**ONLINE EYEWEAR STORE**

**A PROJECT COMPONENT REPORT**

***Submitted by***

|  |  |
| --- | --- |
| **GOWTHAM BABU. E** | **(Reg. No. 202004039)** |
| **HARRISH RAGAVENDAR. S** | **(Reg. No. 202004044)** |

***for the Theory Cum Project Component***

***of***

**19CS694 – WEB USER INTERFACE DESIGN**

***during***

***VI Semester – 2022 – 2023***

**Logo

Description automatically generated**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI**

**(An Autonomous Institution affiliated to Anna University Chennai)**

**April 2023**

**MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI**

**(An Autonomous Institution affiliated to Anna University Chennai)**

**Department of Computer Science and Engineering**

**BONAFIDE CERTIFICATE**

Certified that this project component report titled **ONLINE EYEWEAR STORE** is the bonafide work of **E.GOWTHAM BABU (Reg. No. 202004039),** and **S.HARRISH RAGAVENDAR (Reg. No. 202004044)** who carried out this work under my guidance for the Theory cum Project Component course **“19CS694 – WEB USER INTERFACE DESIGN”** during the sixth semester.

|  |  |
| --- | --- |
| **Dr. M. S. BHUVANESWARI,** M.E.,Ph.D.  Associate Professor  Course Instructor  Department of Computer Science & Engg. Mepco Schlenk Engineering College  Sivakasi. | **Dr. J. RAJA SEKAR,** M.E.,Ph.D.  Professor  Head of the Department  Department of Computer Science & Engg. Mepco Schlenk Engineering College  Sivakasi. |

Submitted for Viva-Voce Examination held at **MEPCO SCHLENK ENGINEERING COLLEGE (Autonomous), SIVAKASI** on **……/05/2023**

**Internal Examiner External Examiner**

**ABSTRACT**

The need for an online eyewear store has increased over the years due to several factors. Firstly, people have become more tech-savvy and prefer to shop online, as it offers convenience and saves time. Secondly, the eyewear industry is growing rapidly, and customers are looking for a wider range of products, styles, and designs to choose from. An online eyewear store can offer a more extensive range of products as compared to a physical store. The online eyewear store project aims to provide a convenient and hassle-free way for customers to purchase high-quality eyewear products from the comfort of their own homes. The website will offer a wide range of eyewear products, including prescription glasses, sunglasses, contact lenses, and other accessories. Customers will be able to browse and filter products based on various parameters such as price, brand, style, and lens type. Moreover, an online eyewear store can offer customers access to high-quality eyewear products at competitive prices, making it more affordable for customers to purchase eyewear. Additionally, customers can also benefit from the convenience of being able to purchase eyewear products from anywhere in the world and have them delivered to their doorstep. The website will also offer personalized recommendations based on the customer's previous purchases and browsing history. The project will utilize secure payment gateways and a user-friendly interface to ensure a smooth shopping experience for customers. The online eyewear store project is a promising initiative that seeks to revolutionize the eyewear industry by offering customers a convenient and hassle-free way to purchase high-quality eyewear products.

**ACKNOWLEDGEMENT**

First and foremost, we thank the **LORD ALMIGHTY** for his abundant blessings that is showered upon our past, present and future successful endeavors.

We extend our sincere gratitude to our college management and Principal **Dr. S. Arivazhagan M.E., Ph.D.,** for providing sufficient working environment such as systems and library facilities. We also thank him very much for providing us with adequate lab facilities, which enable us to complete our project.

We would like to extend our heartfelt gratitude to **Dr. J. Raja Sekar M.E., Ph.D.,** Professor and Head, Department of Computer Science and Engineering, Mepco Schlenk Engineering College for giving me the golden opportunity to undertake a project of this nature and for his most valuable guidance given at every phase of our work.

We would also like to extend our gratitude and sincere thanks to **Dr. M. S. Bhuvaneswari M.E., Ph.D.,** Associate Professor, Department of Computer Science and Engineering, Mepco Schlenk Engineering College for being our Project Mentor. She has put her valuable experience and expertise in directing, suggesting and supporting us throughout the Project to bring out the best.

Our sincere thanks to our revered **faculty members and lab technicians** for their help over this project work.

Last but not least, we extend our indebtedness towards out beloved family and our friends for their support which made the project a successful one.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | **ii** |
|  | **LIST OF TABLES** | **v** |
|  | **LIST OF FIGURES** | **vi** |
|  | **LIST OF ABBREVIATIONS** | **vii** |
| **1** | **INTRODUCTION** | **1** |
| **2** | **REQUIREMENTS DESCRIPTION** | **3** |
|  | 2.1 Functional Requirements | 3 |
|  | 2.2 Non-Functional Requirements | 3 |
| **3** | **SYSTEM DESIGN** | **5** |
|  | 3.1 Architectural design | 5 |
|  | 3.2 Design Components | 7 |
|  | 3.3 Database Description | 7 |
|  | 3.4 Low Level design | 10 |
|  | 3.5 User Interface design | 14 |
| **4** | **SYSTEM IMPLEMENTATION** | **25** |
| **5** | **RESULTS AND DISCUSSION** | **28** |
| **6** | **CONCLUSION AND FUTURE ENHANCEMENT(S)** | **34** |

|  |  |  |
| --- | --- | --- |
| **APPENDIX – A** | **SYSTEM REQUIREMENTS** | **35** |
| **APPENDIX – B** | **SOURCE CODE** | **36** |
|  | **REFERENCES** | **58** |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table Caption** | **Page No.** |
| 3.1 | User Description | 7 |
| 3.2 | Product Description | 8 |
| 3.3 | Review Description | 8 |
| 3.4 | Order Description | 9 |
| 3.5 | Shipping Address Description | 10 |
| 3.6 | Payment Description | 10 |
| 3.7 | Login Details | 10 |
| 3.8 | Register Details | 11 |
| 3.9 | Add Eyewear Details | 11 |
| 3.10 | Add to Cart Details | 12 |
| 3.11 | Place Order Details | 12 |
| 3.12 | Make Payment Details | 13 |
| 3.13 | Search Eyewear Details | 13 |
| 3.14 | Shop by Category Details | 14 |
| 5.1 | Positive Test Case and result for Login | 28 |
| 5.2 | Negative Test Case and result for Login | 28 |
| 5.3 | Positive Test Case and result for Register | 29 |
| 5.4 | Negative Test Case and result for Register | 29 |
| 5.5 | Positive Test Case and result for Add Eyewear | 29 |
| 5.6 | Negative Test Case and result for Add Eyewear | 30 |
| 5.7 | Positive Test Case and result for Add to Cart | 30 |
| 5.8 | Negative Test Case and result for Add to Cart | 30 |
| 5.9 | Positive Test Case and result for Place Order | 31 |
| 5.10 | Negative Test Case and result for Place Order | 31 |
| 5.11 | Positive Test Case and result for Make Payment | 32 |
| 5.12 | Negative Test Case and result for Make Payment | 32 |
| 5.13 | Positive Test Case and result for Search Eyewear | 32 |
| 5.14 | Negative Test Case and result for Search Eyewear | 33 |
| 5.15 | Positive Test Case and result for Shop by Category | 33 |
| 5.16 | Negative Test Case and result for Shop by Category | 33 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure Caption** | **Page No.** |
| 3.1 | Architecture Diagram of Online Eyewear Store | 6 |
| 3.2 | Home Page of Online Eyewear Store | 14 |
| 3.3 | Login Page for Users | 15 |
| 3.4 | Register Page for Users | 15 |
| 3.5 | Product Details Page | 16 |
| 3.6 | Cart Page | 16 |
| 3.7 | Shipping Address Page | 17 |
| 3.8 | Payment Method Page | 17 |
| 3.9 | Place Order Page | 18 |
| 3.10 | Order Summary Page | 19 |
| 3.11 | Admin Dashboard Page | 19 |
| 3.12 | Manage Orders Page | 20 |
| 3.13 | Eyewear Inventory Page | 20 |
| 3.14 | Add Eyewear Page | 21 |
| 3.15 | Manage Users Page | 21 |
| 3.16 | Update User Page | 22 |
| 3.17 | Deliver Order Page | 22 |
| 3.18 | Search Eyewear Page | 23 |
| 3.19 | Shop by Category Page | 23 |
| 3.20 | Category Filter Page | 24 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **ABBREVIATION** | **DESCRIPTION** |
| API  COD  DB  DBMS  JS  TS | Application Programming Interface  Cash On Delivery  Database  Database Management System  JavaScript  TypeScript |
| UI | User Interface |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**CHAPTER 1**

**INTRODUCTION**

* 1. **PERSCPECTIVE**

The online eyewear store project has great potential from various perspectives. From a customer's perspective, it provides a convenient way to purchase high-quality eyewear products without having to visit a physical store. Customers can browse a wider range of products, filter them based on their preferences, and use virtual try-on tools to see how the glasses look on them before making a purchase. Moreover, an online eyewear store can offer competitive prices, making it more affordable for customers to purchase eyewear products.

From a business perspective, an online eyewear store can offer various benefits. It can reach a wider customer base, including those who live in remote areas or have limited mobility. An online store can also reduce overhead costs, such as rent and staffing, as it requires fewer physical resources to operate. Furthermore, an online eyewear store can gather customer data and use it to personalize the shopping experience, increasing customer engagement and loyalty. Overall, the online eyewear store project has great potential to disrupt the eyewear industry by providing a more convenient and personalized shopping experience for customers and reducing overhead costs for businesses.

**1.2 OBJECTIVES**

The objective of the online eyewear store project is to create a user-friendly and convenient platform for customers to purchase high-quality eyewear products online. The project aims to offer a wide range of eyewear products, including prescription glasses, sunglasses, contact lenses, and other accessories, while also providing a personalized shopping experience. The project will utilize data analytics to offer personalized recommendations based on the customer's previous purchases and browsing history. To ensure a safe shopping experience, the project aims to offer secure payment gateways and provide excellent customer service, including prompt responses to customer queries and timely resolution of issues. By offering high-quality products, personalized recommendations, and excellent customer service, the project aims to increase brand awareness and loyalty while generating revenue for the business. Ultimately, the objective of the online eyewear store project is to provide customers with a seamless and hassle-free shopping experience, increasing customer satisfaction and loyalty.

* 1. **SCOPE**

The scope of the online eyewear store project is to provide customers with a convenient and personalized way to purchase eyewear products online. The project aims to offer a wide range of eyewear products, including prescription glasses, sunglasses, contact lenses, and other accessories, to cater to the diverse needs of customers. Customers will be able to browse and filter products based on various parameters such as price, brand, style, and lens type. Moreover, the online eyewear store project aims to ensure secure payment gateways to protect customer's sensitive information and provide a safe shopping experience. The project will also provide excellent customer service, including prompt responses to customer queries and timely resolution of issues, to ensure customer satisfaction. The scope of the project also includes generating revenue for the business by increasing brand awareness and loyalty through high-quality products, personalized recommendations, and excellent customer service. Overall, the online eyewear store project aims to revolutionize the eyewear industry by providing customers with a seamless and hassle-free shopping experience, increasing customer satisfaction, and loyalty.

**CHAPTER 2**

**REQUIREMENT DESCRIPTION**

**2.1 FUNCTIONAL REQUIREMENTS**

The functional requirements of the Online Eyewear Store are the collective information about the core operations that are available in the system.

* The system should be able to manage user accounts, including registration, login, and profile management.
* The system should be able to manage product details, such as product images, descriptions, specifications, and pricing. It should also allow the administrator to add, delete or modify products.
* The system should allow customers to add products to their shopping cart, view the total cost of their order, and proceed to checkout. The checkout process should include options for payment and shipping, as well as order confirmation.
* The system should provide search and filter options to help customers find products that meet their specific requirements.
* The system should ensure secure payment gateways to protect customer's sensitive information.
* The system should offer excellent customer service, including prompt responses to customer queries and timely resolution of issues.

**2.2 NON-FUNCTIONAL REQUIREMENTS**

Non-functional requirements are the quality attributes of the Online Eyewear Store that describe how well it performs its intended functions rather than what functions it performs.

* The system should be user-friendly and easy to navigate, providing a smooth shopping experience for customers.
* The system should be able to handle a large volume of traffic and transactions without any downtime or lag.
* The system should ensure data security by implementing measures such as secure payment gateways, data encryption, and firewall protection.
* The system should be compatible with different devices and platforms, such as desktops, laptops, tablets, and mobile phones.
* The system should be scalable to accommodate future growth in traffic and sales.
* The system should be reliable, ensuring that all transactions and customer data are securely processed and stored.
* The system should be available 24/7 to customers, providing them with access to the website at all times.
* The system should be accessible to all customers, including those with disabilities, through features such as screen readers and alternative text.
* The system should comply with relevant laws and regulations, such as data protection and consumer protection laws.

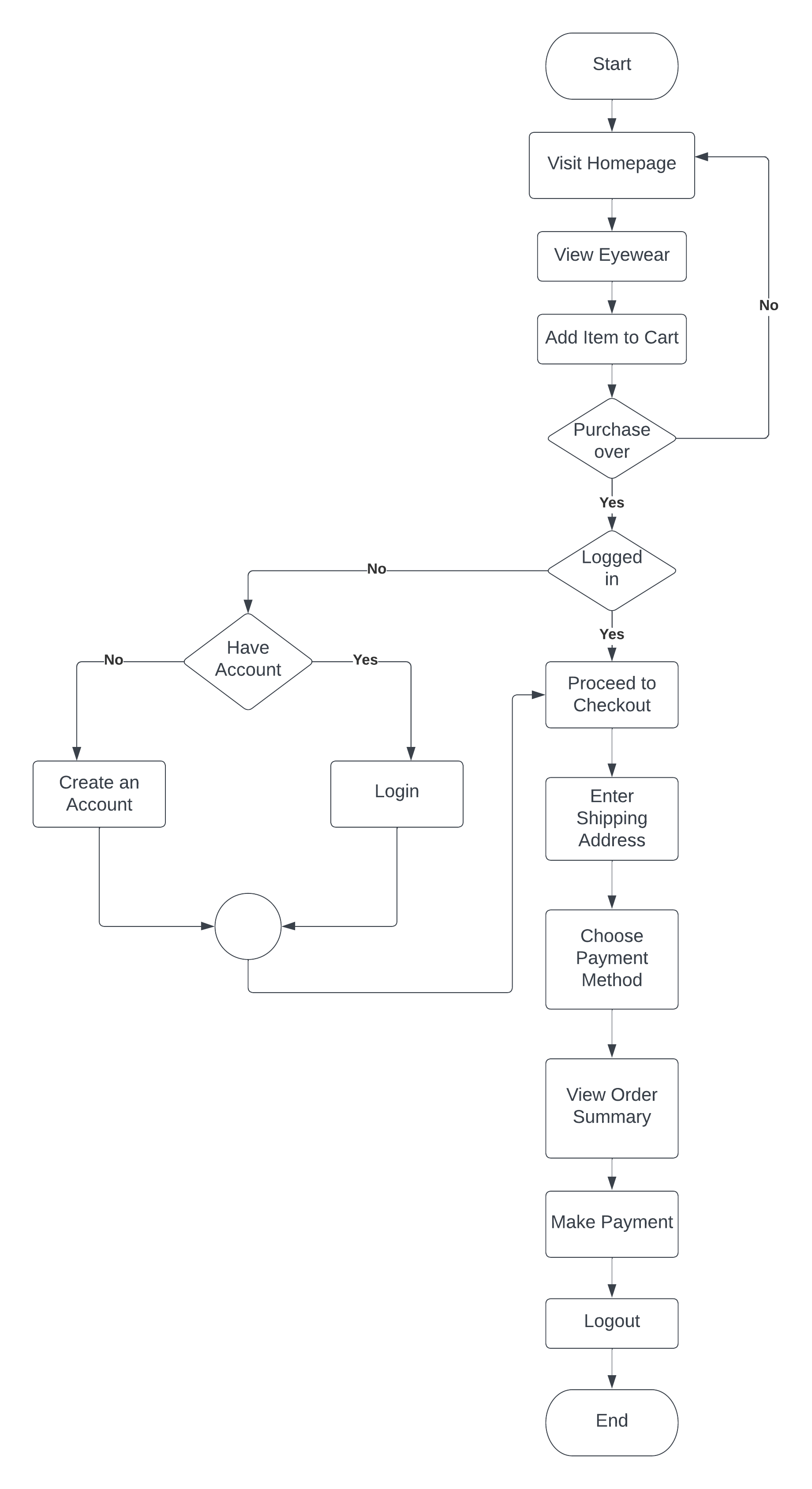
**CHAPTER 3**

**SYSTEM DESIGN**

**3.1 ARCHITECTURE DESIGN**

Architectural flow diagram implies the flow of the system. **Figure 3.1** depicts the architectural flow diagram for the Online Eyewear Store website. The flow starts with the user visiting the homepage of the website. The user can view the various kind of eyewear that are available. The user can also view the detailed description of each eyewear. The user can add the eyewear to the cart that he/she wishes to buy. Once the eyewear is added to the cart, the user can also change the quantity of the eyewear that he/she wishes to buy, that is it can be incremented or decremented. The user can then return to the home page and buy furthermore eyewear.

Once the user has added all the desired eyewear to the cart with the desired quantity, the system checks whether the user is logged in or not. If the user is already logged in, the user proceeds to checkout page. Else, the user is taken to the login page to login to the lensmakers website. If the user is a new user, then the user is taken to the registration page for creating a new account. Then, the user proceeds to check out and enters the shipping address of the products. Then the user chooses the desired payment method, and the order summary is shown to the user. Then the user makes payment online if he has chosen the PayPal option, else the user confirms the order in case of COD option.



**Figure 3.1: Flow Diagram of Online Eyewear Store Website**

**3.2 DESIGN COMPONENTS**

**3.2.1 Front End:**

The front end of the Online Eyewear Store is developed using the Angular JS framework and Angular Material UI Component Library.

**3.2.2 Back End:**

Express JS is used to build the backend server for the Online Eyewear Store and MongoDB is used as the database in the back end.

**3.3 DATABASE DESCRIPTION**

Listed below gives a description of database document schemas used for Online Eyewear Store.

**3.3.1 User Structure**

The details about the users stored in the User table are shown in **Table 3.1.**

**Table 3.1: User Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Constraint(s)** | **Description** |
| User ID | String | UNIQUE | Autogenerated for each user, when a new user registers in the website |
| Name | String | NOT NULL | Name of the user |
| Email-ID | String | Must be of the form <username>@<mailserver>.<domain>  UNIQUE, NOT NULL | E-Mail ID of the User |
| Password | String | NOT NULL | Password which the user uses to login to the website |
| IsAdmin | Boolean | NOT NULL | Indicates whether the user has admin privileges |

**3.3.2 Product Structure**

The details about the products stored in the Product table are shown in **Table 3.2.**

**Table 3.2: Product Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Constraint(s)** | **Description** |
| Product ID | String | UNIQUE | Autogenerated for each product, when it is added to the database |
| Name | String | NOT NULL | Name of the product |
| Slug | String | UNIQUE | Used to navigate to the product description page of that product |
| Image | String | NOT NULL | Path to the image of the eyewear |
| Brand | String | NOT NULL | Brand of the eyewear |
| Category | String | NOT NULL | Type of eyewear |
| Description | String | NOT NULL | Detailed description about the eyewear |
| Price | Number | NOT NULL | Price of the eyewear |
| Count in stock | Number | NOT NULL | Number of that particular eyewear currently in stock |
| Rating | Number | NOT NULL | Star rating out of 5 |
| Number of reviews | Number | NOT NULL | Total number of reviews for that eyewear |
| Reviews | Array | NOT NULL | Collection of usernames, with their corresponding rating and comments |

**3.3.3 Review Structure**

The details about the reviews stored in the Product table are shown in **Table 3.3.**

**Table 3.3: Review Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Constraint(s)** | **Description** |
| Name | String | NOT NULL | Name of the user |
| Rating | Number | NOT NULL | Star rating out of 5 |
| Comment | String | NOT NULL | Comment provided by the user |

**3.3.4 Order Structure**

The details about the orders stored in the Order table are shown in **Table 3.4.**

**Table 3.4: Order Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Constraint(s)** | **Description** |
| Order ID | String | UNIQUE | Autogenerated for each order, when it is added to the database |
| Products | Array | NOT NULL | Name of the user |
| Shipping Address | Object | NOT NULL | Delivery address for the order |
| User ID | String | NOT NULL | Comment provided by the user |
| Payment Method | String | NOT NULL | Way by which customer makes payment for the purchase |
| Items Price | Number | NOT NULL | Price of all the items the customer has purchased |
| Shipping Price | Number | NOT NULL | Shipping charges for the purchase |
| Tax Price | Number | NOT NULL | Tax for the purchase |
| Total Price | Number | NOT NULL | Total price including the items price, shipping charges and the tax price |
| IsPaid | Boolean | NOT NULL | Indicates whether the payment has been made or not |
| Payment Result | Object |  | Status of the payment |
| paidAt | Timestamp |  | Exact time at which payment is made |
| IsDelivered | Boolean |  | Indicates whether the product has been delivered or not |
| deliveredAt | Timestamp |  | Exact time at which order is delivered |

**3.3.5 Shipping Address Structure**

The details about the shipping stored in the Order table are shown in **Table 3.5.**

**Table 3.5: Shipping Address Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Constraint(s)** | **Description** |
| Full Name | String | NOT NULL | Full name of the customer |
| Address | String | NOT NULL | Street address and area |
| City | String | NOT NULL | Delivery City |
| State | String | NOT NULL | Delivery State |
| Postal Code | String | NOT NULL | PIN code of delivery address |
| Country | String | NOT NULL | Delivery country |

**3.3.6 Payment Structure**

The details about the payment stored in the Order table are shown in **Table 3.6.**

**Table 3.6: Payment Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Constraint(s)** | **Description** |
| Payment ID | String | UNIQUE | ID generated for each payment |
| Status | String | NOT NULL | Current state of the payment |

**3.4 LOW LEVEL DESIGN**

The following section illustrates the functionalities of the system. This includes login to the website, register to the website, adding products to the inventory, adding products to the cart, making payment, and placing the order.

**3.4.1 Login**

**Table 3.7** shows the login details of the application.

**Table 3.7 Login Details**

|  |  |
| --- | --- |
| **Files used** | login.component.html, login.component.ts, auth.service.ts, user.routes.ts |
| **Short Description** | Allows the user to login to the online eyewear store website |
| **Arguments** | E-Mail ID, Password |
| **Return** | Success/Failure in login |
| **Pre-Condition** | The user must have an account in the website |
| **Post-Condition** | The home page of the online eyewear store will be displayed |
| **Exception** | Invalid E-Mail ID or password |
| **Actors** | Admin, User |

**3.4.2 Register**

**Table 3.8** shows the login details of the application.

**Table 3.8 Register Details**

|  |  |
| --- | --- |
| **Files used** | register.component.html, register.component.ts, user.service.ts, user.routes.ts |
| **Short Description** | Allows the user to create a new account in the online eyewear store website |
| **Arguments** | Name, E-Mail ID, Password, Confirm Password |
| **Return** | Success/Failure in registration |
| **Pre-Condition** | The user must not have an account with the e-mail id |
| **Post-Condition** | The home page of the online eyewear store will be displayed |
| **Exception** | Invalid e-mail or Weak password |
| **Actors** | User |

**3.4.3 Add Product**

**Table 3.9** shows the add product details of the application.

**Table 3.9 Add Product Details**

|  |  |
| --- | --- |
| **Files used** | admin-product-edit.component.html, admin-product-edit.component.ts,  product.service.ts, product.routes.ts, admin.guard.ts |
| **Short Description** | Allows the admin to add new eyewear to the online eyewear store website |
| **Arguments** | Name, Slug, Price, Path to Image, Brand, Category, Count in stock, Description |
| **Return** | Success/Failure in adding new eyewear |
| **Pre-Condition** | The admin must be logged in to the website |
| **Post-Condition** | The eyewear inventory page of the online eyewear store will be displayed |
| **Exception** | Duplicate slug or Invalid path to image |
| **Actors** | Admin |

**3.4.4 Add to Cart**

**Table 3.10** shows the add to cart details of the application.

**Table 3.10 Add to Cart Details**

|  |  |
| --- | --- |
| **Files used** | cart.component.html, cart.component.ts, cart.service.ts, product.routes.ts |
| **Short Description** | Allows the user add eyewear to the cart in the online eyewear store website |
| **Arguments** | Eyewear name, quantity |
| **Return** | Success/Failure adding eyewear to cart |
| **Pre-Condition** | The user must be logged in to the website |
| **Post-Condition** | The cart page of the online eyewear store will be displayed |
| **Exception** | Product out of stock |
| **Actors** | User |

**3.4.5 Place Order**

**Table 3.11** shows the place order details of the application.

**Table 3.11 Place Order Details**

|  |  |
| --- | --- |
| **Files used** | place-order.component.html, place-order.component.ts, order.routes.ts, cart.service.ts, order.service.ts |
| **Short Description** | Allows the user to login to the online eyewear store website |
| **Arguments** | Eyewear name, Quantity, Shipping Address, Payment Method |
| **Return** | Success/Failure in placing order |
| **Pre-Condition** | The user must be logged in to the website |
| **Post-Condition** | The order summary page of the online eyewear store will be displayed |
| **Exception** | Invalid shipping address or Product out of stock |
| **Actors** | User |

**3.4.6 Make Payment**

**Table 3.12** shows the make payment details of the application.

**Table 3.12 Make Payment Details**

|  |  |
| --- | --- |
| **Files used** | payment-method.component.html, payment-method.component.ts,  order.routes.ts, auth.guard.ts, auth.service.ts |
| **Short Description** | Allows the user to make payment for an order placed in the online eyewear store website |
| **Arguments** | Order ID, Amount, PayPal E-Mail, PayPal Password |
| **Return** | Success/Failure in payment |
| **Pre-Condition** | The user must be logged in to the website |
| **Post-Condition** | The payment status for the order will be displayed to the user |
| **Exception** | Invalid PayPal E-Mail or Password |
| **Actors** | User |

**3.4.7 Search Eyewear**

**Table 3.13** shows the make search eyewear details of the application.

**Table 3.13 Search Eyewear Details**

|  |  |
| --- | --- |
| **Files used** | search.component.html, search.component.ts,  product.routes.ts, product.service.ts |
| **Short Description** | Allows the user to search an eyewear based on the name of the eyewear |
| **Arguments** | Eyewear Name |
| **Return** | Eyewear Details/Eyewear Not Found |
| **Pre-Condition** | The user must know the correct name of the eyewear that he is going to search for |
| **Post-Condition** | All the eyewear that contains the search term in its name is displayed to the user |
| **Exception** | No matching eyewear |
| **Actors** | User |

**3.4.8 Shop by Category**

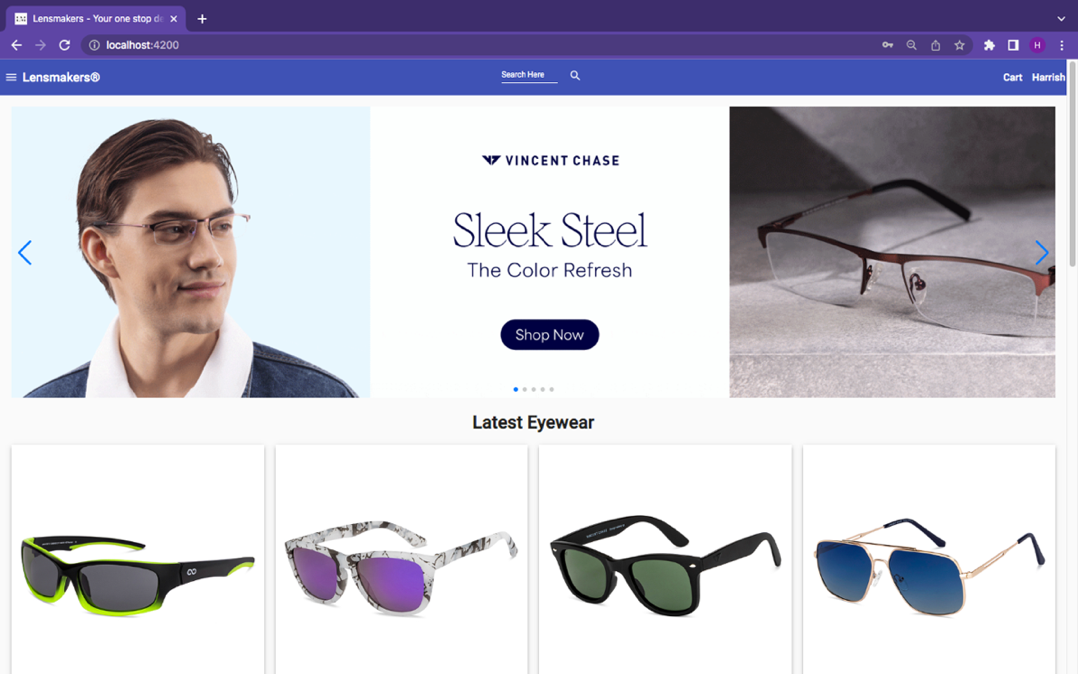
**Table 3.14** shows the shop by category details of the application.

**Table 3.14 Shop by Category Page Details**

|  |  |
| --- | --- |
| **Files used** | search.component.html, search.component.ts,  product.routes.ts, product.service.ts |
| **Short Description** | Allows the user to search an eyewear based on the category of the eyewear |
| **Arguments** | Eyewear Category |
| **Return** | List of Eyewear Details/Eyewear Not Found |
| **Pre-Condition** | The user must know the category or the class of the eyewear that he wishes to buy |
| **Post-Condition** | All the eyewear that belongs to the selected category is displayed to the user. |
| **Exception** | No matching eyewear in the selected category |
| **Actors** | User |

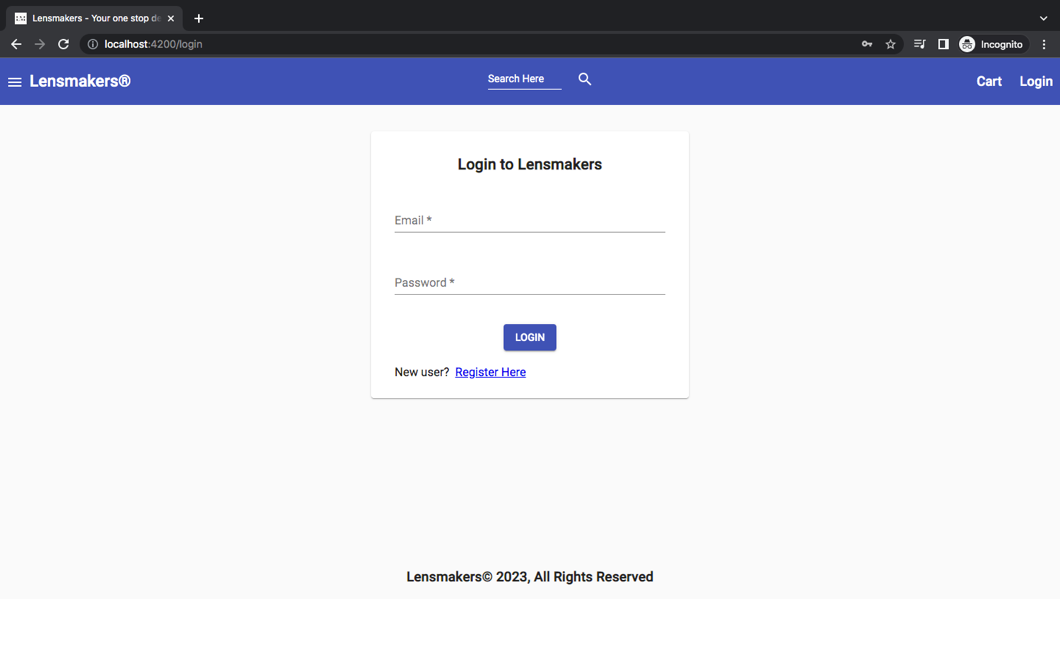
**3.5 USER INTERFACE DESIGN**

**3.5.1 Home Page**

 **Figure 3.2** provides the interface for home page of the website. The user can select form a wide range of eyewear available in the homepage. There is also a search feature to search for a particular eyewear from the website. Also, a filter by category option is available in the side menu, to filter out eyewear based on specific category, for example sunglasses or power glasses.

**Figure 3.2: Home Page of Online Eyewear Store**

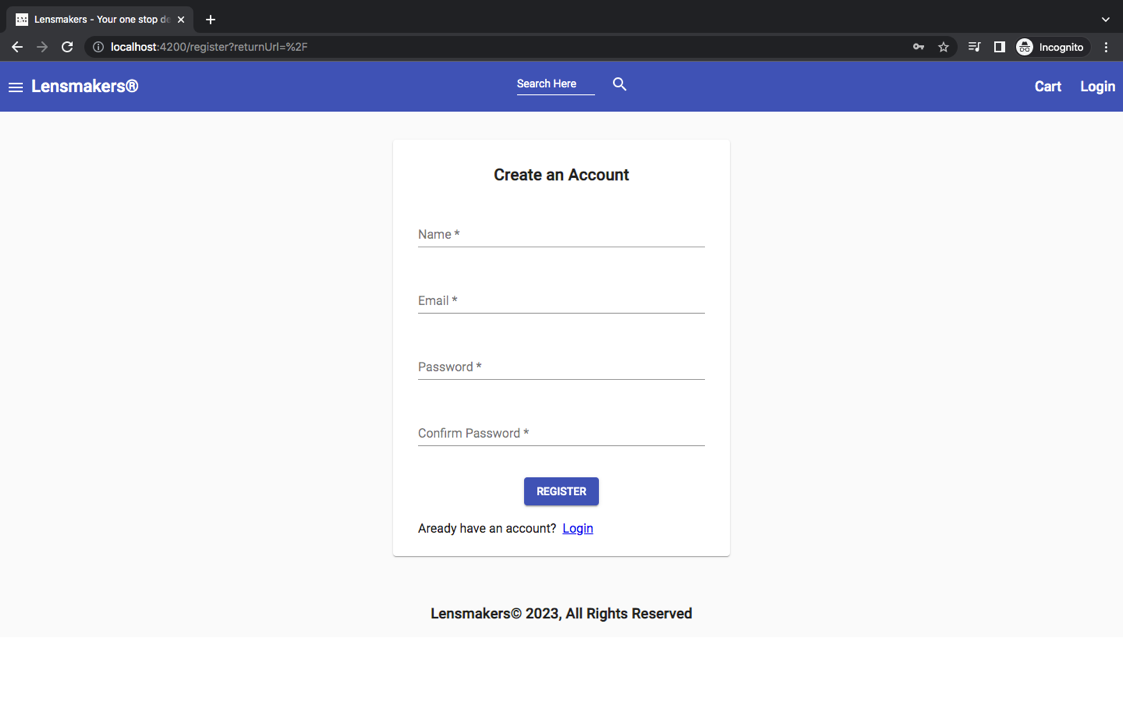
**3.5.2 Login Page**

 **Figure 3.3** provides the interface for login page of the website. If the login e-mail and password are valid, the user is allowed to login to the website. If the user is a new user there is also a register link embedded in the page which takes the user to the registration page.

**Figure 3.3: Login Page of Online Eyewear Store**

**3.5.3 Register Page**

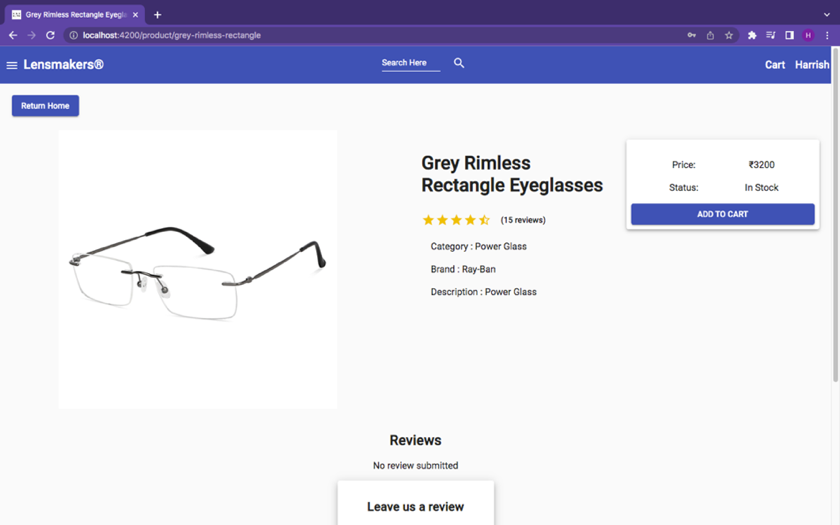
**Figure 3.4** provides the interface for new user registration page of the website. The user can create a new account in the website by using his/her e-mail address, provided the e-mail doesn’t already exist on the database. If the user already has an account, then the user is taken to the login page where the user can login with his/her existing e-mail and password.



**Figure 3.4: Register Page of Online Eyewear Store**

**3.5.4 Product Details Page**

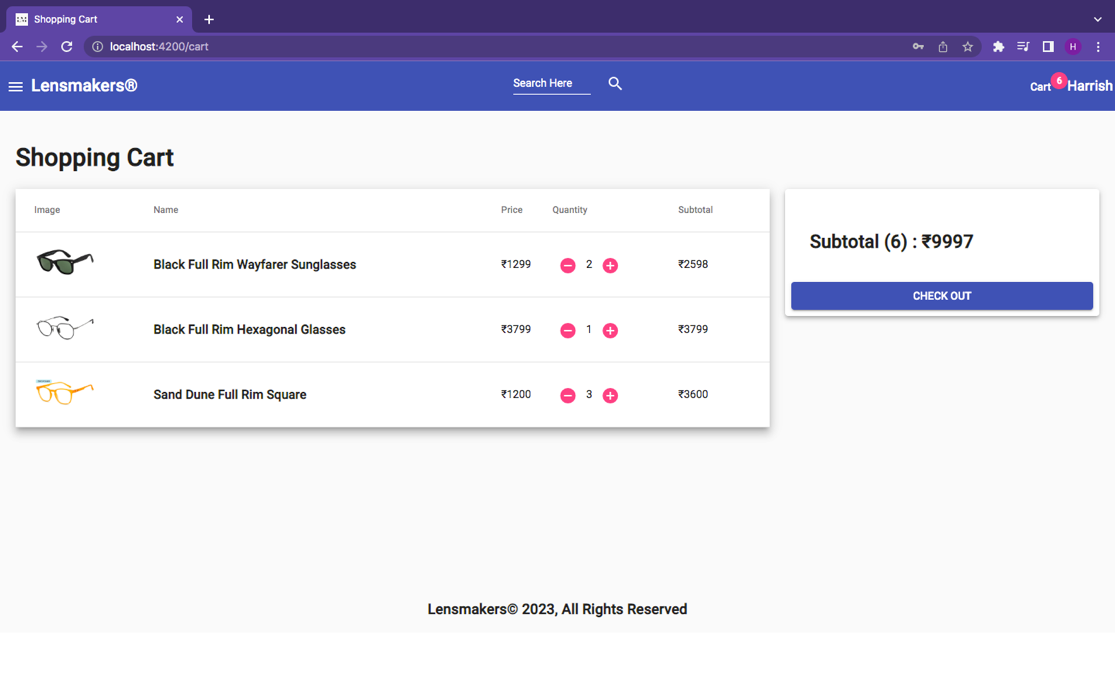
**Figure 3.5** provides the detailed information about an eyewear. It contains the name of the eyewear, category of the eyewear, brand of the eyewear, price of the eyewear, short description about the eyewear, stock status of the eyewear, star rating out of 5, number of reviews. The user can also submit a star rating and review for the eyewear. Add to cart button is provided in the website, which the user can use to add the product to his/her cart, provided the item is in stock, for purchase.



**Figure 3.5: Product Details Page**

**3.5.5 Cart Page**

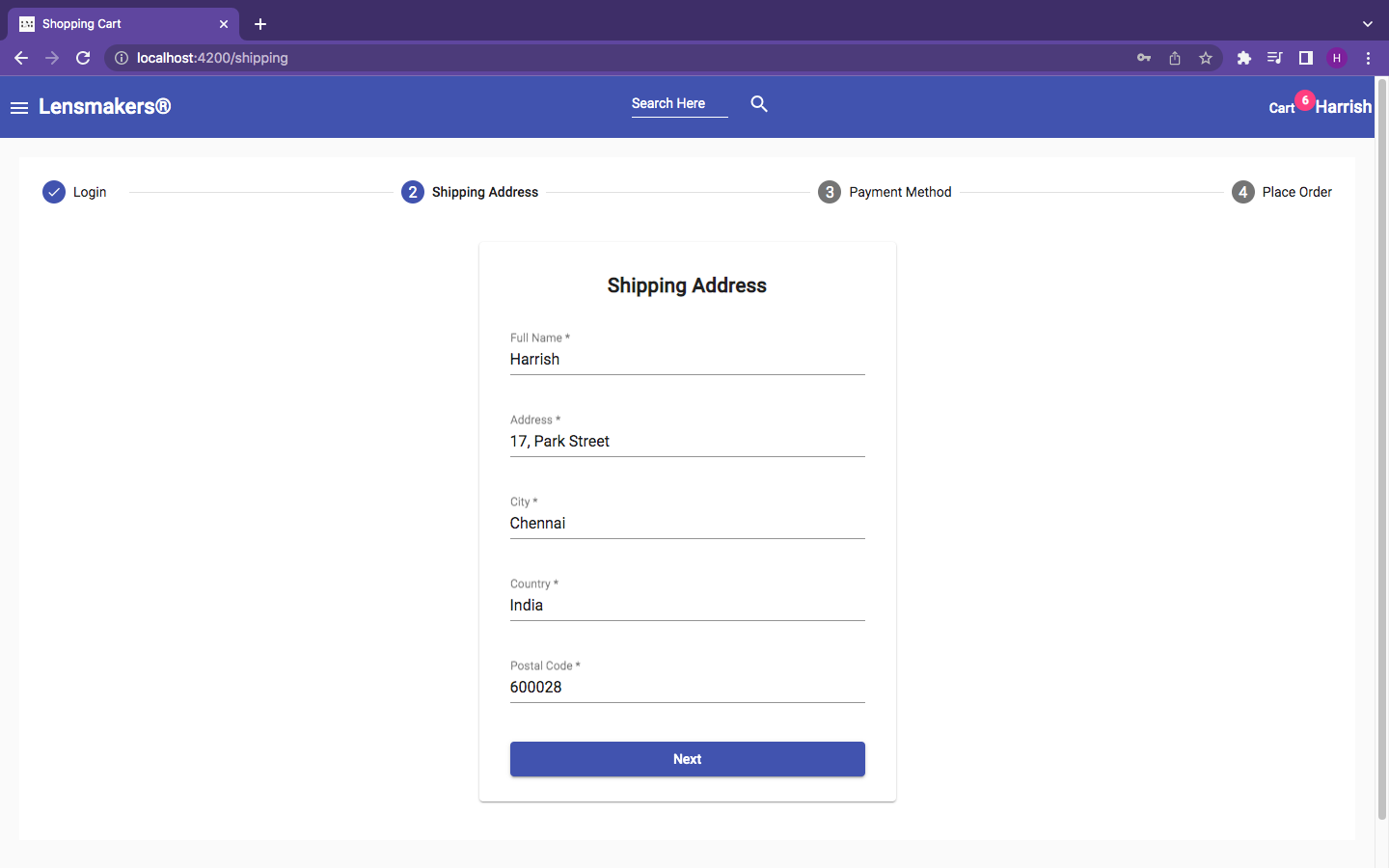
**Figure 3.6** provides the cart page of the website. This page displays all the eyewear that the user has added to his/her cart with the eyewear name, preview image and its price. The user can also increment or decrement the quantity of the eyewear in the cart page. After completing the process, the user can proceed to checkout.



**Figure 3.6: Cart Page**

**3.5.6 Shipping Address Page**

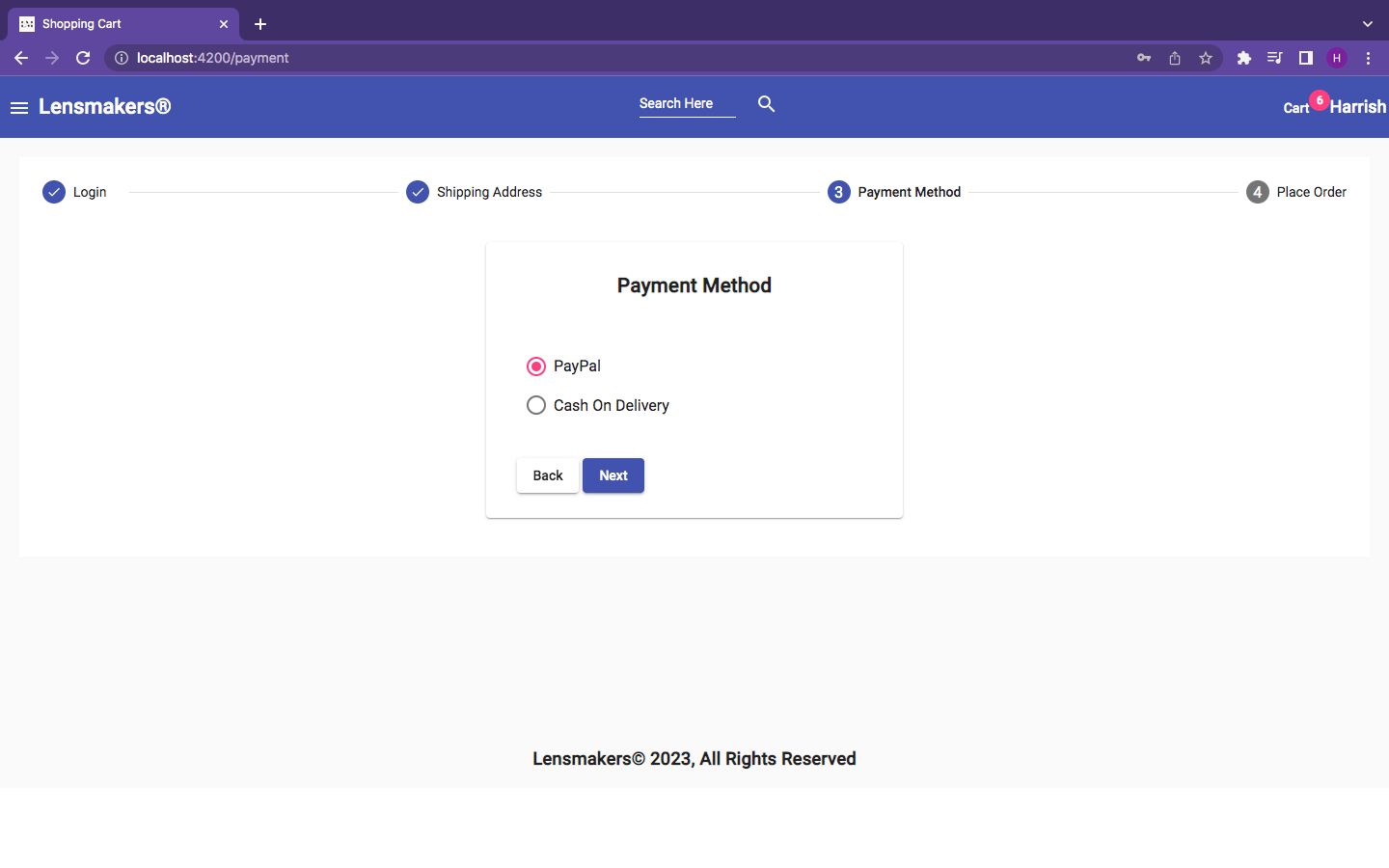
**Figure 3.7** provides the shipping address page of the website. The user must fill the billing name along with the complete address with postal code for the successful delivery of the package.



**Figure 3.7: Shipping Address Page**

**3.5.7 Payment Method Page**

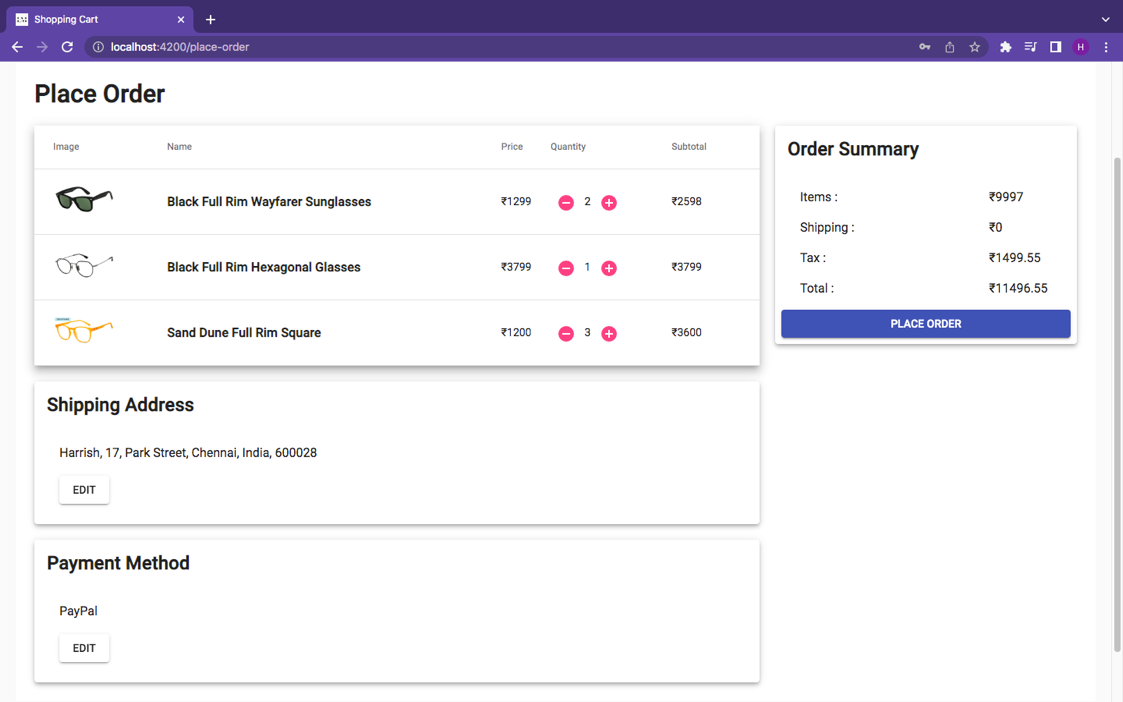
**Figure 3.8** provides the payment method page of the website. The user can choose to pay through the PayPal payment gateway or the Cash on Delivery option.



**Figure 3.8: Payment Method Page**

**3.5.8 Place Order Page**

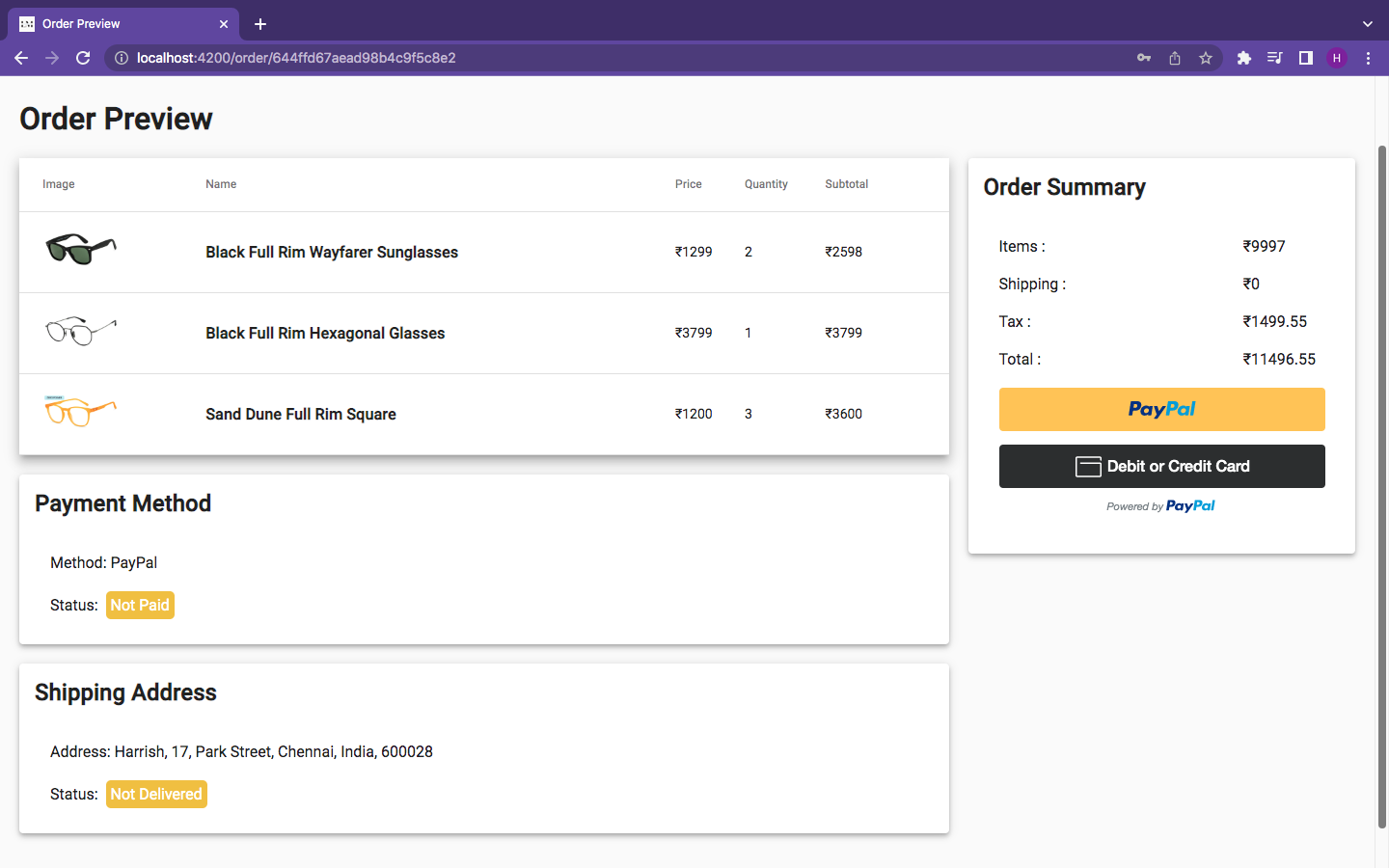
**Figure 3.9** provides the place order page of the website. The user can still increment or decrement the quantity of eyewear that he/she has finally checked out. The user can also edit the shipping address or the mode of payment. Once the user has confirmed all the details, he/she can click the place order button to confirm the order.



**Figure 3.9: Place Order Page**

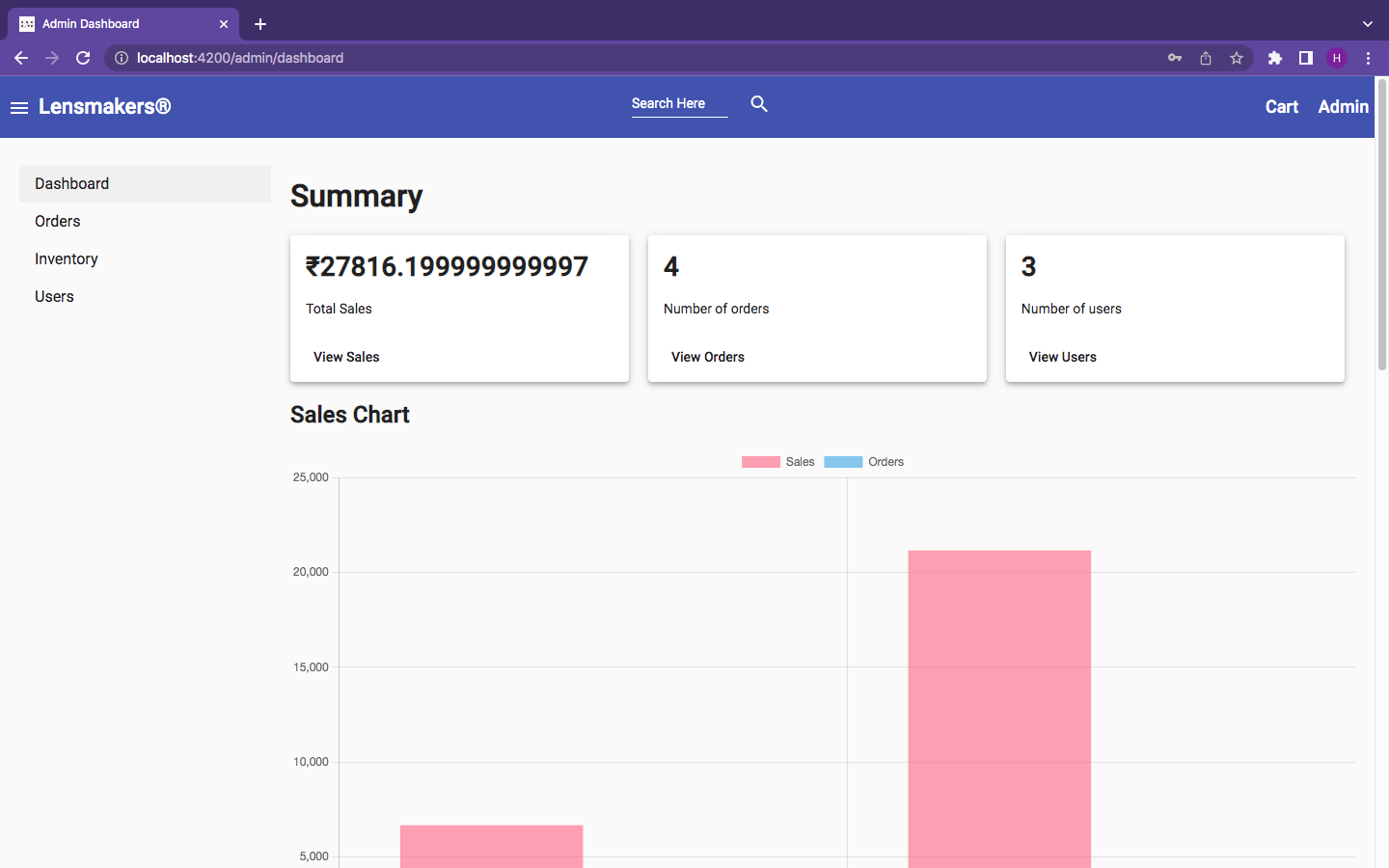
**3.5.9 Order Summary Page**

**Figure 3.10** provides the order summary page of the website. This page displays all the eyewear purchased by the user, with the preview image, eyewear name, price and quantity. It also displays the item price, tax and total amount of the purchase. The user can make payment through the PayPal Payment gateway or using his/her debit/credit card.

**Figure 3.10: Order Summary Page**

**3.5.10 Admin Dashboard Page**

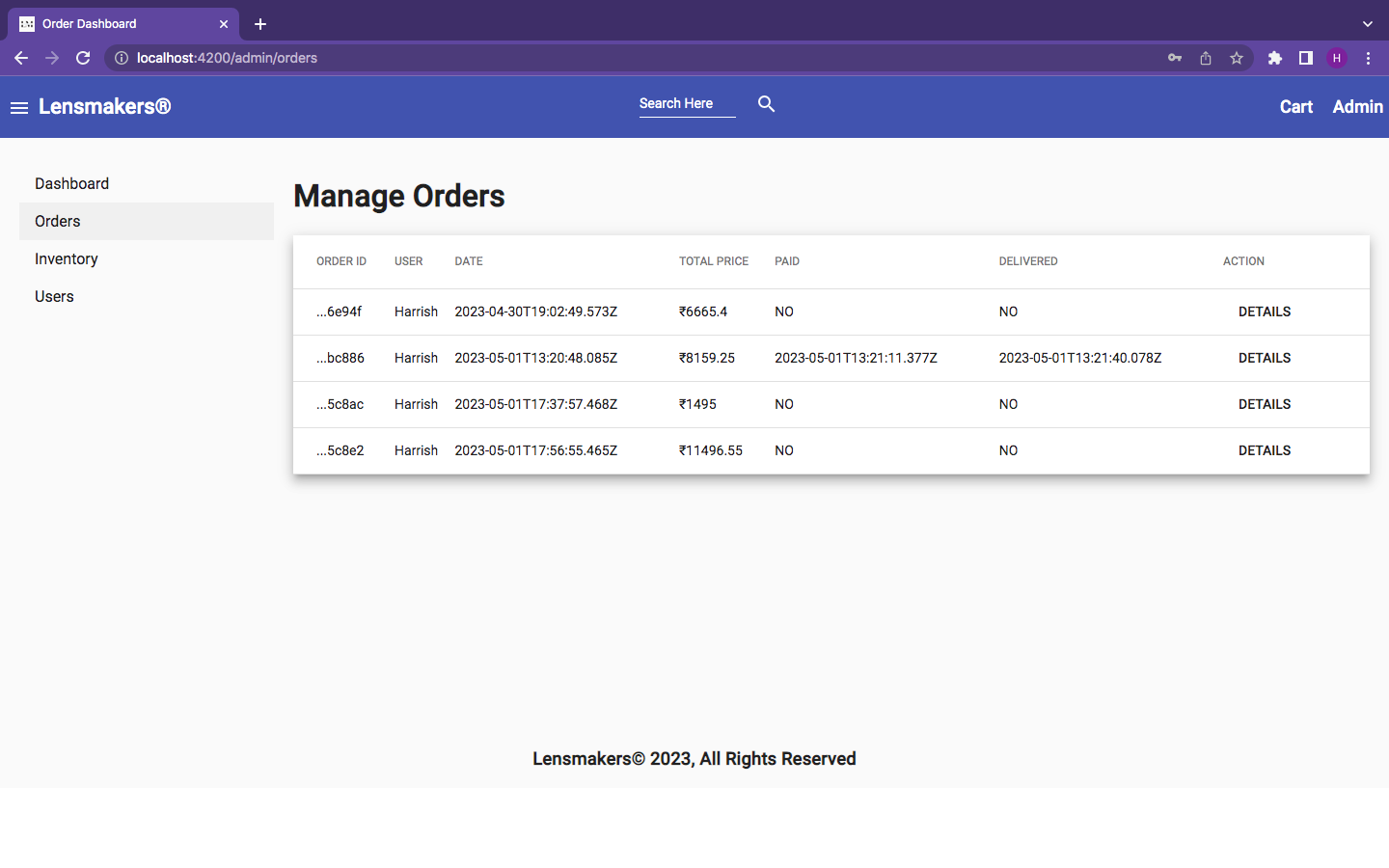
**Figure 3.11** provides the admin dashboard page of the website. This page helps the admin to track the sales, orders and users. The sales is tracked and visually plotted using a histogram and the count of each eyewear category in the inventory is visualized using a pie chart.



**Figure 3.11: Admin Dashboard Page**

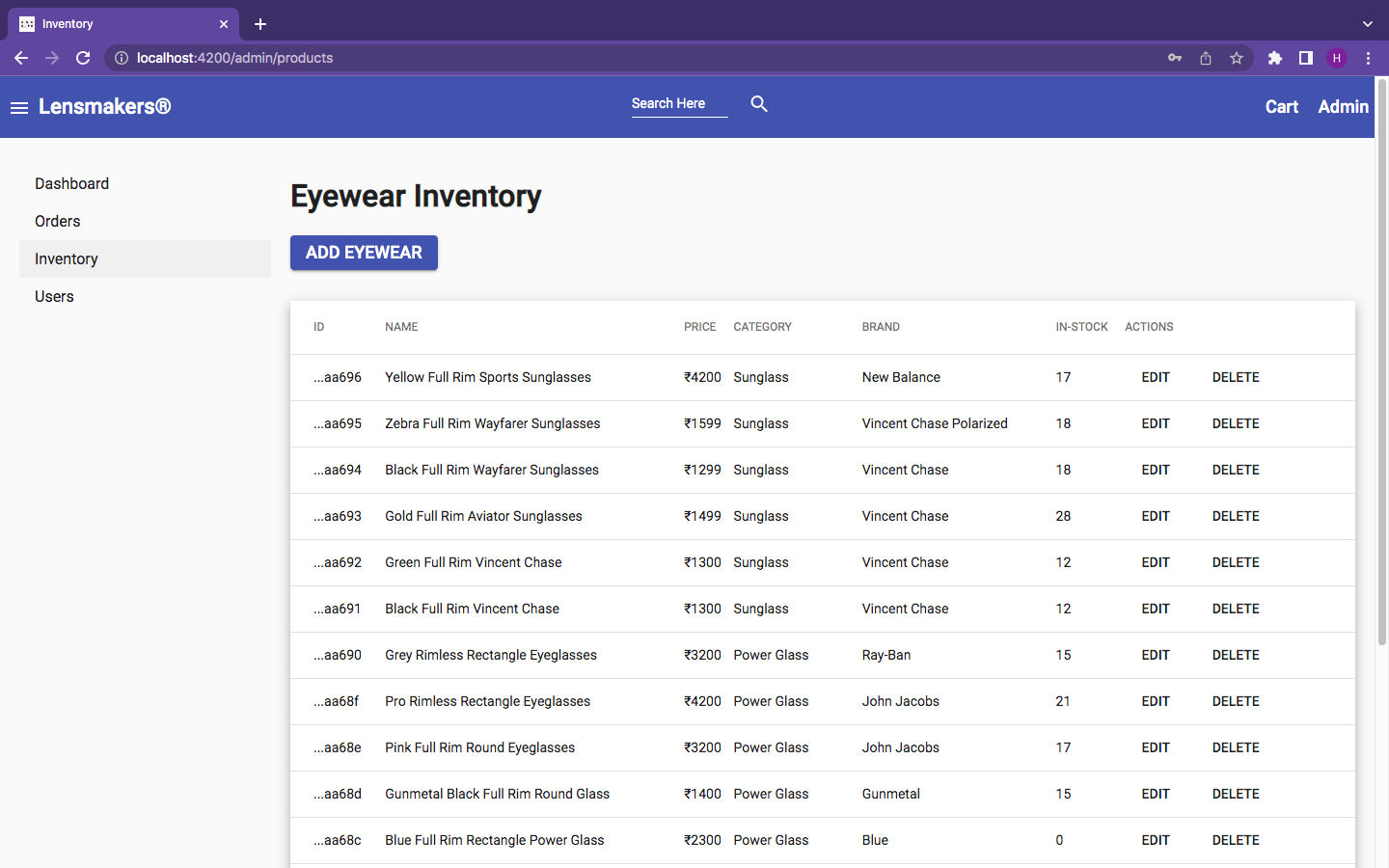
**3.5.11 Manage Orders Page**

**Figure 3.12** provides the order management page of the website. The admin can see and manage all the order details through this page. The admin can also mark a parcel as delivered by clicking the details button which takes the admin to the order details page.



**Figure 3.12: Manage Orders Page**

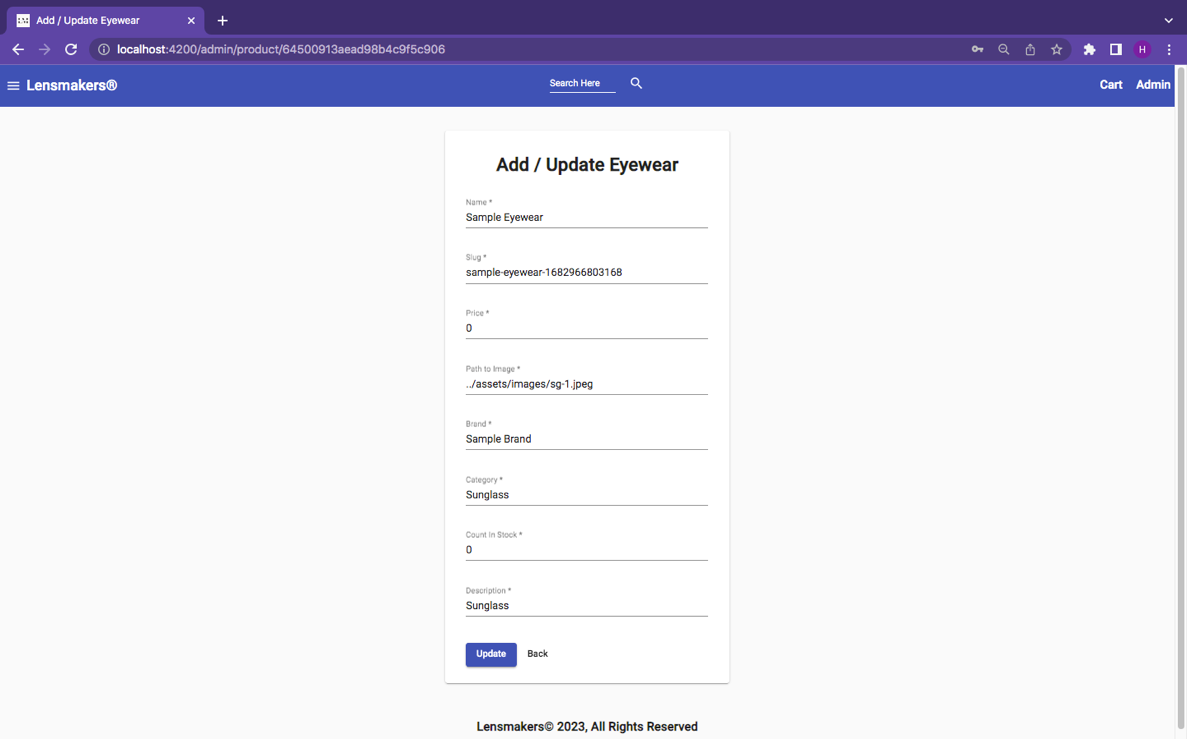
**3.5.12 Eyewear Inventory Page**

 **Figure 3.13** provides the eyewear inventory page of the website. It displays all the eyewear that are currently in the inventory of the eyewear store. The admin can add a new eyewear or modify or delete the existing eyewear details.

**Figure 3.13: Eyewear Inventory Page**

**3.5.13 Add Eyewear Page**

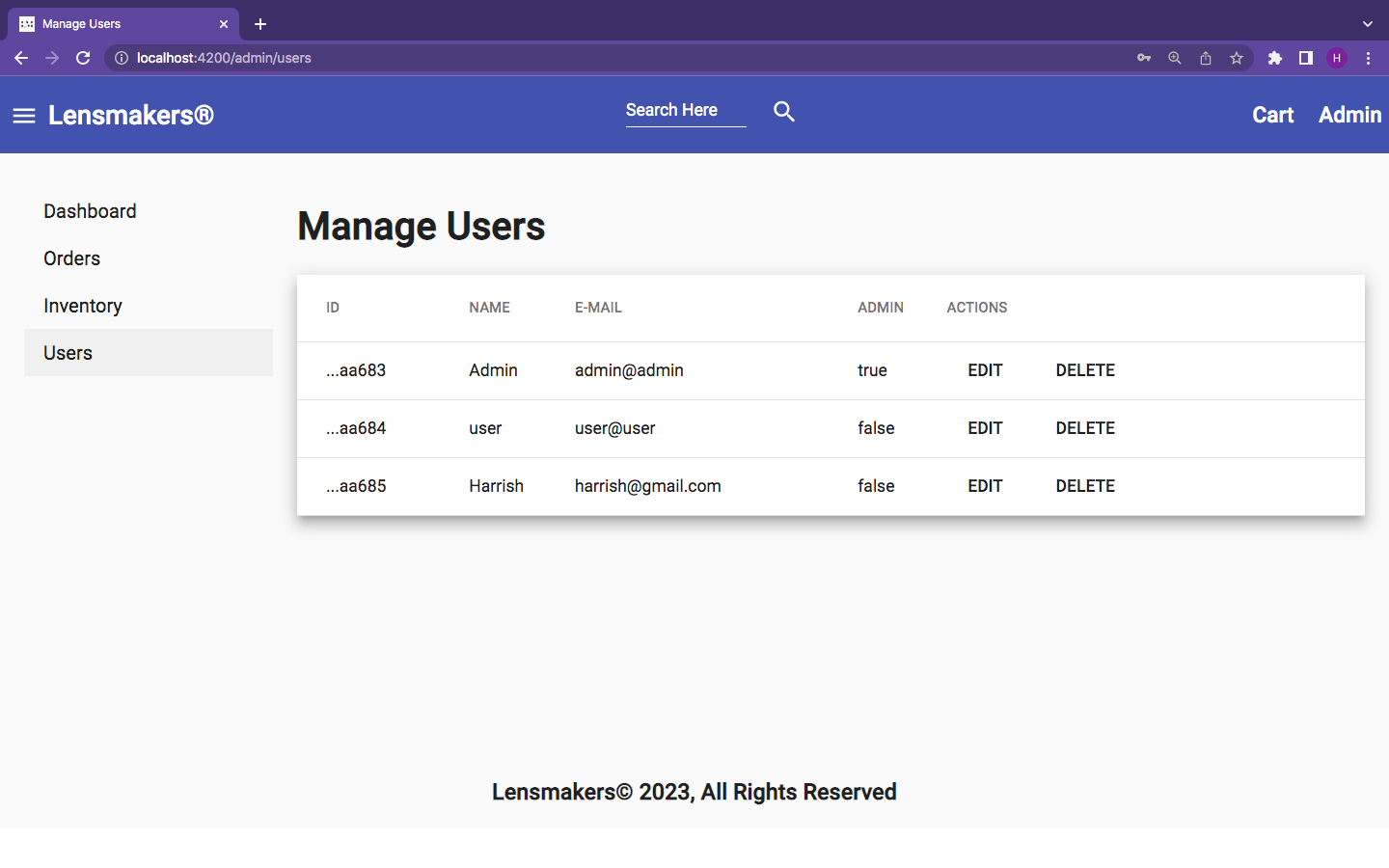
**Figure 3.14** provides the add eyewear page of the website. The Admin can add a new eyewear to the inventory or update the details of an existing eyewear. Once a new eyewear is added or updated, the change is reflected immediately in the home page of the website.



**Figure 3.14: Add Eyewear Page**

**3.5.14 Manage Users Page**

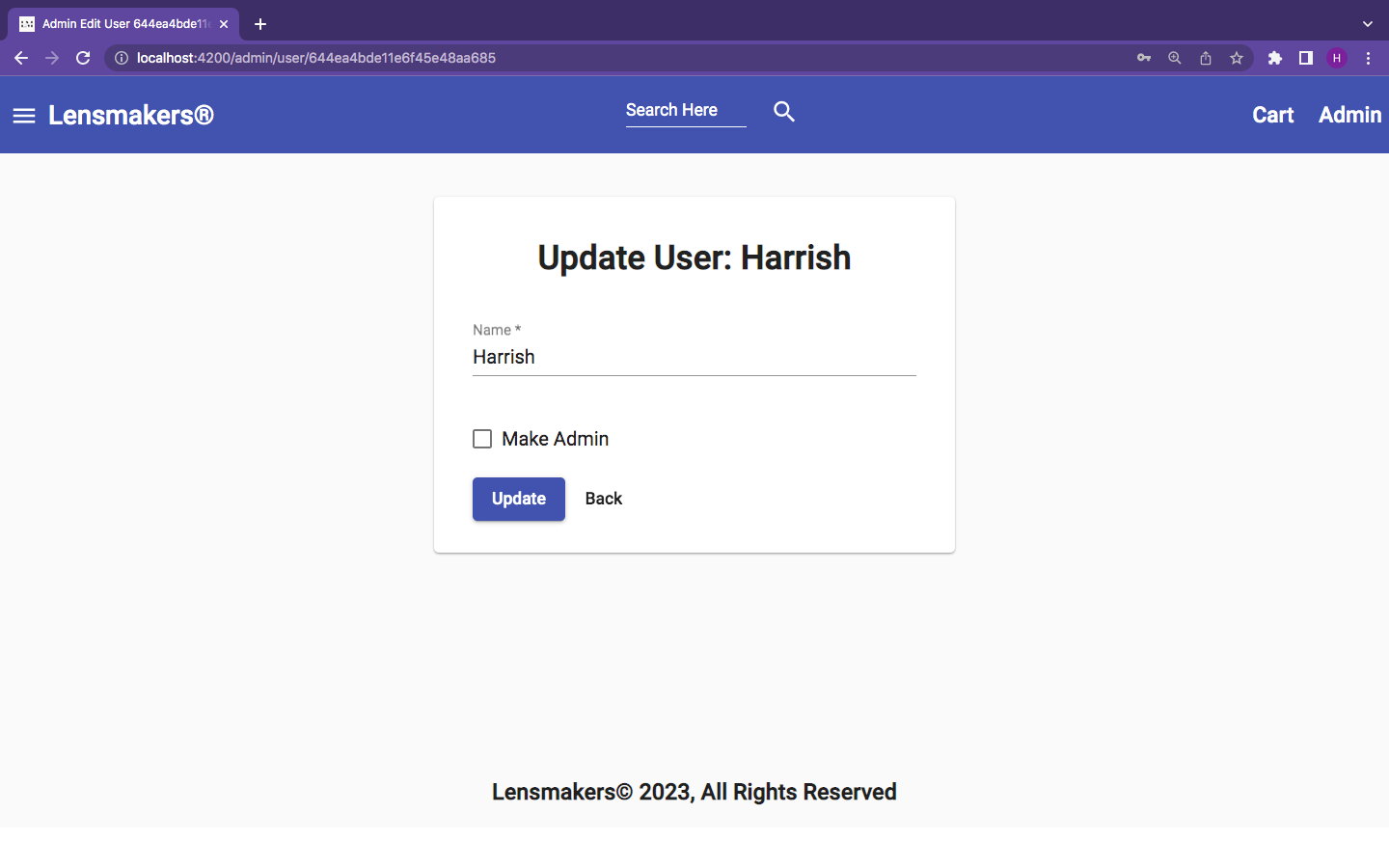
**Figure 3.15** provides the manage users page of the website. The Admin can edit user details or delete user from the website. The admin can also make another user an Admin. That is, the admin can provide admin privileges to other users also.



**Figure 3.15: Manage Users Page**

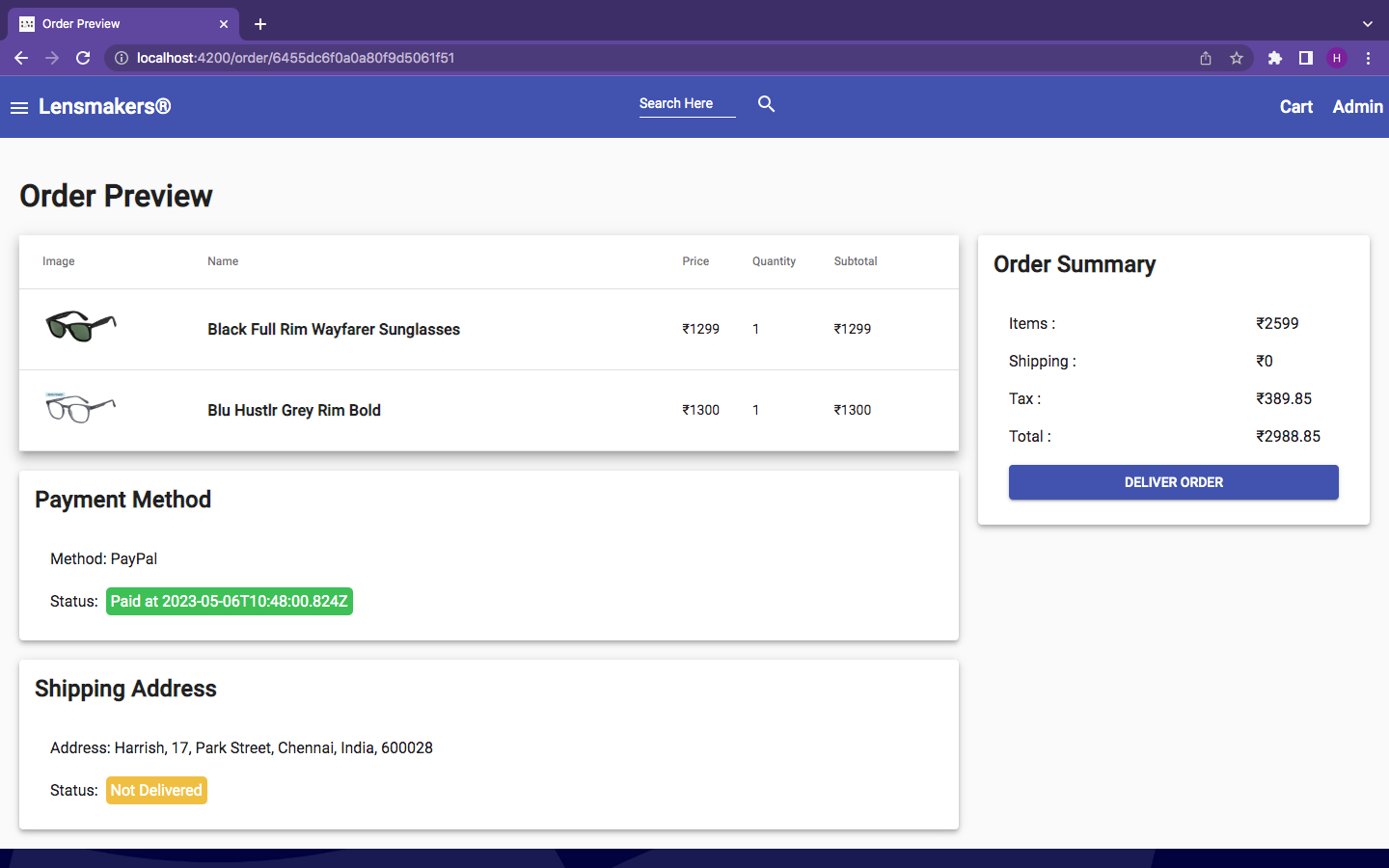
**3.5.15 Update User Page**

**Figure 3.16** provides the update user page of the website. The admin can change the name of the user and can provide a user with Admin privileges.



**Figure 3.16: Update User Page**

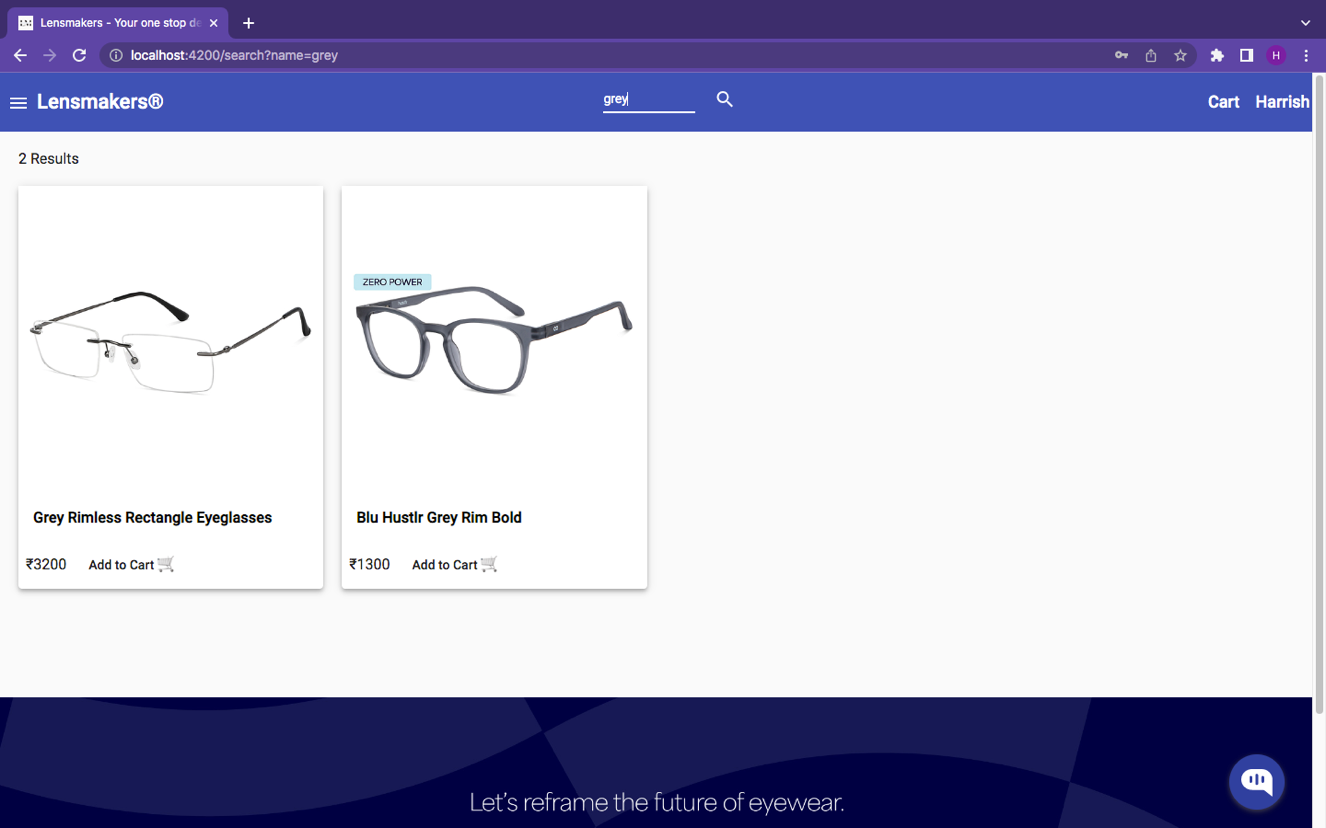
**3.5.16 Deliver Order Page**

** Figure 3.17** provides the deliver order page of the website. Once the payment has been made successfully by the user, and the order has been delivered, the admin can mark the order as delivered in the website.

**Figure 3.17: Deliver Order Page**

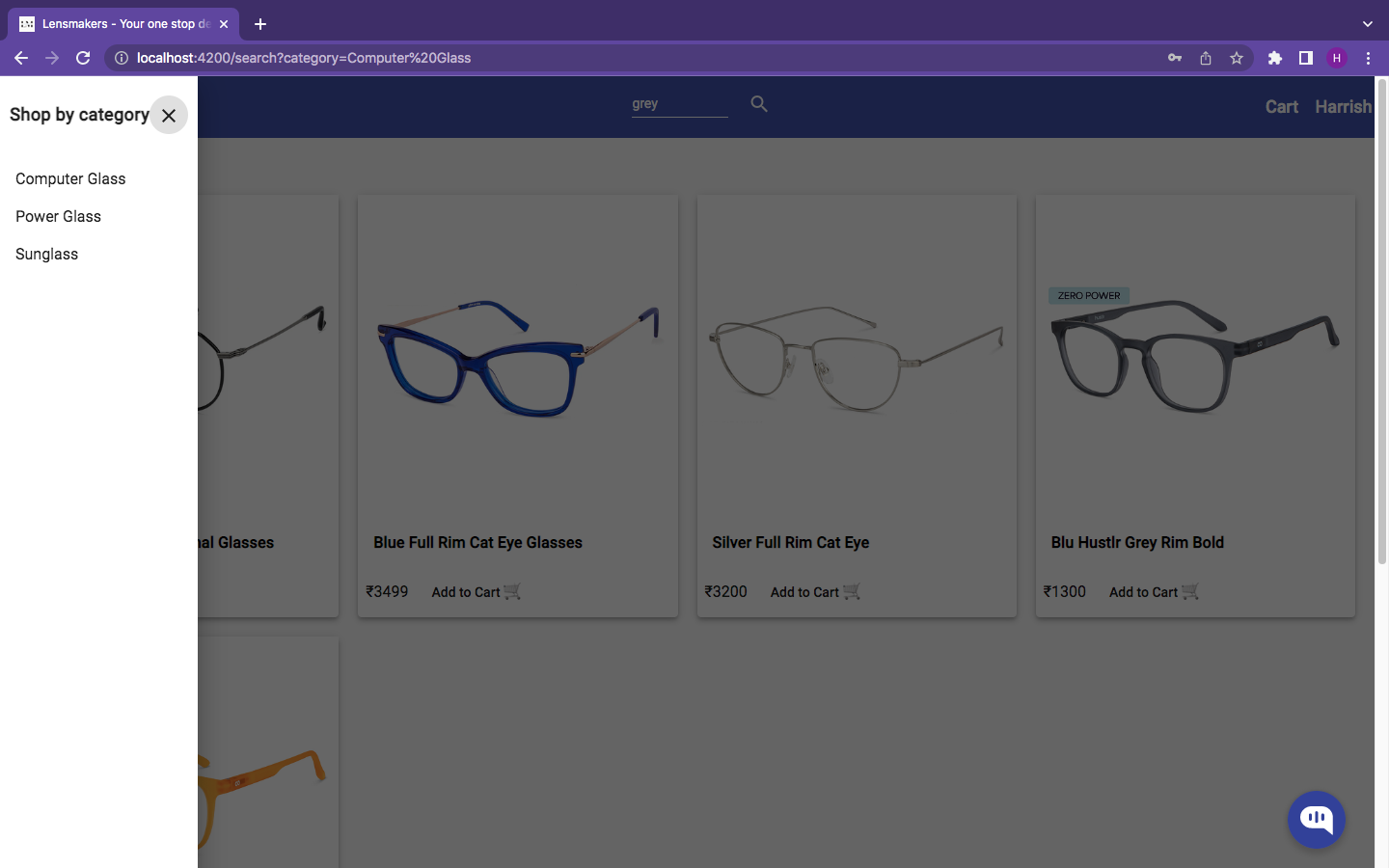
**3.5.17 Search Eyewear Page**

**Figure 3.18** provides the search eyewear page of the website. It enables the user to search for any particular eyewear based on the given eyewear name or a substring of the eyewear name.

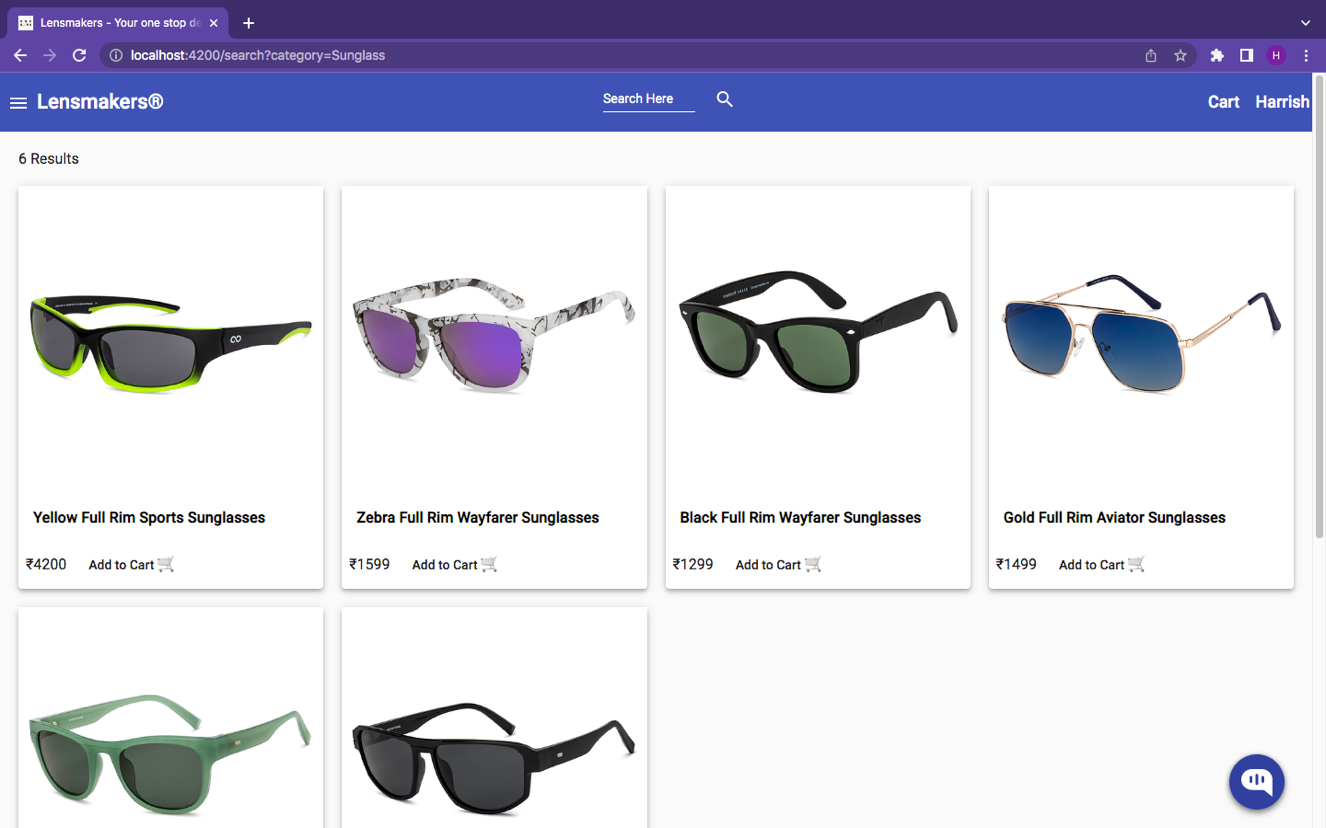


**Figure 3.18: Search Page**

**3.5.18 Shop by Category Page**

 **Figure 3.19** and **Figure 3.20** provides the search by category page and category-based filter results of the website. It enables the user to filter eyewear based on a selected category such as Sunglasses or Computer Glasses.

**Figure 3.19: Shop by Category Menu**

**Figure 3.20: Category Filter Page**

**Conclusion:**

Based on the user interface screenshots presented, it can be concluded that the design of the online eyewear store's interface is visually appealing and user-friendly. The use of high-quality product images, clear typography, and a consistent color scheme creates a cohesive and professional look. The layout of the homepage effectively showcases the store's featured products, while the top navigation bar and search bar make it easy for users to find what they are looking for. The product category pages are well-organized and provide useful filtering options, allowing users to quickly narrow down their search. The product pages themselves are well-designed, with detailed product descriptions and clear pricing information. The "Add to Cart" and "Check Out" buttons are prominently displayed, making it easy for users to complete their purchase. The shopping cart and checkout pages are also well-designed, with clear instructions and easy-to-fill forms. The use of a progress bar during checkout provides users with a clear understanding of the steps required to complete their purchase. Overall, the user interface screenshots demonstrate that the online eyewear store's interface is well-designed and user-friendly, creating a positive user experience.

**CHAPTER 4**

**SYSTEM IMPLEMENTATION**

**4.1 LOGIN IMPLEMENTATION**

The login credentials are obtained. If the credentials are OK, then the user is redirected to the homepage

POST email id, password

IF userid, password valid

RETURN homepage

ELSE

TOAST Invalid Credential

**4.2 REGISTER IMPLEMENTATION**

The form fields are obtained. If they are valid, then the user is added to the database.

POST requestFields

IF requestFields valid

RETURN added to db

ELSE

TOAST enter valid details

**4.3 ADD EYEWEAR IMPLEMENTATION**

The form fields are obtained. If they are valid, then the eyewear is added to the database.

POST requestFields

IF requestFields valid

RETURN added to db

ELSE

TOAST enter valid eyewear details

**4.4 ADD TO CART IMPLEMENTATION**

The form fields are obtained. If eyewear stock is available, then the eyewear is added to the cart.

POST eyewearId

IF eyewear in stock

RETURN added to cart

ELSE

TOAST item out of stock

**4.5 PLACE ORDER IMPLEMENTATION**

The required spectacles are selected by the customer. The total price is calculated and it is displayed.

POST list of eyewear

RETURN total amount

POST shipping address

IF shipping address valid

RETURN Payment page

ELSE

TOAST enter valid details

**4.6 MAKE PAYMENT IMPLEMENTATION**

The payment form fields are obtained. If they are valid, then the payment is made successfully.

POST requestFields

IF requestFields valid

RETURN order successful page

ELSE

TOAST enter valid details

**4.7 SEARCH EYEWEAR IMPLEMENTATION**

The search form fields are obtained. If they are valid, then the eyewear is searched and the result is shown successfully.

POST requestField

IF requestField valid

RETURN all matching eyewear by name

ELSE

TOAST eyewear not found

**4.8 SHOP BY CATEGORY IMPLEMENTATION**

The search by category form fields is obtained. If they are valid, then the all the eyewear of the selected category will be displayed on the website.

POST requestField

IF requestField valid

RETURN all matching eyewear by category

ELSE

TOAST eyewear not found

**CHAPTER 5**

**RESULTS AND DISCUSSION**

**5.1 TEST CASES AND RESULTS**

**5.1.1 Test Cases and Results for Login function:**

The Table 5.1, Table 5.2 shows that the possible test data for the both positive and negative test case given below, if the user is already having account then the output is true otherwise false.

**Table 5.1: Positive Test Case and result for Login**

|  |  |
| --- | --- |
| **Test Case ID** | TC1 |
| **Test Case Description** | It tests whether the given login details are valid or not |
| **Test Data** | test@gmail.com, test@123 |
| **Expected Output** | TRUE |
| **Result** | PASS |

**Table 5.2: Negative Test Case and result for Login**

|  |  |
| --- | --- |
| **Test Case ID** | TC2 |
| **Test Case Description** | It tests whether the given login details are valid or not |
| **Test Data** | test@gmail.com |
| **Expected Output** | FALSE |
| **Result** | PASS |

**5.1.2 Test Case and Results for Register Function:**

The Table 5.3, Table 5.4 shows that the possible test data for both positive and negative test case given below, if the user is already having account then the output is false otherwise true.

**Table 5.3: Positive Test Case and result for Register**

|  |  |
| --- | --- |
| **Test Case ID** | TC3 |
| **Test Case Description** | It tests whether the given registration details are valid or not |
| **Test Data** | Harrish, test@gmail.com, test@123, test@123 |
| **Expected Output** | TRUE |
| **Result** | PASS |

**Table 5.4: Negative Test Case and result for Register**

|  |  |
| --- | --- |
| **Test Case ID** | TC4 |
| **Test Case Description** | It tests whether the given registration details are valid or not |
| **Test Data** | Harrish, test@gmail.com, test@123, test@123 |
| **Expected Output** | FALSE |
| **Result** | PASS |

**5.1.3 Test Case and Results for Add Eyewear:**

The Table 5.5, Table 5.6 shows that the possible test data for both positive and negative test case given below, if the value of the add eyewear fields are valid then the output is true otherwise false.

**Table 5.5: Positive Test Case and result for Add Eyewear**

|  |  |
| --- | --- |
| **Test Case ID** | TC5 |
| **Test Case Description** | It tests whether the new eyewear can be added to the online eyewear store |
| **Test Data** | Sample-eyewear,2500, ../assets/images/sg-1.jpg/, Sample-Brand, Sunglass, 20 |
| **Expected Output** | TRUE |
| **Result** | PASS |

**Table 5.6: Negative Test Case and result for Add Eyewear**

|  |  |
| --- | --- |
| **Test Case ID** | TC6 |
| **Test Case Description** | It tests whether the new eyewear can be added to the online eyewear store |
| **Test Data** | Sample-eyewear-2, ../assets/images/sg-1.jpg/, Sample-Brand-2, Sunglass, 20 |
| **Expected Output** | FALSE |
| **Result** | PASS |

**5.1.4 Test Case and Results for Add to Cart:**

The Table 5.7, Table 5.8 shows that the possible test data for both positive and negative test case given below, if the value of the add to cart fields are valid then the output is true otherwise false.

**Table 5.7: Positive Test Case and result for Add Eyewear**

|  |  |
| --- | --- |
| **Test Case ID** | TC7 |
| **Test Case Description** | It tests whether the eyewear can be added to cart or not |
| **Test Data** | Sample-Eyewear, 10 |
| **Expected Output** | TRUE |
| **Result** | PASS |

**Table 5.8: Negative Test Case and result for Add Eyewear**

|  |  |
| --- | --- |
| **Test Case ID** | TC8 |
| **Test Case Description** | It tests whether the eyewear can be added to cart or not |
| **Test Data** | Sample-Eyewear, 10000 |
| **Expected Output** | FALSE |
| **Result** | PASS |

**5.1.5 Test Case and Results for Place Order**

The Table 5.9, Table 5.10 shows that the possible test data for both positive and negative test case given below, if the order fields are valid then the output is true, else the output is false.

**Table 5.9: Negative Test Case and result for Place Order**

|  |  |
| --- | --- |
| **Test Case ID** | TC9 |
| **Test Case Description** | It tests whether the given place order details are valid or not |
| **Test Data** | Sample-Eyewear, 3, 123 Park Street, Chennai, India, 600-028, PayPal |
| **Expected Output** | TRUE |
| **Result** | PASS |

**Table 5.10: Negative Test Case and result for Place Order**

|  |  |
| --- | --- |
| **Test Case ID** | TC10 |
| **Test Case Description** | It tests whether the given place order details are valid or not |
| **Test Data** | Sample-Eyewear, 3, Chennai, India, 111-111, PayPal |
| **Expected Output** | FALSE |
| **Result** | PASS |

**5.1.6 Test Case and Results for Make Payment:**

The Table 5.11, Table 5.12 shows that the possible test data for both positive and negative test case given below, if the payment details are valid then the output is true otherwise false.

**Table 5.11: Positive Test Case and result for Make Payment**

|  |  |
| --- | --- |
| **Test Case ID** | TC11 |
| **Test Case Description** | It tests whether payment can be made for a given order |
| **Test Data** | sb-uq0jx25570769@personal.example.com, @#$$tytg# |
| **Expected Output** | TRUE |
| **Result** | PASS |

**Table 5.12: Negative Test Case and result for Make Payment**

|  |  |
| --- | --- |
| **Test Case ID** | TC12 |
| **Test Case Description** | It tests whether payment can be made for a given order |
| **Test Data** | harrish-123@personal.example.com, harrish@#$% |
| **Expected Output** | FALSE |
| **Result** | PASS |

**5.1.7 Test Case and Results for Search Eyewear:**

The Table 5.13, Table 5.14 shows that the possible test data for both positive and negative test case given below, if the eyewear name is valid then the matching eyewear is returned otherwise the output is false.

**Table 5.13: Positive Test Case and result for Search Eyewear**

|  |  |
| --- | --- |
| **Test Case ID** | TC13 |
| **Test Case Description** | It searches for the given eyewear based on matching with the eyewear name |
| **Test Data** | blu hustlr sport |
| **Expected Output** | TRUE |
| **Result** | PASS |

**Table 5.14: Negative Test Case and result for Search Eyewear**

|  |  |
| --- | --- |
| **Test Case ID** | TC14 |
| **Test Case Description** | It searches for the given eyewear based on matching with the eyewear name |
| **Test Data** | xyzabcrty |
| **Expected Output** | FALSE |
| **Result** | PASS |

**5.1.8 Test Case and Results for Shop by Category:**

The Table 5.15, Table 5.16 shows that the possible test data for both positive and negative test case given below, if the category details are valid then all the matching eyewear is returned otherwise the output is false.

**Table 5.15: Positive Test Case and result for Shop by Category**

|  |  |
| --- | --- |
| **Test Case ID** | TC15 |
| **Test Case Description** | It retrieves all the eyewear of the selected eyewear category and displays it to the user |
| **Test Data** | Sunglass |
| **Expected Output** | TRUE |
| **Result** | PASS |

**Table 5.16: Negative Test Case and result for Shop by Category**

|  |  |
| --- | --- |
| **Test Case ID** | TC16 |
| **Test Case Description** | It retrieves all the eyewear of the selected eyewear category and displays it to the user |
| **Test Data** | xyzabcrty |
| **Expected Output** | FALSE |
| **Result** | PASS |

**CHAPTER 6**

**CONCLUSION AND FUTURE ENHANCEMENT(S)**

In conclusion, the Online Eyewear Store project has been successfully implemented and offers a user-friendly and efficient platform for customers to purchase eyewear online. The website offers a wide range of products, from sunglasses to prescription glasses, and provides customers with detailed product information, including images, specifications, and customer reviews. The payment and checkout process is simple and secure, with multiple payment options available to customers. The project has been developed using modern web technologies, ensuring that the website is responsive, fast, and accessible across different devices. While the Online Eyewear Store project has been successfully implemented, there are several potential areas for improvement and future enhancements.

These include a virtual try-on feature that enables customers to see how different glasses frames look on their face would be an excellent addition to the website. This feature could be implemented using computer vision and machine learning technologies. The website could implement a personalized recommendation system that suggests glasses frames to customers based on their previous purchases, browsing history, and preferences. The website could integrate with social media platforms to enable customers to share their purchases with friends and family on social media. The website could offer additional payment options, such as cryptocurrency or mobile payment methods, to cater to a broader range of customers. Overall, the Online Eyewear Store project has been successful, but there is always room for improvement and enhancement to provide a better user experience for customers.

**APPENDIX – A**

**SYSTEM REQUIREMENTS**

**HARDWARE REQUIREMENT:**

|  |  |  |
| --- | --- | --- |
| Server | : | 10 TB Storage and 128 GB of Memory |
| Processor | : | Intel Core i3 10th Gen |
| Memory | : | 8 GB |
| Storage | : | 256 GB of Disk Storage |
| Network | : | 100 Mbps Bandwidth |

**SOFTWARE REQUIREMENT:**

|  |  |  |
| --- | --- | --- |
| Operating System | : | Any |
| DBMS | : | MongoDB |
| IDE used | : | Visual Studio Code 2023 |
| NodeJS Version | : | 18.16.0 |
| Angular CLI Version | : | 15.2.4 |

**APPENDIX – B**

**SOURCE CODE**

**index.html**

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Lensmakers - Your one stop destination for all kinds of eyewear</title>

<base href="/">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/x-icon" href="favicon.ico">

</head>

<body>

<app-root></app-root>

</body>

</html>

**app.component.html**

<mat-sidenav-container

class="example-container"

(backdropClick)="sidenav.close()"

>

<mat-sidenav

class="sidenav"

#sidenav

(keydown.escape)="sidenav.close()"

disableClose >

<div class="header" fxLayout="row" fxLayoutAlign="space-between center">

<div>

<h3>Shop by category</h3>

</div>

<div>

<button

(click)="sidenav.close()"

mat-icon-button

aria-label="close menu icon" >

<mat-icon>close</mat-icon>

</button>

</div>

</div>

<mat-nav-list>

<a

(click)="sidenav.close()"

mat-list-item

[routerLink]="['/search']"

[queryParams]="{ category: category }"

\*ngFor="let category of categories"

>

{{ category }}

</a>

</mat-nav-list>

</mat-sidenav>

<mat-sidenav-content>

<div id="app">

<mat-toolbar color="primary">

<button

(click)="sidenav.open()"

mat-icon-button

aria-label="open menu icon"

>

<mat-icon>menu</mat-icon>

</button>

<a mat-button routerLink="/" style="font-size:22px;">Lensmakers®️</a>

<div class="example-spacer">

<form [formGroup]="searchForm" (ngSubmit)="onSubmit()">

<mat-form-field floatLabel="never">

<input

matInput

placeholder="Search Here"

formControlName="name"

name="query"

/>

</mat-form-field>

<button class="hide-small-screen" mat-button color="secondary">

<mat-icon>search</mat-icon>

</button>

</form>

</div>

<a mat-button routerLink="cart">

<span

\*ngIf="itemsCount > 0"

[matBadge]="itemsCount"

matBadgePosition="after"

matBadgeOverlap="false"

matBadgeColor="accent"

>Cart</span>

<span style="font-size:18px" \*ngIf="itemsCount === 0">Cart</span>

</a>

<a mat-button \*ngIf="!currentUser" routerLink="login" style="font-size:18px">Login</a>

<ng-container \*ngIf="currentUser">

<button mat-button [matMenuTriggerFor]="menu" style="font-size:18px">

{{ currentUser.name }}

</button>

<mat-menu #menu="matMenu">

<a mat-menu-item routerLink="profile">Profile</a>

<a mat-menu-item routerLink="order-history">Order Hisotry</a>

<a \*ngIf="currentUser" mat-menu-item routerLink="admin/dashboard"

>Admin</a

>

<button \*ngIf="currentUser" mat-menu-item (click)="logout()">

Logout

</button>

</mat-menu>

</ng-container>

</mat-toolbar>

<div class="container">

<router-outlet></router-outlet>

</div>

<div class="footer"><h3>Lensmakers© 2023, All Rights Reserved</h3></div>

</div>

</mat-sidenav-content>

</mat-sidenav-container>

**app.component.ts**

import { Component, ViewChild } from '@angular/core';

import { Router } from '@angular/router';

import { UserInfo } from './models';

import { AuthService } from './services/auth.service';

import { CartService } from './services/cart.service';

import { MatSidenav } from '@angular/material/sidenav';

import { ProductService } from './services/product.service';

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css'],

})

export class AppComponent {

@ViewChild('sidenav')

sidenav!: MatSidenav;

categories: [] = [];

searchForm: FormGroup;

title = 'Lensmakers - Your one stop destination for all kinds of eyewear';

itemsCount: number = 0;

currentUser: UserInfo | null = null;

constructor(

private formBuilder: FormBuilder,

private router: Router,

private cartService: CartService,

private authService: AuthService,

private productService: ProductService

) {

this.searchForm = this.formBuilder.group({

name: [''],

});

}

ngOnInit() {

this.authService.currentUser.subscribe((x) => (this.currentUser = x));

this.cartService.currentCart.subscribe(

(x) => (this.itemsCount = x.itemsCount)

);

this.productService

.getCategories()

.subscribe((categories) => (this.categories = categories));

}

onSubmit() {

this.router.navigate(['/search'], {

queryParams: { name: this.searchForm.controls.name.value },

});

}

logout() {

this.authService.logout();

this.router.navigate(['/login']);

}

}

**home.component.html**

<div \*ngIf="loading; else result">

<mat-spinner></mat-spinner>

</div>

<ng-template #result>

<swiper

[autoHeight]="true"

[autoplay]="true"

[loop]="true"

[navigation]="true"

[pagination]="true"

>

<ng-template swiperSlide>

<img src="../assets/images/banner1.webp" alt="banner 1" />

</ng-template>

<ng-template swiperSlide>

<img src="../assets/images/banner2.webp" alt="banner 2" />

</ng-template>

<ng-template swiperSlide>

<img src="../assets/images/banner3.jpeg" alt="banner 3" />

</ng-template>

<ng-template swiperSlide>

<img src="../assets/images/banner4.webp" alt="banner 4" />

</ng-template>

<ng-template swiperSlide>

<img src="../assets/images/banner5.webp" alt="banner 5" />

</ng-template>

</swiper>

<h1 style="text-align: center;">Latest Eyewear</h1>

<div

\*ngIf="!error"

fxLayout="row wrap"

fxLayoutGap="20px grid"

fxLayoutGap.xs="10px grid"

>

<div

fxFlex="25%"

fxFlex.sm="33%"

fxFlex.xs="100%"

\*ngFor="let product of products"

>

<mat-card class="mat-elevation-z4">

<a [routerLink]="'/product/' + product.slug">

<img mat-card-image [src]="product.image" />

<mat-card-content>

<h3>

{{ product.name }}

</h3>

<app-rating [rating]="product.rating"></app-rating>

</mat-card-content>

</a>

<mat-card-actions>

<span>₹{{ product.price }}</span>

<button mat-button (click)="addToCart(product)">Add to Cart 🛒</button>

</mat-card-actions>

</mat-card>

</div>

</div>

</ng-template>

**home.component.ts**

import { Component } from '@angular/core';

import { MatSnackBar } from '@angular/material/snack-bar';

import { Title } from '@angular/platform-browser';

import { Product } from 'src/app/models/product';

import { CartService } from 'src/app/services/cart.service';

import { ProductService } from 'src/app/services/product.service';

import SwiperCore, { Navigation, Pagination } from 'swiper';

SwiperCore.use([Navigation, Pagination]);

@Component({

selector: 'app-home',

templateUrl: './home.component.html',

styleUrls: ['./home.component.css'],

})

export class HomeComponent {

loading = true;

error = false;

products!: Product[];

constructor(

private productService: ProductService,

private snackBar: MatSnackBar,

private cartService: CartService,

private titleService: Title

) {

this.titleService.setTitle('Lensmakers - Your one stop destination for all kinds of eyewear');

}

ngOnInit() {

this.productService.getProducts().subscribe(

(products: Product[]) => {

this.loading = false;

this.products = products;

},

(err) => {

this.loading = false;

this.error = true;

this.snackBar.open(err, '', { panelClass: 'error-snackbar' });

}

);

}

addToCart(product: Product) {

const { \_id, image, name, slug, price } = product;

this.cartService

.add({ \_id, image, name, slug, price, quantity: 1 })

.subscribe(

(productName) =>

this.snackBar.open(`${productName} added to the cart`, '', {

panelClass: 'success-snackbar',

}),

(err) => {

this.snackBar.open(err.message, '', { panelClass: 'error-snackbar' });

}

);

}

onSwiper(swiper: any) {

console.log(swiper);

}

onSlideChange() {

console.log('slide change');

}

}

**product-details.component.html**

<div \*ngIf="loading; else result">

<mat-spinner></mat-spinner>

</div>

<ng-template #result>

<div \*ngIf="!error">

<div class="back-to-result">

<button

mat-raised-button

color="primary"

routerLink="/">Return Home</button>

</div>

<div fxLayout="row wrap" fxLayoutGap="20px grid" fxLayoutGap.xs="0 grid">

<div fxFlex="50%" fxFlex.xs="100%">

<img mat-card-image [src]="product.image" />

</div>

<div fxFlex="25%" fxFlex.xs="100%">

<h1>{{ product.name }}</h1>

<app-rating

[rating]="product.rating"

[numReviews]="product.numReviews"

[reviewUrl]="['/product/' + product.slug]"

></app-rating>

<mat-list>

<mat-list-item> Category : {{ product.category }}</mat-list-item>

<mat-list-item> Brand : {{ product.brand }}</mat-list-item>

<mat-list-item> Description : {{ product.description }}</mat-list-item>

</mat-list>

</div>

<div fxFlex="25%" fxFlex.xs="100%">

<mat-card class="mat-elevation-z4">

<mat-list>

<mat-list-item>

<div class="full-width" fxLayout="row">

<div fxFlex="50%">Price:</div>

<div fxFlex="50%">₹{{ product.price }}</div>

</div>

</mat-list-item>

<mat-list-item>

<div class="full-width" fxLayout="row">

<div fxFlex="50%">Status:</div>

<div fxFlex="50%">

{{ product.countInStock > 0 ? "In Stock" : "Unavailable" }}

</div>

</div>

</mat-list-item>

</mat-list>

<mat-card-actions>

<button

class="full-width"

mat-raised-button

color="primary"

(click)="addToCart()"

>

ADD TO CART

</button>

</mat-card-actions>

</mat-card>

</div>

</div>

<div name="reviews">

<h2 style="text-align: center;">Reviews</h2>

<p \*ngIf="product.reviews.length === 0" style="text-align: center;">No review submitted</p>

<mat-list \*ngFor="let review of product.reviews">

<mat-list-item>

<div>

<p>

<strong>{{ review.name }}</strong>

</p>

<app-rating [rating]="review.rating"></app-rating>

<p>

<strong>{{ review.comment }}</strong>

</p>

<mat-divider></mat-divider>

</div>

</mat-list-item>

</mat-list>

<p \*ngIf="!currentUser" style="text-align: center;">

Please

<a

routerLink="/login"

[queryParams]="{ returnUrl: '/product/' + product.slug }"

>login</a

>

to submit a review

</p>

<div class="main-div">

<mat-card \*ngIf="currentUser" class="mat-elevation-z7">

<mat-card-content>

<form [formGroup]="form" (ngSubmit)="onSubmit()">

<h2>Leave us a review</h2>

<mat-list>

<mat-list-item>

<mat-form-field class="full-width">

<input

matInput

placeholder="Comment"

formControlName="comment"

required

/>

<mat-error> Please provide a comment </mat-error>

</mat-form-field>

</mat-list-item>

<mat-list-item>

<mat-form-field appearance="fill">

<mat-label>Choose an option</mat-label>

<select formControlName="rating" matNativeControl>

<option value=""></option>

<option value="1">⭐ Poor</option>

<option value="2">⭐⭐ Okay</option>

<option value="3">⭐⭐⭐ Good</option>

<option value="4">⭐⭐⭐⭐ Very good</option>

<option value="5">⭐⭐⭐⭐⭐ Excellent</option>

</select>

</mat-form-field>

</mat-list-item>

<mat-list-item>

<button

[disabled]="createReviewLoading"

mat-raised-button

color="primary"

style="display:block;margin:auto"

>

Submit

</button>

<mat-spinner

[diameter]="30"

\*ngIf="createReviewLoading"

></mat-spinner>

</mat-list-item>

</mat-list>

</form>

</mat-card-content>

</mat-card>

</div>

</div>

</div>

</ng-template>

**product-details.component.ts**

import { Component, OnInit } from '@angular/core';

import { Title } from '@angular/platform-browser';

import { ActivatedRoute, Router } from '@angular/router';

import { Product } from '../../models/product';

import { AuthService } from '../../services/auth.service';

import { ProductService } from '../../services/product.service';

import { CartService } from 'src/app/services/cart.service';

import { MatSnackBar } from '@angular/material/snack-bar';

import { UserInfo } from 'src/app/models';

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({

selector: 'app-product-details',

templateUrl: './product-details.component.html',

styleUrls: ['./product-details.component.css'],

})

export class ProductDetailsComponent implements OnInit {

form: FormGroup;

currentUser: UserInfo | null = null;

submitted = false;

error = false;

loading = true;

createReviewLoading = false;

product: Product = {

\_id: '',

name: '',

price: 0,

image: '',

brand: '',

category: '',

description: '',

countInStock: 0,

slug: '',

rating: 0,

numReviews: 0,

reviews: [],

};

constructor(

private formBuilder: FormBuilder,

private authService: AuthService,

private titleService: Title,

private route: ActivatedRoute,

private productService: ProductService,

private router: Router,

private cartService: CartService,

private snackBar: MatSnackBar

) {

this.form = this.formBuilder.group({

comment: ['', Validators.required],

rating: ['', Validators.required],

});

}

async ngOnInit() {

this.authService.currentUser.subscribe((x) => (this.currentUser = x));

this.getProduct();

}

getProduct() {

const routeParams = this.route.snapshot.paramMap;

const slug = routeParams.get('slug');

if (slug) {

this.productService.getProductBySlug(slug).subscribe(

(data) => {

this.loading = false;

this.product = data;

this.titleService.setTitle(this.product.name);

},

(err) => {

this.error = true;

this.loading = false;

this.snackBar.open(err, '', {

panelClass: 'error-snackbar',

});

}

);

} else {

this.error = true;

this.loading = false;

this.snackBar.open('Product not found', '', {

panelClass: 'error-snackbar',

});

}

}

addToCart() {

const { \_id, image, name, slug, price } = this.product;

this.cartService

.add({ \_id, image, name, slug, price, quantity: 1 })

.subscribe(

(productName) => {

this.snackBar.open(`${productName} added to the cart`, '', {

panelClass: 'success-snackbar',

});

this.router.navigate(['/cart']);

},

(err) => {

this.snackBar.open(err.message, '', { panelClass: 'error-snackbar' });

}

);

}

onSubmit() {

this.submitted = true;

if (this.form.invalid) {

return;

}

const { comment, rating } = this.form.controls;

this.createReviewLoading = true;

this.productService

.createReview(this.product.\_id, comment.value, rating.value)

.subscribe(

(data) => {

this.getProduct();

this.createReviewLoading = false;

},

(error) => {

this.snackBar.open(error, '', { panelClass: 'error-snackbar' });

this.createReviewLoading = false;

}

);

}

}

**login.component.html**

<mat-card>

<mat-card-content>

<form [formGroup]="form" (ngSubmit)="onSubmit()">

<h2>Login to Lensmakers</h2>

<mat-list>

<mat-list-item>

<mat-form-field class="full-width">

<input

matInput

placeholder="Email"

formControlName="email"

required

/>

<mat-error> Please provide a valid email address! </mat-error>

</mat-form-field>

</mat-list-item>

<mat-list-item>

<mat-form-field class="full-width">

<input

matInput

type="password"

placeholder="Password"

formControlName="password"

required

/>

<mat-error> Please enter your password! </mat-error>

</mat-form-field>

</mat-list-item>

<mat-list-item>

<button [disabled]="loading" mat-raised-button color="primary"

style="display:block;margin:auto">

LOGIN

</button>

<mat-spinner [diameter]="30" \*ngIf="loading"></mat-spinner>

</mat-list-item>

<mat-list-item>

New user? &nbsp;

<a

[routerLink]="['/register']"

[queryParams]="{ returnUrl: returnUrl }"

>Register Here</a

>

</mat-list-item>

</mat-list>

</form>

</mat-card-content>

</mat-card>

**login.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router, ActivatedRoute } from '@angular/router';

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

import { first } from 'rxjs/operators';

import { AuthService } from '../../services/auth.service';

import { MatSnackBar } from '@angular/material/snack-bar';

@Component({

selector: 'app-login',

templateUrl: './login.component.html',

styleUrls: ['./login.component.css'],

})

export class LoginComponent implements OnInit {

form: FormGroup;

loading = false;

submitted = false;

returnUrl: string;

error = '';

constructor(

private formBuilder: FormBuilder,

private route: ActivatedRoute,

private router: Router,

private snackBar: MatSnackBar,

private authService: AuthService

) {

if (this.authService.currentUserValue) {

this.router.navigate(['/']);

}

this.form = this.formBuilder.group({

email: ['', Validators.email],

password: ['', Validators.required],

});

this.returnUrl = this.route.snapshot.queryParams['returnUrl'] || '/';

}

ngOnInit() {}

onSubmit() {

this.submitted = true;

if (this.form.invalid) {

return;

}

const { email, password } = this.form.controls;

this.loading = true;

this.authService

.login(email.value, password.value)

.pipe(first())

.subscribe(

(data) => {

this.router.navigate([this.returnUrl]);

},

(error) => {

this.snackBar.open(error, '', { panelClass: 'error-snackbar' });

this.loading = false;

}

);

}

}

**cart.component.html**

<h1>Shopping Cart</h1>

<div fxLayout="row wrap" fxLayoutGap="20px grid" fxLayoutGap.xs="10px grid">

<div \*ngIf="cart" fxFlex="70%" fxFlex.xs="100%">

<table mat-table [dataSource]="cart.items" class="mat-elevation-z8">

<!-- Image Column -->

<ng-container matColumnDef="image">

<th mat-header-cell \*matHeaderCellDef>Image</th>

<td class="thumbnail-cell" mat-cell \*matCellDef="let element">

<img [src]="element.image" class="thumbnail" />

</td>

</ng-container>

<ng-container matColumnDef="name">

<th mat-header-cell \*matHeaderCellDef>Name</th>

<td mat-cell \*matCellDef="let element">

<h3 [routerLink]="'/product/' + element.slug"> {{ element.name }}</h3>

</td>

</ng-container>

<ng-container matColumnDef="price">

<th mat-header-cell \*matHeaderCellDef>Price</th>

<td mat-cell \*matCellDef="let element">₹{{ element.price }}</td>

</ng-container>

<ng-container matColumnDef="quantity">

<th mat-header-cell \*matHeaderCellDef>Quantity</th>

<td mat-cell \*matCellDef="let element">

<button

(click)="remove(element)"

mat-icon-button

color="accent"

aria-label="remove icon"

>

<mat-icon>remove\_circle</mat-icon>

</button>

{{ element.quantity }}

<button

(click)="add(element)"

mat-icon-button

color="accent"

aria-label="add icon"

>

<mat-icon>add\_circle</mat-icon>

</button>

</td>

</ng-container>

<ng-container matColumnDef="subtotal">

<th mat-header-cell \*matHeaderCellDef>Subtotal</th>

<td mat-cell \*matCellDef="let element">

₹{{ element.quantity \* element.price }}

</td>

</ng-container>

<tr mat-header-row \*matHeaderRowDef="displayedColumns"></tr>

<tr mat-row \*matRowDef="let row; columns: displayedColumns"></tr>

</table>

</div>

<div fxFlex="30%" fxFlex.xs="100%" class="mt-20">

<mat-card class="mat-elevation-z4">

<mat-list>

<mat-list-item>

<h2>Subtotal ({{ cart.itemsCount }}) : ₹{{ cart.itemsPrice }}</h2>

</mat-list-item>

</mat-list>

<mat-card-actions>

<button

(click)="checkout()"

class="full-width"

mat-raised-button

color="primary"

>

CHECK OUT

</button>

</mat-card-actions>

</mat-card>

</div>

</div>

**cart.component.ts**

import { Component, OnInit } from '@angular/core';

import { MatSnackBar } from '@angular/material/snack-bar';

import { Title } from '@angular/platform-browser';

import { ActivatedRoute, Router } from '@angular/router';

import { Product } from 'src/app/models/product';

import { CartService } from 'src/app/services/cart.service';

import { Cart, Item } from '../../models/cart';

@Component({

selector: 'app-cart',

templateUrl: './cart.component.html',

styleUrls: ['./cart.component.css'],

})

export class CartComponent implements OnInit {

cart!: Cart;

displayedColumns: string[] = [

'image',

'name',

'price',

'quantity',

'subtotal',

];

error: string = '';

constructor(

private titleService: Title,

private router: Router,

private route: ActivatedRoute,

private snackBar: MatSnackBar,

private cartService: CartService

) {}

ngOnInit() {

this.titleService.setTitle('Shopping Cart');

this.cartService.currentCart.subscribe((x) => (this.cart = x));

}

add(item: Item) {

this.cartService.add(item).subscribe(

(productName) =>

this.snackBar.open(`${productName} added to the cart`, '', {

panelClass: 'success-snackbar',

}),

(err) => {

this.snackBar.open(err.message, '', { panelClass: 'error-snackbar' });

}

);

}

remove(item: Item) {

this.snackBar.dismiss();

this.cartService.remove(item.\_id);

}

checkout() {

if (this.cart.itemsCount === 0) {

this.snackBar.open('Cart is empty', '', { panelClass: 'error-snackbar' });

return;

}

this.router.navigate(['/shipping']);

}

}

**server.ts**

import cors from 'cors';

import path from 'path';

import dotenv from 'dotenv';

import express, { Request, Response } from 'express';

import mongoose from 'mongoose';

import { userRouter } from './routers/user.routes';

import { orderRouter } from './routers/order.routes';

import { productRouter } from './routers/product.routes';

import { uploadRouter } from './routers/upload.routes';

dotenv.config();

const app = express();

app.use(

cors({

credentials: true,

origin: ['http://localhost:4200'],

})

);

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

const MONGODB\_URI = process.env.MONGODB\_URI || 'mongodb://localhost/ts-backend';

mongoose

.connect(MONGODB\_URI)

.then(() => {

console.log('Successfully connected to MongoDB!');

})

.catch((err) => {

console.log('Error in database connection!');

});

app.use('/api/uploads', uploadRouter);

app.use('/api/users', userRouter);

app.use('/api/products', productRouter);

app.use('/api/orders', orderRouter);

app.get('/api/config/paypal', (req: Request, res: Response) => {

res.send({ clientId: process.env.PAYPAL\_CLIENT\_ID || 'sb' });

});

const \_\_dirname = path.resolve();

app.use(express.static(path.join(\_\_dirname, './frontend/dist/frontend')));

app.get('\*', (req, res) =>

res.sendFile(path.join(\_\_dirname, './frontend/dist/frontend/index.html'))

);

app.use((err: Error, req: Request, res: Response, next: Function) => {

res.status(500).send({ message: err.message });

});

const PORT: number = parseInt((process.env.PORT || '5000') as string, 10);

app.listen(PORT, () => {

console.log(`Server started at http://localhost:${PORT}`);

});

**utils.ts**

import { Request, Response } from 'express';

import jwt from 'jsonwebtoken';

import { User } from './models/user.model';

export const generateToken = (user: User) => {

return jwt.sign(

{

\_id: user.\_id,

name: user.name,

email: user.email,

isAdmin: user.isAdmin,

},

process.env.JWT\_SECRET || 'somethingsecret',

{

expiresIn: '30d',

}

);

};

export const isAuth = (req: Request, res: Response, next: Function) => {

const authorization = req.headers.authorization;

if (authorization) {

const token = authorization.slice(7, authorization.length); // Bearer XXXXXX

const decode = jwt.verify(

token,

process.env.JWT\_SECRET || 'somethingsecret'

);

req.user = decode;

next();

} else {

res.status(401).send({ message: 'No Token' });

}

};

export const isAdmin = (req: Request, res: Response, next: Function) => {

if (req.user && req.user.isAdmin) {

next();

} else {

res.status(401).send({ message: 'Invalid Admin Token' });

}

};

**user.service.ts**

import { Injectable } from '@angular/core';

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Observable } from 'rxjs';

import { User } from '../models/user';

import { environment } from 'src/environments/environment';

const httpOptions = {

headers: new HttpHeaders({ 'Content-Type': 'application/json' }),

};

@Injectable({

providedIn: 'root',

})

export class UserService {

constructor(private http: HttpClient) {}

getUser(userId: string): Observable<any> {

return this.http.get(`${environment.apiUrl}/api/users/${userId}`, {

responseType: 'json',

});

}

update(user: User) {

return this.http.put<User>(

`${environment.apiUrl}/api/users/${user.\_id}`,

user

);

}

deleteUser(userId: string): Observable<any> {

return this.http.delete(`${environment.apiUrl}/api/users/${userId}`, {

responseType: 'json',

});

}}

**product.service.ts**

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

import { environment } from '../../environments/environment';

import { Product, ProductFilter } from '../models/product';

@Injectable({

providedIn: 'root',

})

export class ProductService {

getAdminProducts() {

return this.http.get(`${environment.apiUrl}/api/products`, {

responseType: 'json',

});

}

constructor(private http: HttpClient) {}

getProducts(): Observable<any> {

return this.http.get(`${environment.apiUrl}/api/products`, {

responseType: 'json',

});

}

createProduct(): Observable<any> {

return this.http.post(`${environment.apiUrl}/api/products`, {

responseType: 'json',

});

}

createReview(

productId: string,

comment: string,

rating: number

): Observable<any> {

return this.http.post(

`${environment.apiUrl}/api/products/${productId}/reviews`,

{ comment, rating },

{

responseType: 'json',

}

);

}

deleteProduct(productId: string): Observable<any> {

return this.http.delete(`${environment.apiUrl}/api/products/${productId}`, {

responseType: 'json',

});

}

searchProducts(productFilter: ProductFilter): Observable<any> {

let qs = '';

if (productFilter.category) {

qs += `category=${productFilter.category}&`;

}

if (productFilter.name) {

qs += `name=${productFilter.name}&`;

}

return this.http.get(`${environment.apiUrl}/api/products?${qs}`, {

responseType: 'json',

});

}

getCategories(): Observable<any> {

return this.http.get(`${environment.apiUrl}/api/products/categories`, {

responseType: 'json',

});

}

getProduct(productId: string): Observable<any> {

return this.http.get(`${environment.apiUrl}/api/products/${productId}`, {

responseType: 'json',

});

}

getProductBySlug(slug: string): Observable<any> {

return this.http.get(`${environment.apiUrl}/api/products/slug/${slug}`, {

responseType: 'json',

});

}

postFile(fileToUpload: File): Observable<any> {

const formData: FormData = new FormData();

formData.append('image', fileToUpload, fileToUpload.name);

return this.http.post(`${environment.apiUrl}/api/uploads`, formData);

}

update(product: Product) {

return this.http.put<Product>(

`${environment.apiUrl}/api/products/${product.\_id}`,

product

);

}

}

**REFERENCES**

1. David Herron, “Node.js Web Development: Create real-time server-side applications with this practical, step-bystep guide”, 3rd Edition, 2016

2. Agus Kurniawan, “AngularJS Programming by Example”, First Edition, Kindle, 2014

3. "Full-Stack Web Development with MongoDB, Express, Angular, and Node.js" by Uttam Agarwal

4. "MEAN Machine: A beginner's practical guide to the JavaScript stack" by Chris Sevilleja and Holly Lloyd

5. "Getting MEAN with Mongo, Express, Angular, and Node" by Simon Holmes

6. Angular Documentation - <https://angular.io/docs>

7. Angular Material Component Dev Kit - <https://material.angular.io/cdk/categories>

8. Node.js and Express Tutorial by freeCodeCamp - <https://youtu.be/Oe421EPjeBE>

9. TypeScript by Infosys Springboard- [https://infyspringboard.onwingspan.com/typescript](https://infyspringboard.onwingspan.com/web/en/app/toc/lex_9436233116512678000_shared/overview)

10. Angular by Infosys Springboard - <https://infyspringboard.onwingspan.com/angular>