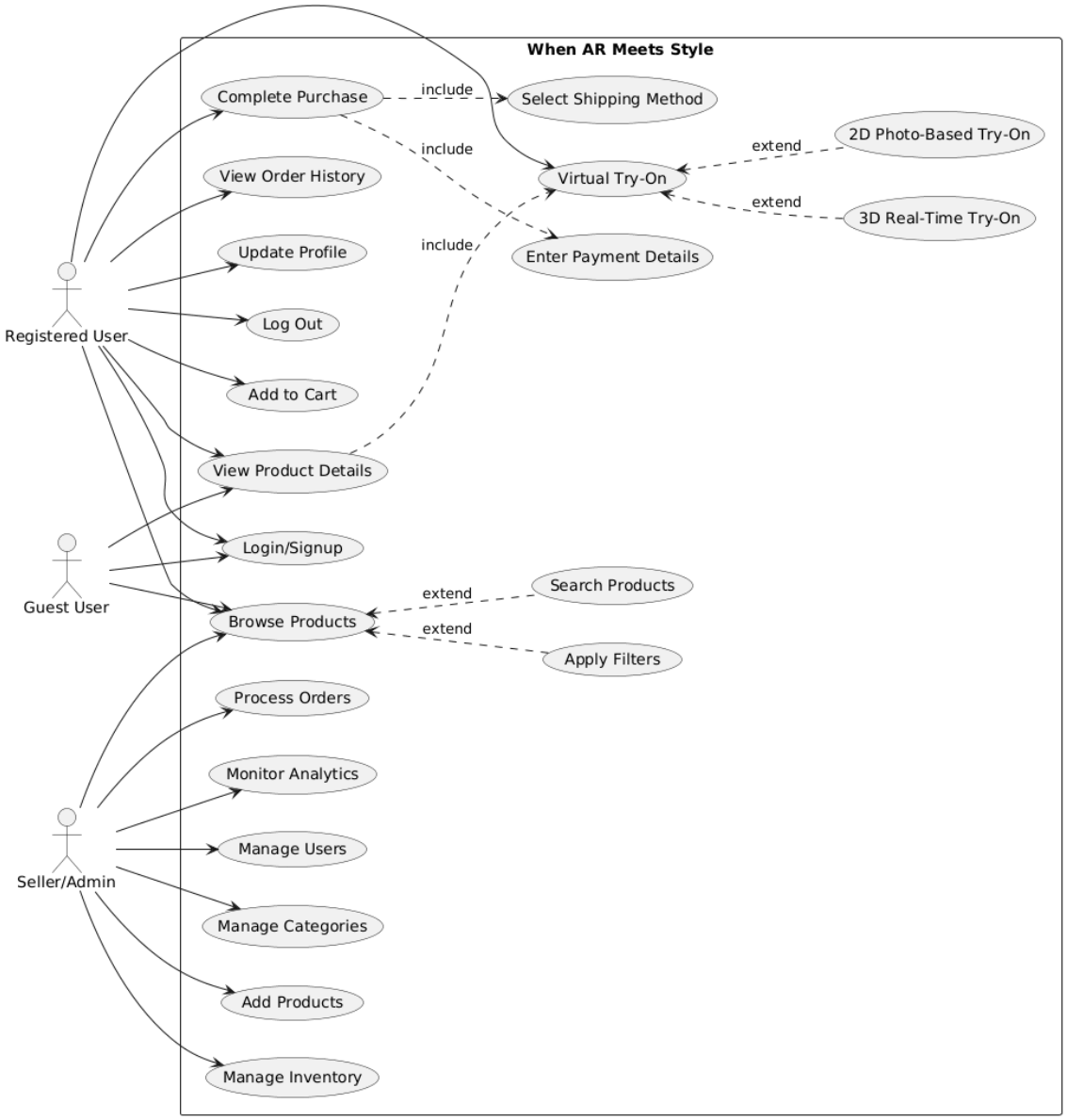

Figure 3.1: Use Case Diagram

3.5.1 Use Case Descriptions

Table 3.1: Use Case Description: Login/Signup

Use Case ID	**UC-1**
Use Case Name	Login/Signup
Actors	Guest User, Registered User
Data	User credentials (email, password), registration details (name, contact, address)
Trigger	User navigates to login or signup page
Precondition	User is on the platform's main interface
Assumptions	User has a valid email address and internet connection
Normal Flow of Events	1. User clicks on Login/Signup button. 2. System displays login form or registration form. 3. For login: User enters email and password. 4. For signup: User enters name, email, password. 5. System validates the input data. 6. System authenticates user and redirects to home page.
Alternative Flow of Events	1. If login credentials are incorrect, display error message and allow retry. 2. If email already exists during signup, prompt user to login instead. 3. If validation fails, display specific error messages for each field. 4. User can reset password through "Forgot Password" option.
Main Success Scenario	User successfully logs in or creates account and is redirected to home page
Postcondition	User is authenticated and can access personalized features

Table 3.2: Use Case Description: Browse Products

Use Case ID	**UC-2**
Use Case Name	Browse Products
Actors	Registered User, Guest User
Data	Product catalog data, user search/filter inputs
Trigger	User navigates to the product catalog or enters a search query
Precondition	User is on the platform's main interface
Assumptions	MongoDB product database is accessible and up-to-date
Normal Flow of Events	1. User browses the catalog or enters search/filter criteria (e.g., size, category). 2. System queries MongoDB for matching products. 3. Results are displayed with images, prices, and details.
Alternative Flow of Events	1. If no products match, display "no results found" and suggest broader filters. 2. If database query fails, show "unable to load products" and retry. 3. If search input is invalid, prompt user to refine their query.
Main Success Scenario	User views a list of relevant products for try-on or purchase
Postcondition	Product views are logged, and user can proceed to try-on or purchase

Table 3.3: Use Case Description: View Product Details

Use Case ID	**UC-3**
Use Case Name	View Product Details
Actors	Registered User, Guest User
Data	Product information (name, price, description, images, sizes, colors, reviews, seller info)
Trigger	User clicks on a product from browse or search results
Precondition	User has selected a product to view
Assumptions	Product data is available in database
Normal Flow of Events	1. User clicks on product card or image. 2. System retrieves complete product details from MongoDB. 3. System displays product page with images, price, description, available sizes, colors. 4. User can view multiple product images. 5. User can see customer reviews and ratings. 6. User can view seller information. 7. User can access Virtual Try-On feature from this page. 8. User can add product to cart.
Alternative Flow of Events	1. If product is out of stock, display "Out of Stock" message. 2. If images fail to load, display placeholder image.
Main Success Scenario	User views complete product information and can proceed to try-on or purchase
Postcondition	Product details are displayed and user can take further actions

Table 3.4: Use Case Description: Virtual Try-On

Use Case ID	**UC-4**
Use Case Name	Virtual Try-On
Actors	Registered User, Guest User
Data	Product 3D model, user camera feed or uploaded photo, body tracking data
Trigger	User clicks on "Try On" button on product details page
Precondition	User is viewing a product and chooses to try it virtually
Assumptions	User's device has camera capability
Normal Flow of Events	1. User clicks "Virtual Try-On" button. 2. User allows camera access, system uses Three.js for real-time body tracking and overlays garment. 3. User can see how the product looks on them. 5. User can proceed to add to cart or continue browsing.
Alternative Flow of Events	1. camera access is denied, enable it on browser. 2. User can exit try-on and return to product page anytime.
Main Success Scenario	User successfully tries on product virtually
Postcondition	User has visualized the product on themselves and can make informed purchase decision

Table 3.5: Use Case Description: Add to Cart

Use Case ID	**UC-5**
Use Case Name	Add to Cart
Actors	Registered User
Data	Product details, selected size, color, quantity, cart information
Trigger	User clicks "Add to Cart" button on product page
Precondition	User is viewing product details and is logged in
Assumptions	User has selected size and color options if applicable
Normal Flow of Events	1. User selects size and color options. 2. User specifies quantity. 3. User clicks "Add to Cart" button. 4. System validates selection and availability. 5. System adds product to user's cart in database. 6. System displays confirmation message. 7. User can continue shopping or proceed to checkout.
Alternative Flow of Events	1. If size/color not selected, prompt user to select. 2. If product is out of stock, display error message. 3. If item already in cart, prompt to update quantity. 4. User can view cart anytime to review items.
Main Success Scenario	Product is successfully added to cart
Postcondition	Product is stored in user's cart and available for checkout

Table 3.6: Use Case Description: Complete Purchase

Use Case ID	**UC-6**
Use Case Name	Complete Purchase
Actors	Registered User
Data	Cart items, shipping address, payment details, order information
Trigger	User clicks "Proceed to Checkout" from cart page
Precondition	User has items in cart and is logged in
Assumptions	Payment gateway (Stripe) is operational
Normal Flow of Events	1. User reviews cart items and clicks checkout. 2. System displays order summary with total amount. 3. User enters or confirms shipping address. 4. User selects shipping method. 5. User enters payment details (credit card). 6. System processes payment through secure gateway. 7. System creates order in database. 8. System displays order success page with order ID.
Alternative Flow of Events	1. If payment fails, display error and allow retry. 2. User can edit cart items before final purchase. 3. If address is in missing details, prompt for complete information.
Main Success Scenario	Order is successfully placed and payment is processed
Postcondition	Order is created, payment is confirmed, seller is notified, and user receives confirmation

Table 3.7: Use Case Description: Add Products

Use Case ID	**UC-7**
Use Case Name	Add Products
Actors	Seller
Data	Product details (name, description, price, images, sizes, colors, category, stock quantity)
Trigger	Seller clicks "Add New Product" in seller panel
Precondition	Seller is logged in and verified
Assumptions	Seller has product information and images ready
Normal Flow of Events	1. Seller clicks "Add Product" button. 2. System displays product creation form. 3. Seller enters product name, description, price. 4. Seller uploads product images. 5. Seller selects category. 6. Seller specifies available sizes and colors. 7. Seller enters stock quantity. 8. Seller clicks "Submit" button. 9. System validates all fields. 10. System saves product to database. 11. System displays success message and product listing.
Alternative Flow of Events	1. If required fields are missing, display validation errors. 2. If image upload fails, allow retry.
Main Success Scenario	Product is successfully added and visible to customers
Postcondition	Product is listed on platform and available for purchase

Table 3.8: Use Case Description: Manage Inventory

Use Case ID	**UC-8**
Use Case Name	Manage Inventory
Actors	Seller
Data	Product list, stock quantities, product status
Trigger	Seller navigates to "Manage Inventory" section
Precondition	Seller is logged in and has products listed
Assumptions	Seller wants to update product information or stock
Normal Flow of Events	1. Seller clicks "Manage Inventory" option. 2. System displays list of all seller's products. 3. Seller can view current stock levels. 4. Seller selects product to edit. 5. Seller can update price, description, images, or stock quantity. 6. Seller can mark products as out of stock or in stock. 7. Seller can delete discontinued products. 8. System saves changes to database. 9. System displays confirmation message.
Alternative Flow of Events	1. Seller can search for specific products.2. If update fails, display error and allow retry.
Main Success Scenario	Inventory is successfully updated
Postcondition	Product information and stock are updated

Table 3.9: Use Case Description: Process Orders

Use Case ID	**UC-9**
Use Case Name	Process Orders
Actors	Seller
Data	Order details, customer information, product details, order status
Trigger	Seller receives new order notification or navigates to orders section
Precondition	Seller is logged in and has received orders
Assumptions	Orders are available in seller's dashboard
Normal Flow of Events	1. Seller navigates to "Orders" section. 2. System displays list of all orders (pending, confirmed, shipped, delivered). 3. Seller views order details including customer info and products. 4. Seller confirms order and updates status. 5. Seller marks order as delivered when completed. 6. System notifies customer of status updates. 7. System updates order status in database.
Alternative Flow of Events	1. Seller cancel order. 2. Seller can filter orders by status or date. 3. Seller can search for specific orders. 4. If order details are unclear, seller can contact customer.
Main Success Scenario	Order is successfully processed and delivered to customer
Postcondition	Order status is updated and customer is notified

Table 3.10: Use Case Description: Monitor Analytics

Use Case ID	**UC-10**
Use Case Name	Monitor Analytics
Actors	Seller
Data	Sales data, order statistics, revenue metrics, product performance, user activity
Trigger	User navigates to analytics or dashboard section
Precondition	User is logged in as seller or admin
Assumptions	Analytics data is being collected and stored
Normal Flow of Events	1. User clicks on "Analytics" or "Dashboard" option. 2. System retrieves analytics data from database. 3. For Seller: System displays total sales, number of orders, best-selling products, revenue trends. 4. For Admin: System displays total users, active sellers, platform revenue, order statistics.
Alternative Flow of Events	1. If no data available, display "No data to show" message. 2. User can customize dashboard view.
Main Success Scenario	User views comprehensive analytics and insights
Postcondition	User has access to performance data for decision making

Table 3.11: Use Case Description: Manage Users and Categories

Use Case ID	UC-11
Use Case Name	Manage Users and Categories
Actors	Seller/Admin
Data	User accounts (customers and sellers), user status, activity logs, product categories, subcategories, category hierarchy
Trigger	navigate to user management or category management section
Precondition	logged in to the system
Assumptions	Platform has registered users and needs organized product categorization
Normal Flow of Events	1. Seller clicks on "Manage Users" or "Manage Categories" option. 2. For Users: System displays list of all users with their details. Admin can view user profile information and activity, search for specific users, suspend accounts that violate policies, or delete accounts if necessary. 3. For Categories: System displays current category structure. Admin can add new categories (e.g., Men's Clothing, Women's Clothing, Shoes), add subcategories under main categories, edit existing category names or descriptions, delete unused categories, or reorder categories for better organization. 4. System updates user status or category structure in the database. 5. System sends notifications to affected users . 6. System updates category display throughout the platform.
Alternative Flow of Events	1. For Users: can view user order history and complaints, remove suspicious accounts 2. For Categories: If a category has products, warn before deletion. Admin can move products to different categories, set category images or icons. If category name already exists, prompt for a unique name.
Main Success Scenario	User accounts or categories are successfully managed according to admin actions.
Postcondition	User account status or product categories are updated and changes are reflected throughout the system.

Table 3.12: Use Case Description: Log Out

Use Case ID	**UC-11**
Use Case Name	Log Out
Actors	Registered User AND Seller
Data	User/Seller session data
Trigger	User/Seller clicks "Log Out" button
Precondition	User/Seller is logged in to the system
Assumptions	User/Seller wants to end their session
Normal Flow of Events	1. User/Seller clicks "Log Out" option from profile menu. 2. System terminates user session. 3. System clears authentication tokens. 4. System redirects user to home page or login page. 5. System displays logout confirmation message.
Alternative Flow of Events	1. System can prompt confirmation before logging out if there are unsaved changes. 2. User can cancel logout and continue session.
Main Success Scenario	User/Seller is successfully logged out
Postcondition	User/Seller session is terminated and user must login again to access personalized features

Table 3.13: Requirement Traceability Matrix

Requirement	**Description**	**Mapped Use Cases**
C1-C3 (Browse)	Authentication, browse and search products, view details	UC-1, UC-2, UC-3
C4-C5 (Try-On)	virtual try-on	UC-4
C6-C8 (Purchase)	Shopping cart, secure payment, order tracking	UC-5, UC-6
S1-S3 (Products)	Seller authentication, add products, manage inventory	UC-1, UC-7, UC-8
S4-S6 (Orders)	View and process orders, sales analytics	UC-9, UC-10
A1-A3 (Management)	Admin authentication, user and category management	UC-1, UC-11
A4-A5 (System)	Monitor performance, configure platform settings	UC-10, UC-11

Chapter 4

Design

Designing an efficent e-commerce platform with virtual try-on features is requiring careful planning to ensure smooth interaction between different parts of the system and good user experience. The design approach for "When AR Meets Style" is focusing on making the system modular, easy to maintain, and able to expand when more users are using the platform. Our architecture is designed in a way that different components can work together easily while supporting a dynamic shopping experience with real-time virtual visualization. The system is built to be user-friendly for customers who want to browse and try on products, and also simple for sellers to manage their products and orders without requiring technical knowledge.

4.1 System Architecture

- **Frontend (React.js):**A responsive web that is rendering the user interface, controlling the state of the application, as well as responding to the user input. React is providing dynamism and interactivity to browse products, virtual try-on, checkout process, shopping cart and checkout process. The front end is interacting with the backend by means of REST API calls

- **API Layer (Node.js / Express):** We are using REST endpoints which is then implementing authentication and authorization using JWT tokens for tasks like input validation, and handling all business logic.

- **Virtual Try-On Engine :**A 3D visualization module that is facilitating real-time. virtual try-on experiences. Three.js is visualising 3D models of garments and superimposing. record videos of them on the body of the user through camera feed.

- **2D Try-On :** With an AI model trained on a large dataset, we are able to provide human. dimensions and clothing dimensions in order to produce an output of the user wearing. the selected clothing.
- **Database (MongoDB):** database that is storing all entities including user accounts, seller information, orders and product information. MongoDB is offering flexible schema. and effective information search of the site.
- **Seller Panel :** We created a seller panel which assists in creating a seller panel that is easy to add and maintain. products need not have a lot of technical knowledge and are easy to interact with.
- **Payment Integration:** Stripe gateway which will provide safe and secure transaction incompliance to industry standards

This is a high-level architecture diagram that illustrates the entire system structure of our project and it is divided into several layers. The Client Layer shows how the users communicate with one another through web browsers, mobile browsers, and cameras in a secure way through the HTTPS. The React.js Frontend which is written in the Presentation Layer has two main interfaces, the Customer Interface where the shopper interface is and the Seller (Admin) Dashboard where the business operator interface is. The Express.js Server is located in the API Layer and responds to all the requests using REST API and, in turn, addresses these requests to the respective services, including Order API, Payment API, Authentication API, Product API, and Seller (Admin) API. The Business logic layer contains the primary services: Order Service which is used to process purchases, Payment Service which is used to process transactions, Authentication Service which is used to log in users and secure them, and Product Service which is used to manage the product catalog. The Virtual Try-On Service of this layer is related to the camera stream and processes product data to be visualized. Data Layer is the MongoDB database that contains five large collections, including Orders, Cart, Payments,Users, and Products. The External Services section reflects three essential integrations, which are Stripe Payment Gateway to provide secure payments, Camera Input Service to access the camera of the users to make virtual try-on, and Three.js Engine to render models in 3D and produce effect of visualization. All the layers are connected by arrows showing data flow between the components: the frontend works with a REST API, the services interact with the database, and external services are called whenever they are required to provide a particular feature.

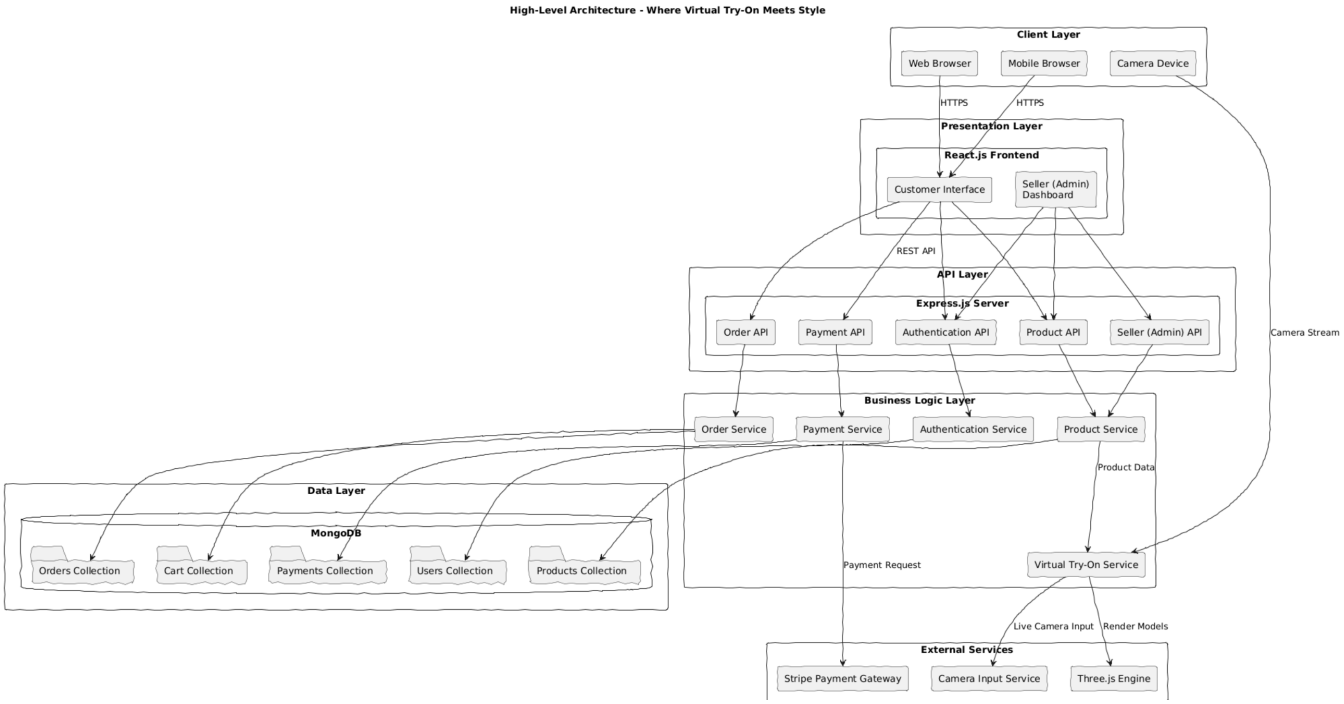

Figure 4.1: High Level Architecture Diagram

4.2 Design Constraints

- Virtual try-on accuracy is constrained by current AR technology limitations, which cannot provide 100% accurate fit representation and cannot replicate the actual feeling of wearing clothes.
- The platform must work within web browser limitations, which are restricting access to device hardware and camera capabilities compared to native mobile applications.
- Creating custom 3D models for all clothing products is not feasible due to resource, time, and cost constraints, which is limiting virtual try-on availability to products with pre-made sample models only.
- The system must function on low-end devices with poor cameras and limited processing power, which is constraining the complexity of 3D rendering and requiring simpler 2D alternatives which could cater diffrent user preferences
- The system relies on third-party services : Stripe to take payments, zrok to access the camera, and so on. Three.js the availability and performance of which is not controllable by the project, which is limiting system reliability and necessitating fallback.

4.3 Design Methodology

Our design takes a cyclic strategy thus all the elements are collaborating effectively:

- **Modular Architecture:** The system will be designed in distinct modules of. React.js/frontend, Node.js/ backend, MongoDB/database, Three.js/ virtual tryon engine, and Stripe/payment processing. Each module is developed applied independently and integrated together with clearly defined APIs, and therefore easier to maintain.
- **Iterative Development:**The platform is being built in cycles and the first phase is. basic authentication, product browsing, basic cart and feature by feature. innovative capabilities try-on, seller analytics. This is the approach that is permitting. easy changes to the stages and testing.

4.4 High Level Design

4.5 High Level Design

Here, we are discussing in more detail aspects discussed in the Architecture. Major technologies: React SPA as the frontend interface, Express API with REST endpoints

as API. Domain Services (Authentication, Product Management, Virtual Try-On Engine, backend, Orders, Payments, Seller Panel, Admin Operations), MongoDB as a database, and External Integrations (Stripe to take payment, Three.js to render 3D, AI Model to render 2D). try-on, zrok to avoid accessing the camera).

4.5.1 Data flow diagram

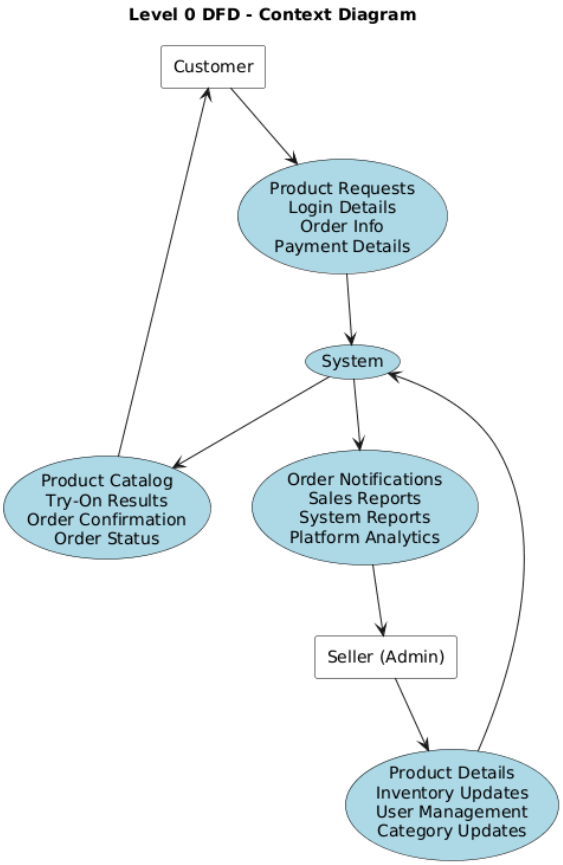

Figure 4.2: DFD Level 0

The context view of the whole system is being presented by this Level 0 Data Flow Diagram, it depicts external entities interacting with Where AR Meets Style system. The diagram contains two classes of objects, which are the rectangles of the outside bodies (Customer and Seller/Admin), and the oval of the system. The Customer entity is providing data streams to the system such as Product Requests in the process of browsing the items, Login Details in the process of authentication, Order Info in the course of buying the products and Payment Details in the course of the checkout process. The system is in turn sending Product Catalog which indicates available items, Try-On Results which is a virtual try-on option, Order Confirmation once a purchase is successful and Order Status as the delivery is in progress. The Seller (Admin) entity is communicating with the system via

Product Details when adding new items, Inventory Updates when tracking the stock and User Management commands that are administrative in nature. The system is reacting to sellers with Order Notifications to new purchases, Sales Reports that display business performance, System Reports that report about platform health, and Platform Analytics that can offer information about user behavior and sales patterns. Through this context diagram the system boundary is being defined and all of the external interactions are being displayed giving a high level overview of what data is entering and leaving the system but not internal processing details.

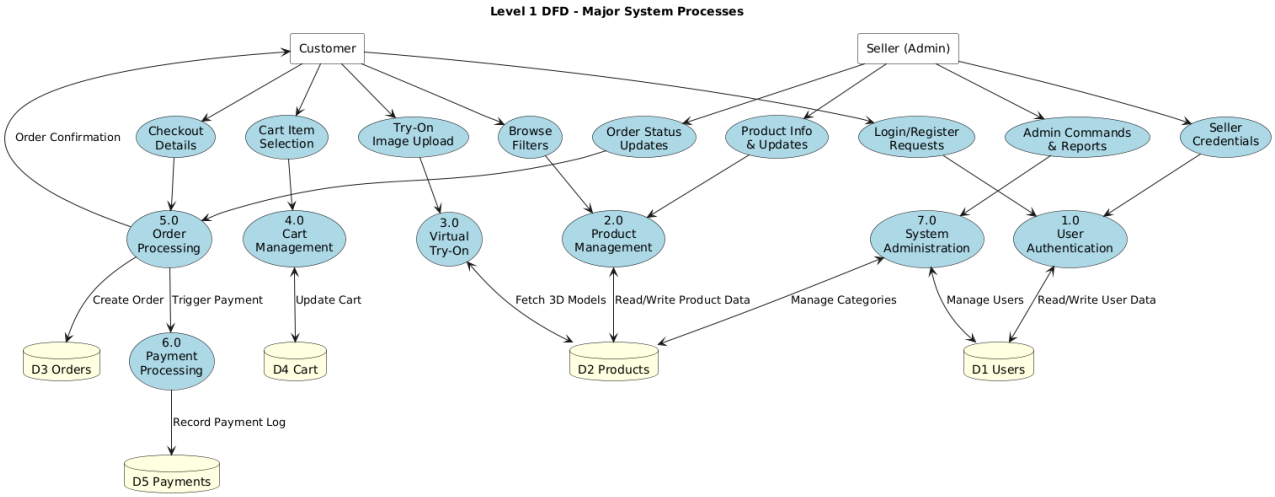

Figure 4.3: DFD Level 1

This Level 1 DFD divides this system into seven broad processes and demonstrates information flow between the entities, processes, and data stores. Process 1.0 (User Authentication) handles logins/Registration and communicates with D1 Users, and generates Auth Tokens. Process 2.0 (Product Management) involves product browsing and updates on sellers, with the help of D2 Products. The Process 3.0 (Virtual Try-On) uses 3D models on D2 to Fetch Try-On Requests. Cart Management (Process 4.0) is used to maintain and retrieve cart data in D4 Cart. Process 5.0 (Order Processing) takes care of checking out, generating order in D3, and liaising with sellers. Process 6.0 (Payment Processing) is used to process payment through Stripe and records the data into D5 Payments. Process 7.0 (System Administration) includes the handling of administration activities with the help of D1 and D2 and reports and analytics.

This Level 2 Data Flow Diagram will break down three large processes Virtual Try-On (3.0), Order Processing (5.0), and Product Management (2.0) into the specifics of their internal workflow. In Virtual Try-On, the Try-On Request of the customer is sent to Process 3.1 (Select Try-On Type) where the customer will pick between 3D or 2D. In the case of 3D, Process 3.2 connects to the Camera service of live video to trigger Process 3.4 to fetch product data provided by D2 Products and 3D garment models provided by D6, which is then rendered in real time by Process 3.5 using Three.js. In 2D, the request is sent to Process 3.3, where the user uploads a photo to it; the process then pulls model data to Process 3.4 and transfers both inputs to the external AI Model and then an output image result is returned. In the case of Order Processing, Process 5.0 is subdivided into the five steps, which are: validating cart items (5.1), calculating totals using D2 (5.2), processing payments and creating orders in D3 with logs in D5 (5.3), updating inventory in D2 (5.4) and sending confirmations and status updates in 5.5. Sellers have an opportunity to update the order status that is logged in D3 and sends notifications until delivery. In Product Management, Process 2.1 processes search and filter requests, Process 2.2 retrieves the list of products and maintains the list, Process 2.3 retrieves specific product data on the selected ones, and Process 2.4 supports inventory updates (add, edit, delete) in D2 with the notification to Process 2.2 of the necessary update of the catalog. On the whole, the figure shows how these sub-processes align to keep the data accurate, maintain the operations of customers and sellers and the normal operation of the systems.

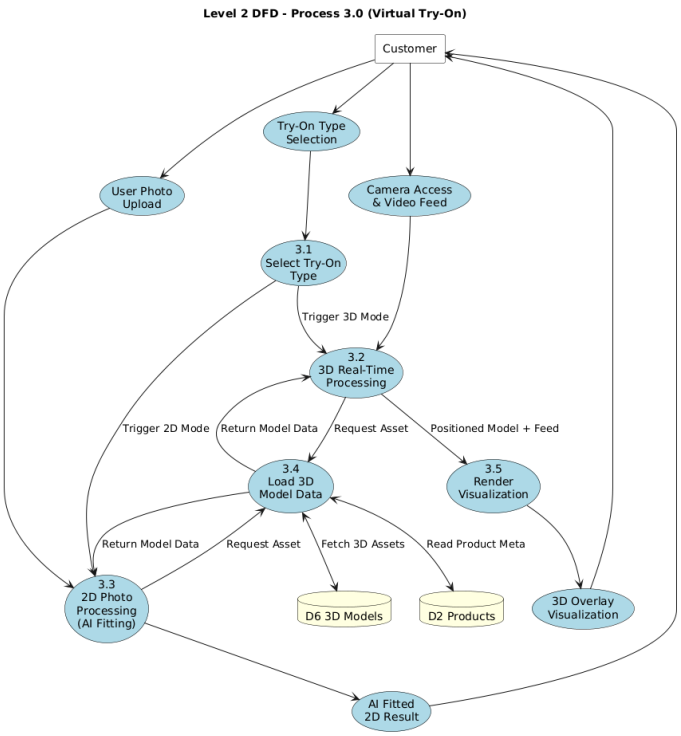

Figure 4.4: DFD Level 2 - Virtual try on

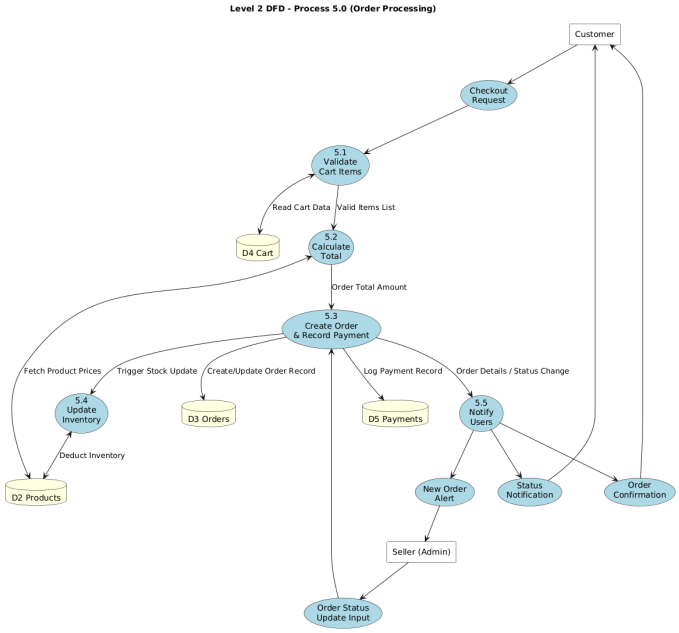

Figure 4.5: DFD Level 2 - Order Processing

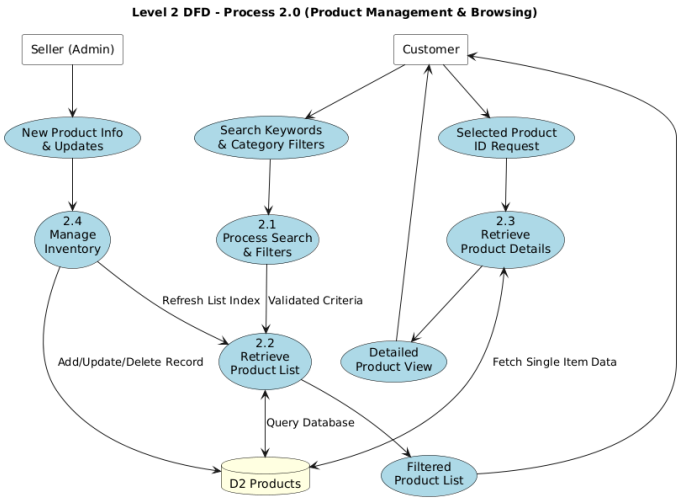

Figure 4.6: DFD Level 2 - Product Management and Browsing

4.6 Low Level Design

4.6.1 Component Diagram

This is a component diagram illustrating the system architecture in four key segments namely User Interface, Business Logic, External APIs, and Database. The User Interface section has Seller (Admin) Panel and Customer View which links to React Frontend component which is the primary presentation layer of all interactions with the users. The

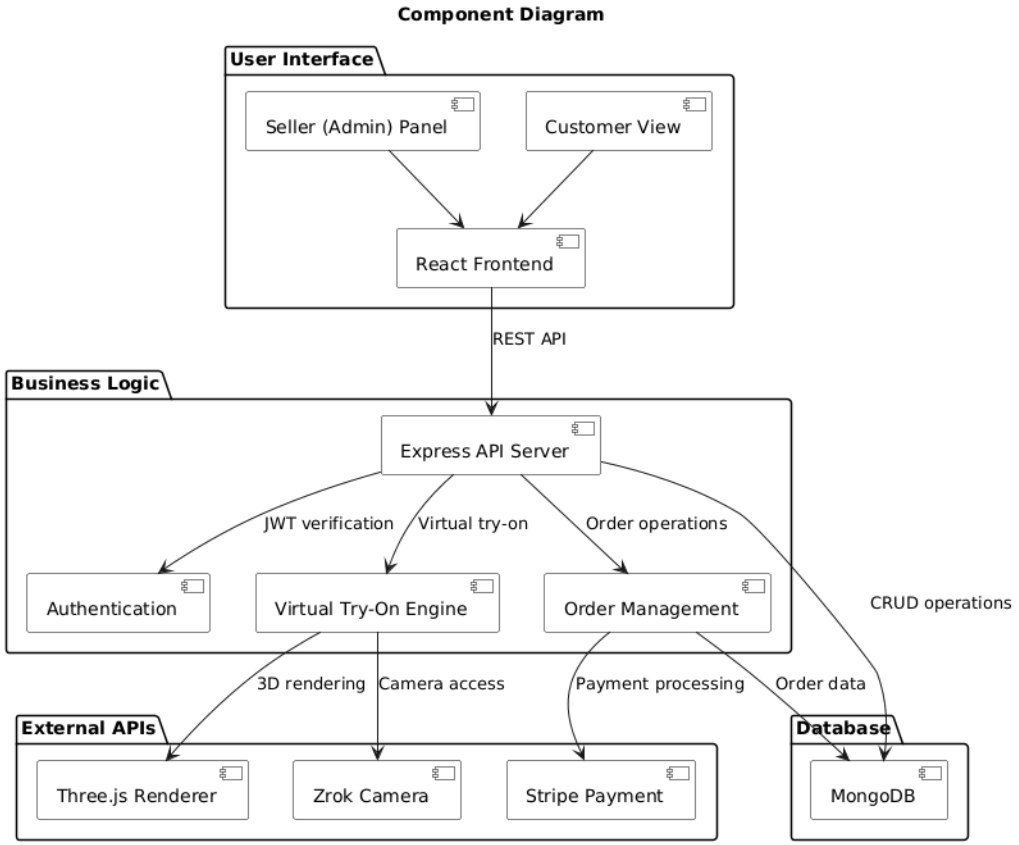

Figure 4.7: Component Diagram

React Frontend is communicating with the Business Logic layer by way of the REST API protocol. The Business Logic section depicts Express API Server as the backbone component that is handling all the activities at the back end and directing the request to the right services. Express API Server is linked to three primary business elements, where Authentication component deals with JWT verification and user login, Virtual Try-On Engine deals with the try-on functionality being connected to external services, and Order Management component deals with all the tasks related to the orders. Authentication component is executing JWT verification in secure access control. The Virtual Try-On Engine is communicating with Three.js Renderer performing the 3D rendering functions and Zrok Camera to access camera feed in order to provide real-time try-on functionality. The Order Management aspect is taking care of payment processing via Stripe Payment gateway and order data storage in the database. These are three essential third party services: Three.js Renderer which will be used to create a 3D representation, Zrok Camera which will be used to provide a secure access to the camera, and Stripe Payment that will be used to provide a secure environment with transactions. The Database section depicts the MongoDB as the primary data storage platform, which processes all CRUD operations of users, products, orders, payments, and carts. The diagram is showing arrows in which

the direction of all the arrows represents the direction of data flow and dependencies among various components of the system.

4.6.2 Deployment Diagram

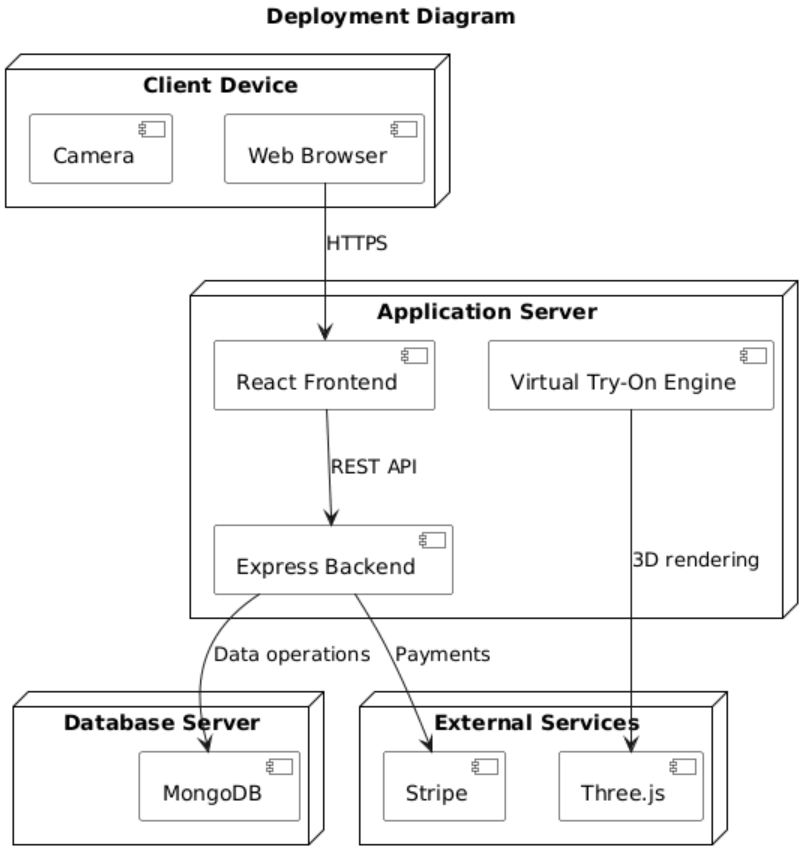

Figure 4.8: Deployment Diagram

This deployment diagram is a physical layout of system components. across servers and devices. Camera and Web Browser are found in the Client Device node, as the device of the user, which connects to the application via a browser and can include the use of the camera as a virtual try-on, ensuring the connection to the Application. Server via HTTPS. There are three main components found in the Application Server: React. Frontend, also known as the user interface; Virtual Try-On Engine, also known as AR. visualization and 3D rendering; and as well as Express Backend, that controls business logic and. REST API requests. The Express Backend is linked to Database Server and External Services. Users, products, orders are stored in the Database Server in MongoDB. and requests, and gives queries to the backend. The node of External Services includes Stripe to make payments and Three.js to render 3D. The diagram focuses on segregation of concerns:

the presentation is done on the client devices and processing on the application server. persistent data in the database server, and bespoke operations in the external services.

4.6.3 Activity Diagram

These five activity diagrams are the workflows that are significant in the system. Activity Diagram 1 (User Authentication) addresses the entry points of a user to the system through the user visiting the webpage and then splitting into different pathways depending on whether the user is a registered user or not and then converging again on the successful entry of a user into the system through a successful log in or sign up entry. Activity Diagram 2 (Product Discovery and Cart) depicts the browsing categories, viewing the details of products, adding items to the cart or loop shopping until users move to the cart. Activity Diagram 3 (Virtual Try-On) demonstrates the try-on flow in which users have an option to process 3D (camera-based) or 2D (photo upload) which both lead to visualization results which then result in either adding to cart or re-trying. Activity Diagram 4 (Checkout and Payment) shows how to input address, choose a payment method, verify order information, and pay via Stripe and in case of success, the order is created, and in case of failure, the system retries. Activity Diagram 5 (Product Management Seller Side) describes the process of adding, editing, and deleting product that includes the form input, validation, saving changes into the database, and displaying successful messages.

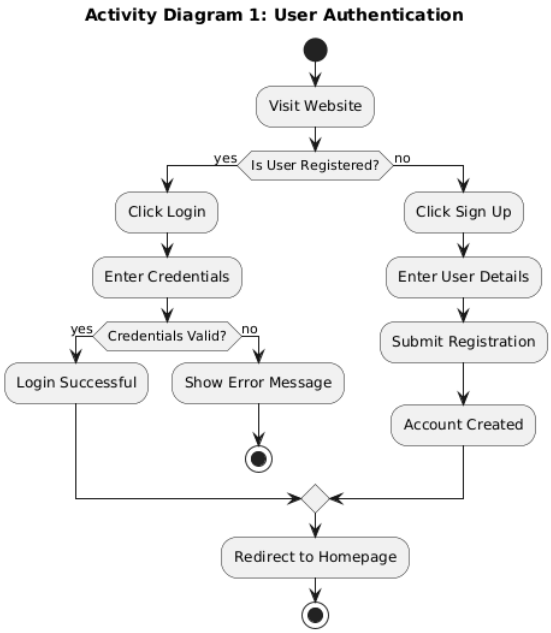

Figure 4.9: Activity Diagram-User Authentication

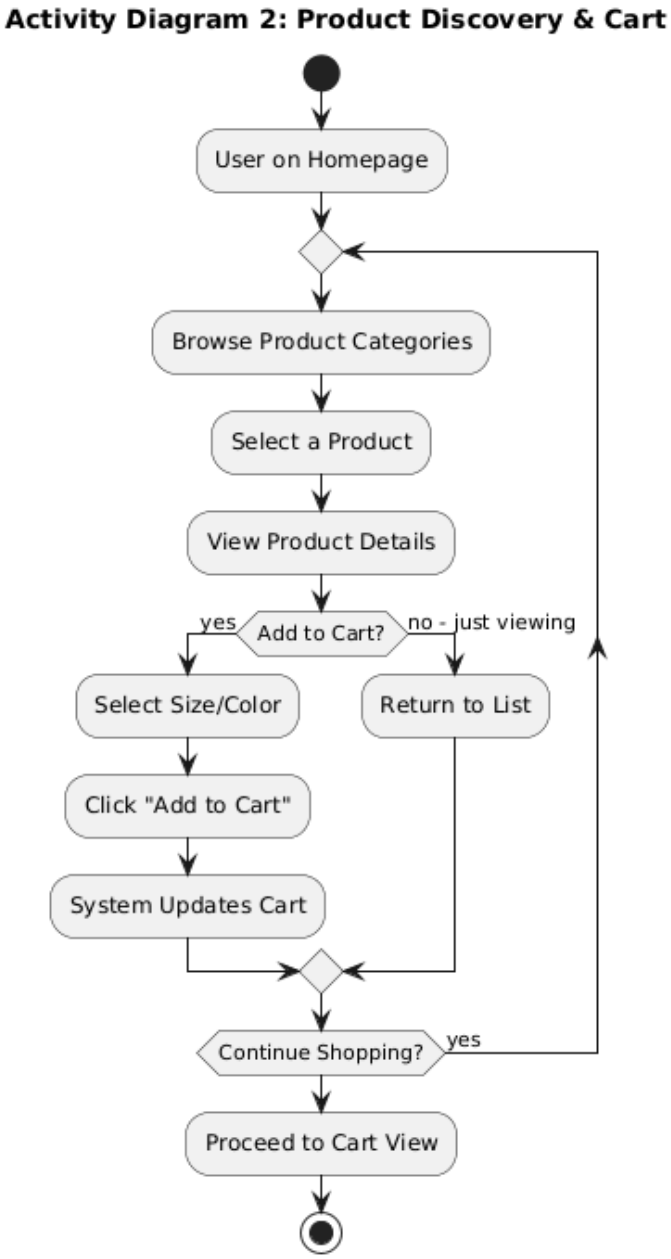

Figure 4.10: Activity Diagram-Browsing And Adding Products to cart

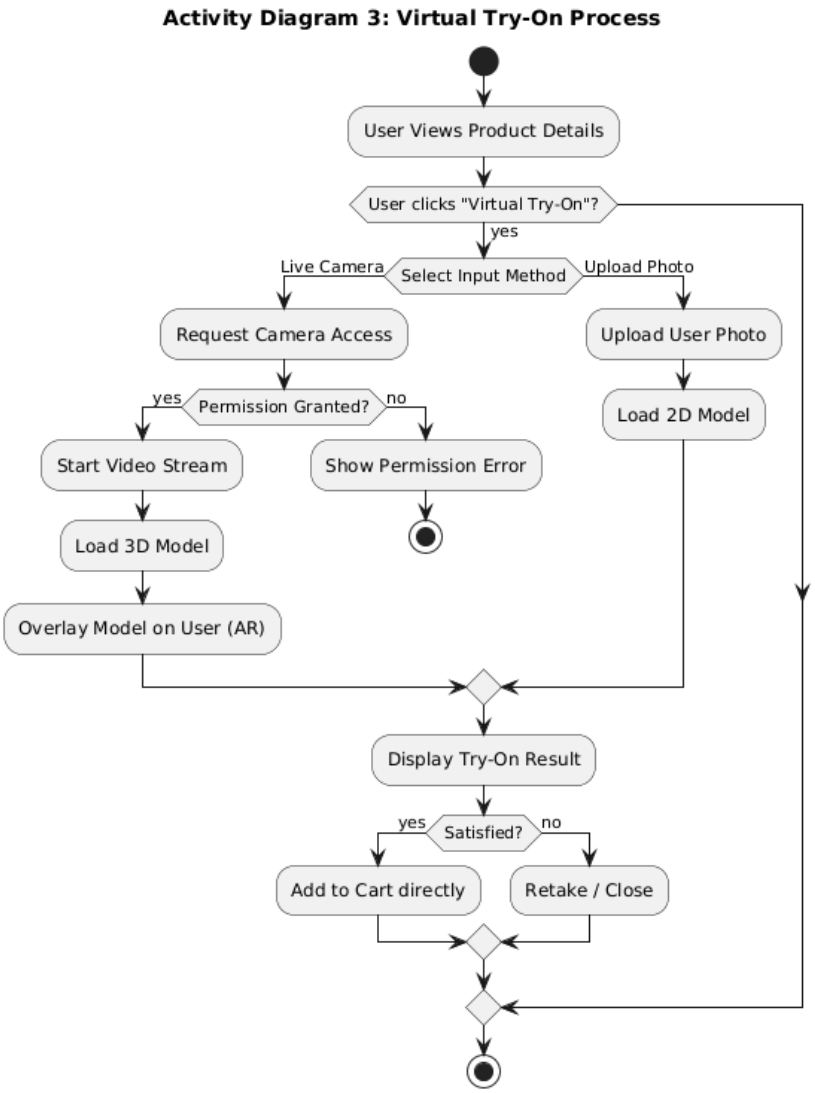

Figure 4.11: Activity Diagram-Virtual Try-on

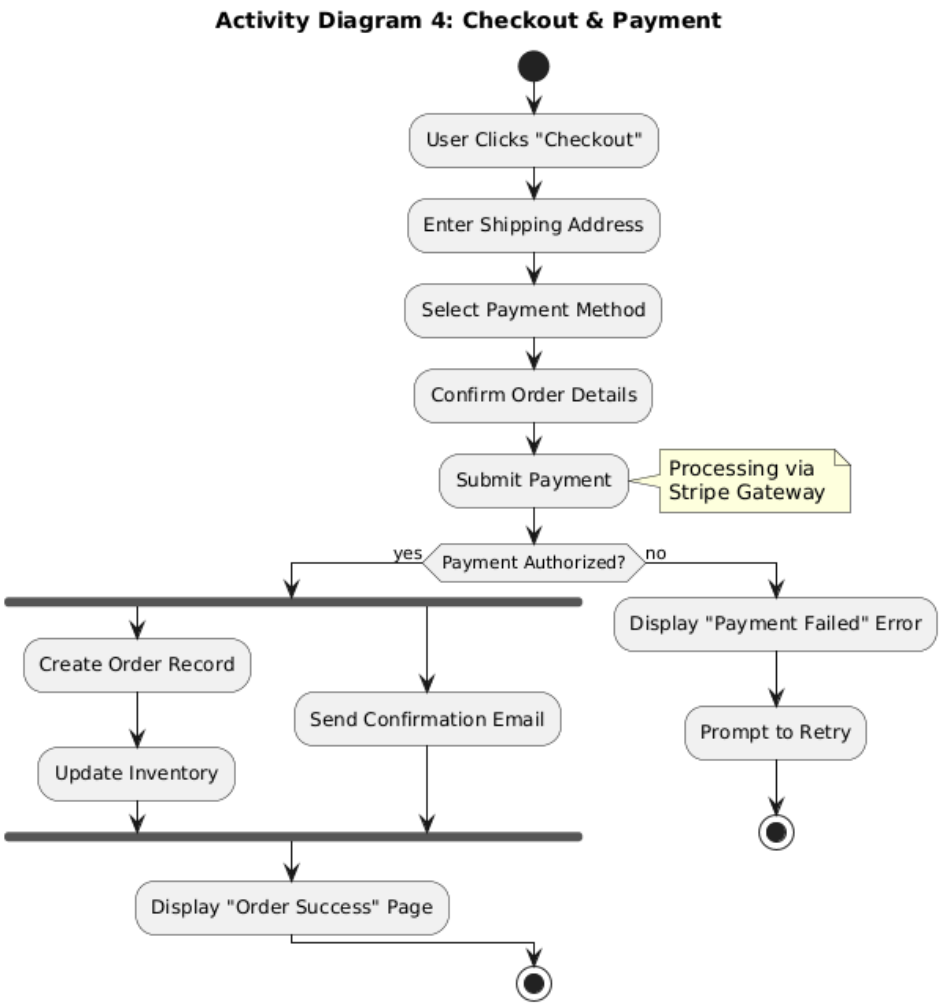

Figure 4.12: Activity Diagram-Checkout and Payment

4.6.4 Sequence Diagram

These sequence diagrams are aimed at showing the time-sequenced interactions of the system components with respect to key use cases. According to the Sequence Diagram 1 (User Authentication), the Customer is shown as logging in after clicking on Browse Products by sending a request to the frontend which in turn sends the request to the backend which sends the request to MongoDB and the Customer is shown as logged in due to the issue of a JWT Token. Sequence Diagram 2 (Browsing Products) shows that Customer sends a request to the frontend which sends a request to the backend which sends a request to MongoDB and the user is shown as logged in. The Virtual Try-On has two edges of its Sequence Diagram 3, the frontend requests access to the camera, receives live stream, fetches 3D model on the backend, and renders with Three.js after which the results are displayed to the customer, although in 2D, the user uploads a photo, which the backend sends to the AI Model Service, which sends the results back to the customer. The shop

shopping and checkout are depicted in the Sequence Diagram 4 (Shopping and Checkout), and it is described through the following way: the backend adds the items to the cart and sends them to MongoDB and the backend creates orders and receives payment by Stripe, changes the status of the order, and the frontend displays confirmation. In the Seq5 (Seller Management) it is indicated that sellers will add the products as per the frontend which sends the request to the backend which in turn stores the products in the MongoDB and the success will be verified. The figures take the conventional sequence representation that utilizes activating bars, lifelines and return arrows to illustrate the request-response patterns and timing order in the system.

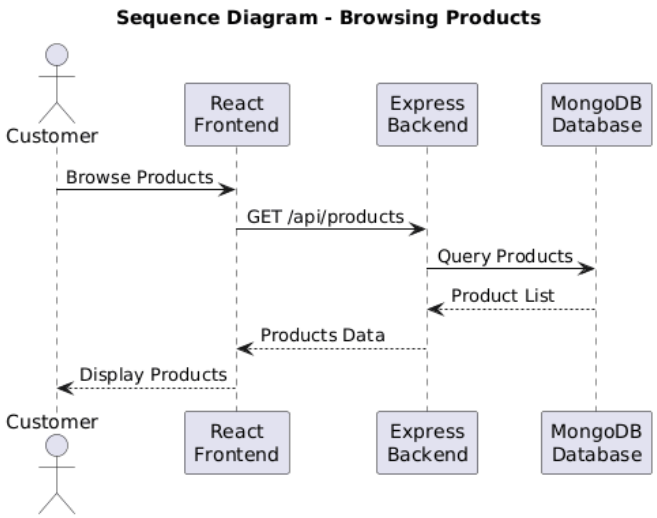

Figure 4.13: Product Browsing - Sequence Diagram

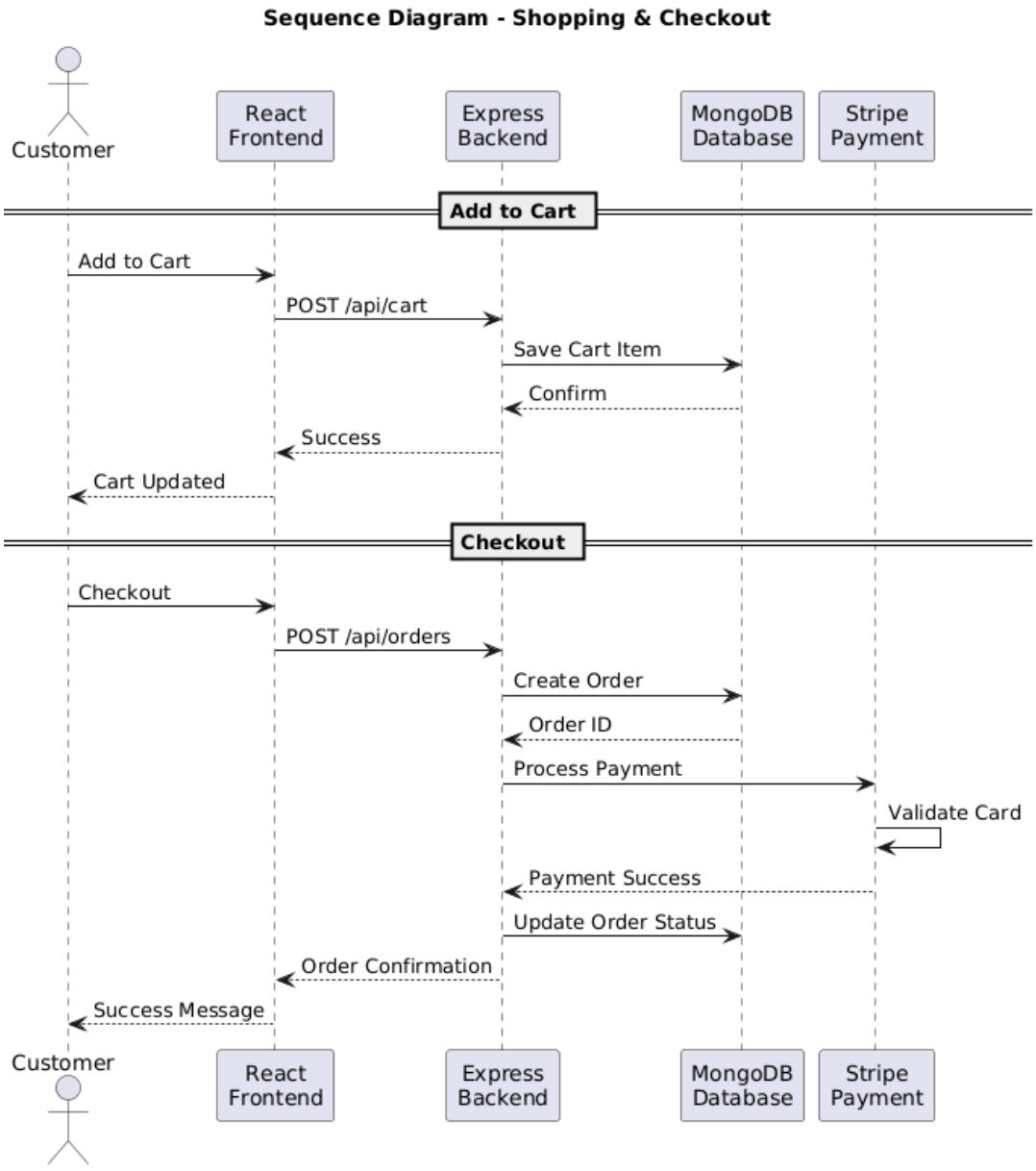

Figure 4.14: Shopping and Checkout - Sequence Diagram

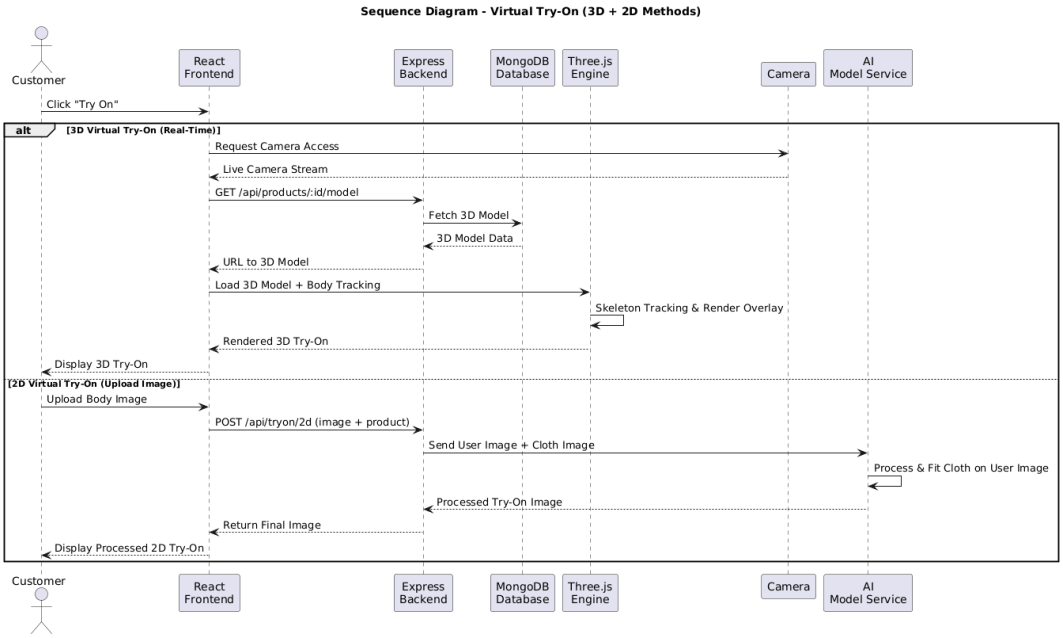

Figure 4.15: Virutal Try On - Sequence Diagram

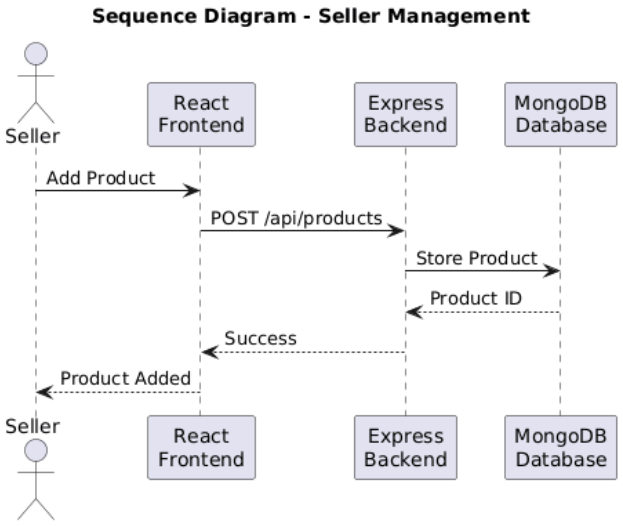

Figure 4.16: Shopping and Checkout Sequence Diagram

4.6.5 Entity-Relationship Diagram

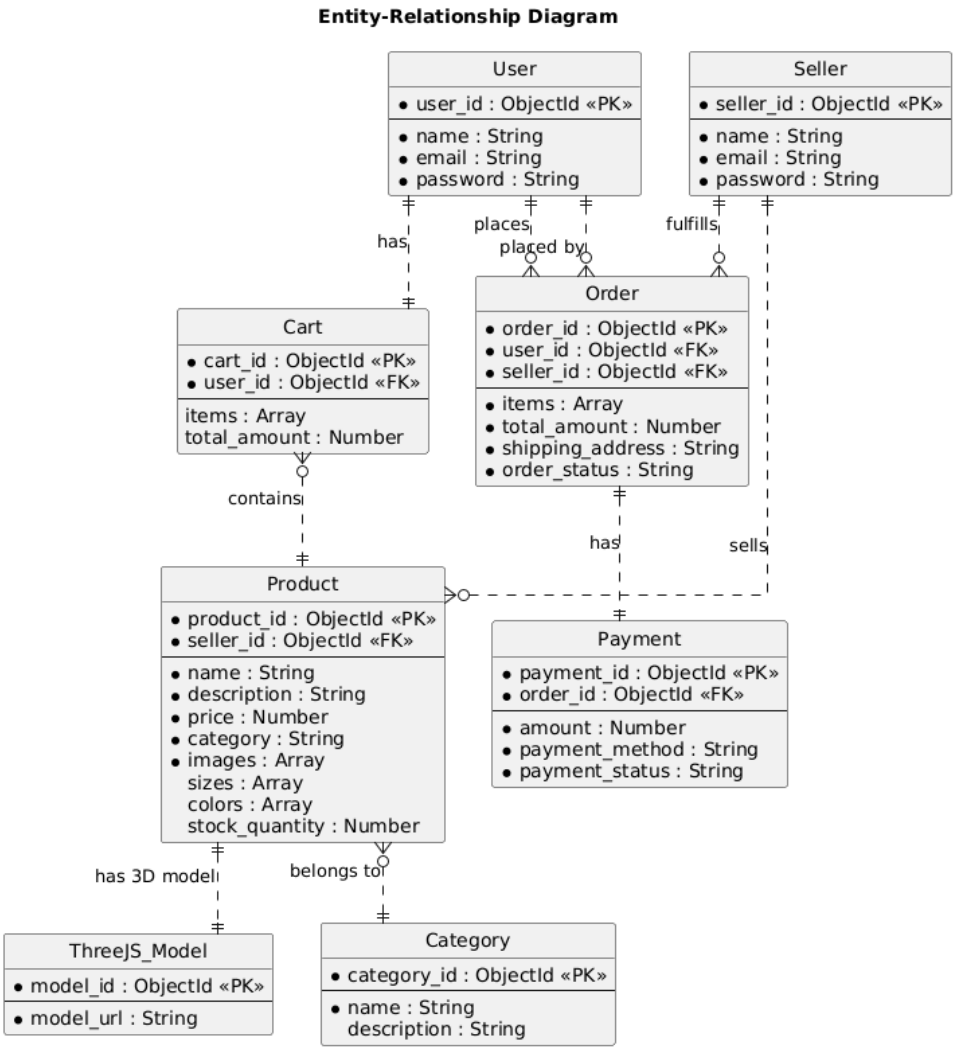

Figure 4.17: Entity-Relationship Diagram

This entity-relationship illustration depicts the database schema structure proposed for our project, showing entities, attributes, and links between data objects. The schema is built around two major actors: the User and Seller entities, each having similar records with user_id as Primary Key (PK), name, email, and password. The User has a one-to-one relationship with the Cart entity and a one-to-many relationship with the Order entity. The Order entity is a central hub of transactions between user_id and seller_id as Foreign Key (FK), including shipping address and order status, and connects directly to the Payment entity, which stores transaction details such as payment method and payment status. The Product entity models inventory maintained by the seller and includes attributes such as price, category, stock quantity, images, sizes, and colors arrays. This Product entity has three critical relationships: it belongs to a Category entity for classification, it is sold by a

Seller, and it is linked to a 3D model in the ThreeJS_Model entity.

4.7 Database Design

The database design for "When AR Meets Style" is using MongoDB, a NoSQL database that is providing flexible schema and efficient data storage for the e-commerce platform. We are using MongoDB because as its part of MERN and allows dynamic schema changes, supporting complex data structures like arrays and nested objects, and scaling horizontally as the platform grows. The database is organized into several collections that are storing information about users, sellers, products, orders, payments, shopping carts, categories, and 3D models. Each collection is having a unique identifier (ObjectId) as the primary key, and relationships between collections are maintained through foreign key references. This design is ensuring data integrity while providing the flexibility needed for an e-commerce platform with virtual try-on capabilities. The database structure is supporting all functional requirements including user authentication, product management, order processing, and virtual try-on features.

Table 4.1: Core Collections and Schema

Collection	**Key Fields / Notes**
users	_id ObjectId PK, email UNIQUE, password String (hashed), name String, role String (customer/admin)
sellers	_id ObjectId PK, email UNIQUE, password String (hashed), name String, business_name String, verification_status String
products	_id ObjectId PK, seller_id ObjectId FK sellers(_id), model_id ObjectId FK threejs_models(_id), name String, description String, price Number, category String, images Array, sizes Array, colors Array, stock_quantity Number
threejs_models	_id ObjectId PK, model_url String, model_type String, file_size Number, texture_url String
categories	_id ObjectId PK, name String UNIQUE, description String, parent_category ObjectId
carts	_id ObjectId PK, user_id ObjectId FK users(_id), items Array (product_id, quantity, size, color), total_amount Number
orders	_id ObjectId PK, user_id ObjectId FK users(_id), seller_id ObjectId FK sellers(_id), payment_id ObjectId FK payments(_id), items Array (product details), total_amount Number, shipping_address String, order_status String, tracking_info String
payments	_id ObjectId PK, order_id ObjectId FK orders(_id), amount Number, payment_method String, payment_status String, stripe_payment_id String, transaction_date Date

4.8 GUI Design

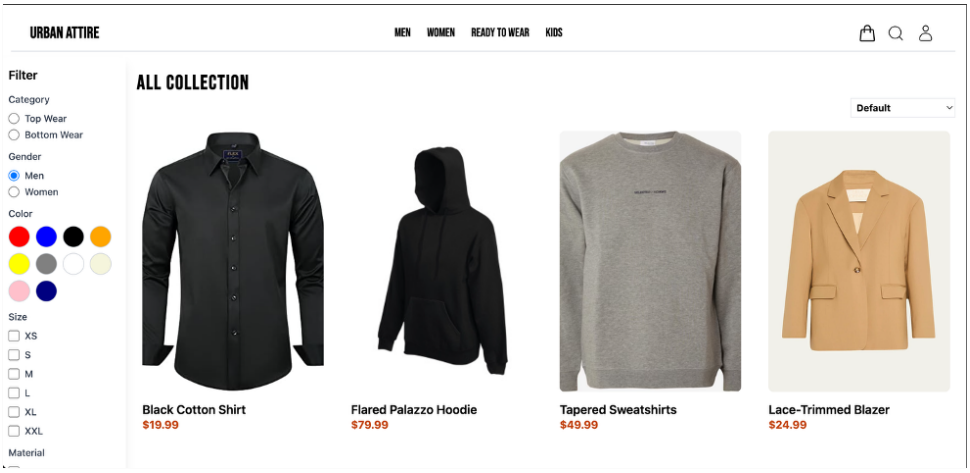

Figure 4.18: Products Page

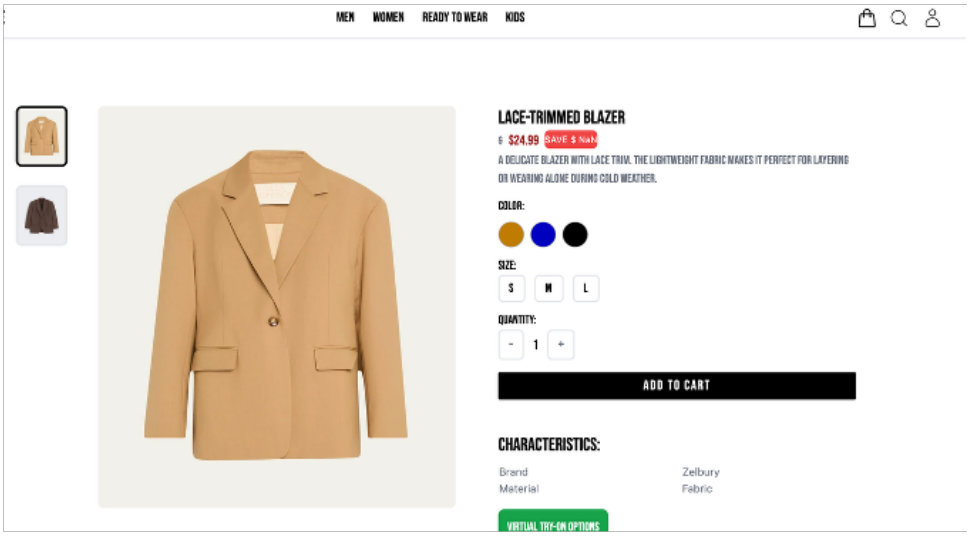

Figure 4.19: Indiviual Product Interface

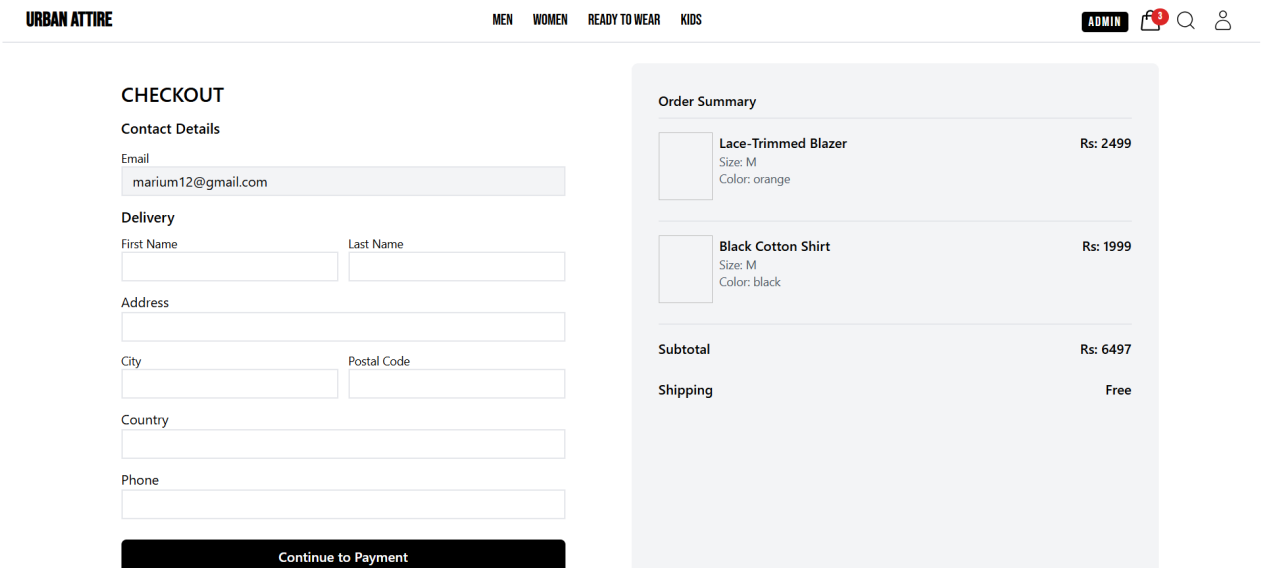

Figure 4.20: CheckOut Page

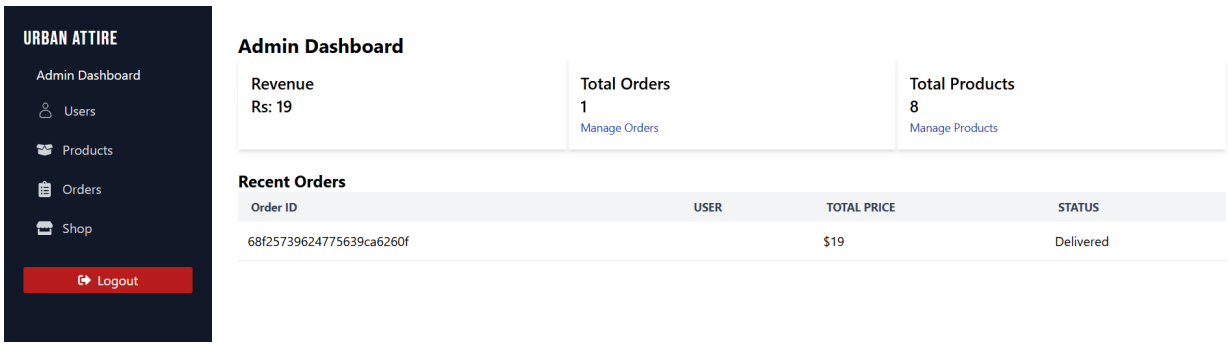

Figure 4.21: Seller Panel Interface

4.9 External Interfaces

Our system is communicating with a number of external systems to offer full e-commerce. functionality. The major external interface is with Stripe Payment Gateway which is. processing all payments with secure API calls . When a customer checkout is initiated, the backend sends payment information to the API endpoint of Stripe, and receives JSON response of payment confirmation or failure. Another critical external rendering engine is Three.js and this is a JavaScript library that processes 3D models of garments and view them in real-time, which is used in the virtual try-on feature. The system is also integrating with Zrok tunneling service to allow safe access into the camera. the system to receive camera feeds of users to track their bodies in real-time without complicated network configurations. All external interfaces are on the protocol of HTTPS to ensure data transmission security

Chapter 5

System Implementation

5.1 Technology Stack

- **Frontend:** React.js to develop user interface elements, develop responsive. customer, seller and administration layouts.
- **Backend:** Node.js with express software to develop REST APIs, process. classification, and administration of business logic.
- **Database:** MongoDB NoSql database for storing both the customer and user data
- **Virtual Try-On:** Three.js library of 3D garment model rendering and creation. try-on experiences, which are interactive virtual
- **Camera Access:** Zrok to provide secure tunneling and camera access, which provides real-time. body tracking: Virtual try-on.
- **Payment Integration:** Stripe payment gateway to do payment safely with. security standards of the industry.
- **Authentication:** JWT (JSON Web Tokens) to make user authentication and session a secure one. management of the platform.
- **Development Tools:** VS Code for code and Postman for API testing

5.2 Architecture and Services

5.2.1 Modules

The system is organized into several independent modules that are handling specific functionalities:

- **Authentication and Users:** User registration, user login, generation of JWT tokens, Hashing of passwords, customer, seller, and administrator role-based access control.
- **Products and Catalog:** creating, editing, deleting, product category. Seller management, seller inventory tracking.
- **Virtual Try-On Engine:** Processing of camera feed or uploaded photos loading 3D models, 3D visualizing of garments with Three.js, and visualizing as try-on.
- **Shopping Cart:** Operations involving the add, update and delete of cart items and computation of total value.
- **Orders:** Processing order creation, order status updates, order history and customer-seller communications.
- **Payments:** Making payments using Stripe gateway, receiving payment confirmation, and changing the order status according to the payments.
- **Admin Operations:** Handiling customer complain , checking on products if there are any inconsistencies and making sure site is working

5.3 Representative Endpoints

The following are key API endpoints that are demonstrating the system's functionality:

5.3.1 Authentication Endpoints

POST /api/auth/register - User Registration

```
{
  "method": "POST",
  "path": "/api/auth/register",
  "body": {
    "name": "Ali Ahmed",
    "email": "ali@example.com",
    "password": "securePassword123",
    "role": "customer"
  },
  "expected": {
    "status": 201,
    "response": {
      "userId": "usr123",
      "token": "jwt_token_here",
```

```
      "message": "User registered successfully"
    }
  }
}
```

POST /api/auth/login - User Login

```
{
  "method": "POST",
  "path": "/api/auth/login",
  "body": {
    "email": "ali@example.com",
    "password": "securePassword123"
  },
  "expected": {
    "status": 200,
    "response": {
      "userId": "usr123",
      "token": "jwt_token_here",
      "role": "customer"
    }
  }
}
```

5.3.2 Product Endpoints

GET /api/products - Get All Products

```
{
  "method": "GET",
  "path": "/api/products?category=shirts&size=M",
  "headers": {
    "Authorization": "Bearer jwt_token"
  },
  "expected": {
    "status": 200,
    "response": {
      "products": [
        {
          "productId": "prd456",
          "name": "Cotton T-Shirt",
```

```
          "price": 1500,
          "category": "shirts",
          "images": ["url1", "url2"],
          "sizes": ["S", "M", "L"],
          "modelUrl": "3d_model_url"
        }
      ]
    }
  }
}
```

POST /api/products - Add New Product (Seller)

```
{
  "method": "POST",
  "path": "/api/products",
  "headers": {
    "Authorization": "Bearer seller_jwt_token"
  },
  "body": {
    "name": "Summer Dress",
    "description": "Beautiful summer dress",
    "price": 3500,
    "category": "dresses",
    "sizes": ["S", "M", "L", "XL"],
    "colors": ["Red", "Blue"],
    "stockQuantity": 50,
    "images": ["image_url_1", "image_url_2"]
  },
  "expected": {
    "status": 201,
    "response": {
      "productId": "prd789",
      "message": "Product added successfully"
    }
  }
}
```

5.3.3 Order Endpoints

POST /api/orders - Create Order

```
{
  "method": "POST",
  "path": "/api/orders",
  "headers": {
    "Authorization": "Bearer jwt_token"
  },
  "body": {
    "userId": "usr123",
    "items": [
      {
        "productId": "prd456",
        "quantity": 2,
        "size": "M",
        "color": "Blue"
      }
    ],
    "shippingAddress": "House 123, Street 5, Islamabad",
    "paymentMethod": "stripe"
  },
  "expected": {
    "status": 201,
    "response": {
      "orderId": "ord789",
      "status": "PENDING",
      "totalAmount": 3000
    }
  }
}
```

5.3.4 Express Handler Example

Express route handler for creating orders:

```
// POST /api/orders
app.post("/api/orders", authenticateToken, async (req, res) => {
  try {
    const { userId, items, shippingAddress, paymentMethod } = req.body;

    // Calculate total amount
    let totalAmount = 0;
```

```
    for (const item of items) {
      const product = await Product.findById(item.productId);
      totalAmount += product.price * item.quantity;
    }

    // Create order
    const order = new Order({
      userId,
      items,
      totalAmount,
      shippingAddress,
      paymentMethod,
      status: 'PENDING'
    });

    await order.save();

    res.status(201).json({
      orderId: order._id,
      status: order.status,
      totalAmount: order.totalAmount
    });
  } catch (error) {
    res.status(500).json({ error: 'Order creation failed' });
  }
});
```

5.4 Database Layer

5.4.1 Core Collections

The MongoDB database is structured as follows and the collections are

- **Users:**Storing customer account data
- **Products:** storing product information, pictures, and the 3D models.
- **Categories:** Product category hierarchy storing
- **Carts:** Saving shopping carts data on users behalf
- **Orders:** Generation of order data and status.

- **Payments:**Storing payment transaction details

5.4.2 Schema Examples

User Schema (Mongoose):

```
const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  role: { type: String, enum: ['customer', 'admin'],
          default: 'customer' }
});

const User = mongoose.model('User', userSchema);
```

Product Schema (Mongoose):

```
const productSchema = new mongoose.Schema({
  sellerId: { type: mongoose.Schema.Types.ObjectId,
              ref: 'Seller', required: true },
  name: { type: String, required: true },
  description: { type: String },
  price: { type: Number, required: true },
  category: { type: String, required: true },
  images: [{ type: String }],
  sizes: [{ type: String }],
  colors: [{ type: String }],
  stockQuantity: { type: Number, default: 0 }
});

const Product = mongoose.model('Product', productSchema);
```

Order Schema (Mongoose):

```
const orderSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId,
            ref: 'User', required: true },
  sellerId: { type: mongoose.Schema.Types.ObjectId,
              ref: 'Seller', required: true },
  paymentId: { type: mongoose.Schema.Types.ObjectId,
               ref: 'Payment' },
```

```
  items: [{
    productId: mongoose.Schema.Types.ObjectId,
    name: String,
    quantity: Number,
    size: String,
    color: String,
    price: Number
  }],
  totalAmount: { type: Number, required: true },
  shippingAddress: { type: String, required: true },
  orderStatus: {
    type: String,
    enum: ['PENDING', 'CONFIRMED', 'SHIPPED',
           'DELIVERED', 'CANCELLED'],
    default: 'PENDING'
  },
  trackingInfo: { type: String }
});

const Order = mongoose.model('Order', orderSchema);
```

5.5 Security

The platform is implementing multiple security measures to protect user data and ensure safe transactions:

- **Password Hashing:** User passwords are hashed using bcrypt with salt rounds before storing in MongoDB, ensuring passwords are never stored in plain text.
- **JWT Authentication:** JSON Web Tokens are used for authenticating users across API requests, with tokens expiring after a set period for security.
- **Input Validation:** All user inputs are validated on the backend before processing to prevent injection attacks and malicious data.
- **HTTPS Encryption:** All data transmission between frontend and backend is encrypted using HTTPS protocol.
- **Stripe Security:** Payment processing follows PCI DSS compliance standards through Stripe integration, ensuring payment data is handled securely.

5.6 Deployment and Observability

5.6.1 Deployment Strategy

The platform can be implemented based on the following methods:

- **Local Development:** to run MongoDB locally, the Express backend server and spawning React development server to test
- **Cloud Deployment:** Backend deployment to such platforms as Heroku, AWS, or Google. Frontend on the cloud using Vercel or Netlify and back office on MongoDB Atlas. database.

5.6.2 Monitoring and Logging

- **Error Logging:** Backend is recording errors and key events to be debugged and monitoring system health.
- **API Response Times:** Checking if api endpoints are giving correct responses and within the correct timeframe .
- **Database Backups:** MongoDB database automated backups to prevent regular. data loss.
- **Health Checks:** Health check endpoint implementation to check whether services are. running properly.

Chapter 6

System Testing and Evaluation

6.1 Graphical User Interface Testing

Scope: React pages and components (home page, product browsing, product details, virtual try-on, shopping cart, checkout, seller panel, admin panel).
Method: Manual testing of user interface components, navigation flows, form validations, and visual elements across different screen sizes.

Table 6.1: Test Module: Home Page

Test ID	Test Description	Expected Result	Status
TC-HOME-01	Verify Navigation Menu functionality	Menu links redirect to correct pages	Pass
TC-HOME-02	Verify Product Categories display	Categories are visible and clickable	Pass
TC-HOME-03	Verify Search Bar functionality	Relevant products appear upon search	Pass

Table 6.2: Test Module: Product Browsing

Test ID	**Test Description**	**Expected Result**	**Status**
TC-BRW-01	Filter products by Category	Only items in selected category display	Pass
TC-BRW-02	Filter products by Size/Color	List updates based on selected attributes	Pass
TC-BRW-03	Filter products by Price Range	Products fall within selected price limits	Pass

Table 6.3: Test Module: Product Details Page

Test ID	**Test Description**	**Expected Result**	**Status**
TC-DTL-01	Verify Image Gallery interaction	Images switch correctly on click/hover	Pass
TC-DTL-02	Select Size and Color variants	User can select available variants	Pass
TC-DTL-03	Test 'Add to Cart' functionality	Item adds to cart with correct details	Pass

Table 6.4: Test Module: Virtual Try-On Feature

Test ID	**Test Description**	**Actual Result**	**Status**
TC-VTO-01	Request Camera Access	Browser prompts for permission	Pass
TC-VTO-02	Render clothing Model	clothing loads on screen	Pass
TC-VTO-03	Garment Overlay Accuracy	Clothing aligns with user body	Pass*
**Note: cloth aligns and moves with body but doesnt blend in with user body*			

Table 6.5: Test Module: Shopping Cart

Test ID	Test Description	Expected Result	Status
TC-CART-01	Add/Remove Items	Item count updates correctly	Pass
TC-CART-02	Update Item Quantities	Total price recalculates immediately	Pass
TC-CART-03	Verify Cart Total Calculation	Subtotal + Tax = Total is correct	Pass

Table 6.6: Test Module: Checkout Process

Test ID	Test Description	Expected Result	Status
TC-CHK-01	Shipping Form Validation	Error shows if required fields empty	Pass
TC-CHK-02	Enter Shipping Address	System accepts valid address format	Pass
TC-CHK-03	Select Payment Method	User can select Stripe/Card option	Pass

Table 6.7: Test Module: Seller Panel (Admin)

Test ID	Test Description	Expected Result	Status
TC-SEL-01	Test 'Add Product' Form	Product saves to DB with images	Pass
TC-SEL-02	Manage Inventory	Stock count updates on edit	Pass
TC-SEL-03	View Order List	Recent orders display in table	Pass

6.2 Usability Testing

Method: Task-based tests with 5 participants (3 customers, 1 seller). Time taken, errors made, and overall satisfaction were recorded.

Tasks for Customers:

- Browse products and apply filters
- Use virtual try-on feature
- Add items to cart and proceed to checkout
- Complete a purchase with minimum payment

Tasks for Seller:

- Add a new product with images and details
- Update inventory for existing products
- View and process customer orders

Table 6.8: Usability Testing Outcomes

Task	Success Rate	Avg Time (s)
Browse products	100%	30
Use virtual try-on	80%	65
Add to cart	100%	20
Complete checkout	90%	80
Add product (seller)	100%	120
Process order (seller)	90%	45

6.3 Software Performance Testing

Table 6.9: Performance Testing Results

Operation	Time (seconds)	Notes
Home page load	1.8	Good
Product list (GET /api/products)	2.0	Good with MongoDB indexing
Product details (GET /api/products/:id)	1.8	Decent response
AR Virtual try-on initialization	3.5	Depends on model size
Virtual try-on (2D) Image processing	3.5	Depends on model size
Add to cart (POST /api/cart)	0.9	Very fast
Checkout (POST /api/orders)	2.8	Includes payment processing
Seller add product (POST /api/products)	3.7	Image upload takes time

6.4 Compatibility Testing

Browsers Tested: Chrome,Safari(on Mac and Ios), Edge (latest versions)

Devices Tested: Desktop (1920x1080), Laptop (1366x768), Mac (2560x1600), Mobile (375x667)

Network Conditions: WiFi, 4G, 3G

Table 6.10: Compatibility Testing Results

Scenario	Configuration	Result
Desktop Chrome	1920x1080, WiFi	Pass - All features working
Mobile Safari	375x667, 4G	Pass - Responsive design works well
Tablet Firefox	768x1024, WiFi	Pass - Touch interactions smooth
Mobile Chrome	375x667, 3G	Partial - Slow model loading and heavy models took too long
Desktop Edge	1366x768, WiFi	Pass - No issues

6.5 Exception Handling

Bugs Injected: Payments failed, cannot access the camera, no valid product data, MongoDB. connection loss,network failure.

Behavior Requirements: Our system should clearly display error messages to users, error logging, system. remains stable, no data loss.

- **Payment Failure:** The status of the order is PENDING; a user receives an error notification like: Sorry, the payment was not successful, re-trying one more time is possible; re-trying order. *Result: Pass*
- **Camera Access Denied:** System displays message of camera access needed to real. time try-on" and recommends 2D photo upload instead.*Result: Pass*
- **Invalid Product Data:** Backend Checks Data Error and shows product does not exist error messages. *Result: Pass*
- **Confirming MongoDB Connection Loss:** AAPI response is an error; error has been logged by the system; frontend displays temporary unavailability of service.*Result: Pass*
- **JWT Token Expired:** The time-out (expired) of JWT token redirects the user to the login page with the message of Session.. *Result: Pass*
- **Network Timeout:** Frontend shows loading state and retry option; user can retry the action. *Result: Pass*

6.6 Load Testing

We lacked enough resources to perform load testing by multiple users and could do only load testing with 7 users at a time

Scenario: 7 simultaneous users browsing, adding to cart and checking out products at the

same time.

- System was able to withstand 7 simultaneous users.
- No order or cart item discrepancies in data
- MongoDB worked well with optimal indexing
- Virtual try-on was also running without any problem in all users concurrently.

Limitations: It was not conducted with full-scale load testing with a large number of users at the same time due to resource constraints.

6.7 Security Testing

Areas Under Testing: Authentication, authorization, input validation, password security, payment,security and data protection.

- **Authentication/Authorization:**Checking of the integrity of the JWT tokens; the expired tokens are rejected correctly; role-based access control is functioning correctly. *Result: Pass*
- **Input Validation:** everything on the backend had to be validated; bad scripts, malicious scripts were blocked. *Result: Pass*
- **Payment Security:** Stripe deals with payment information; HTTPS encryption is on. *Result: Pass*

Table 6.11: Installation Testing Results

Check	**Result**
Install on fresh machine	Pass
Environment variable configuration	Pass
Database connection	Pass
Backend server startup	Pass
Frontend server startup	Pass
All features functional after installation	Pass

6.8 Evaluation Metrics and Results

6.8.1 Quantitative Results

- **Average Page Load Time:** 2.5 seconds
- **API Response Time:** 1.5–3.5seconds
- **Virtual Try-On Rendering:** 3–5 seconds
- **Error Rate:** Less than 2% (very rare)

6.8.2 Qualitative Results

The metrics were taken from the 7 users we asked for testing

- **User Satisfaction:** 4.4/5 stars
- **Ease of Use:** Simple and Elegant interface
- **Virtual Try-On Feature:** Appreciated but accuracy could improve
- **Seller Panel:** Easy to use

Chapter 7

Conclusions

In conclusion, “When AR Meets Style” shows that using modern web technologies with virtual try-on capabilities can create a more engaging e-commerce experience for fashion retail in Pakistan. Current AR has several limitations, and it is designed for mobile devices; hence, implementing it on a website limits it further. By offering both 3D and 2D visualization options, the platform addresses usability challenges while providing value to customers . The use of the MERN stack and a modular architecture has proven effective for building a scalable, maintainable platform. The following points summarize the key technical takeaways and lessons learned during the development and testing of the system, highlighting areas of success and opportunities for further improvement.

7.1 Key Technical Takeaways

- **Virtual Try-On Limitations:** Although the virtual try-on is an attractive feature and gives customers a concept of how products would look on them, However AR technology is not so developed that it can be considered to provide the experience of real try-on. The AR state nowadays cannot be used to simulate the touch of clothes. like texture of fabrics, comfort, or measurements of the precise fit. Users should understand that virtual try-on is an innovative preview tool instead of an ideal display of real-world fitting.

- **2D AR vs 3D AR:** 2D photo-based virtual try-on proved to be more practical and reliable compared to 3D real-time AR. The 3D approach is requiring good lighting conditions, proper camera positioning, and higher device performance, which is making it difficult for many users. On the other hand, 2D AR is simpler to use, The use of AI in body detection and garment overlay for 2D try-on is more mature and accurate than real-time 3D body tracking, making it a better choice for most users.

- **MERN Stack Benefits:** The MERN stack (MongoDB, Express.js, React.js, Node.js) is assisting in the development of an effective and convenient web site. Applying JavaScript in both frontend and backend are simplifying and increasing the speed of development since we have to use the same language across the project. React is providing a vibrant and dynamic user interface which provides fast reload. Express and Node.js are processing API requests with ease, and MongoDB is storing data in a flexible manner. Such combination is rendering it a fast loading and user friendly platform

- **Component Modularity:** Designing the project in the modular manner was found most useful in the way of maintenance and future development. Separating authentication, product management, order processing, virtual try-on and payment. The modules is ensuring it becomes easy to update or repair individual parts and not entire system. In case something is wrong with any module, it can be fixed quickly

- **Web-Based 3D AR Challenges:** Implementing full 3D AR with real-time body tracking on web browsers is proving to be very challenging and often produces poor results. Web browsers have limited access and don't support full-body 3D tracking or access device-level AR frameworks like ARKit/ARCore. They also can't use depth sensors, which are required for accurate body mesh, occlusion, and alignment. which is restricting the accuracy of body detection and 3D rendering capabilities. These technical limitations are making web-based 3D AR unreliable and inconsistent on a web page That is why we had to implement 2D virtual try on beside it

7.2 Future Work

7.2.1 Technical Enhancements

- **Improved 2D Virtual Try-On:** Focusing development efforts on enhancing the 2D photo-based try-on feature with better AI models for body detection and garment overlay. Since 2D is proving more reliable, investing in its improvement will benefit more users than advancing 3D technology.

- **AI-Powered Size Recommendations:** Developing an intelligent system that is analyzing user body measurements from uploaded photos and recommending the best size, reducing returns and improving customer satisfaction.

- **Performance Optimization:** Further optimizing image loading, 3D model rendering, and database queries to ensure the platform works smoothly even on slower internet connections common in Pakistan.

- **Mobile Application:** Developing native mobile apps for better camera integration and offline browsing capabilities, as most Pakistani users are accessing e-commerce through mobile devices.

7.2.2 Feature Additions

- **Multiple Payment Methods:** Adding cash on delivery, JazzCash, Easypaisa, and bank transfer options to cater to Pakistani payment preferences.

- **Multiple Sellers:** Adding multiple sellers and giving local brands and boutiques a chance to sell on our website and help benifit and increace their business

- **Order Tracking:** Adding detailed order tracking features so customers can see real-time updates on their shipment status.

References

[1] Tingting Zhang, William Yu Chung Wang, Ling Cao, and Yan Wang. The role of virtual try-on technology in online purchase decision from consumers aspect. *Internet Research*, 29(3):529–550, 2019. Cited on pp. i and 7.

[2] BetterMirror. Benefits of virtual try-on for online shopping. *BetterMirror Insights*, 2023. Cited on p. 1.

[3] X. Han, J. Wang, and Y. Zhang. Comparative analysis of 2d and 3d virtual try-on systems in fashion e-commerce. *Journal of Fashion Technology*, 15(2):45–62, 2021. Cited on pp. 2 and 5.

[4] Vinamra Singh. Augmented reality on mobile browsers with webgl. *International Journal of Future Computer and Communication*, 13(4):67–75, 2024. Cited on p. 4.

[5] Q. Li and A. Martinez. User preference in virtual try-on: 2d vs 3d interfaces. *International Journal of Human-Computer Studies*, 158:102732, 2022. Cited on p. 5.

[6] Danny Parisi. Virtual try-on is the antidote to the pandemic-fueled rise in returns. Apr 2021. Cited on p. 6.

[7] Dharmendra Patel and Kaushal Shah. Mobile app or responsive website: Best option for e-commerce. *International Journal of Innovative Research in Management, Pharmacy and Sciences (IJIRMPS)*, 6(5):1–6, 2018. Cited on pp. 8 and 9.

[8] Nabil El Barhoumi, Rafika Hajji, Zakaria Bouali, Youssef Ben Brahim, and Abderrazzaq Kharroubi. Assessment of 3d models placement methods in augmented reality. *Applied Sciences*, 12(20):10620, 2022. Cited on p. 8.