Palestine Technical University - Kadoorie

Faculty of Engineering and Technology

Computer Systems Engineering Department

AR GLASSES STORE

Prepared by:

Mahmoud Qasem

Omar Abbadi

Supervised by:

Nael Salman, Ph.D.

A graduation project submitted in partial fulfillment of the requirements for the bachelor's degree in computer systems engineering.

Tulkarm, Palestine

January, 2023

ACKNOWLEDGEMENT

We begin by expressing our gratitude to Allah for His blessings and guidance throughout this journey. We would like to extend our heartfelt appreciation to our parents and families for their unwavering support and encouragement that enabled us to successfully complete this work. We are also deeply grateful to our university Palestine Technical University - Kadoori and specifically the Faculty of Engineering and Technology for providing us with the opportunity to pursue our studies and engage in this project.

Our utmost appreciation goes to our supervisor, Dr. Nael Salman, for his invaluable guidance, continuous support, and dedication. His expertise and insightful feedback have greatly contributed to the success of this project. We would also like to extend our sincere thanks to all the faculty members of the Department of Computer Systems Engineering for their valuable teachings and assistance, which have been instrumental in shaping our knowledge and skills.

Lastly, we would like to acknowledge the efforts of everyone involved in this project, whose contributions have been instrumental in its completion. We are grateful for their collaboration and support throughout this endeavor.

ABSTRACT

This thesis presents the development of a glasses store application with a virtual try-on feature, aiming to revolutionize the eyewear shopping experience.

The application utilizes *Flutter* for the mobile app, *Spring Boot* with PostgreSQL for the REST API, and *Unity 3D* with AR Foundation for the virtual try-on functionality.

Through a user-friendly interface, customers can browse and virtually try on glasses in real-time, providing a more accurate and engaging shopping experience. The application focuses on usability, accuracy, performance, and security, and envisions future expansion to a global market and exploration of alternative low-level AR services. This project introduces an innovative solution that enhances the way customers shop for glasses online, utilizing cutting-edge technologies for a seamless and immersive experience.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In this chapter, we will discuss the outlines of our project.

1.1 Background

The e-commerce industry has undergone a remarkable transformation, revolutionizing the way people shop by providing convenience and accessibility. Eyeglasses, being both functional and aesthetic, have traditionally required physical store visits for trying on frames. However, this traditional approach poses limitations such as geographical constraints and a limited range of frame selections.

This project aims to develop an augmented reality (AR) based e-commerce platform for eyeglasses, addressing these limitations. Augmented reality technology overlays virtual objects onto the real world, allowing users to virtually try on glasses and make well-informed purchase decisions. The "Eyeglasses Store" platform will cater to guests, customers, and store account holders.

The platform supports multiple stores, enabling eyeglass retailers from various locations to participate. Each store account holder has the capability to manage their inventory through CRUD operations, ensuring flexibility and control over their product offerings. By integrating augmented reality, the platform enhances the shopping experience by enabling users to visualize glasses on their own faces, thereby overcoming the limitation of physical try-on.

Users will have access to a wide range of eyeglasses, along with detailed descriptions and the ability to add items to their carts. Leveraging the device's camera, users can virtually try on different glasses, providing an interactive and immersive shopping experience. The project's main objectives include the technical implementation of the e-commerce platform, ensuring usability, enhancing user experience, and evaluating the impact of augmented reality on customer satisfaction.

The "Eyeglasses Store" project aspires to revolutionize the eyeglass shopping landscape, offering customers a novel and enhanced experience through augmented reality-based e-commerce. By providing a seamless and convenient platform, it aims to overcome geographical constraints, expand frame selections, and empower customers to make confident and informed purchase decisions from multiple eyeglass stores.

1.2 Problem Statement

The problem that this e-commerce project aims to solve is the lack of a convenient and an efficient way for customers to try on glasses before making a purchase. Customers may struggle with finding the right fit, style, or color without trying it on, which can lead to dissatisfaction and an increase in returns. Additionally, many customers may not have access to physical stores or may prefer to shop online. This can make it difficult for them to try on glasses before making a purchase, leading to a suboptimal buying experience. By developing an online platform for selling glasses with a virtual try-on feature, this project aims to provide customers with a more accurate representation of how glasses will look on their faces, improve customer satisfaction and make the purchasing process more convenient and efficient.

In general, in this century, technology plays an important role in our daily activities of life. people are looking for what can save their time and effort, so, using online services that are available for everybody at all times the want will achieve the goal of technology.

A lot of people nowadays need eyeglasses not just those who suffer from visual deficiencies, but there are glasses for protection from sunlight or computer lights, so facilitating the process of choosing and buying glasses efficiently via the internet will make life better for customers and shop owners.

1.3 Objectives

1. To develop an online platform for selling eyeglasses that includes a virtual try-on feature using augmented reality (AR).
2. To allow customers to try on a variety of frames in a convenient and efficient manner, without the need to physically visit a store.

3. To reduce the number of returns due to poor fit by allowing customers to try on glasses virtually before making a purchase.

4. To improve customer satisfaction by providing a more convenient and efficient buying experience.

5. To increase sales and revenue for the e-commerce company by providing a more user-friendly buying experience.

1.4 Definition of Terms

Terms	**Definition**
UML	Unified Modelling Language Software Development Life Cycle.
ER-Diagram	A diagram of the Relationship between Entities in the database.
API	Application Programming Interface, It allows different systems to access the functionality of the backend application without having to know the internal structure and logic of the application.

Table 1:1 Definition of Terms

1.5 Procedures

To achieve our project goals, we focused on developing a mobile app-based system that allows users to interact with our glasses store and access information conveniently. To ensure we captured the requirements of our customers and stakeholders accurately, we conducted interviews to gather detailed information about the desired functionalities and capabilities of the system.

After gathering the requirements, we analyzed and organized the information using UML (Unified Modeling Language) diagrams. These diagrams represented the structure and behavior of each functionality in the system, providing a blueprint for how they would be implemented. Additionally, we utilized an ER (Entity-Relationship) diagram to visualize the relationships between entities in the database, ensuring data integrity and efficiency.

Once the requirements and design phase were completed, our team transitioned into the development phase. For this, we adopted the agile approach to the Software Development Life Cycle (SDLC). The agile model is widely recognized in the software development industry, as it promotes flexibility and iterative development strategies. Our workflow progressed through cycles of planning, designing, coding, and testing specific sets of features or functionality. This allowed us to deliver software quickly and efficiently while remaining adaptable to changing requirements and customer needs.

By adopting the agile methodology, we were able to ensure regular communication and collaboration within our development team. This facilitated efficient decision-making, effective problem-solving, and continuous feedback loops, ultimately resulting in a high-quality product that met the expectations of our users.

Throughout the development process, we prioritized regular testing to verify the functionality, performance, and usability of the mobile app. This iterative approach allowed us to identify and resolve issues early, ensuring a smooth and reliable user experience.

Overall, the combination of requirements gathering, UML and ER diagram representation, and agile development practices allowed us to successfully develop our glasses store mobile app with a virtual try-on feature, meeting the needs of our customers and stakeholders while maintaining a high level of quality and adaptability.

1.5 Organization of the Study

This thesis is divided into five chapters each one describing a part of our project.

The chapters are as the following:

- Chapter 1: Introduction: This chapter outlines the idea of the project, the main objectives of the project, and the outline of the project.
- Chapter 2: Literature review: This chapter discusses similar previous projects.
- Chapter 3: System Requirements: This chapter discusses the System Requirements (functional and non-functional).
- Chapter 4: Methodology And Technologies: This chapter discusses the methodology we used during the construction of this project, as well as the methods of collecting information and representing it through ER and UML diagrams.
- Chapter 5: This chapter reviews the conclusion and Future Vision for this project.
- Chapter 6: References.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we will discuss similar previous systems

2.1 Similar Work and Differences

Eyeconic [1.7] and Zenni Optical [2.2] are two websites that offer a virtual try-on experience for glasses. However, there are distinct differences between these websites and our glasses store mobile app, highlighting the unique aspects of our project.

Firstly, both Eyeconic and Zenni Optical employ a video recording method to allow users to visualize glasses on their faces. Users record a video, and then the websites augment the selected glasses onto the recorded video. In contrast, our mobile app offers a real-time try-on experience, where users can view glasses on their faces instantaneously without the need for pre-recorded videos. This real-time feature enhances user convenience and provides a more interactive and seamless try-on experience.

Secondly, Eyeconic and Zenni Optical are specific online stores that offer their own products and services. In contrast, our glasses store mobile app aims to serve as a public platform where multiple sellers can showcase and sell their glasses, similar to popular e-commerce platforms like Amazon and AliExpress. This key distinction positions our app as a versatile marketplace that offers a wide range of glasses from various sellers, providing users with a greater selection and potentially better pricing options.

CHAPTER 3

SYSTEM REQUIREMENTS

This chapter discusses the System Requirements (functional and non-functional).

3.1 Functional Requirements

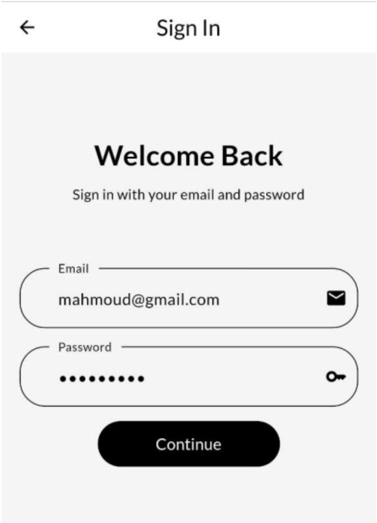
It’s important to consider the needs and goals of both the business and the users when defining functional requirements for any software application such as:

3.1.1 Sign in

the users can sign in the system by enter the password and email, this functionality enables users to access their accounts and use the remaining functional requirement. this functionality guarantees the security and privacy to user account, if the user not pass the requirement of this functionality the user cannot interactive with any functionality else.

Step 1: the user enters his/her data in the field of the form.

Step 2: the user clicks continue.



Sign In

Welcome Back

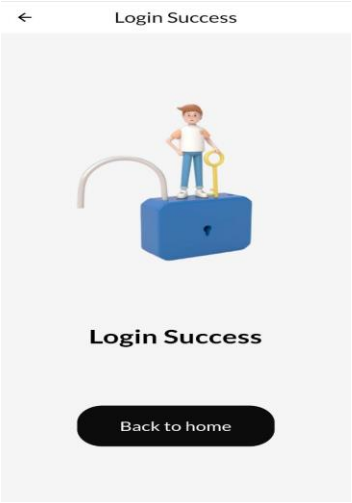
Sign in with your email and password

Email: mahmoud@gmail.com

Password: •••••••••

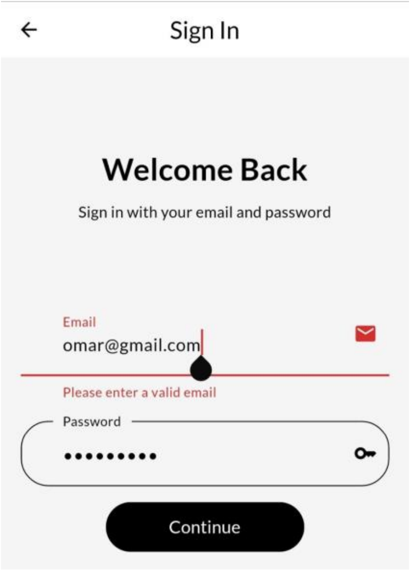
Step 3: the system checks if the user is on the database or not.

Step 4: if found pass and system move user to successful screen,



Login Success

otherwise the system refuses the request and send to the user invalid data message.



Welcome Back

Sign in with your email and password

Email
omar@gmail.com

Please enter a valid email

Password
•••••••••

3.1.2 Integrations with 3rd Parties

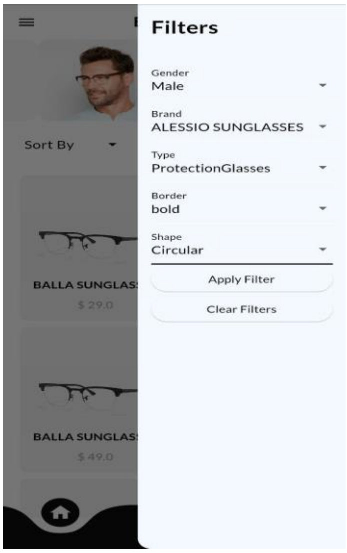
Integration with 3D Glasses Models: The system should support the integration of 3D glasses models into the Unity AR environment, allowing users to visualize and interact with eyeglasses.

3.1.3 Product Specification

It defines the specific features and capabilities that a product or system should have in order to meet the needs of its users and the business. A product specification typically includes a detailed description of the functionality that the product is expected to provide, along with any constraints or requirements that must be met in order to achieve that functionality.

For e-commerce for selling glasses, product specification might include functional requirements such as:

- An ability for users to browse and filter glasses by various criteria, such as type, brand, price, border, gender, or color.



Filters

Gender: Male

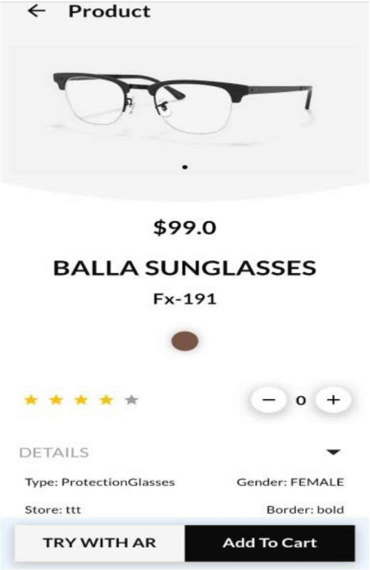
Brand: ALESSIO SUNGLASSES

Type: ProtectionGlasses

Border: bold

Shape: Circular

- An ability for users to view detailed product information for each pair of glasses, including photos, descriptions, and technical specifications.



Product

$99.0

BALLA SUNGLASSES

Fx-191

DETAILS

Type: ProtectionGlasses

Gender: FEMALE

Store: ttt

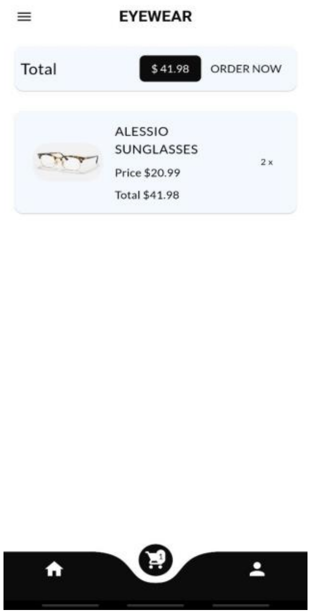
Border: bold

Product specification can also include non-functional requirements, which are requirements that relate to the overall quality or performance of the product, rather than specific features or functionality.

3.1.4 The Flow of Orders & Checkout

The flow of order and checkout is typically considered a functional requirement for an e-commerce application, as it defines the specific steps and processes that a user must follow in order to complete a purchase. In the case of e-commerce for selling glasses, functional requirements related to the flow of orders and checkout might include:

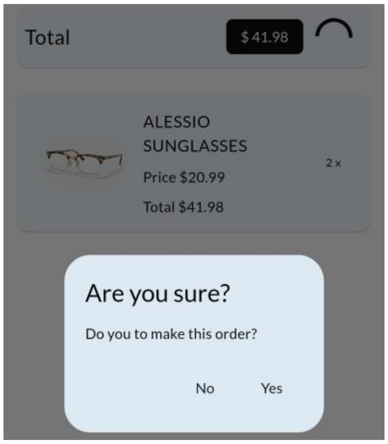
1. An ability for users to browse, and to view detailed product information for each pair of glasses.
2. An ability for users to add glasses to their shopping cart and proceed to checkout.



Total $ 41.98

ALESSIO SUNGLASSES
Price $20.99
Total $41.98
2 x

3. An ability for the application to process the order, and to confirm the order details and total cost to the user.



Total $ 41.98

ALESSIO SUNGLASSES

Price $20.99

Total $41.98

2 x

Are you sure?

Do you to make this order?

No Yes

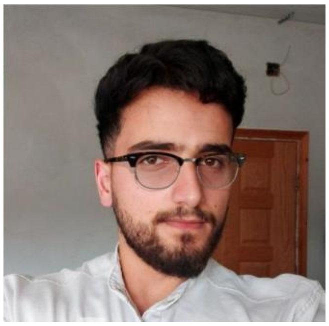
4. An ability for the user to confirm their order before submitting it.

These are just a few examples of functional requirements that might be relevant for the flow of orders and checkout in the project.

3.1.5 Product Visualization and customization

Functional requirements for a virtual try-on feature might include:

- An ability for the application to detect the user's face and align the glasses with the appropriate position on the user's face.



3.2 Non-Functional Requirements

3.2.1 Usability

Usability plays a significant role in the success and adoption of any software application, and our app is no exception. We have strongly emphasized ensuring a user-friendly and intuitive interface that enhances the overall user experience. The following key points highlight the usability goals and considerations incorporated into the design and development of our app:

- Intuitive Navigation: We aim to provide users with a clear and easy-to-understand navigation structure throughout the app. By organizing the content and features in a logical manner, users can effortlessly browse through different sections, such as product categories, search functionality, shopping cart, and account management.
- Minimal Learning Curve: To reduce the learning curve for new users, we have strived to create a user interface that is familiar and consistent with common mobile app design patterns. By employing recognized design principles, such as using standard icons and labels, providing visual cues, and maintaining consistency across different screens, we aim to facilitate a smooth onboarding experience.
- Clear and Concise Product Presentation: Presenting the glasses in a visually appealing and informative manner is crucial for engaging users. We have

focused on providing high-quality product images, accurate descriptions, and relevant details (e.g., material, dimensions, pricing) to help users make informed purchasing decisions.

- Efficient Search and Filtering: We recognize the importance of an efficient search and filtering system to help users find the glasses they are looking for quickly. Our app incorporates robust search functionality that allows users to search by various criteria, such as brand, style, frame shape, and price range. Additionally, we provide advanced filtering options to further refine search results based on specific preferences.
- Responsive and Adaptive Design: Our app is designed to be responsive and adapt seamlessly to different screen sizes and orientations, ensuring a consistent user experience across various mobile devices. This responsiveness enhances usability by optimizing the app's layout and content presentation for different screen resolutions.
- Feedback and Error Handling: We have implemented precise feedback mechanisms throughout the app to inform users about their interactions and the status of their actions. Furthermore, we have developed effective error-handling strategies that provide informative error messages and guidance in case of any issues, allowing users to recover from errors easily and continue their interactions smoothly.

By considering these usability aspects, we aim to create an engaging and user-friendly glasses store mobile app that not only provides a seamless virtual try-on experience but also ensures a delightful overall user experience.

3.2.2 Accuracy

Since our system provides glasses with virtual try-on features, such features should be accurate to match the real-life glasses experience, therefore, customers will have a better overall experience. We can say that we achieved this requirement if the result of the try-on feature is accurate at a high rate, meaning that the result has correct attributes such as normal fitting of the glasses on the human face, natural-looking, and closer to reality.

3.2.3 Performance

Performance is a critical aspect of our app, considering the ever-increasing demand for fast and responsive internet and mobile applications. Maximizing performance is essential for enhancing usability and ensuring a seamless user experience.

Our primary objective is to ensure that our system operates at an optimal level of performance, maintaining a stable and responsive state at all times. To achieve this goal, we have focused on the following aspects:

1. Responsiveness: Our app aims to deliver near-instantaneous responses to user interactions. Actions such as browsing products, searching, adding items to

the cart, and completing purchases should be executed swiftly, minimizing any noticeable delays. By prioritizing responsiveness, we enhance user satisfaction and engagement with our app.

2. Comparative Performance: To evaluate our app's performance, we have benchmarked it against current industry standards and common apps and web services. Our aim is to ensure that our system is on par with or outperforms similar apps in terms of speed and responsiveness. By meeting or surpassing these benchmarks, we provide users with a high-quality experience and avoid being perceived as slow compared to other systems in the current technological landscape.
3. Scalability: Our app has been designed with scalability in mind to handle increased user demand and growing data volumes. By employing scalable architecture and infrastructure, we can maintain optimal performance even during periods of high traffic or expanding user base. This ensures that our app remains responsive and stable as the user base grows.

3.2.4 Security

Security is a paramount consideration in our glasses store mobile app, as it involves handling sensitive user data and conducting monetary transactions. We have implemented various tools and techniques to ensure the security and confidentiality of our users' information.

One of the key security measures we have adopted is the use of JSON Web Tokens (JWT). JWT allows us to securely transmit and verify the identity of users during their interactions with the server. Each HTTP request coming to the server is assigned a unique JWT, which represents the user's identity and authentication status. Based on this token, the server can authenticate and authorize the user, allowing them to access their respective functionalities and perform authorized actions. This helps prevent unauthorized access to sensitive information and ensures that only authenticated users can carry out specific operations within the app.

Furthermore, we have established a role-based access control system to enforce granular permissions and authorities for different user roles. The three main roles in our app are the CUSTOMER, STORE, and ADMIN. Each role has a set of predefined authorities that determine the actions and operations they are permitted to perform within the system. This ensures that users can only access and manipulate data and functionality that are relevant to their role and responsibilities. By implementing role-based access control, we limit the risk of unauthorized access or malicious actions by users.

CHAPTER 4

METHODOLOGY AND TECHNOLOGIES

This chapter discusses the methodology we used during the construction of this project, as well as the methods of collecting information and representing it through Software Analysis & Design.

4.1 Data Collection & Analysis

Data collection methods depend on several methods, including interviews with the supervisor, clients, and stakeholders, this stage contained questions that helped us identify problems and difficulties facing clients. The previous requirement is collecting making interviews with the Customers and Stakeholders. Methodology means the method that will be used to build this system. In addition, the methodology is the most important part of the system development. In our project, we use System Development Life Cycle (SDLC). SDLC is a series of phases in the development process. It provides a model for the development and lifecycle of an application. The SDLC process will help to produce an effective, cost-efficient, and high-quality system.

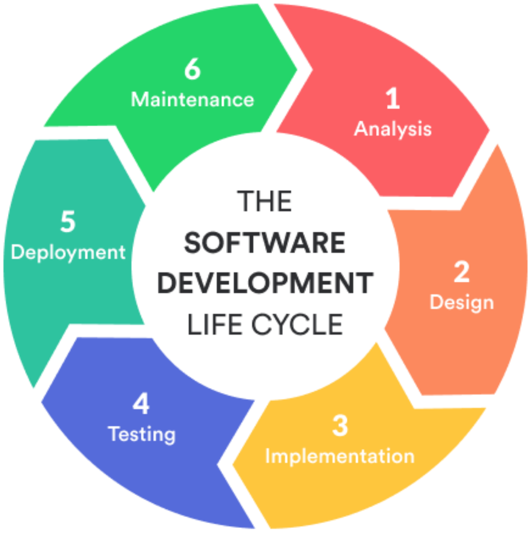


Figure 4:1 System Development Life Cycle

One of several types of SDLC is the Agile model and that is what we depended on when we started work on this project Because requirements may vary based on the demands of customers and stakeholders, using the agile technique will be easier because it is flexible and iteratively developing. The agile method can help in plan and scheduling the system development. The development process moves from planning, development, testing, and deployment then the feedback from customers and stakeholders is collected and fit back into the cycle, this help the development team to keep up to date with the customer needs and the constantly evolving world.

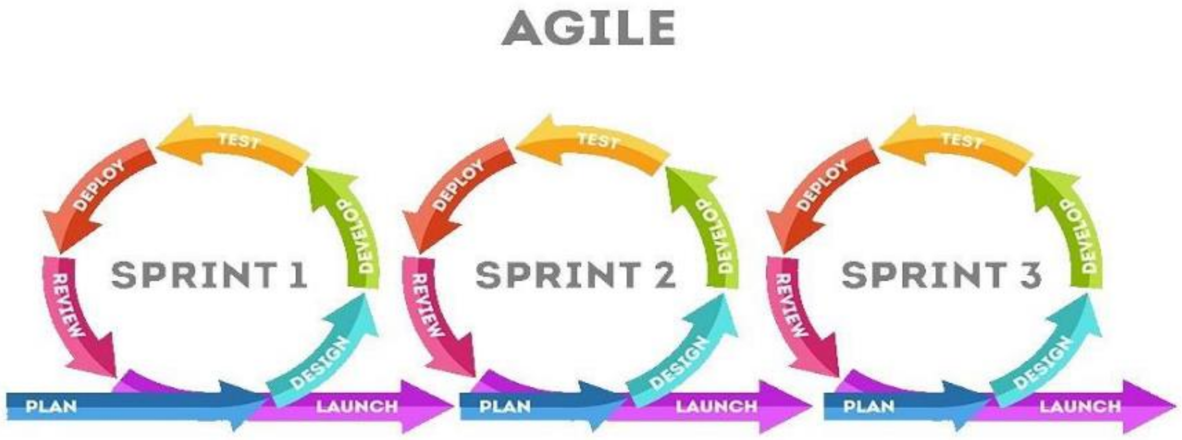


Figure 4:2 Agile Model

4.2 Technologies

4.2.1 Front-End Technologies

Flutter:

Flutter is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase.[1.1]

The reasons why we choose React are below:

- **Fast:** Flutter code compiles to ARM or Intel machine code as well as JavaScript, for fast performance on any device.
- **Productive**: Build and iterate quickly with Hot Reload. Update code and see changes almost instantly, without losing state.
- **Flexible**: Control every pixel to create customized, adaptive designs that look and feel great on any screen.
- **Stable & Reliable**: Flutter is supported and used by Google, trusted by well-known brands around the world, and maintained by a community of global developers.

4.2.2 Back-End Technologies

Spring Boot

Spring Boot is a powerful and widely used framework that we have employed to develop the REST API component of our app. As a framework built on top of the popular Spring framework, Spring Boot provides a streamlined and efficient approach to building robust and scalable web applications.[1.2]

One of the key advantages of using Spring Boot is its ability to simplify the development process. It eliminates the need for extensive configuration by providing default settings and conventions that promote rapid application development. With Spring Boot, we can quickly set up a fully functional RESTful API with minimal boilerplate code, allowing us to focus more on the actual implementation of business logic and functionalities.

Spring Boot offers a comprehensive set of features and libraries that facilitate the development of REST APIs. It provides built-in support for key functionalities such as request mapping, request/response serialization, exception handling, authentication, and authorization mechanisms. These features enable us to implement the required API endpoints, handle incoming requests, and respond with the appropriate data or error messages efficiently.

Furthermore, Spring Boot integrates seamlessly with other Spring projects and frameworks, such as Spring Data JPA for database access and management. This integration simplifies the implementation of data persistence and allows us to leverage the powerful features of Spring Data JPA, such as object-relational mapping (ORM) and database query generation.

Additionally, Spring Boot offers extensive documentation, a vibrant community, and active development, which ensures continuous support and updates. This allows us to stay up-to-date with the latest best practices and security patches, thereby enhancing the overall stability and reliability of our REST API.

By utilizing Spring Boot for our REST API development, we have leveraged its simplicity, productivity, and powerful features to build a robust and efficient backend foundation for our glasses store mobile app. The framework's comprehensive toolset, seamless integration capabilities, and active community support have contributed to the successful implementation of our REST API component.

PostgreSQL

PostgreSQL is a highly compatible and reliable open-source relational database management system. It seamlessly integrates with Spring Boot, making it an ideal choice for storing and managing data in our glasses store mobile app. With its strong performance, robust security features, and active community support, PostgreSQL enhances the functionality and scalability of our Spring Boot-based REST API.[1.3]

Unity 3D

Unity 3D is a powerful and widely used game development engine that we have leveraged to create the virtual try-on feature in our glasses store mobile app. With Unity 3D, we can create immersive and interactive experiences by combining 3D graphics, animations, physics, and more.[1.4]

To enable the virtual try-on functionality, we utilized Unity's AR Foundation, which is a cross-platform framework for building augmented reality (AR) applications. AR Foundation simplifies the development process by providing a unified API that supports multiple AR platforms, such as ARKit for iOS and ARCore for Android. This allows our app to offer virtual try-on capabilities on a wide range of devices.

By leveraging AR Foundation, we can track the user's facial features in real-time and overlay virtual glasses onto their face, creating a realistic and interactive try-on experience. Unity's powerful rendering capabilities ensure that the virtual glasses seamlessly blend with the user's surroundings, providing a convincing and immersive visual representation.

AR Foundation integrates seamlessly with Unity's existing workflow and leverages its robust features, such as the visual editor, asset pipeline, and scripting capabilities. Here's an overview of how Unity works with AR Foundation:

1. Scene Setup: Developers begin by setting up their Unity project and scene. They can import 3D models, textures, and other assets into the project, and arrange them within the scene to create the desired environment for the AR experience.
2. AR Foundation Package: Developers need to import the AR Foundation package from the Unity Package Manager. This package provides the necessary scripts, prefabs, and components for AR development.
3. AR Session: An AR session is created in the scene, representing the AR environment. It manages the tracking of the user's physical space, such as detecting planes and anchors, and provides the necessary data for AR interactions.
4. AR Session Origin: The AR Session Origin acts as the reference point for the AR scene. It provides the coordinate system and transformation information required to position virtual objects in the real world accurately.
5. AR Trackables: AR Trackables are objects in the physical world that can be tracked by AR technology, such as detected planes, images, or 3D objects.

Developers can set up event handlers to respond to changes in trackable states, such as when a plane is detected or lost.

6. AR Anchors: Anchors allow developers to attach virtual objects to specific locations in the real world. For example, they can place a virtual object on a detected plane or align it with a physical object using anchors.
7. AR Raycasting: Unity provides raycasting capabilities for AR, allowing developers to cast rays into the AR scene and interact with virtual objects. This enables features like tap or gesture-based interactions with the AR elements.
8. Scripting and Interactions: Developers can write scripts in C# to define the behavior and interactions of virtual objects in the AR scene. These scripts can handle input events, perform calculations, update object positions, trigger animations, and interact with other game components.
9. Building and Deploying: Once the AR experience is developed, developers can build and deploy the app for specific platforms, such as iOS or Android. Unity's build settings and platform-specific settings ensure that the app is optimized and compatible with the target devices.

AR Foundation simplifies the process of developing AR applications by providing a unified API that abstracts the complexities of different AR platforms. It allows developers to create immersive and interactive AR experiences using Unity's powerful features, scripting capabilities, and visual editor while ensuring compatibility and ease of deployment across multiple platforms.

Railway

Railway.app is a cloud hosting platform that we have utilized to deploy our REST API, allowing seamless integration with our Flutter front-end. Railway simplifies the deployment process by providing a user-friendly interface and automation tools, enabling us to quickly set up and manage our API's hosting environment.[2.1]

Here's an overview of how Railway works with our REST API deployment:

1. Easy Setup: Railway offers a straightforward setup process, where we can connect our code repository (e.g., GitHub) and select the appropriate deployment settings. With Railway's integration, our code changes can be automatically deployed to the hosting environment.
2. Seamless Integration: Railway seamlessly integrates with our Spring Boot application, providing the necessary infrastructure to run our REST API. It automatically configures the environment, including setting up the required dependencies, environment variables, and runtime configurations.
3. Scalability and Performance: Railway's hosting infrastructure ensures scalability and performance for our REST API. It leverages cloud-based

technologies to dynamically allocate resources based on demand, allowing our API to handle a large number of concurrent requests efficiently.

4. Continuous Deployment: Railway supports continuous deployment, allowing us to streamline the deployment process and ensure that our REST API is always up to date with the latest code changes. This enables us to iterate and improve our API quickly without manual intervention in the deployment process.
5. Monitoring and Analytics: Railway provides monitoring and analytics features that allow us to track the performance and usage of our deployed REST API. We can gain insights into response times, traffic patterns, and other key metrics to optimize our API's performance and user experience.
6. Security and Reliability: Railway prioritizes security and reliability, ensuring that our deployed REST API is protected and available to users. It implements security measures such as HTTPS encryption and provides monitoring and alerts to address potential issues proactively.

By leveraging Railway to deploy our REST API, we have streamlined the deployment process and ensured a reliable hosting environment for our Flutter front-end to consume. Railway's simplicity, scalability, continuous deployment capabilities, and monitoring features have contributed to the seamless integration of our front-end and back-end components, enabling a smooth and efficient user experience.

4.3 Software Analysis & Design

4.3.1 Use Case Diagram

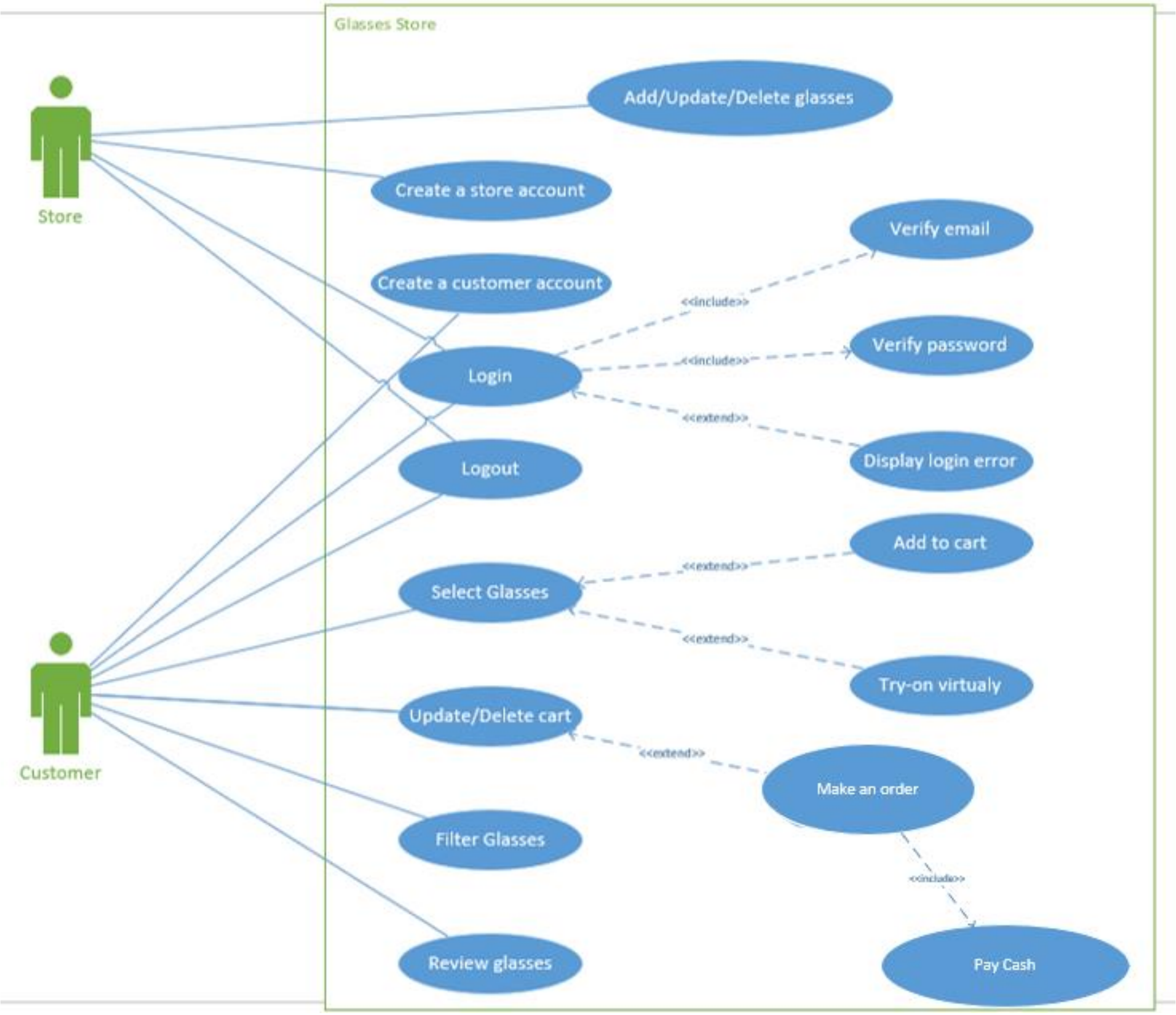


Figure 4.3 Use Case Diagram

4.3.2 Use Case Description

Here we describe the workflow for some use cases that the users interact with.

<table>
<tr><td>Use case name:</td><td colspan="2">Create a customer account.</td></tr>
<tr><td>Scenario:</td><td colspan="2">Create an online customer account.</td></tr>
<tr><td>Triggering event:</td><td colspan="2">A new customer wants to set up an account online.</td></tr>
<tr><td>Brief description:</td><td colspan="2">Online customer creates customer account by entering basic information and then following up with more addresses.</td></tr>
<tr><td>Actors:</td><td colspan="2">Customer.</td></tr>
<tr><td>Related use cases:</td><td colspan="2">Might be invoked by the *Make an Order* use case.</td></tr>
<tr><td>Stakeholders:</td><td colspan="2">Accounting, Marketing, Sales.</td></tr>
<tr><td>Preconditions:</td><td colspan="2">Customer account must be available.</td></tr>
<tr><td>Postconditions:</td><td colspan="2">Customer must be created and saved.
Addresses must be created and saved.
Account must be created and saved.
Address and Account must be associated with the Customer.</td></tr>
<tr><td rowspan="4">Flow of activities:</td><td>**Actor**</td><td>**System**</td></tr>
<tr><td>1. Customer indicates a desire to create a customer account and enters basic customer information.</td><td>1.1 System creates a new customer.
1.2 System prompts for customer addresses.</td></tr>
<tr><td>2. Customer enters Addresses</td><td>2.1 System creates addresses.</td></tr>
<tr><td>3. Customer finishes filling in information.</td><td>3.1 System creates an account.
3.2 System associates customer, address, and account.
3.3 System returns valid customer account details.</td></tr>
<tr><td>Exception conditions:</td><td colspan="2">1.1 Basic customer data are incomplete.
2.1 The address isn’t valid.</td></tr>
</table>

Table 4.1 Create Customer Account

<table>
<tr><td>Use case name:</td><td>Make an order</td></tr>
<tr><td>Scenario:</td><td>Make an order to buy glasses.</td></tr>
<tr><td>Triggering event:</td><td>The customer wants to buy glasses.</td></tr>
<tr><td>Brief description:</td><td>The customer makes an order after confirming the order operation.</td></tr>
<tr><td>Actors:</td><td>Customer.</td></tr>
<tr><td>Related use cases:</td><td>Glasses must be added to the cart by <i>Update/Delete Cart</i></td></tr>
<tr><td>Stakeholders:</td><td>Marketing, Sales, Accounting.</td></tr>
<tr><td>Preconditions:</td><td>Glasses must exist in their stores.
Customer Account must be available.</td></tr>
<tr><td>Postconditions:</td><td>New Order must be added in the Orders Screen.</td></tr>
<tr><td>Flow of activities:</td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td><b>1.</b> Customer indicates desire to view all glasses and select one of them.
<b>2.</b> Customer see all details of the glasses and may add them to the cart or try it on face virtually.
<b>3.</b> Customer adds the Glasses to the cart
<b>4.</b> Customer Make an order</td><td>1.1 System moves screen to product details screen.
2.1 System adds the glasses to the cart if the customer selects them.
2.2 System opens the camera and shows glasses to try it by customer.
3.1 System adds the Glasses to the cart Screen.
4.1 System adds the Glasses to the Orders Screen.</td></tr>
</table>
</td></tr>
<tr><td>Exception conditions:</td><td>-</td></tr>
</table>

Table 4.2 Make an order

<table>
<tr><td>Use case name:</td><td>Select Glasses.</td></tr>
<tr><td>Scenario:</td><td>Select a glasses to view it.</td></tr>
<tr><td>Triggering event:</td><td>The customer wants to buy glasses.</td></tr>
<tr><td>Brief description:</td><td>The customer views a glasses by selecting it and viewing all attributes.</td></tr>
<tr><td>Actors:</td><td>Customer.</td></tr>
<tr><td>Related use cases:</td><td>Might be invoked after <i>Filter Glasses</i> use case</td></tr>
<tr><td>Stakeholders:</td><td>Marketing, Sales.</td></tr>
<tr><td>Preconditions:</td><td>Glasses must exist in their stores.</td></tr>
<tr><td>Postconditions:</td><td>New Screen must be shown to view product details.</td></tr>
<tr><td>Flow of activities:</td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Customer indicates a desire to view all glasses and select one of them.</td><td>1.2 System moves screen to product details screen.</td></tr>
<tr><td>2. Customer see all details of the glasses and may add them to the cart or try it on face virtually.</td><td>2.1 System adds the glasses to the cart if the customer selects them.
2.2 System opens the camera and shows glasses to try it by customer.</td></tr>
</table>
</td></tr>
<tr><td>Exception conditions:</td><td>-</td></tr>
</table>

Table 4.3 Select Glasses

<table>
<tr><td>Use case name:</td><td>Add / Update/ Delete glasses.</td></tr>
<tr><td>Scenario:</td><td>Add/ update/ delete Glasses.</td></tr>
<tr><td>Triggering event:</td><td>A Store wants to update its Products.</td></tr>
<tr><td>Brief description:</td><td>Online Store updates its products or deletes glasses or adds glasses.</td></tr>
<tr><td>Actors:</td><td>Store.</td></tr>
<tr><td>Related use cases:</td><td>-</td></tr>
<tr><td>Stakeholders:</td><td>Sales.</td></tr>
<tr><td>Preconditions:</td><td>A store account must be available.</td></tr>
<tr><td>Postconditions:</td><td>Products on the overview page must be updated.</td></tr>
<tr><td>Flow of activities:</td><td>
<table>
<tr><th>Actor</th><th>System</th></tr>
<tr><td>1. Store indicates a desire to update glasses by modifying glasses attributes.</td><td>1.1 System open manage store screen.</td></tr>
<tr><td>2. Store confirms editing glasses in the store</td><td>2.1 System update glasses information in the store</td></tr>
</table>
</td></tr>
<tr><td>Exception conditions:</td><td>1.1 Basic glasses information is invalid.</td></tr>
</table>

Table 4.4 Add/ Update/ Delete Glasses

4.3.3 Activity Diagrams

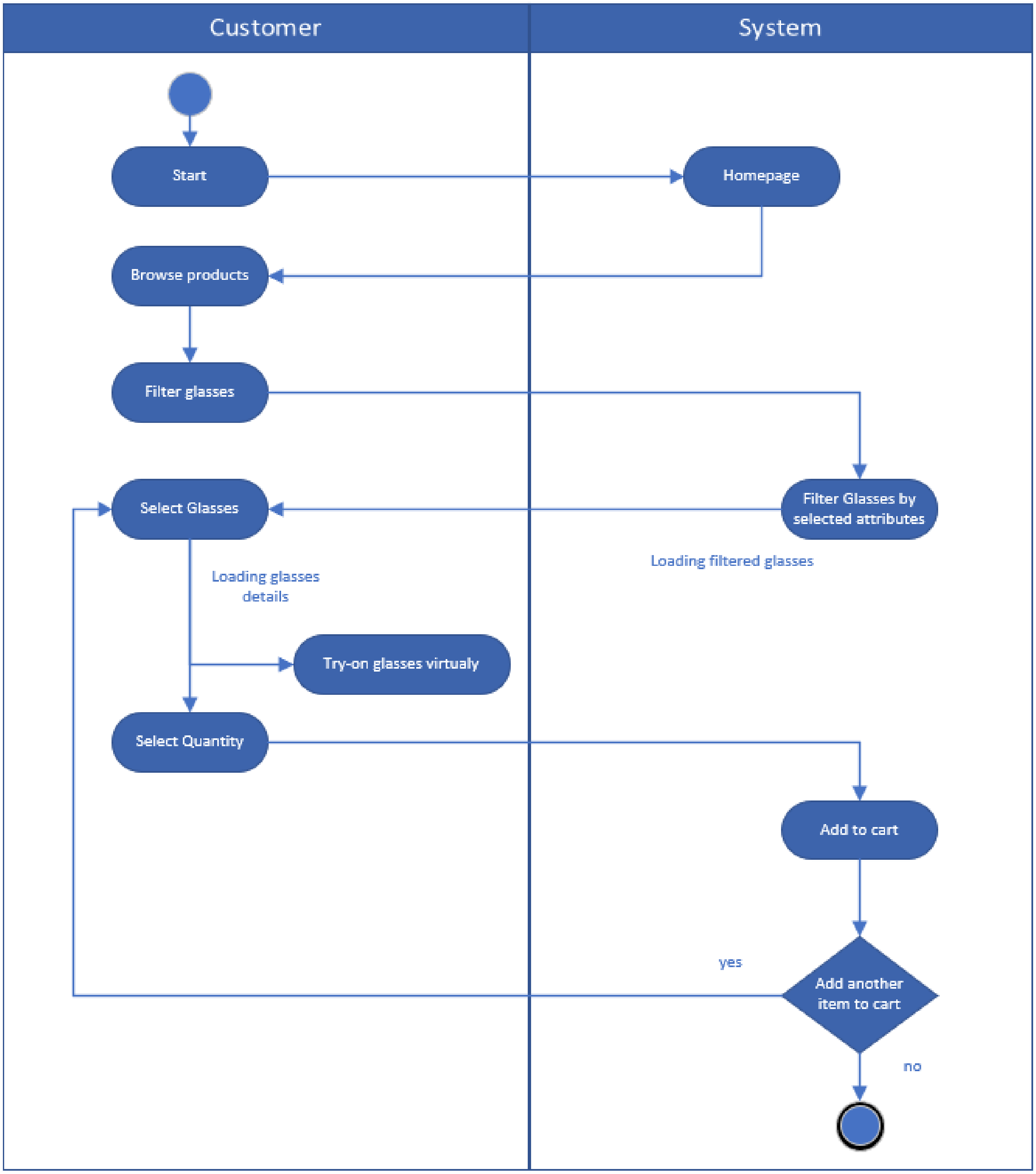


Figure 4.4 Activity diagram for update cart

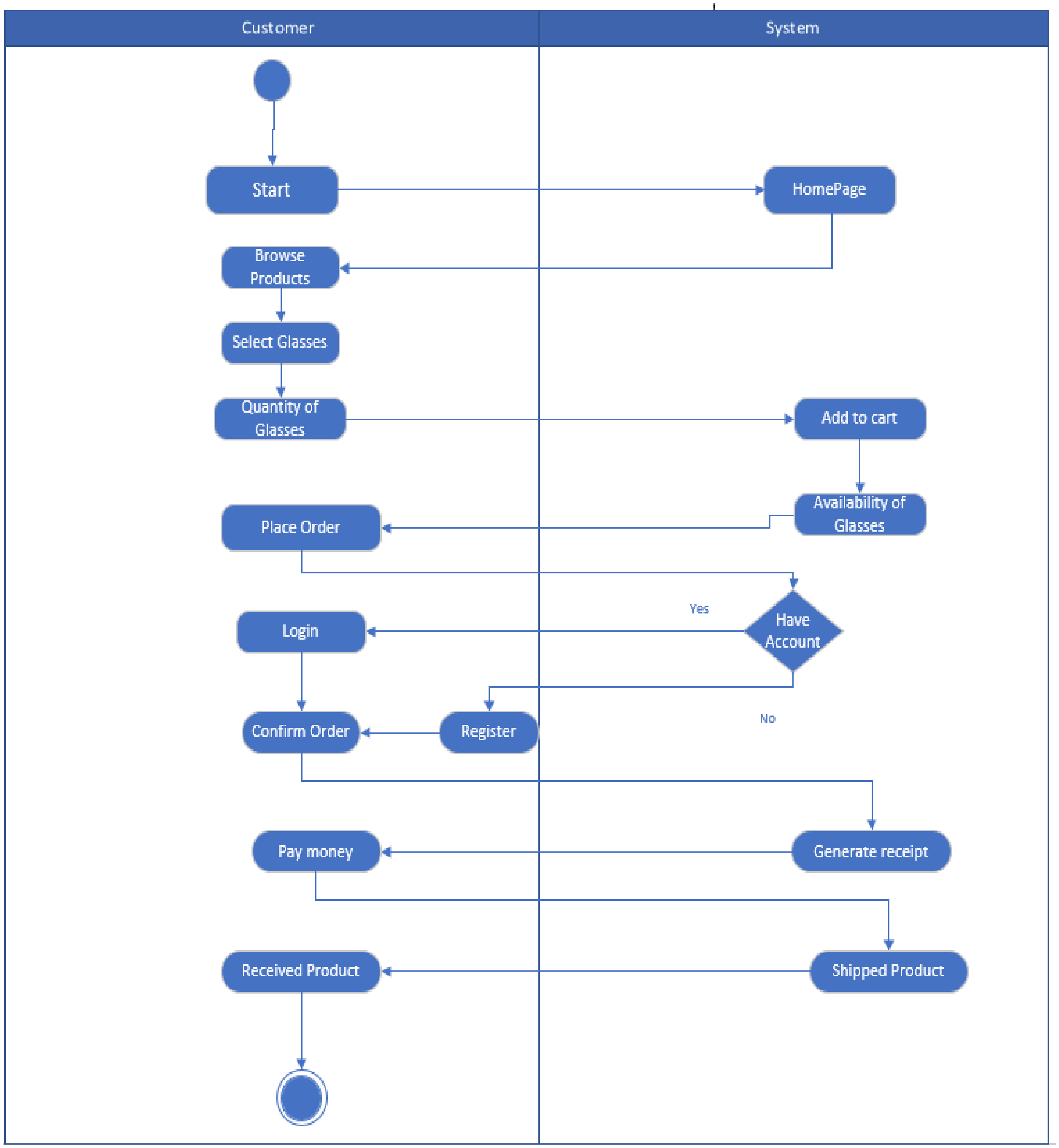


Figure 4.5 Activity diagram for Make Order

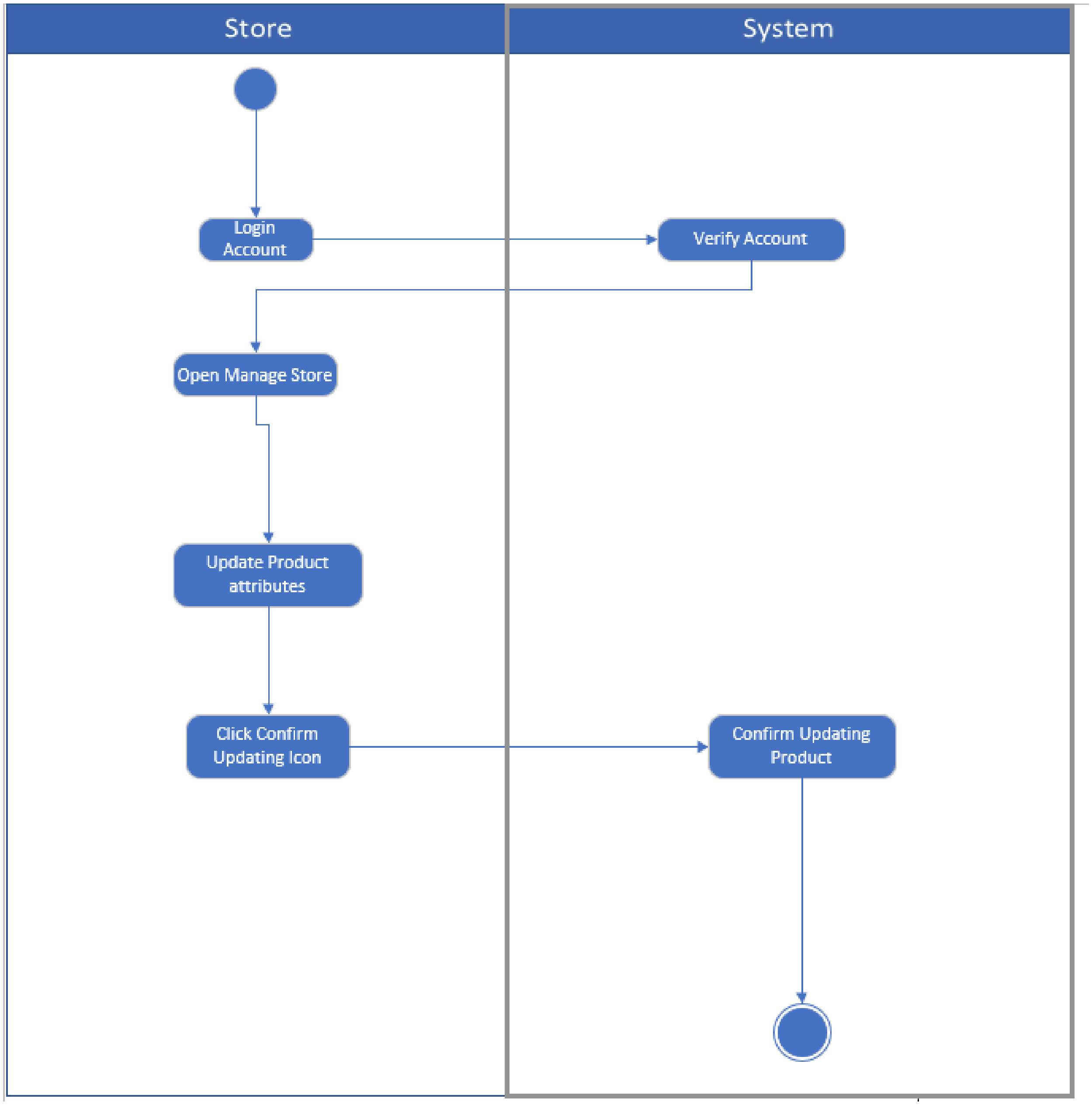


Figure 4.6 Activity diagram for Update Glasses

4.3.4 System Sequence Diagrams

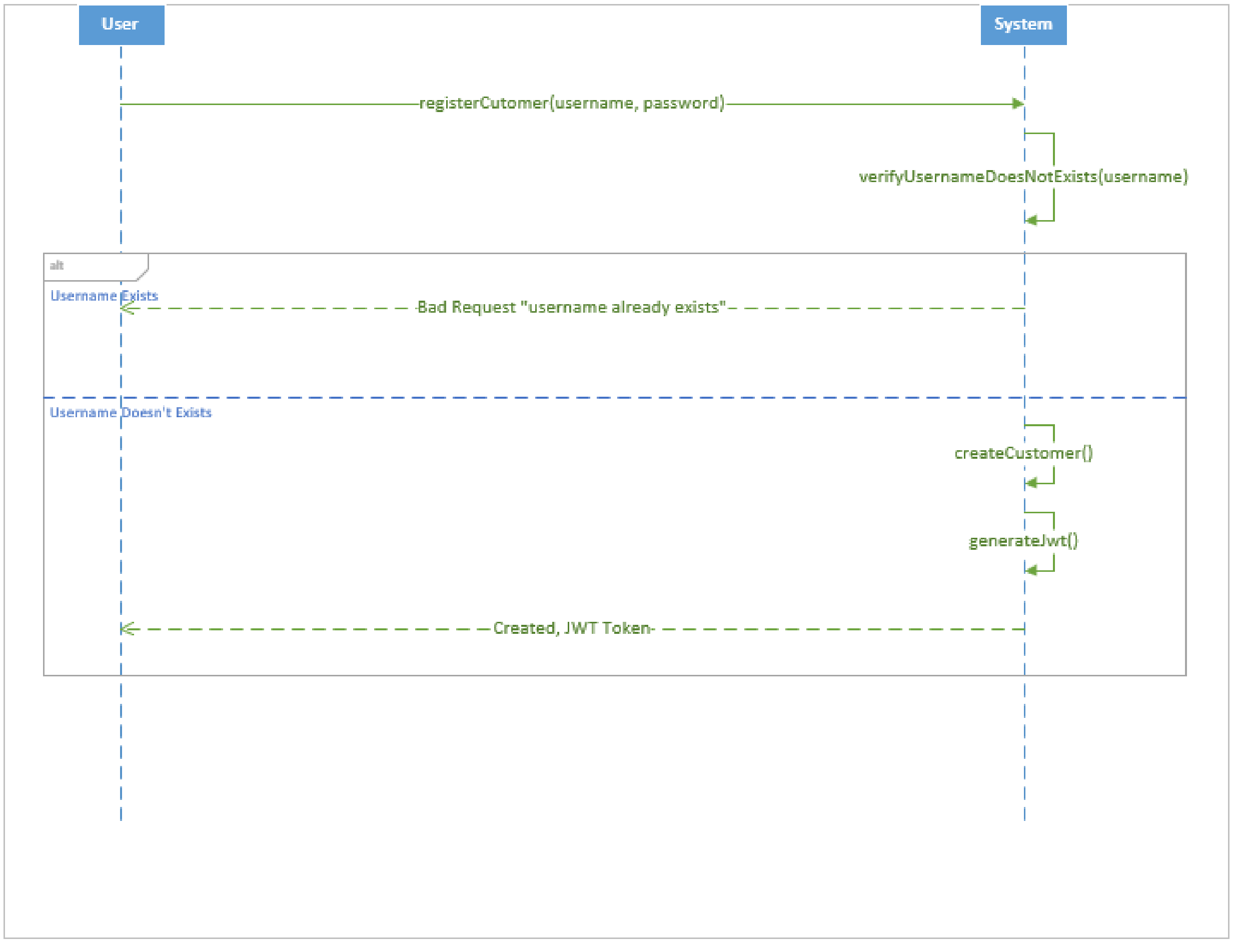

User
System
registerCutomer(username, password)
verifyUsernameDoesNotExists(username)
alt
Username Exists
Bad Request "username already exists"
Username Doesn't Exists
createCustomer()
generateJwt()
Created, JWT Token

Figure 4.7 System Sequence diagram for Customer Registration

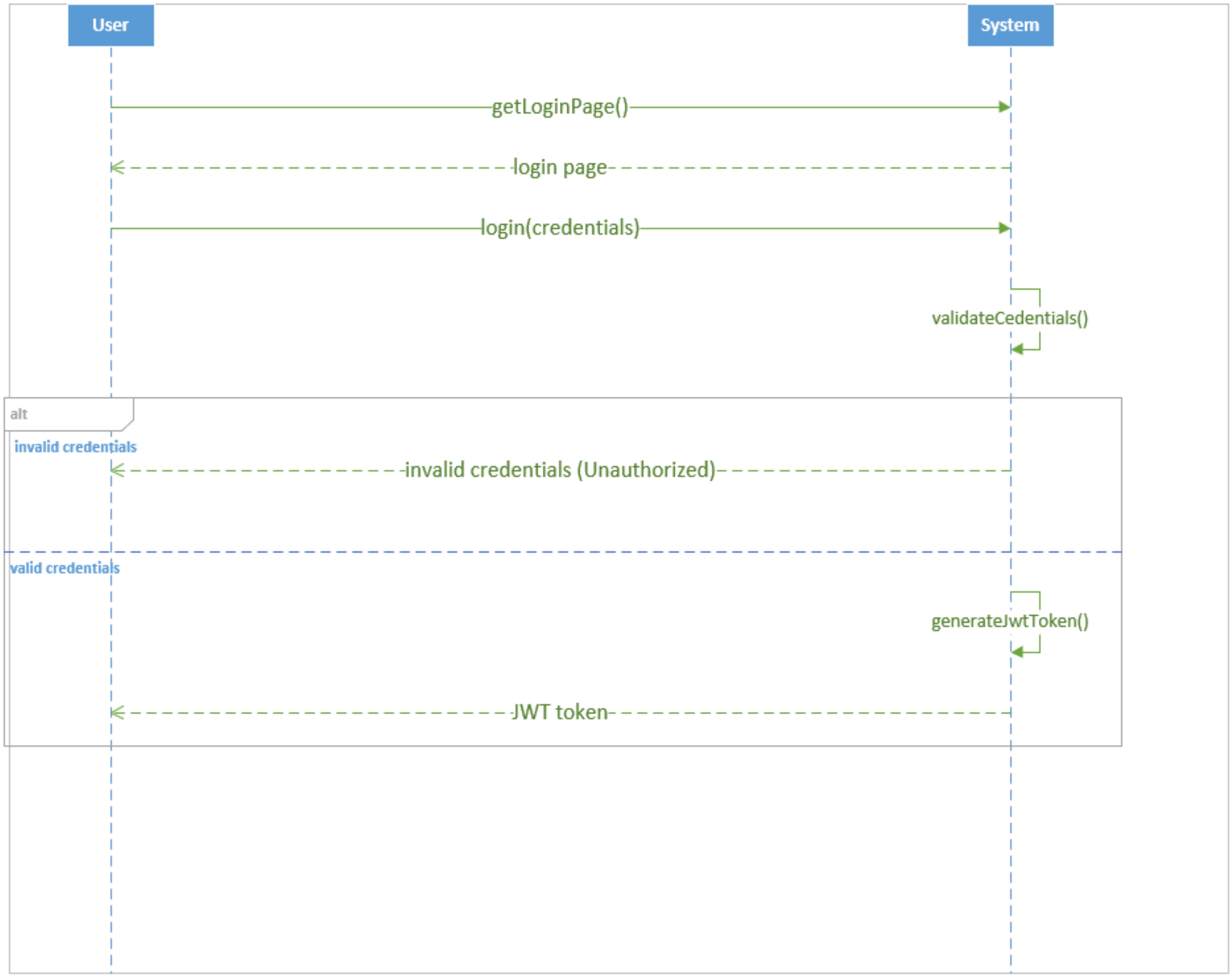


Figure 4.8 System Sequence diagram for Login

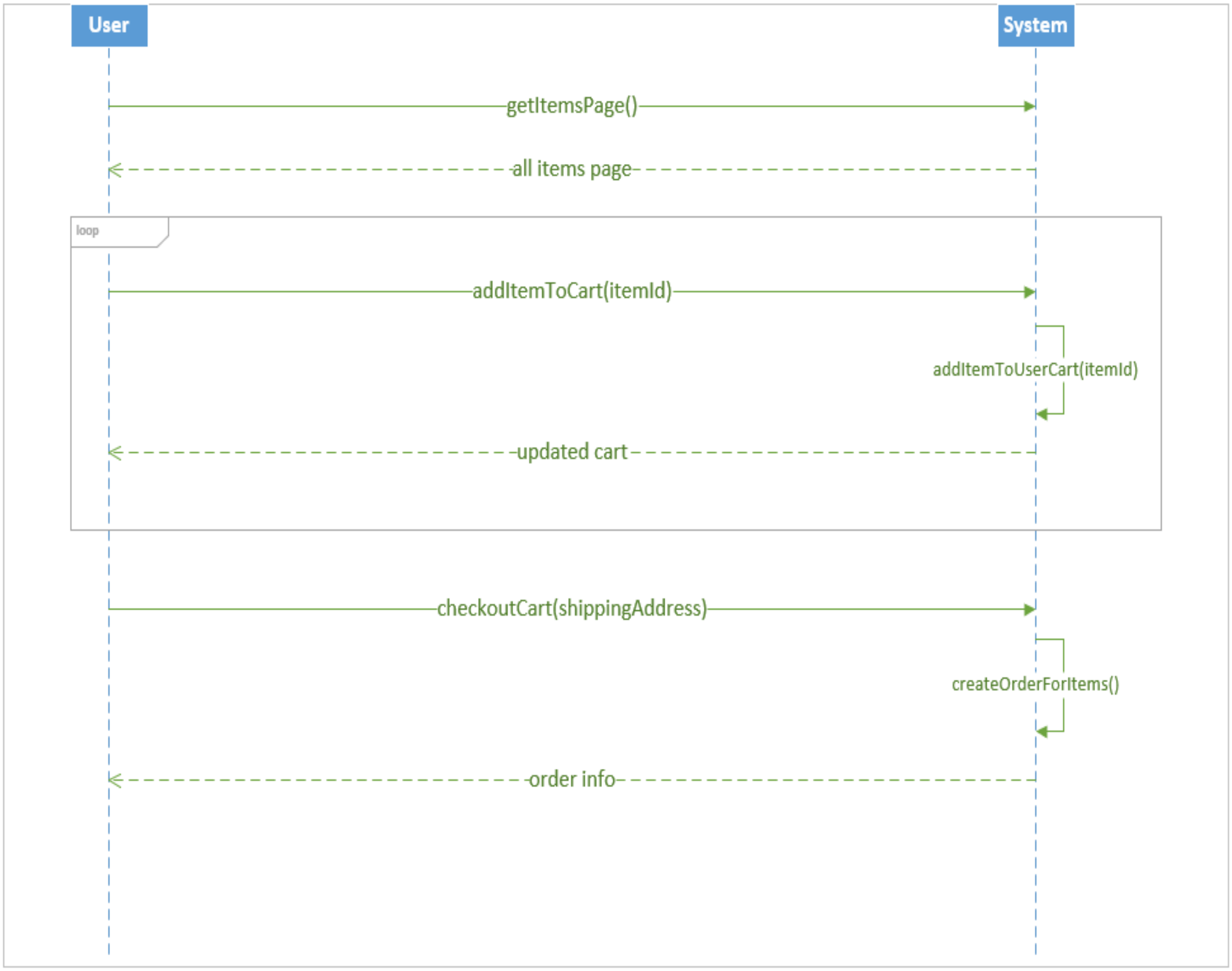


Figure 4.9 System Sequence diagram for Make an order

4.3.5 State Machine Diagrams

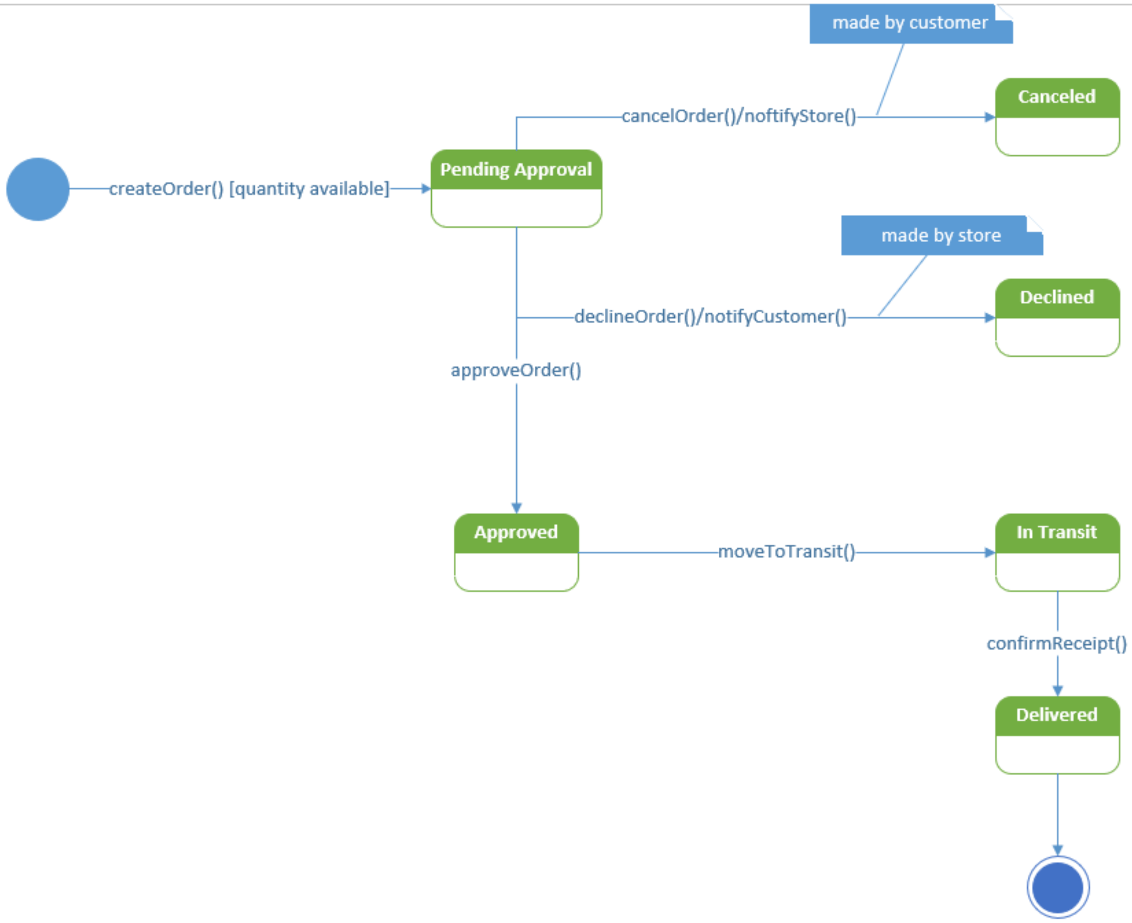


Figure 4.10 State Machine diagram for Make an order

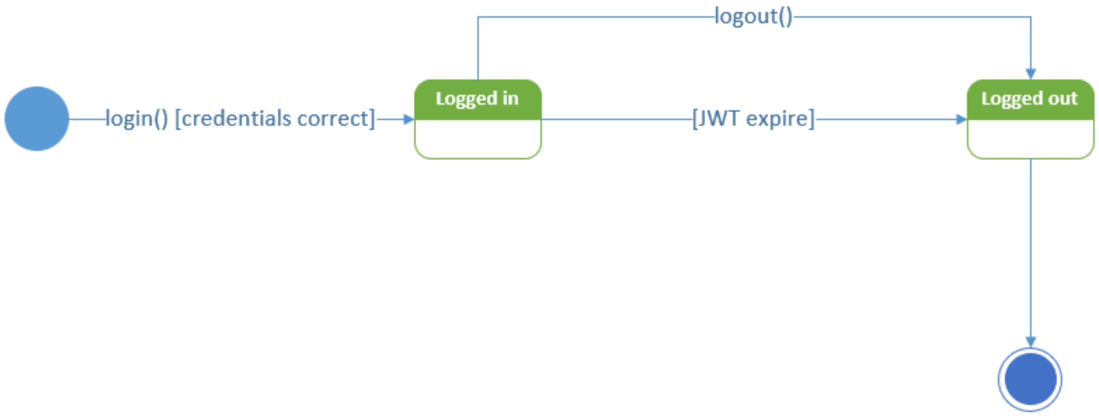


Figure 4.11 State Machine diagram for Login

4.3.6 Entity Relationship Diagram

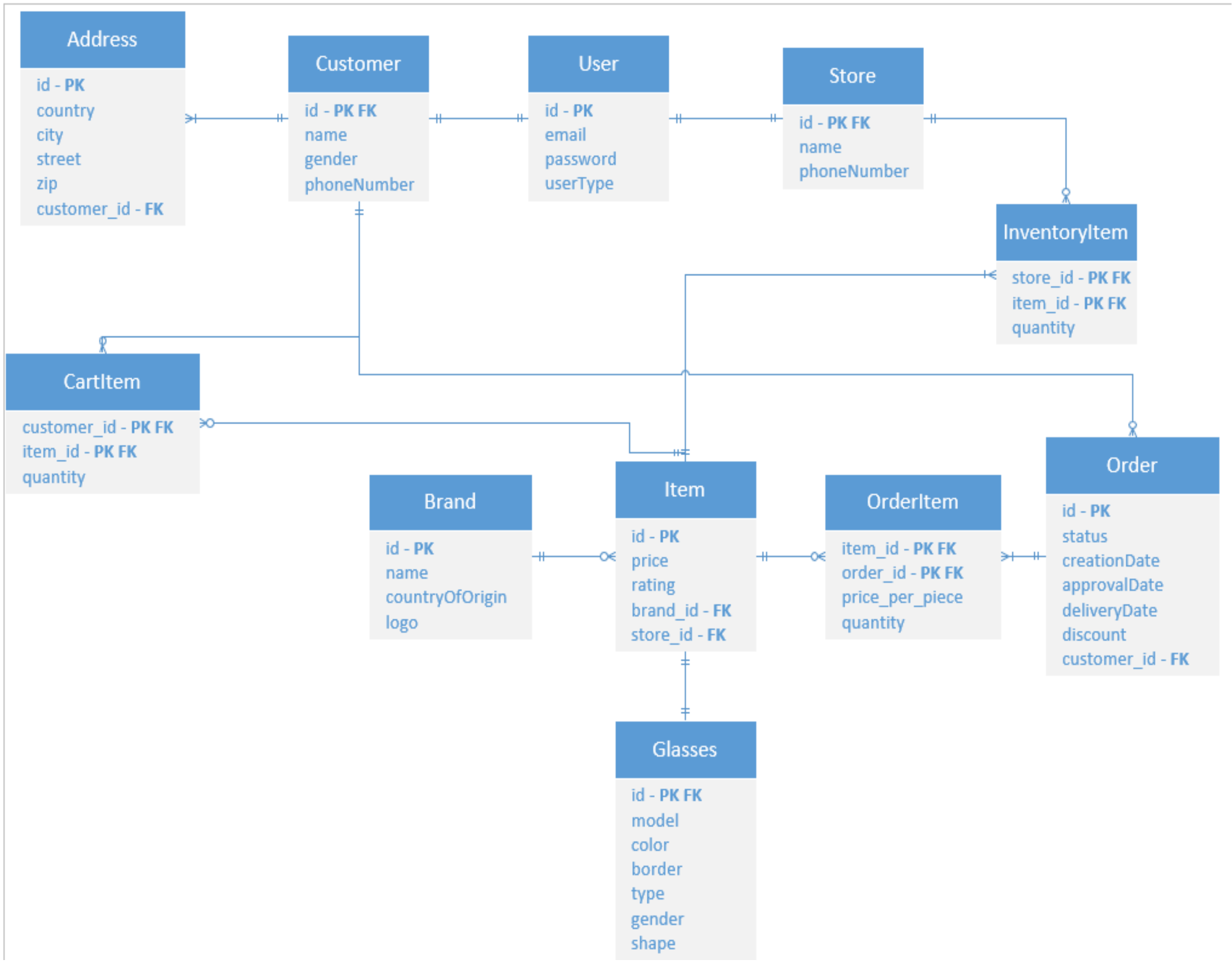


Figure 4.12 Entity Relationship diagram

4.3.7 Domain Model Class Diagram

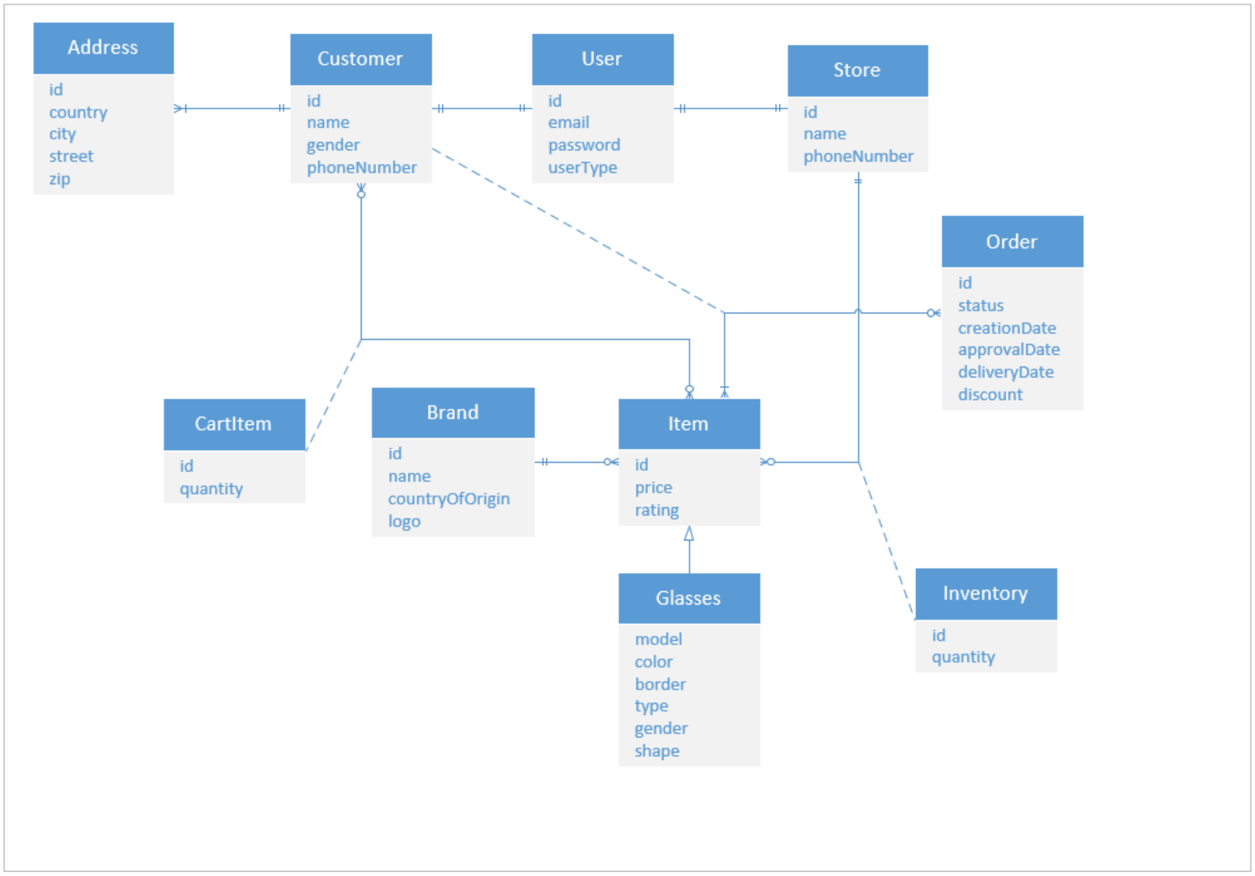
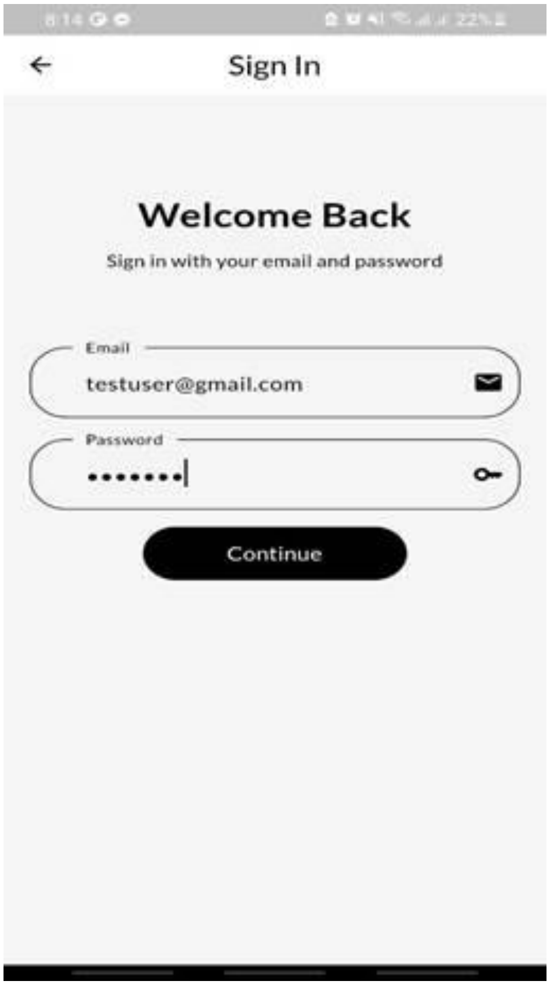


Figure 4.13 Domain Model Class diagram

4.3.8 Dialog Design and GUI interfaces



Welcome Back

Sign in with your email and password

Email: testuser@gmail.com

Password: •••••••

Login Dialog

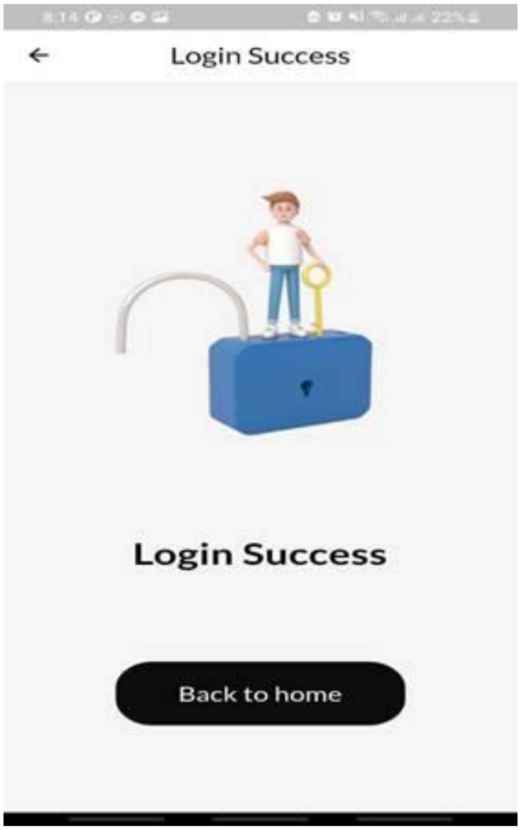
User: Hello, I need to log in to my account. Can you assist me with that?

System: Of course! I'm here to help. Please provide me with your email and password, and I'll verify your credentials.

User: Sure. My email is “testuser@gmail.com”and my password is "password123".

System: Thank you for providing your information. Let me check if your credentials are valid.

[The system verifies the email and password against the stored data.]

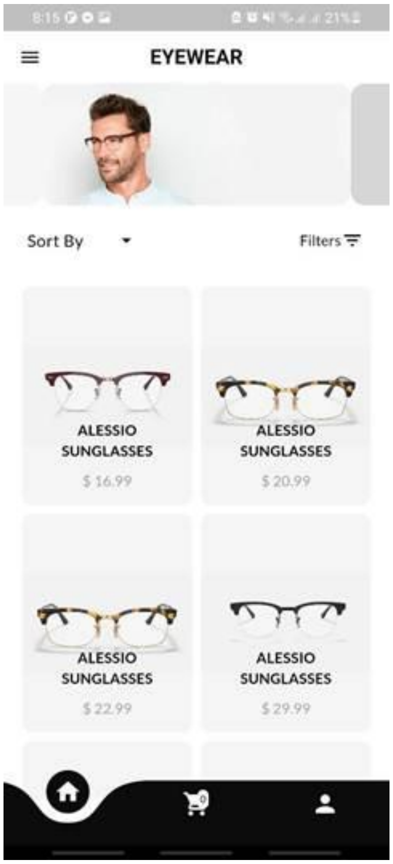


System: Congratulations! Your login was successful.

User: Thank you! It's good to be back. What would you like to do next?

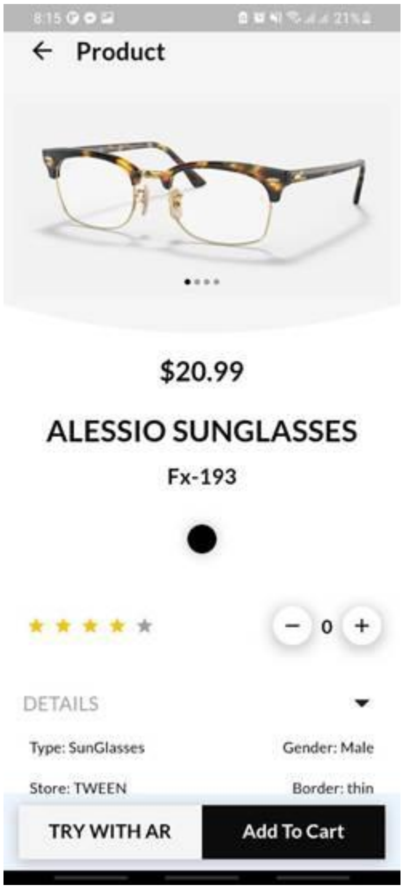
System: Now that you're logged in, you have access to all the features and functionalities of your account. You can back to home.

Make an order Dialog



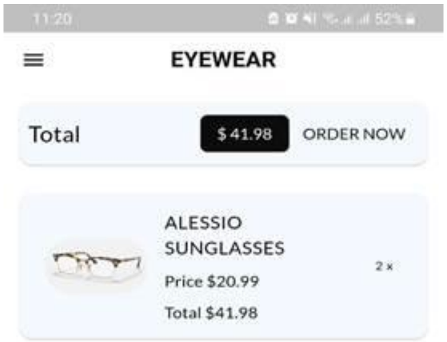
System: you can select an item to buy.

User: sure, I will choose "Alessio sunglasses"



System: You can choose quantity and add them to cart.

User: I take two glasses and add them to cart



ALESSIO SUNGLASSES

Price $20.99

Total $41.98

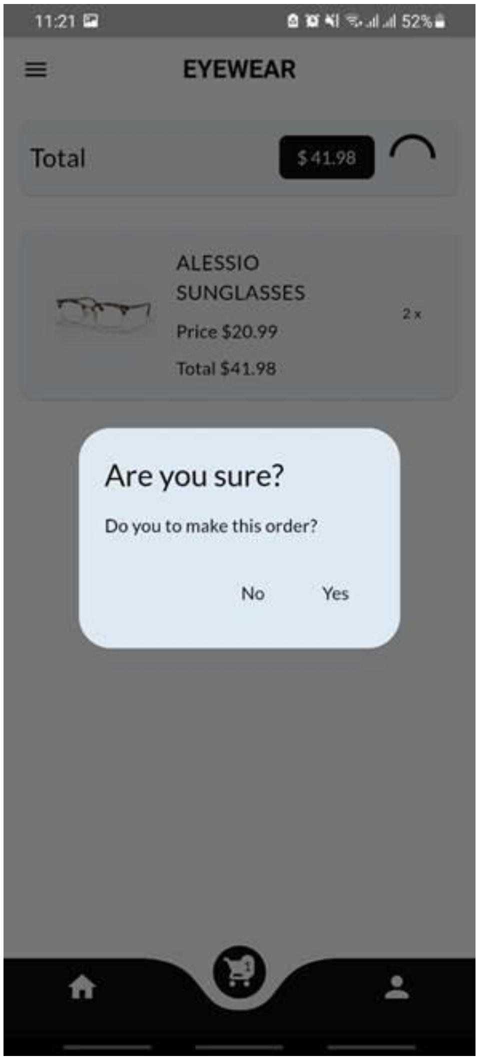
2 x

System: the glasses added to cart .

User: I want to make an order .

System: press on “order now” to make an order

User: ok, well done.



ALESSIO SUNGLASSES

Price $20.99

Total $41.98

2 x

Are you sure?

Do you to make this order?

System: are you sure that you want to make this order?

User: Yes, I want

System: your order added successfully , you can see your order from orders section.

CHAPTER 5

CONCLUSION & FUTURE VISION

This chapter reviews the conclusion and Future Vision for this project.

5.1 Conclusion

In conclusion, our glasses store application with a virtual try-on feature introduces a transformative approach to eyewear shopping. By leveraging technologies such as Flutter, Spring Boot with PostgreSQL, and Unity 3D with AR Foundation, we have created a powerful platform that revolutionizes the way customers can browse and try on glasses.

While Unity 3D and AR Foundation enabled us to accomplish our goals, it is important to note that Unity's level of flexibility may have limitations in certain areas. Although it served its purpose in developing the virtual try-on feature, alternative frameworks could potentially provide even greater flexibility and customization options.

Nonetheless, our application effectively combines these technologies to deliver a user-friendly interface that allows customers to seamlessly try on virtual glasses in real-time. The emphasis on accuracy ensures that the virtual try-on experience closely mirrors the real-life fitting process, providing customers with a reliable and realistic representation.

In summary, our glasses store application with a virtual try-on feature represents a significant advancement in the realm of eyewear shopping. While Unity 3D and AR

Foundation successfully facilitated the development of the application, there is room for improvement in terms of flexibility. Nevertheless, the application excels in providing usability, accuracy, performance, and security, paving the way for an enhanced and immersive glasses shopping experience. With a forward-thinking approach and a commitment to innovation, we are confident that our application will continue to redefine the eyewear shopping landscape.

5.2 Future Vision

1. Expanding to the Global Market: Our vision for the glasses store application extends beyond the Palestinian market, aiming to reach a global audience. To achieve this, we will focus on localization and internationalization efforts. This includes providing multi-language support, accommodating different currencies and payment methods, and tailoring the user experience to suit various cultural preferences and norms. Additionally, we will explore partnerships with global eyewear brands and suppliers to expand our product offerings and ensure worldwide availability.
2. Exploring Alternative AR Frameworks: While Unity 3D and AR Foundation offer a comprehensive solution for AR development, there are alternative frameworks that provide more low-level control and flexibility. One such example is ARCore for Android and ARKit for iOS.

CHAPTER 6

REFERENCES

6.1 References

[1.1] Flutter. "Flutter - Beautiful native apps in record time." Accessed April 12, 2023. [Online]. Available: https://flutter.dev/

[1.2] Spring Boot. "Spring Boot." Accessed March 7, 2023. [Online]. Available: https://spring.io/projects/spring-boot

[1.3] PostgreSQL. "PostgreSQL." Accessed 22 May, 2023. [Online]. Available: https://www.postgresql.org/

[1.4] Unity 3D. "Unity 3D Features." Accessed 1 March, 2023. [Online]. Available: https://unity.com/unity/features/3d

[1.5] AR Foundation. "Unity - AR Foundation." Accessed 2 March, 2023. [Online]. Available: https://unity.com/unity/features/arfoundation

[1.6] JWT (JSON Web Token). "JWT (JSON Web Token)." Accessed April 4, 2023. [Online]. Available: https://jwt.io/

[1.7] Eyeconic. "Men's Sunglasses - Eyeconic." Accessed March 12, 2023. [Online]. Available: https://www.eyeconic.com/eyewear/sunglasses/mens

[2.1] Railway. "Railway." Accessed February 2, 2023. [Online]. Available: https://railway.app/

[2.2] Zenni Optical. "Zenni Optical." Accessed February 2, 2023. [Online].

Available:

https://www.zennioptical.com/?gclid=CjwKCAjwp6CkBhB_EiwAlQVyxerDlUmA2HtSgbU80an-Pz6d0TvKcXHoL4nipNMN9q4GyRjitUvoGxoCWYEQAvD_BwE&gclsrc=aw.ds