

**IT 309 SOFTWARE ENGINEERING**

PROJECT DOCUMENTATION

E-commerce

Prepared by:

**Sanjin Kurtalija**

**Edvin Halilović**

Proposed to:

**Nermina Durmić, Assist. Prof. Dr.**

**Aldin Kovačević, Teaching Assistant**

21.06.2023.

TABLE OF CONTENTS

Contents

[1. Introduction 3](#_Toc138287078)

[1.1. About the Project 3](#_Toc138287079)

[1.2. Project Functionalities and Screenshots 3](#_Toc138287080)

[2. Project Structure 9](#_Toc138287081)

[2.1. Technologies 9](#_Toc138287082)

[2.2. Database Entities 10](#_Toc138287083)

[2.3. Design Patterns 10](#_Toc138287084)

[2.4. Tests 10](#_Toc138287085)

[3. Conclusion 11](#_Toc138287086)

# Introduction

## About the Project

Our project was to develop an E-commerce site as a project for Software Engineering course.

**REMINDER**

This project is consisting of two different github repositories which are both deployed and functional.

* E-commerce admin

Github link: <https://github.com/EdvinHalilovic/E-commerce>

Deployment Link: <https://e-commerce-46b9.vercel.app>

* E-commerce front

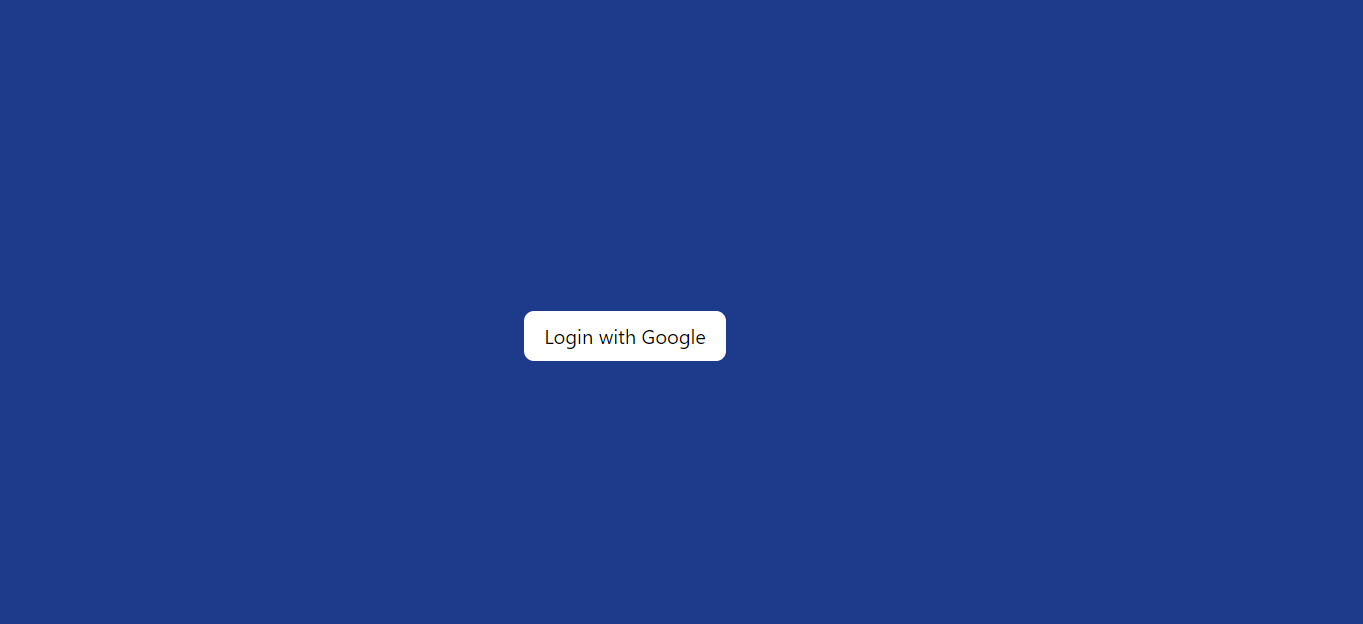
Github link: <https://github.com/Sanjin888/ecommerce-front>

Deployment Link: <https://ecommerce-front-sanjin888.vercel.app>

## Project Functionalities and Screenshots

Admin:

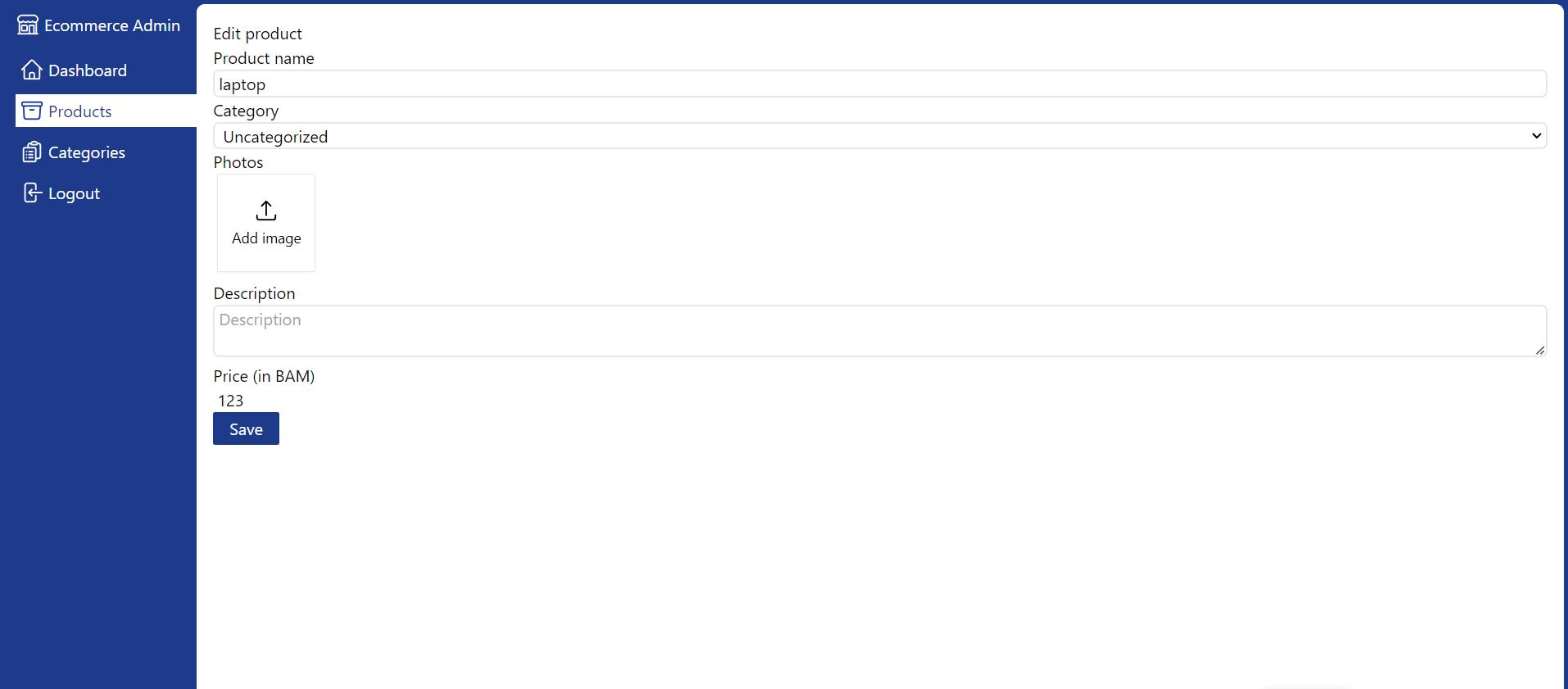
LOGIN



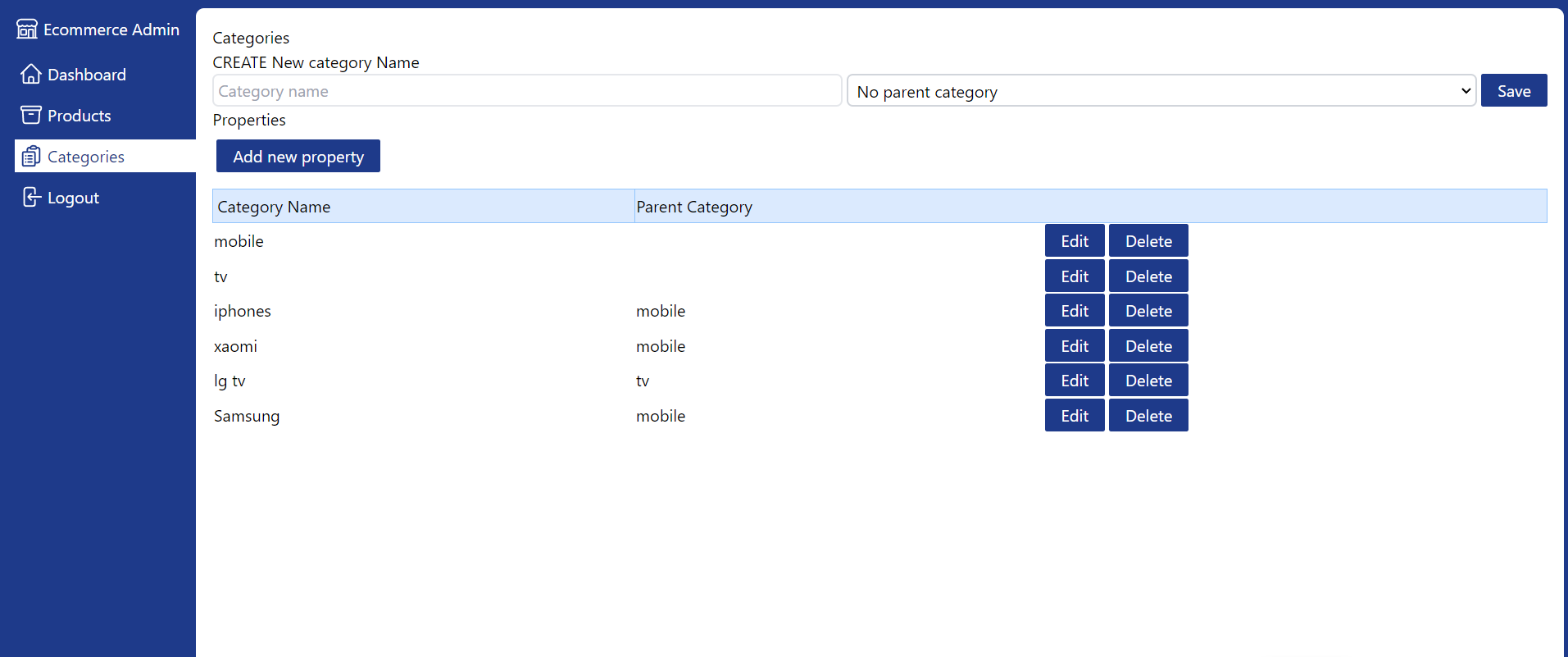
DASHBOARD

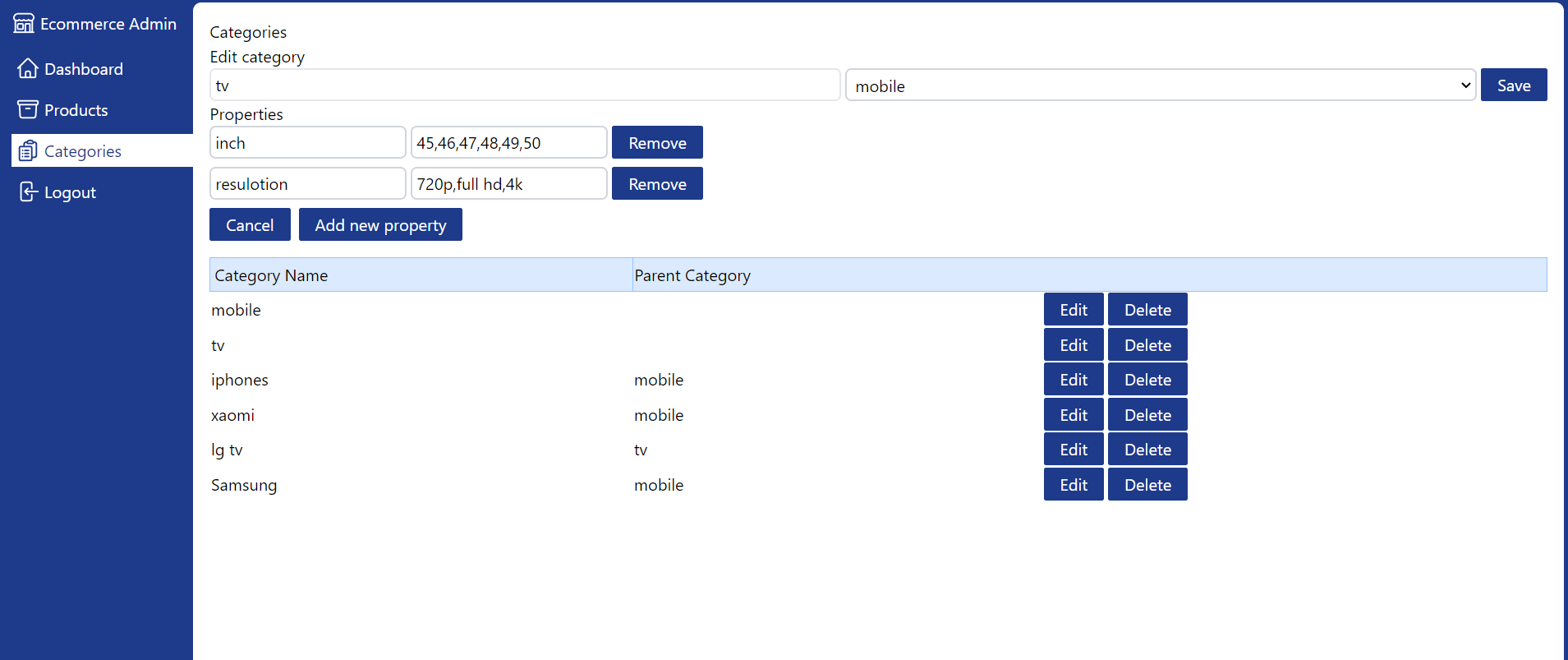


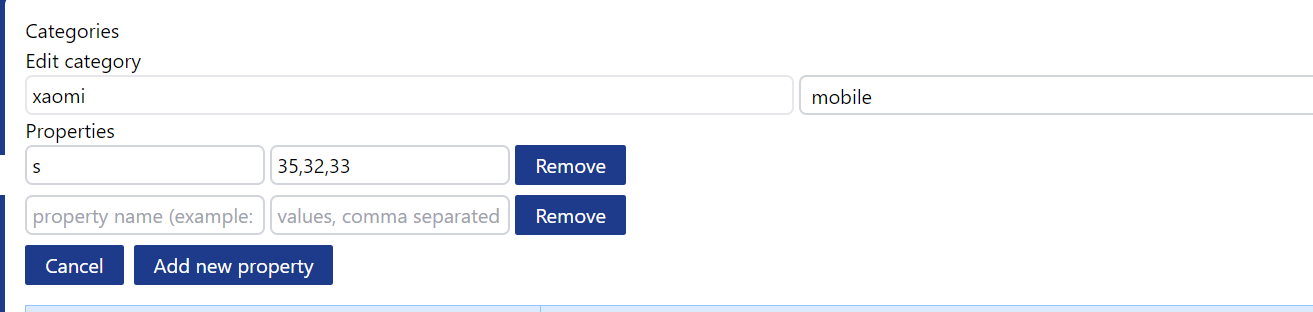
PRODUCTS PAGE



CATEGORIES PAGE

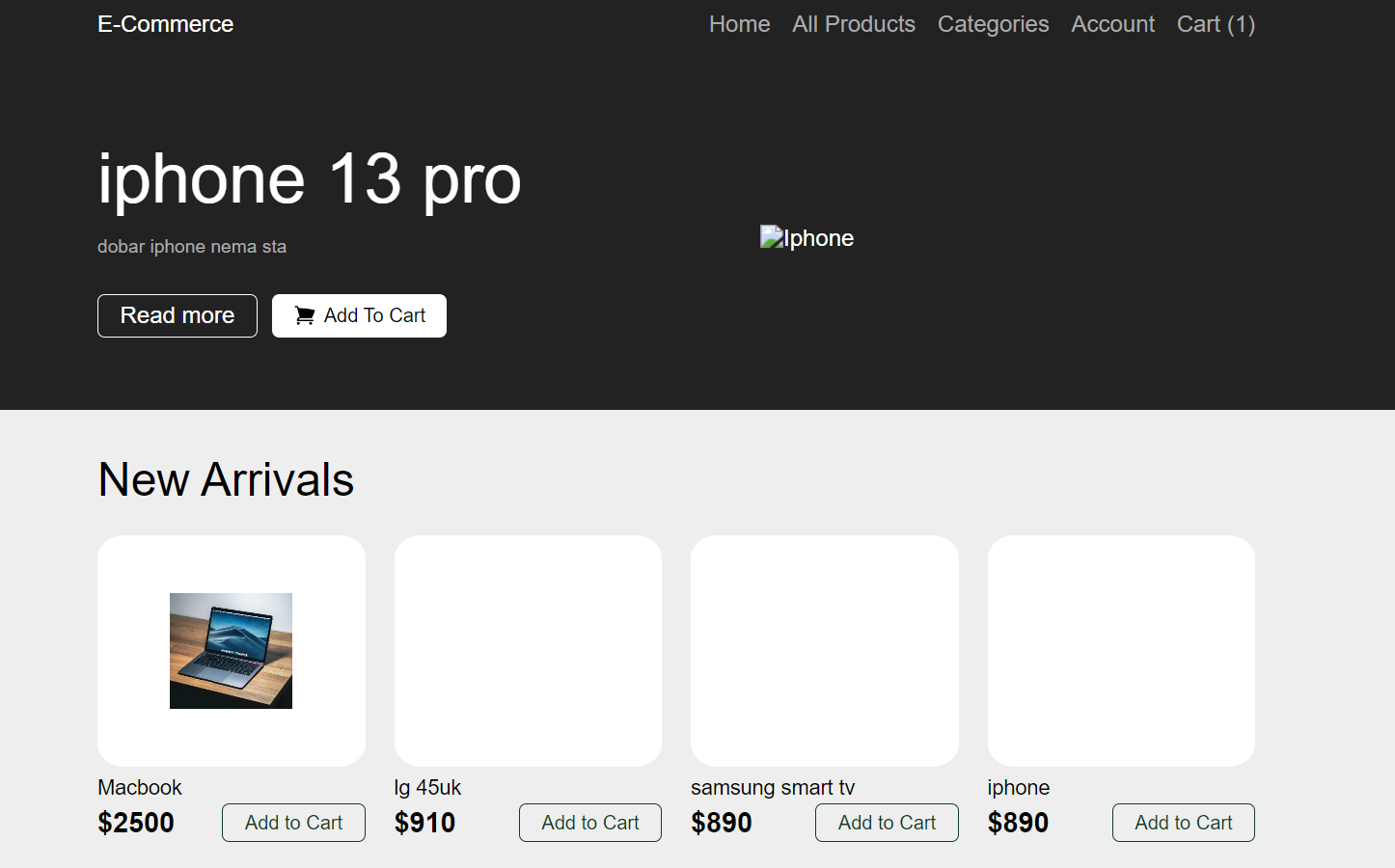


Ability to add new categories, edit existing ones or edit/add/remove any of the properties associated with the category

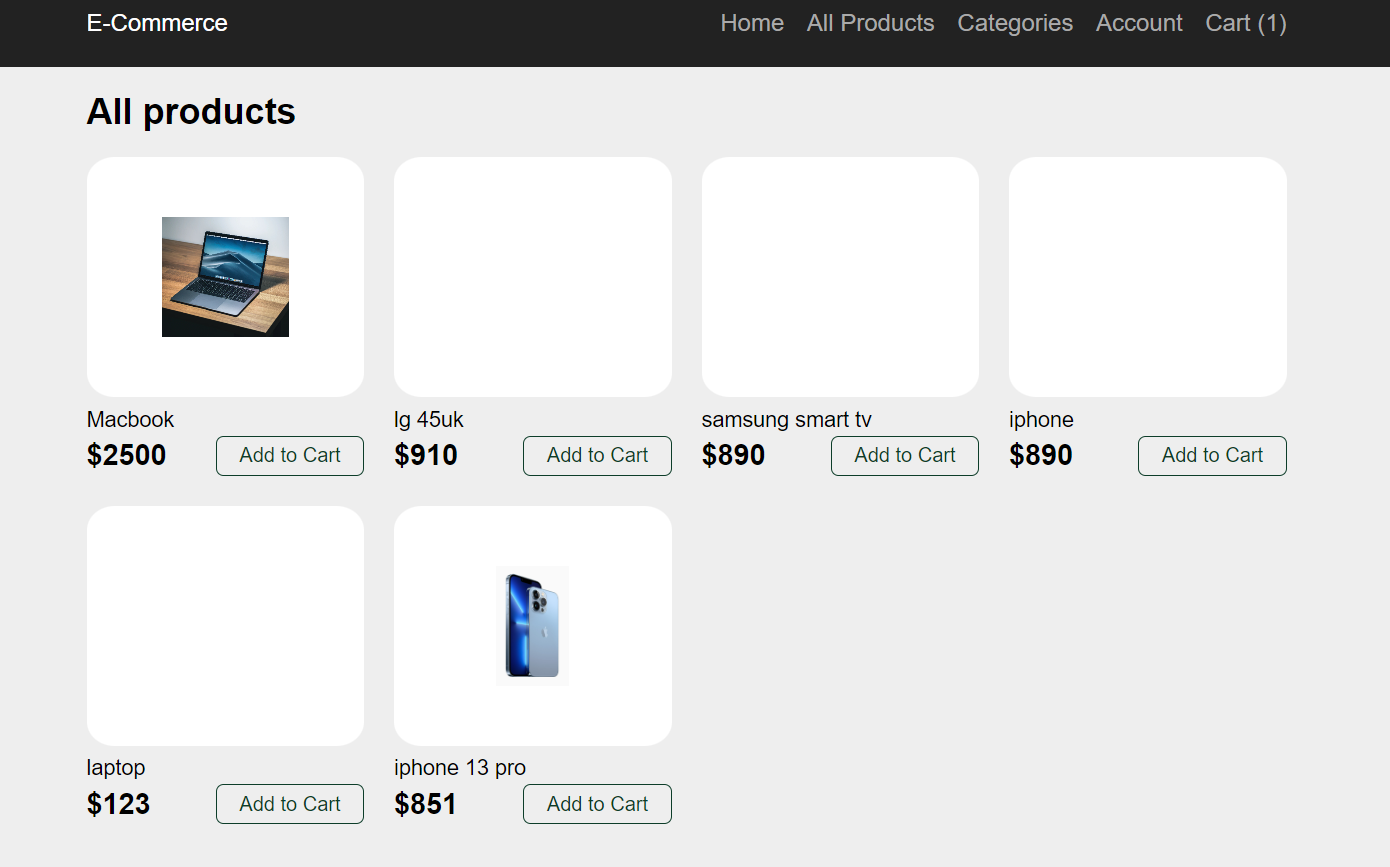


Front:

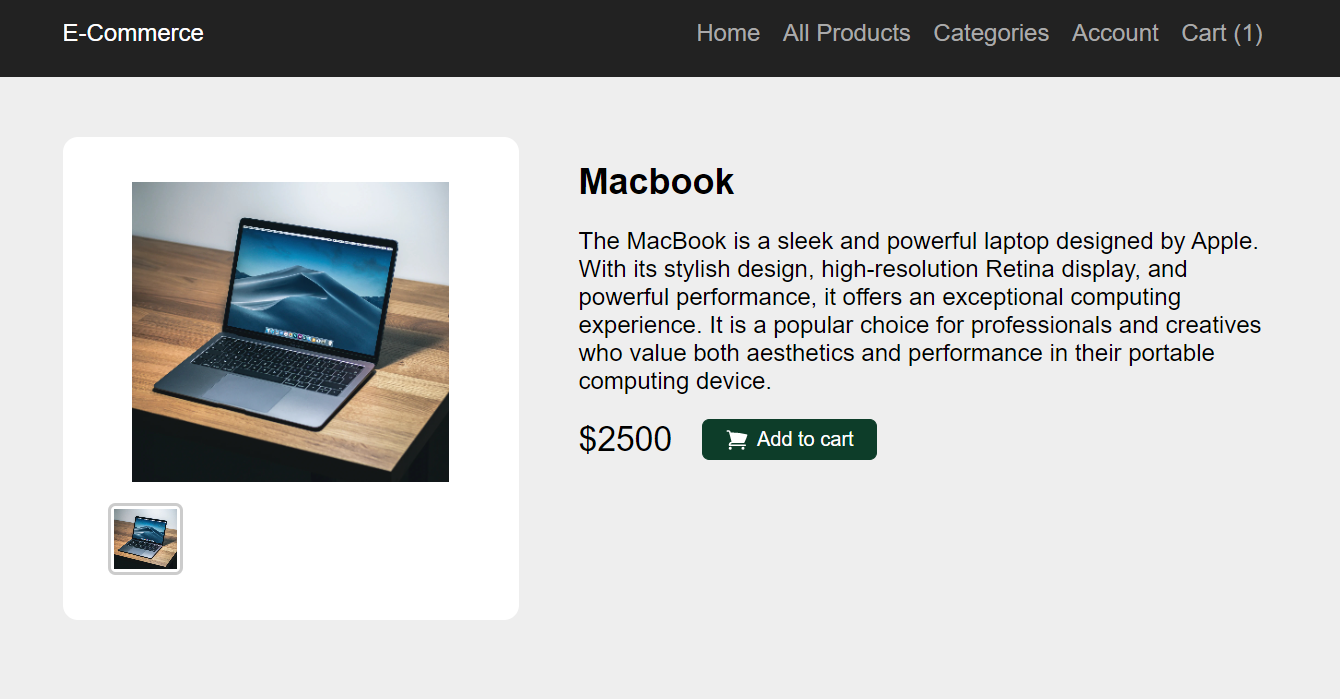
HOME PAGE



ALL PRODUCTS PAGE

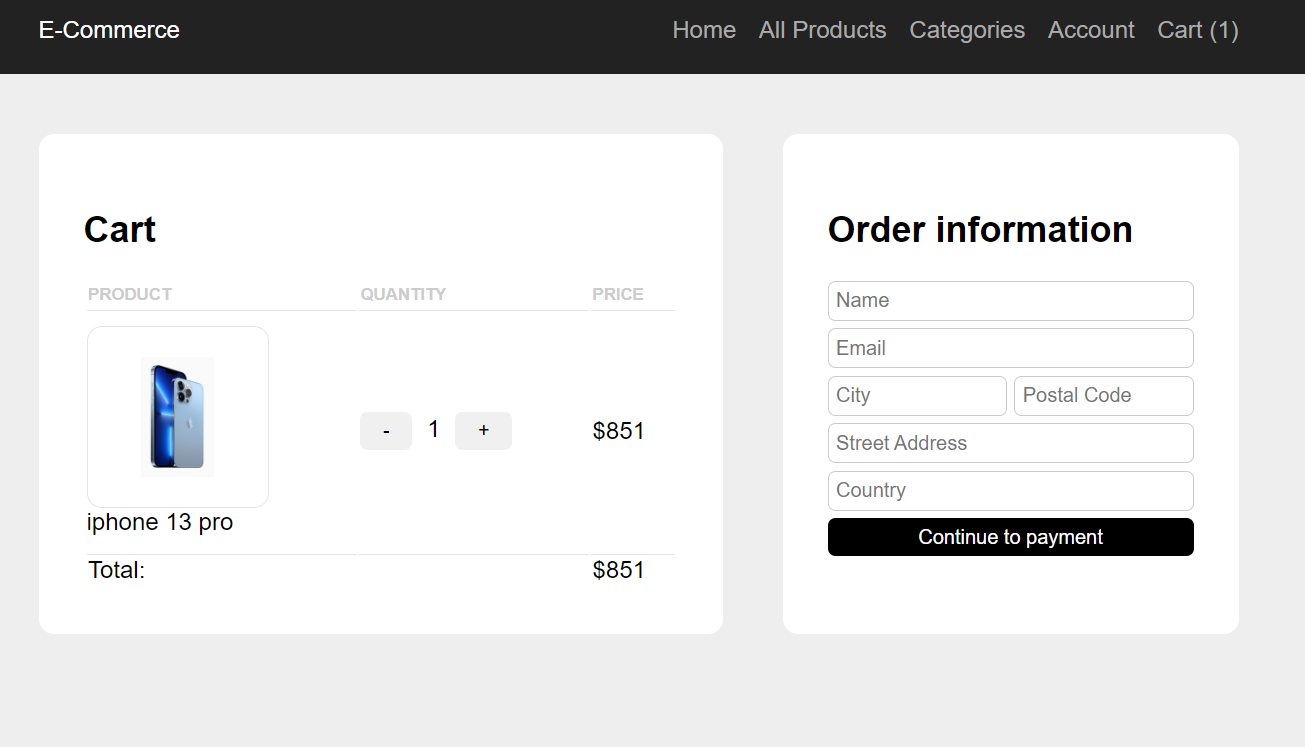


SINGLE PRODUCT PAGE



CART PAGE

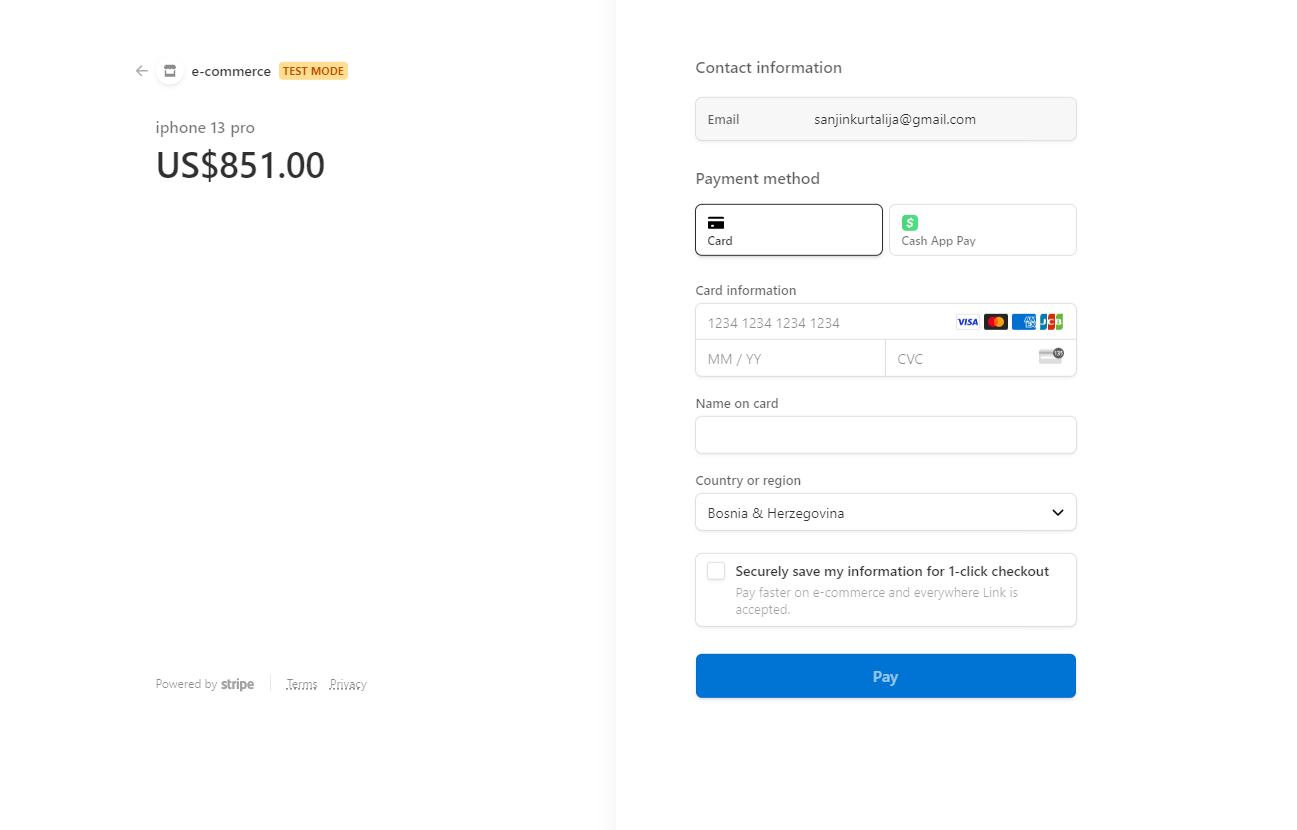
Ability to add or remove items on the previous pages by clicking add to cart. You can check your quantity, price and change the quantity to your desire.



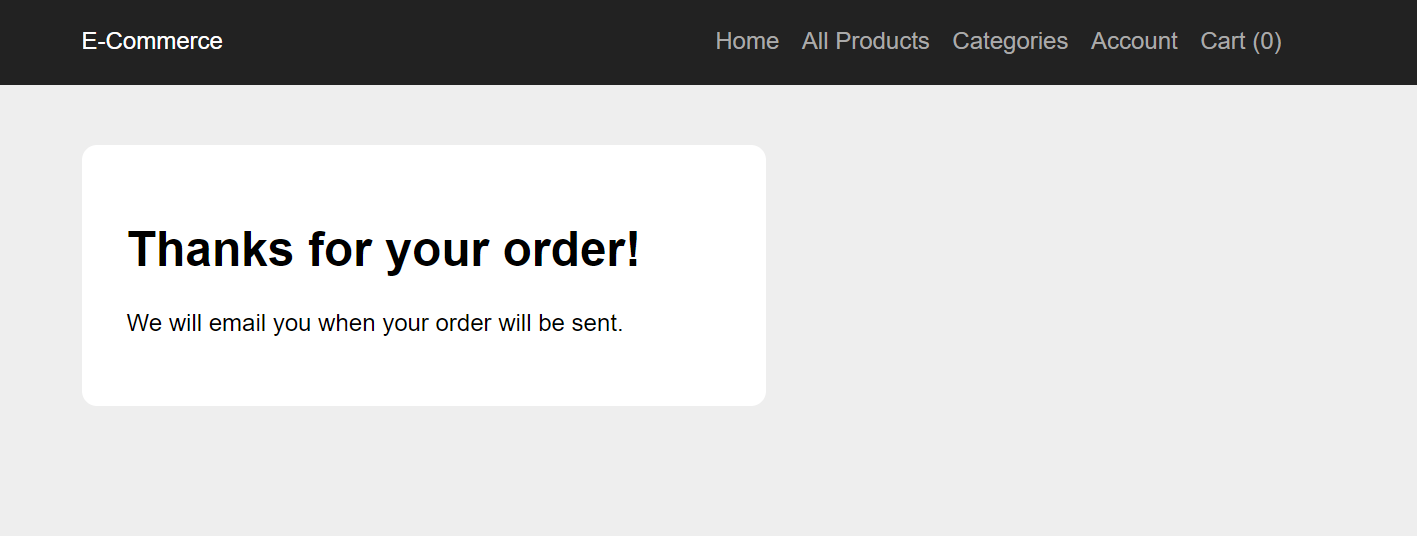
PAYMENT PAGE

Use 4242424242424242 as card number for the successful payment.

Use 4000000000009995 as card number for the failed payment.



SUCCESSFUL PAYMENT PAGE



# 2. Project Structure

## 2.1. Technologies

Technologies that we used are: React.JS, Next.js, MongoDB as our database. For the Admin we have used Tailwind CSS, while in Front we have used styled-components. Also we have used Stripe to implement our Payments on the Front-End part.

## 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:

* accounts
* categories
* orders
* products
* sessions
* users

## 2.3. Design Patterns

* **Component-based architecture:**

The code follows a component-based architecture, where different UI components such as Header, Button, and Table are defined as separate components. This approach promotes code reusability, modularity, and separation of concerns.

* **Conditional Rendering:**

The Link components within the navigation are conditionally rendered based on the current pathname obtained from the router object. This approach allows for rendering different styles based on the active or inactive state of the link.

* **Router Integration** :

The code integrates the Next.js useRouter hook to access the router object, allowing for programmatic navigation by using router.push('/');. It ensures a seamless navigation experience within the application.

## 2.4. Tests

All of these test are done on Front part of our project

* CartContext

This test suite verifies the functionality of the CartContext component. It ensures that the CartContext properly provides cart data to its children, and that the addProduct, removeProduct, and clearCart functions are called correctly. These tests validate the CartContext's ability to manage the cart products and handle cart-related actions accurately, ensuring the expected behavior of the component.

* Featured

The test "renders the featured product correctly" verifies that the Featured component displays the featured product accurately. It begins by mocking the featured product data, including its name, price, and description. The component is then rendered with the mocked data. The test proceeds to assert that the rendered output contains the expected product name, price, and description. This ensures that the Featured component properly renders the provided product information.

* ProductsPage

The test "renders all products correctly" verifies that the ProductsPage component renders the header, title, and products grid components accurately. It checks if the header is rendered and if the title component displays the expected text "All products". Additionally, it ensures that the products grid component is rendered with the correct number of products, matching the length of the mocked products array. The nested test "getServerSideProps" validates that the server-side data fetching function successfully retrieves products from the server. It confirms that mongooseConnect and Product.find are called with the correct parameters, and verifies that the returned server props contain the expected products data

# 3. Conclusion

The project showcases our understanding of essential e-commerce features, including product listing, cart management, and a streamlined checkout process. The incorporation of technologies like Mongoose, Axios, and server-side rendering exemplifies our ability to integrate backend functionality and improve performance.

Overall, our Next.js e-commerce project demonstrates their competence in full-stack development, showcasing our ability to apply theoretical knowledge to practical software engineering projects.