



CEN 308 SOFTWARE ENGINEERING

PROJECT DOCUMENTATION

PROJECT NAME

Prepared by:
Amar Mujagić
Adnan Mujagić

Proposed to:
Nermina Durmić, Assist. Prof. Dr.
Aldin Kovačević, Teaching Assistant

21.6.2022

TABLE OF CONTENTS

[1. Introduction](#)

[1.1. About the Project](#)

[1.2. Project Functionalities and Screenshots](#)

[2. Project Structure](#)

[2.1. Technologies](#)

[2.2. Database Entities](#)

[2.3. Architectural Pattern](#)

[2.4. Design Patterns](#)

[Factory](#)

[Facade](#)

[3. Conclusion](#)

1. Introduction

1.1. About the Project

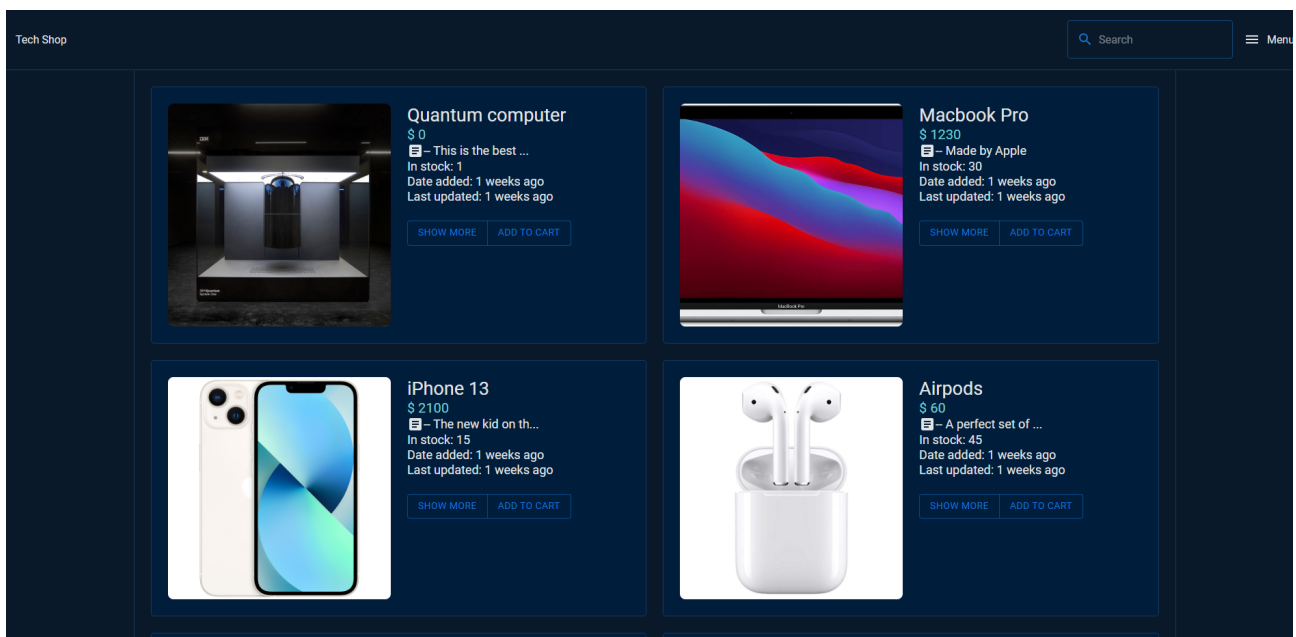
The application that we were working on for this project is an online tech shop, where users can easily search for and buy various devices. Once they buy a product, they can also leave a review for it. The project also has an admin dashboard, where the shop owners and/or other administrators can monitor the current state of their inventory as well as reports on the most sold items.

For the deployment of our project, we used two platforms: Heroku and Vercel, for our backend and frontend respectively. The application is available at the following link:

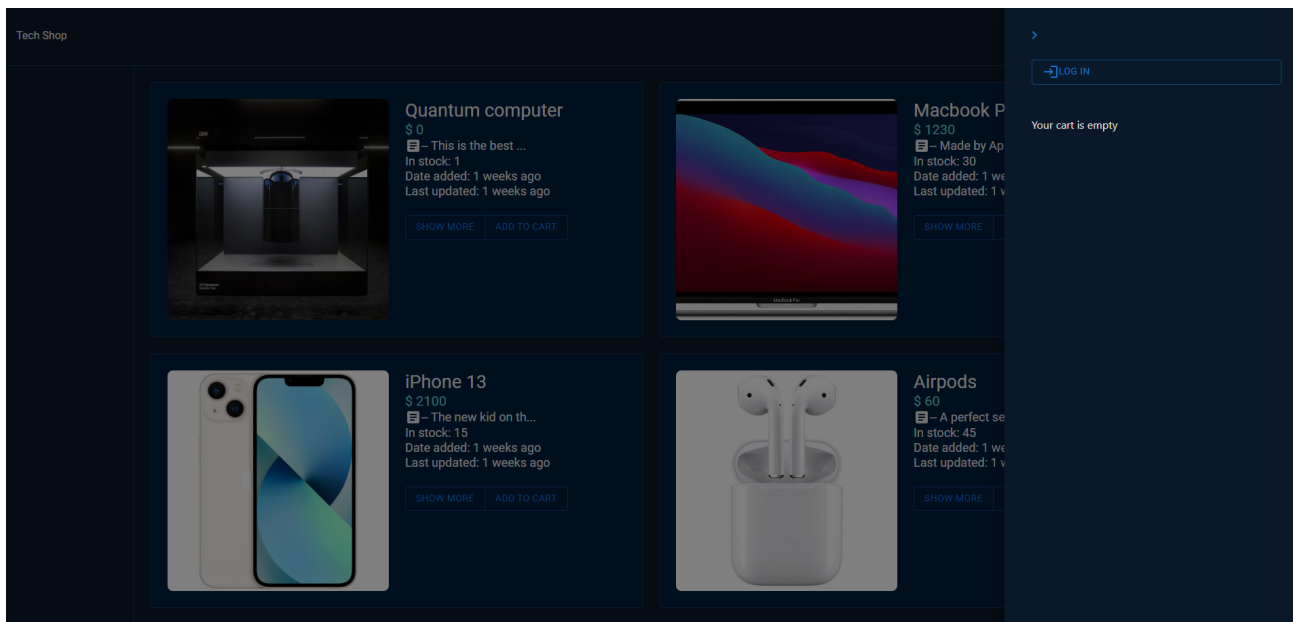
<https://tech-shop-frontend.vercel.app/>

1.2. Project Functionalities and Screenshots

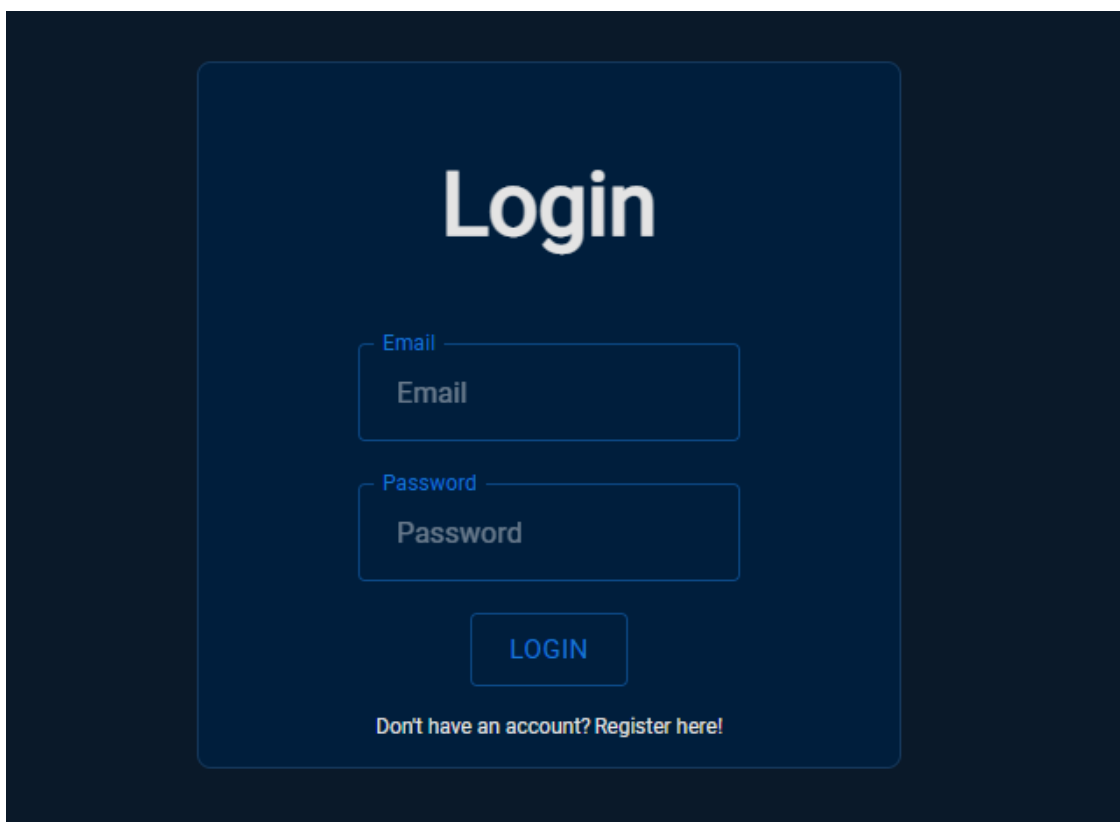
Landing page – this is the page that will load by default when the user opens the website. They do not have to be logged in to access this route. From here, the users can easily search for and read more details about the products that interest them. However, if the user wishes to buy a product (make an order), they will not be able to do so, until they log in.



If a user is not logged in, a link to the login page will be provided in the drawer menu on the right hand side, which can be opened by pressing on the “burger” icon in the top right corner of the website.



The login page allows the user to log in or navigate to the register page in case they do not already have an account.



By pressing on the “Register here” link, a user is taken to the register page that looks like this:

Register

First name

First name

Last name

Last name

Email

Email

Username

Username

Password

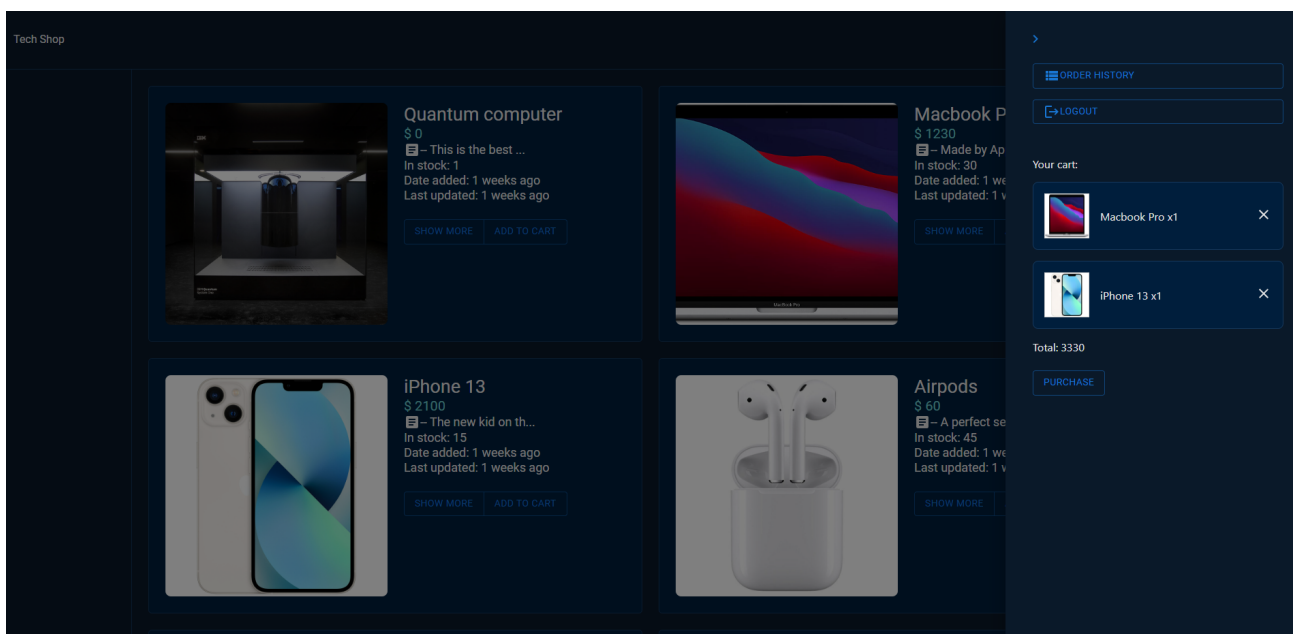
Password

REGISTER

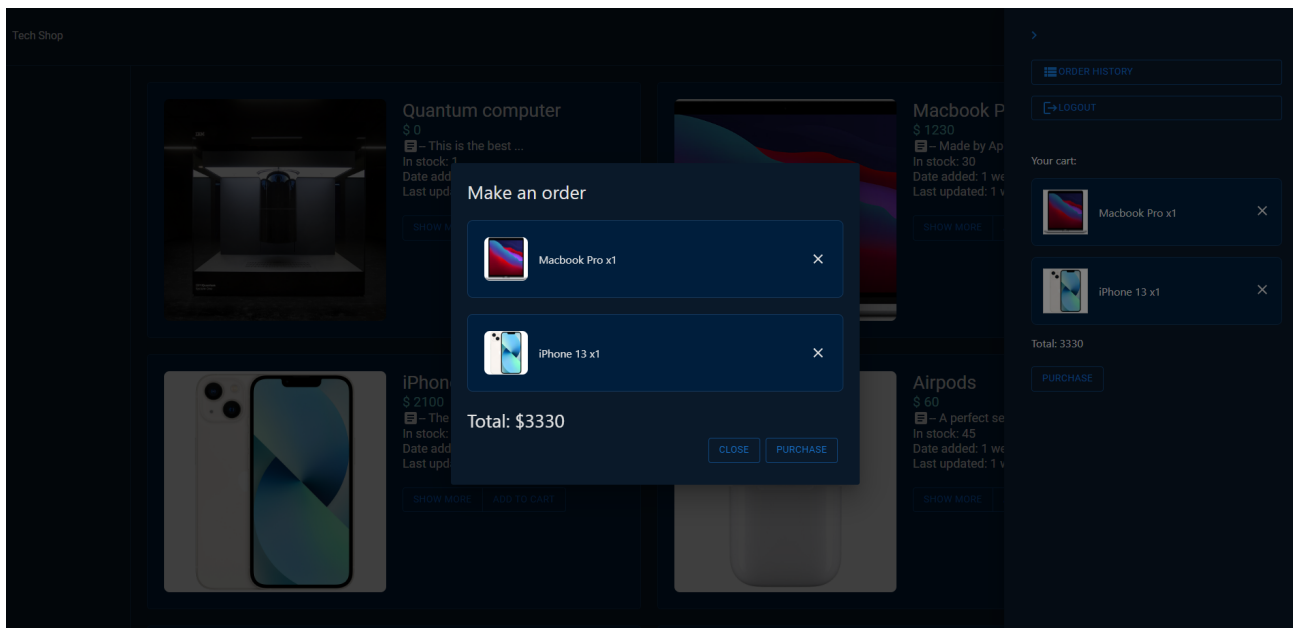
Already have an account? Login here!

Once the user has logged in, they will be taken back to the landing page. Now they have a few more options available.

First of all, they can now add items to the cart by pressing the “Add to cart” button on a desired product card. The added items will then show up in the drawer menu which looks like this:



Once the user has items in the cart, they will be able to make an order (by clicking on the “Purchase” button), which displays a confirmation dialog that looks like this:

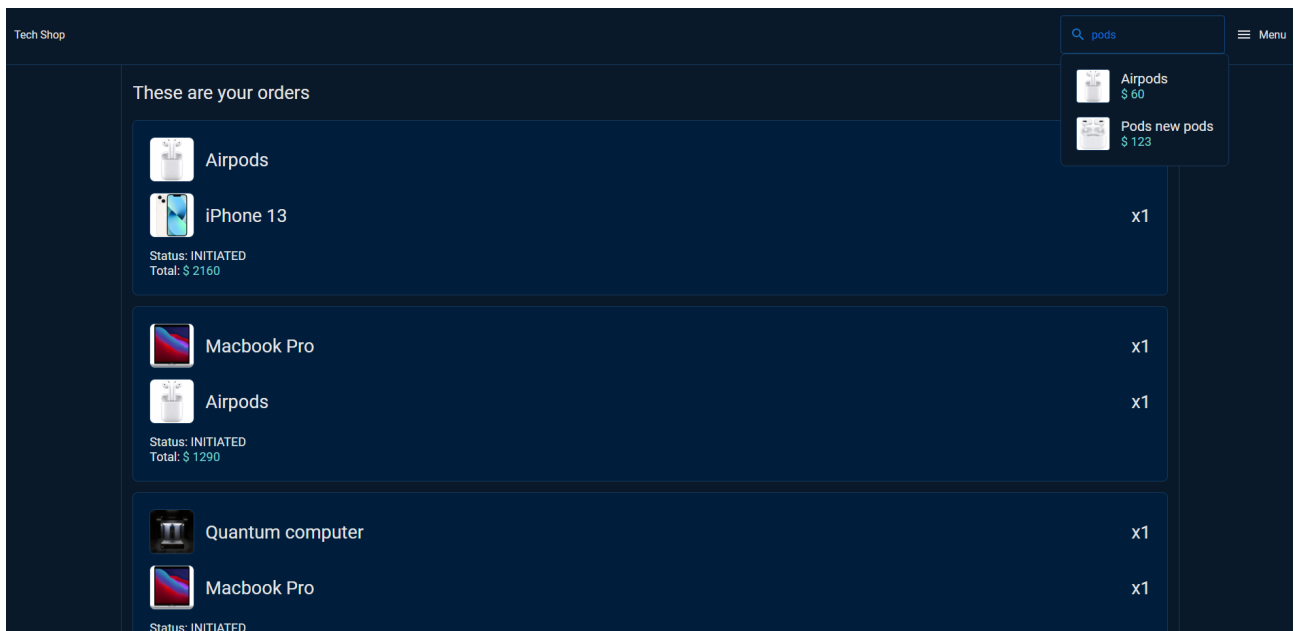


From this dialog, the users can modify their order (by removing certain items), they can confirm the order by clicking on the “Purchase” button, or close the dialog. If the user wishes to view their order history, as well as the status of their order, they can do so by clicking on the “Order history” button in the drawer, which opens the following page:

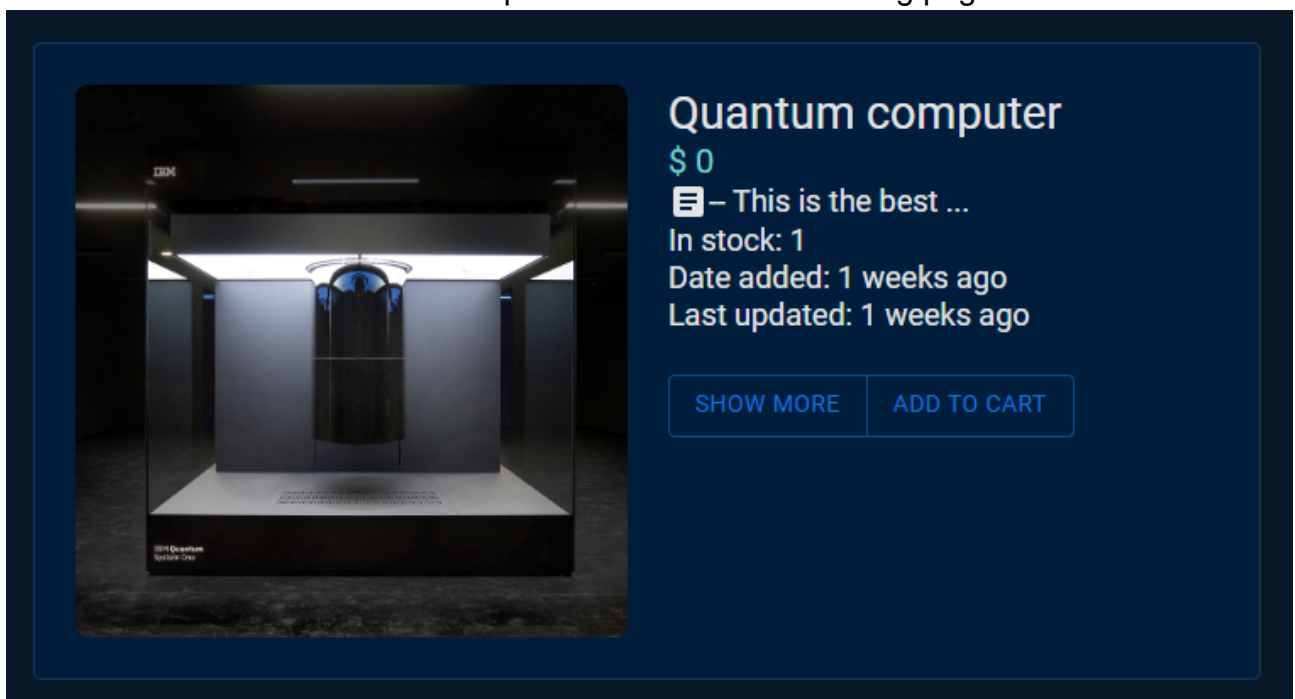


Once the user has bought a specific product, they will be allowed to leave a review for it. They can do so by navigating to the detailed view of the product they wish to review, which can be done in multiple ways:

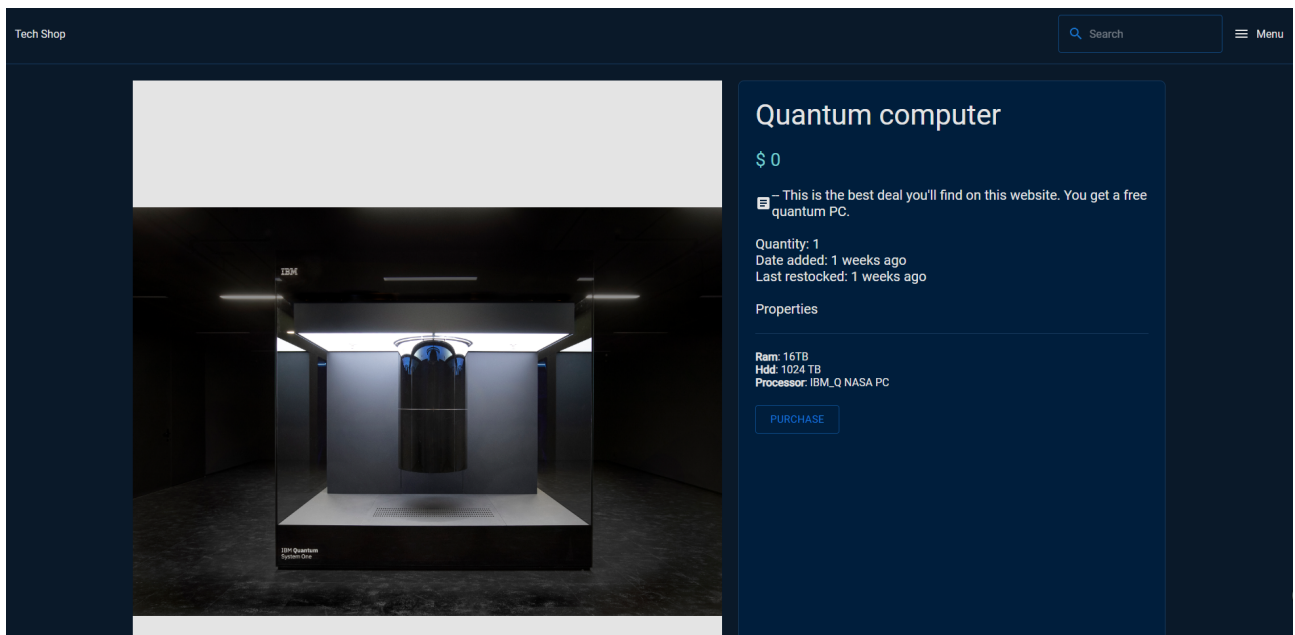
For example, a user can search for that product in the search bar, and then pick it from one of the suggested items:



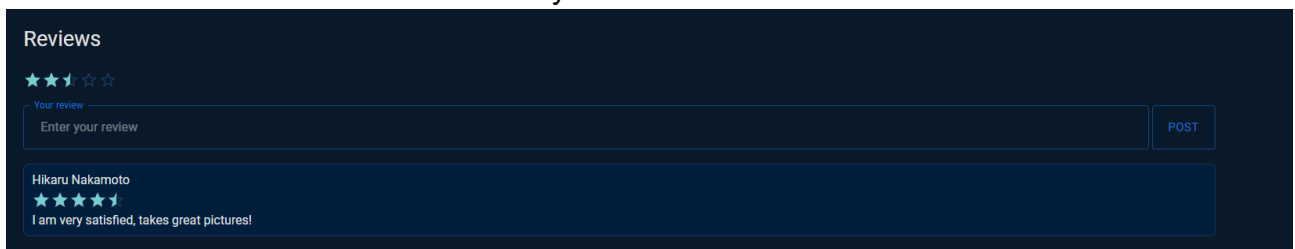
The search bar can be found in the header of the application, right next to the burger menu in the top right corner. Another way to get to the product details view is by clicking on the “Show more” button on the desired product’s card on the landing page:



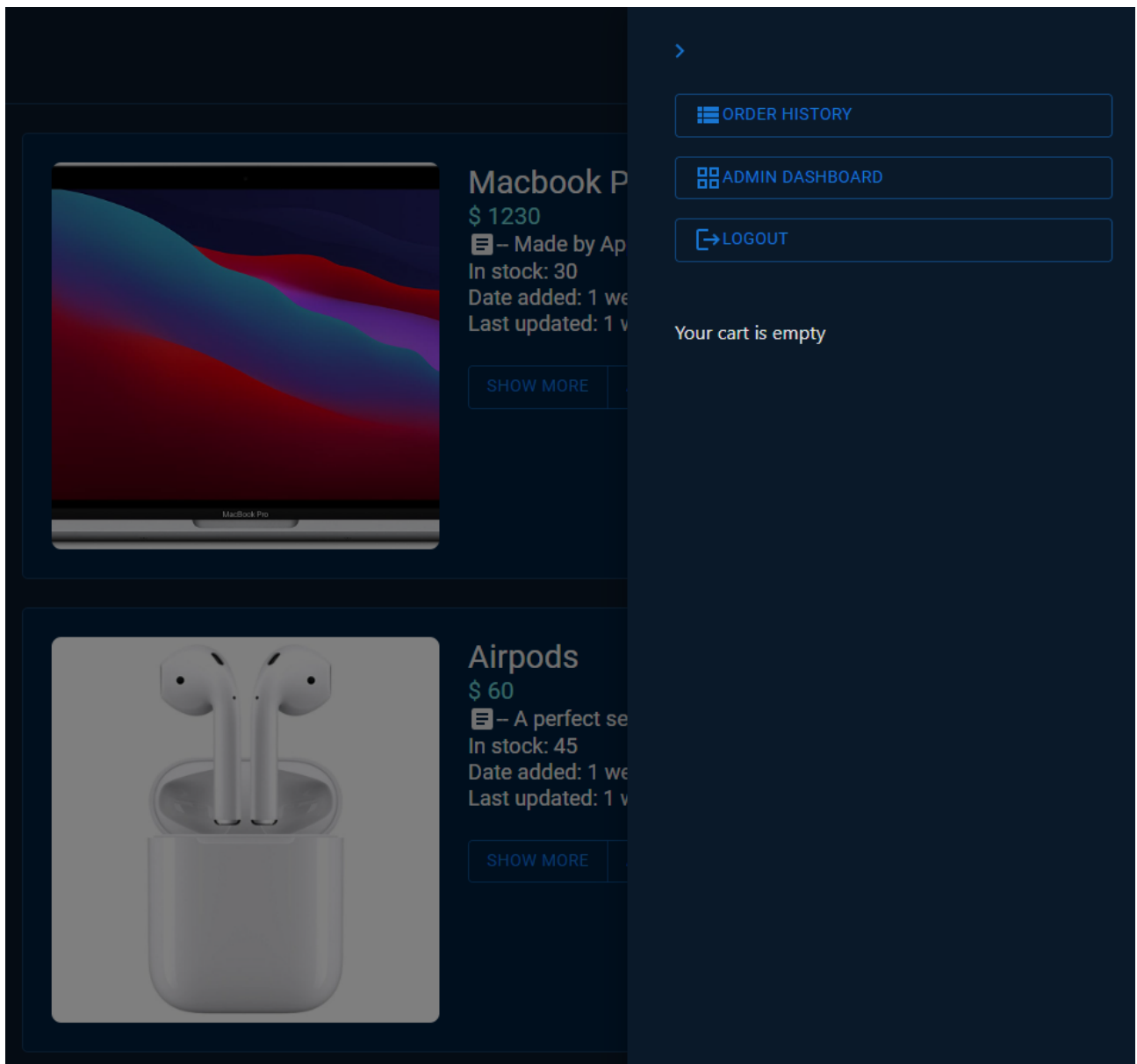
After performing one of the two most recently mentioned actions, the following page should be presented to the user:



Here the user can swipe through all of the product photos (given that there are multiple), and also if they scroll down, they can read the reviews, or potentially leave a review of their own (given that they have ordered the product before). The review section is shown right under the images. In the photo below, you can see the form for adding a review, as well as a list of reviews which are left by other users.



If the user is an admin, they will have access to an additional page, which is known as the admin dashboard. A link to it will be provided in the drawer menu:



After clicking on the “Admin dashboard” button in the picture above, the user should be taken to the following page:







Tech Shop

Search

Menu

> ADD PRODUCT

Lowest in stock

ID	Name	Amount in stock
62a5f21838f58d860f53d961	 Quantum computer	1
62a7b152afd7f37357c7b4e7	 iPhone 13	15
62a7ae4eafd7f37357c7b4d8	 Macbook Pro	30
62af5ed0879bd54359a42d61	 Huawei P30 lite	32
62a7b234afd7f37357c7b4f0	 Airpods	45
62b0cc5246e7d3d7ee18fa38	 Pods new pods	123

Most sold items

Place	ID	Name	Amount sold
-------	----	------	-------------

This page contains a lot of interesting data tables. For example, administrators can see items with the lowest amount in stock (inventory), or the most sold items. The admin also has the ability to add new products to the inventory.

You can use the following credentials to check out the admin features:

- Email: admin@admin.com
- Password: Admin123!

2. Project Structure

2.1. Technologies

The main programming language that we used throughout the whole project is JavaScript.

When it comes to markdown languages and other languages we used: HTML, SCSS.

The technology stack that we used to make this application is the following:

- MongoDB - a non relational database that we used to save our data.
- Express - a node package that we use to create a server
- Node.js - cross-platform, back-end JavaScript runtime environment
- React.js - a frontend library used for creating and rendering reusable components

Another key library that we used throughout our frontend is MaterialUI, which provides commonly used components such as buttons, menus, drawers, text fields, toast messages (alerts) etc.

For seamless transitions between the pages, we used the React Router.

On the backend we used a wrapper around the MongoDB's API, called "mongoose" which is an abstraction and makes development a little bit easier.

Since this is a Node.js project, we used Node Package Manager (NPM) to manage our dependencies.

The coding standard that we used for both frontend and backend development is [AirBNB JavaScript style guide](#).

Finally, we used Nightwatch.js to perform UI automation testing.

2.2. Database Entities

Provide a list of *tables or entities* you have in your database/schema. If it is not obvious from the name of the table/entity what it is used for, also provide a brief explanation next to it. For example:

In our database we have four main entities that are used:

- users the data about the registered users such as name, surname, email, password etc.
- products details about different products such as their name, price, quantity in stock, when they were last restocked, when they were added etc.
- orders details about orders such as their buyer and a list of products for that order, their quantities etc.
- reviews data about reviews that users leave for bought products, which includes the rating, textual review, data about the reviewer, date published etc.

2.3. Architectural Pattern

For this project we used Express specific architectural pattern which includes the following main components:

- Model - used for defining the structure of each entity in the database
- Route - used for defining the endpoints for the RESTful API
- Controller - functions that separate the logic that processes the requests from the logic that routes the requests

[Here](#) is a basic demo of how that architecture is supposed to work.

The main reason we used this pattern is because it's commonly used in MERN applications and it's a standard for which it was easy to find examples on the internet.

2.4. Design Patterns

Factory

Throughout our backend, in order to avoid a lot of boilerplate code when it comes to the creation of instances of specific entities, we used the factory design pattern. It was used for creation of [order](#) instances, as well as [reviews](#). Inside both of the factory methods we used an "updateEntity" function, which maps the properties of the body to the desired object. Finally, we returned the created object. This way we avoided manually assigning each property from the request body to the object, and polluting our controller

unnecessarily, we just delegated that responsibility to our factories, and called their static methods instead.

Facade

In the frontend part of the project, we made use of a facade design pattern in order to provide a simple to use interface for making HTTP requests. In the example [here](#), we created an Api constant, which contains basic functions for HTTP requests, such as GET, POST, PUT and DELETE, all of which handle more complex logic for creating a request in the background, such as adding a request body, or an “Authorization” header.

3. Conclusion

Our goal for this project was to put all of the pieces of software engineering together, such as the database design, backend and frontend implementation, working with architectural and design patterns, as well continuous integration and deployment (CI/CD) and create an appealing, functional and easy to use website in the process.

Overall, we are pretty happy with what we have managed to build, although there are a few more features that we had in mind, but weren’t implemented due to time constraints.

One of the things that we could improve is actually allowing the user to upload images from their device when adding a new product (currently we only support providing an image URL that we save to the database).

The most challenging task to implement is allowing the user to add custom properties for each product, which we wanted to have since not all the products are the same and we wanted to give the admin more control over it. We managed to implement it due to one of the pros of using MongoDB, where the documents aren’t strictly constrained to a schema.