Cricket Shelf E-commerce site

UP2022742

May 8, 2023

Contents

Abstract

This e-commerce site targets cricket book enthusiasts with a user-friendly interface for browsing and ordering books. It accommodates various user roles, including unregistered customers, registered customers, and administrators. Customers can search for books, add them to a shopping basket, and complete orders.

1 Introduction

Introducing my new e-commerce site, designed to provide an easy and efficient shopping experience for cricket book enthusiasts. With a comprehensive database, this system stores information about each book, including the title, author(s), publisher, edition, year of publication, and sales price. The software also manages customer information, including their name, email address, last-used delivery address, last-used payment information, and the timestamp of their last login in accordance with the Data Protection Act (DPA).

This software caters to different user roles, including not-yet-registered customers, registered customers, and system administrators. Only registered customers can complete an order and access previous orders, while system administrators can see all orders and status of orders, timestamps and with financial transactions.

Customers can search for books based on any data stored in the system and add them to a shopping basket. At checkout, the shopping basket contents, along with the customer's identity, delivery address, payment information, timestamp of the order, and order status, are saved to the database.

The software provides flexibility to customers who can reuse their last-used delivery address and payment information. However, credit cards are not verified in order to keep the scope of the product manageable in the time period set.

2 Design

The Cricket Store software was designed using the MVC architecture, which separates the application logic into three interconnected components: the model, view, and controller. The model manages data and business logic, the view presents data to the user, and the controller handles user input and data flow. This design aims to promote maintainability and flexibility, allowing for easier management of complex systems, code reuse, and efficient testing and debugging (Gupta & Govil, 2010).

In a real-world scenario, the application would require that the session information be encrypted as well as the password that gets uploaded to the database. However, considering such wasn't on the initial requirements and it's not going to be a deploy-able product, I will avoid such functionality.

2.1 System Context Diagram

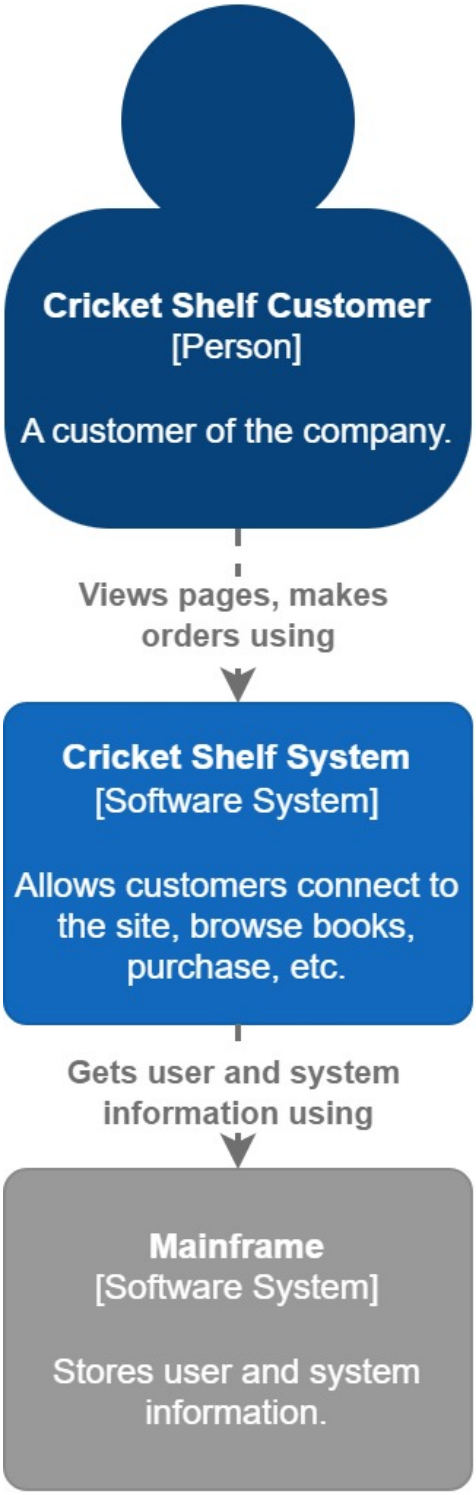

Figure 1: System context diagram

2.2 System Container Diagram

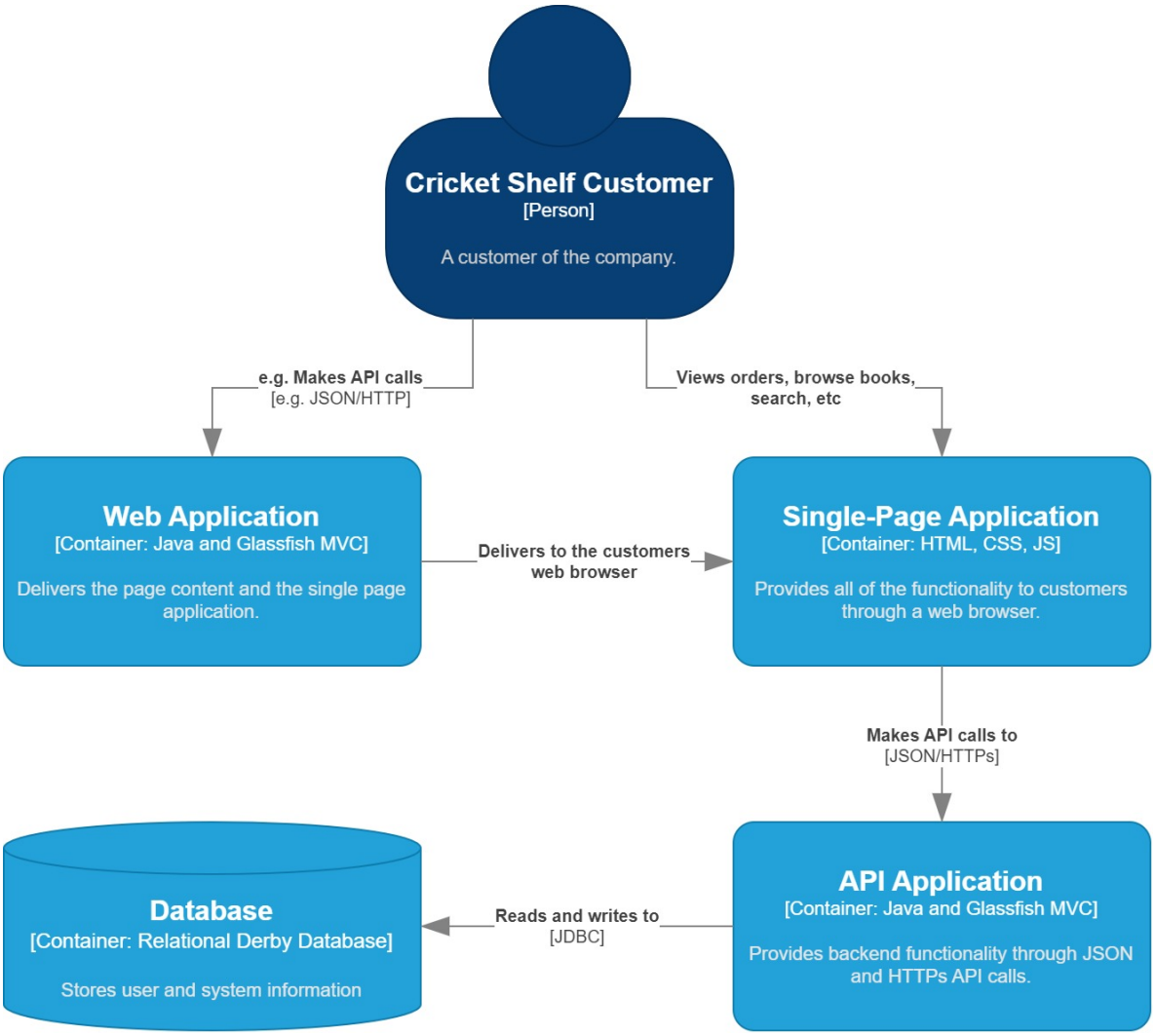

Figure 2: System Container Diagram

2.3 System API component diagram

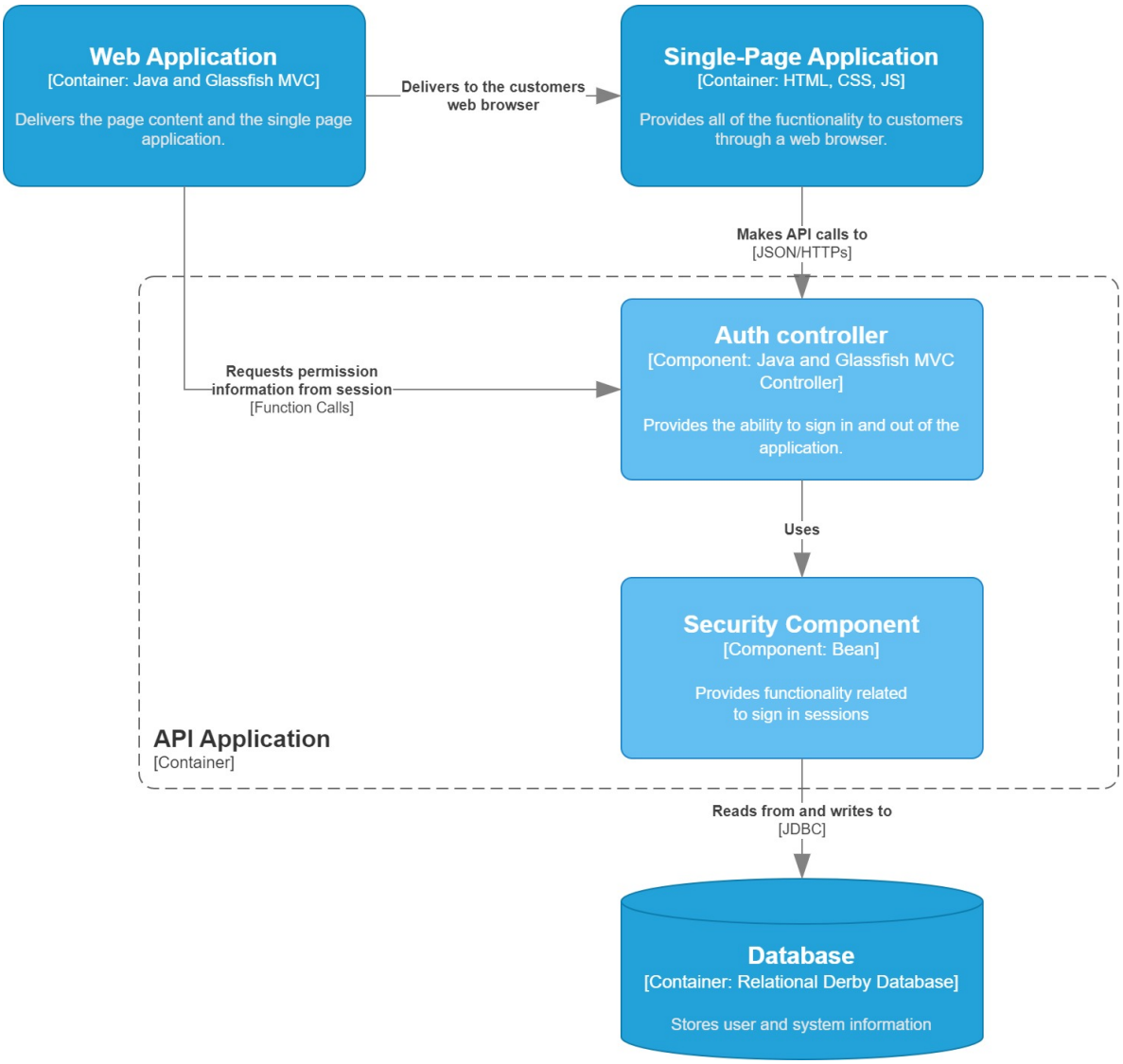

Figure 3: System API component diagram

2.4 Entity Relationship Diagram

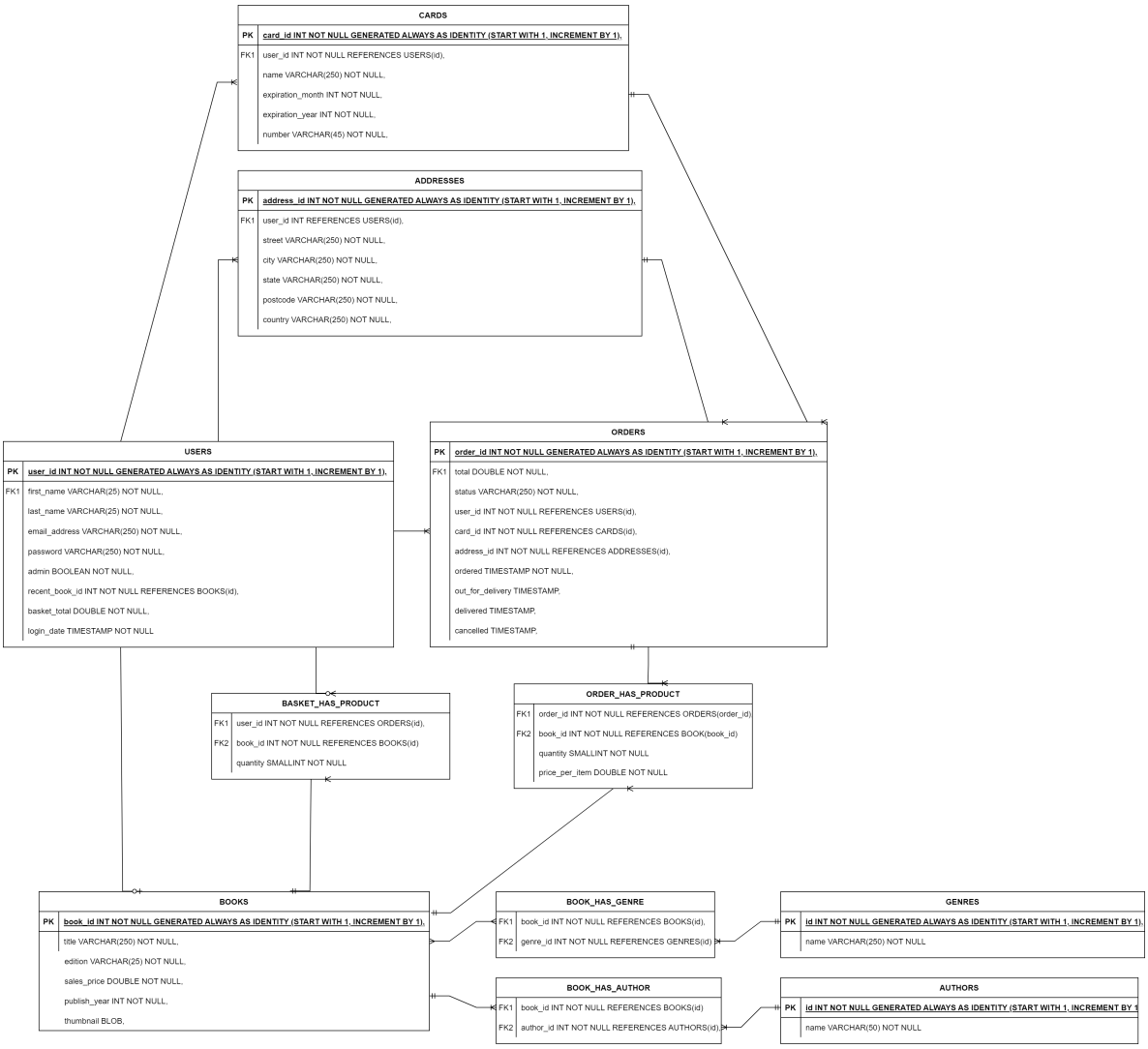

Figure 4: Entity relationship diagram (ERD)

3 Implementation

3.1 Documentation convention

I was aware that JavaDoc used code comments in order to generate documentation pages. In foresight, I made sure to peruse the Oracle documentation convention guide (Oracle, 2019) in order to make sure I didn't miss anything.

3.2 Persistence

After researching several online examples, I discovered that most of them stored basket data in a user session that would expire after a certain period of time. This could be frustrating for users who returned to the site after a while only to find that their search results had disappeared. To avoid this issue, I decided to use persistent storage to save basket information. All the data would be stored in the database. However, this approach also presented a challenge. When I posted data using a request, the application wouldn't commit and flush until the session ended. To address this, I utilised the .flush() and .persist() functions provided by NetBeans' autogenerated AbstractFacade class. With these functions, data would persist to the database, but subsequent calls still retrieved

cached information. This was problematic if the data had been updated before the call. To resolve this, I created a function in the AbstractFacade that cleared the cache, forcing a query to the database. I found the method for this on Stack Overflow (Youcef, 2017).

3.3 Debugging Create

At first, I didn't realise the extent to which JPA relied on the database configuration when generating entity and session classes with NetBeans. During debugging, I frequently encountered constraint violation messages that provided insufficient details about the violation. This made debugging difficult. To resolve this problem, I discovered a solution (Iomanip, 2014) that involved inserting a few lines of code into the creation function of the AbstractFacade class. This code checks for any constraint validations and, if found, reads the output message and writes the entire message to the console.

3.4 Create return object

3.5 EJB Entity Classes

When developing, I realised that the getter and setter functions in the automatically generated Enterprise JavaBeans (EJB) classes were missing and that this can be attributed to the absence of references to foreign keys in the tables created during the SQL database creation process.

In this context, the EJB classes facilitate the manipulation of data in a database and are dependent on the quality of the underlying data model. Without these references, the EJB classes lack the necessary information to generate the appropriate getter and setter methods for the corresponding data attributes. Therefore, it is essential to ensure that the SQL database creation process includes the proper definition of foreign key references to avoid issues such as the one encountered with the automatically generated EJB classes.

4 Testing

4.1 Google Lighthouse

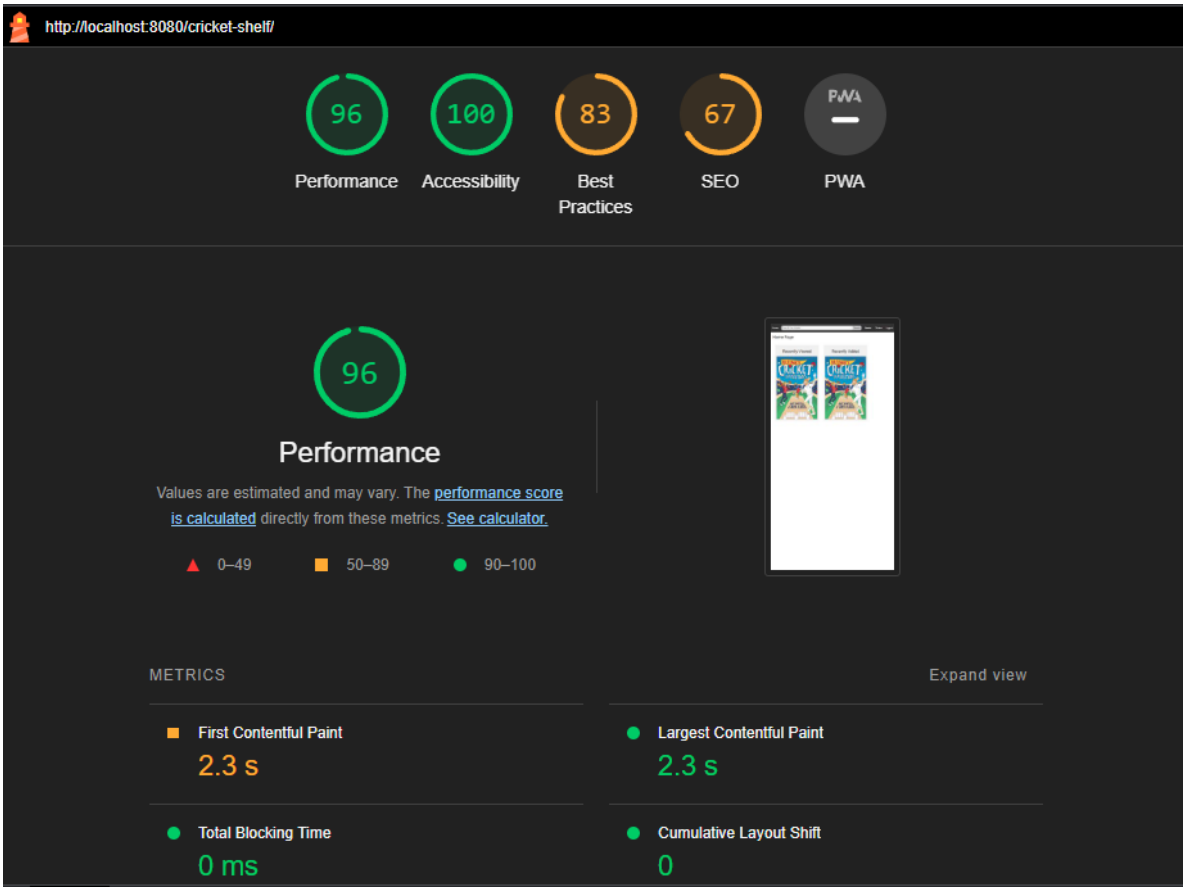

Figure 5: Google Lighthouse Report

4.2 Functional Requirements

Functional Requirements			
Index	Requirement	Status	Description
1	Enable the user to be able to search for a book by its title	Passed	By clicking on the search bar located at the top of every page, the user will have the ability to type their book request and view all available books with a similar name by submitting that query.
2	Upon clicking on a book icon, a new page opens up displaying the rest of the book information.	Passed	Once a user clicks on a page with the suffix "/book?id=x", they will be greeted with information about the selected book.
3	Display a list of books of a similar category when searching for particular books. They will be displayed at the bottom of the page with clickable links	Passed	On the book page, the users will be able to see a section below which lists potentially similar books based on similar names and genres.

4	Add an input box which allows the user to specify the quantity of a book the wish to add to their cart.	Passed	On the book page, there will be a input box and a button saying "Add to cart" where the user can add their desired book and quantity to the shopping cart.
5	Enable users to click on a button which adds a book and its quantity to their basket.	Passed	A button is available on the book pages, once a user clicks said button. A request is made to add the book and the specified quantiy (default 1) to their basket
6	Add the ability to request delivery addresses or offer previously specified delivery addresses.	Passed	Once a user has requested checkout, they will be greeted with a page where they can either select previously used addresses or create a new one through the use of a form. Upon adding the new address, the page will update showing the newly added address.
7	Add a form in which the user can add payment information or reuse previously specified payments.	Passed	As per the address. Upon reaching the payment page, the user will have the option to choose a previously entered payment method or to create a new one.
8	All users should be able to view a list of all their previously placed orders in the order of newest first.	Passed	When the user clicks on the options page, they can search the orders by; time/date, total cost, status, id, book names, etc.
9	Non-administrators should have the ability to cancel previously placed orders as long as they aren't already delivered.	Passed	The user can double click on the order on the orders page and a dropdown appears allowing them to change the status to cancelled if the status is anything but delivered.
10	Display the status of the order	Passed	On the order screen, there will be a table of orders will the status of their order and the timestamps that each status was updated. E.g. "Ordered 2023-04-29T20:06:41" and "Out for delivery 2023-04-30T09:00:51".
11	Upon changing the status of an order, save a timestamp stating the time and date the order status was changed.	Passed	When an admin changes the status of an order, the timestamp is saved and appended to the relevant column in the table as well as the status box changing the status.
12	A page showing the audit trail of the status changes for an order.	Passed	An audit trail can be followed by the status and timestamps in the table columns.

14	System administrations should have the ability to see all user orders including the most recent first with the ability to sort and filter by user	Passed	The system administrator order page is different than the user one. The page will include a slider which enables them to load in the data for all user orders. This will display a separate table in which they can filter as normal.
14	System administrators should have the ability to see the audit summary for all user orders.	Passed	The system administrator table will show the users orders and their corresponding order statuses.
15	System administrators should have the ability to search financial transactions that have occurred between two input dates	Passed	There is a time period at the top of the table in which the administrator can apply the time between two dates to see which orders where made.

4.3 Non-Functional Requirements

Non-Functional Requirements			
Index	Requirement	Status	Description
1	Data Protection Act (2018)	Passed	Upon accessing the site, the user will be greeted with a cookie prompt explaining to the user that cookies are necessary for an optimal user experience, and if the user proceeds, it will be understood that they consent to the use of cookies.
2	Simplicity	Undecided	No usability testing has occured in order to assume that the interface is inherently simple.
3	Performant Backend	Passed	The back-end runs well with response times being pseudo-instant, thus we can safely assume that it is performant enough for this site.
4	Performant Frontend	Passed	As you can see from the google lighthouse report, the performance score was rated 96/100.
5	Responsive to devices	Passed	The website was made with bootstrap in mind, thus resulting in a inherently responsive design.
6	Accessibility	Passed	As you can see from the Google Lighthouse report, it received an accessability rating of 100%.
7	Simultaneous clients	Passed	Not only does glassfish have the ability to inherently support multiple simultanous connections (eginnovations, 2023). It also has the ability to scale servers for increased load.

5 Summary

To sum up, due to my unfamiliarity with this framework, I faced a challenging learning curve, resulting in some inconsistency in coding standards. Despite this, I made a conscious effort to follow the MVC pattern as closely as I could. Given additional development time, I would take the opportunity to reorganise the code and deepen my knowledge of JPA and Glassfish to better align myself with the best coding practices.

6 References

eginnovations. (2023). Application performance monitoring & infrastructure monitoring. https://www.eginnovations.com/documentation/GlassFish/GlassFish-Thread-Pool-Test.htm

Gupta, P., & Govil, M. (2010). Spring web mvc framework for rapid open source j2ee application development: A case study. *International Journal of Engineering Science and Technology*, *2*(6), 1684–1689.

Iomanip. (2014). Bean validation constraint(s) violated while executing automatic bean validation on callback event: "prepersist." https://stackoverflow.com/questions/7579882/bean-validation-constraints-violated-while-executing-automatic-bean-validation

Oracle. (2019). How to write doc comments for the javadoc tool. https://www.oracle.com/uk/technical-resources/articles/java/javadoc-tool.html

Youcef, L. (2017). Java - difference between evictall and refresh. https://stackoverflow.com/questions/41565014/difference-between-evictall-and-refresh