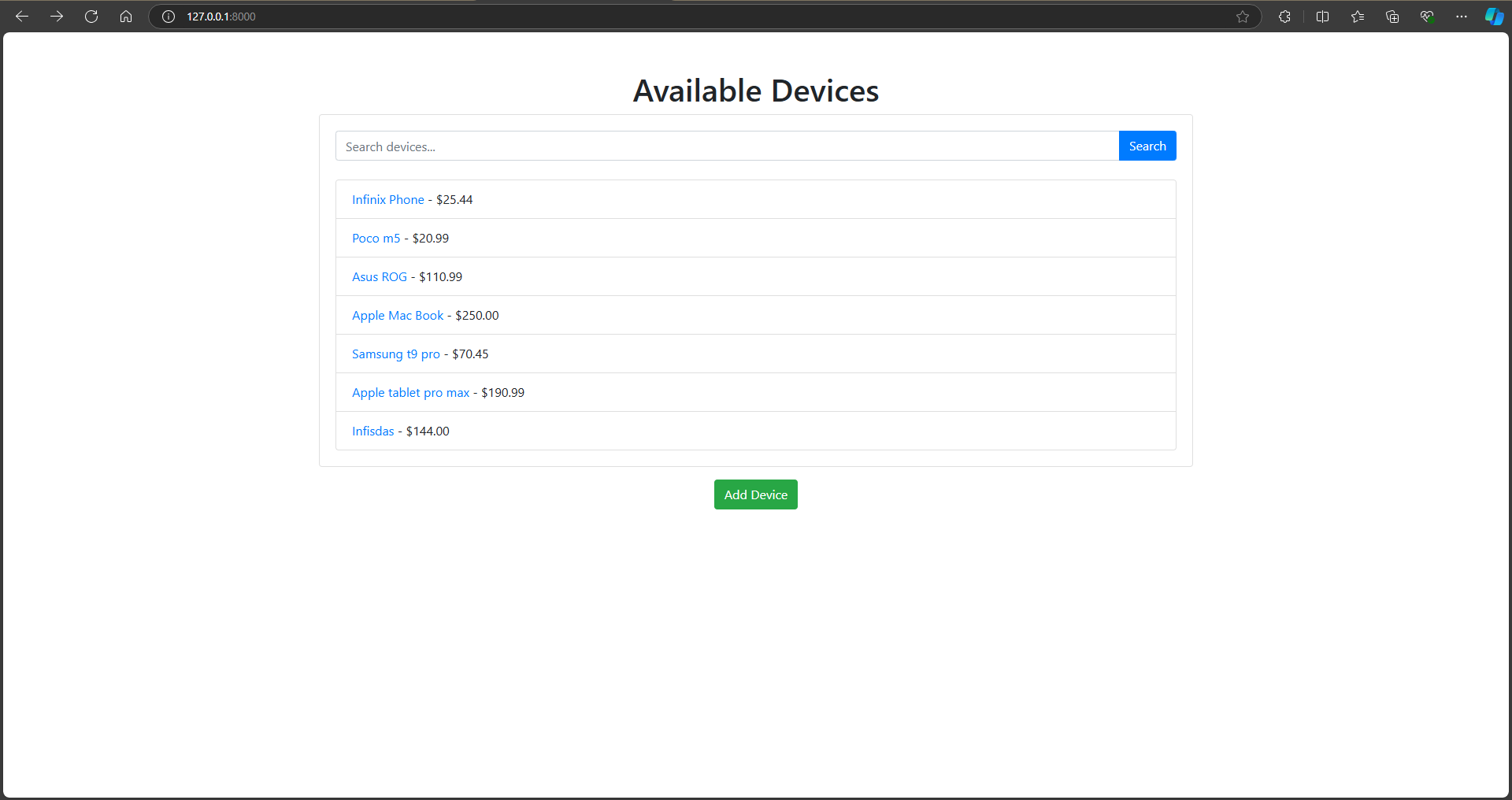
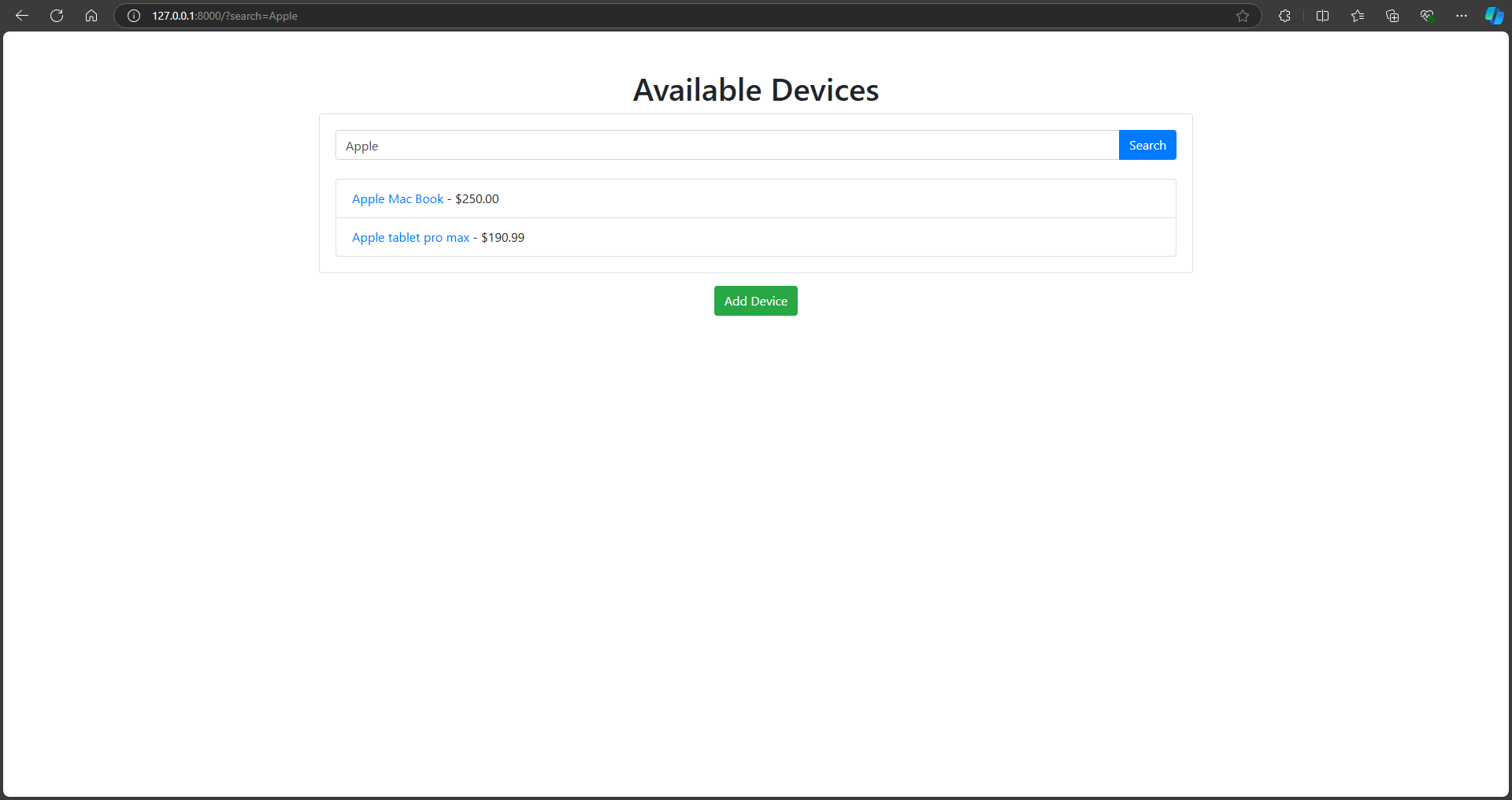
**Build a Model :**

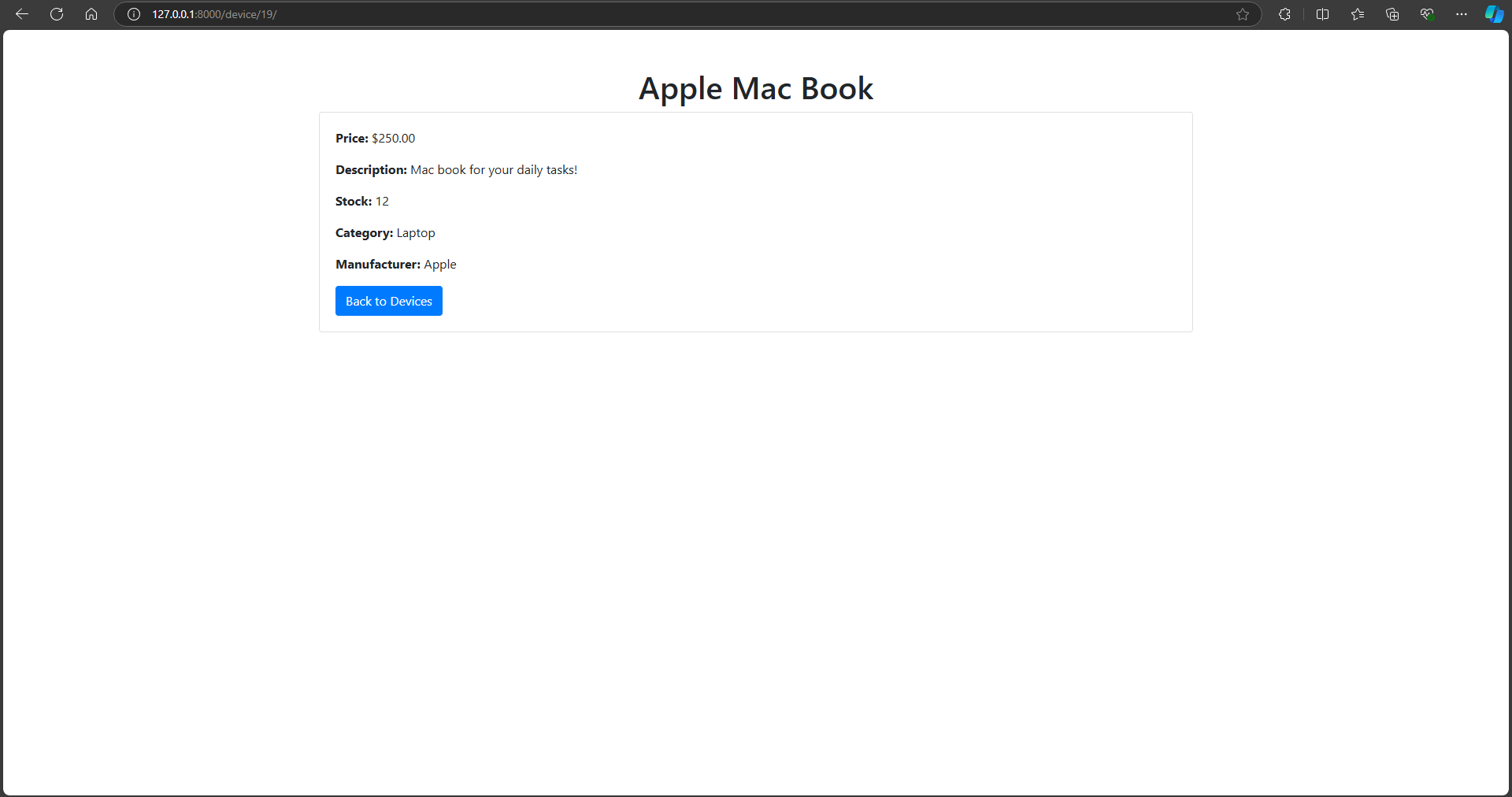
1. **Task Description**

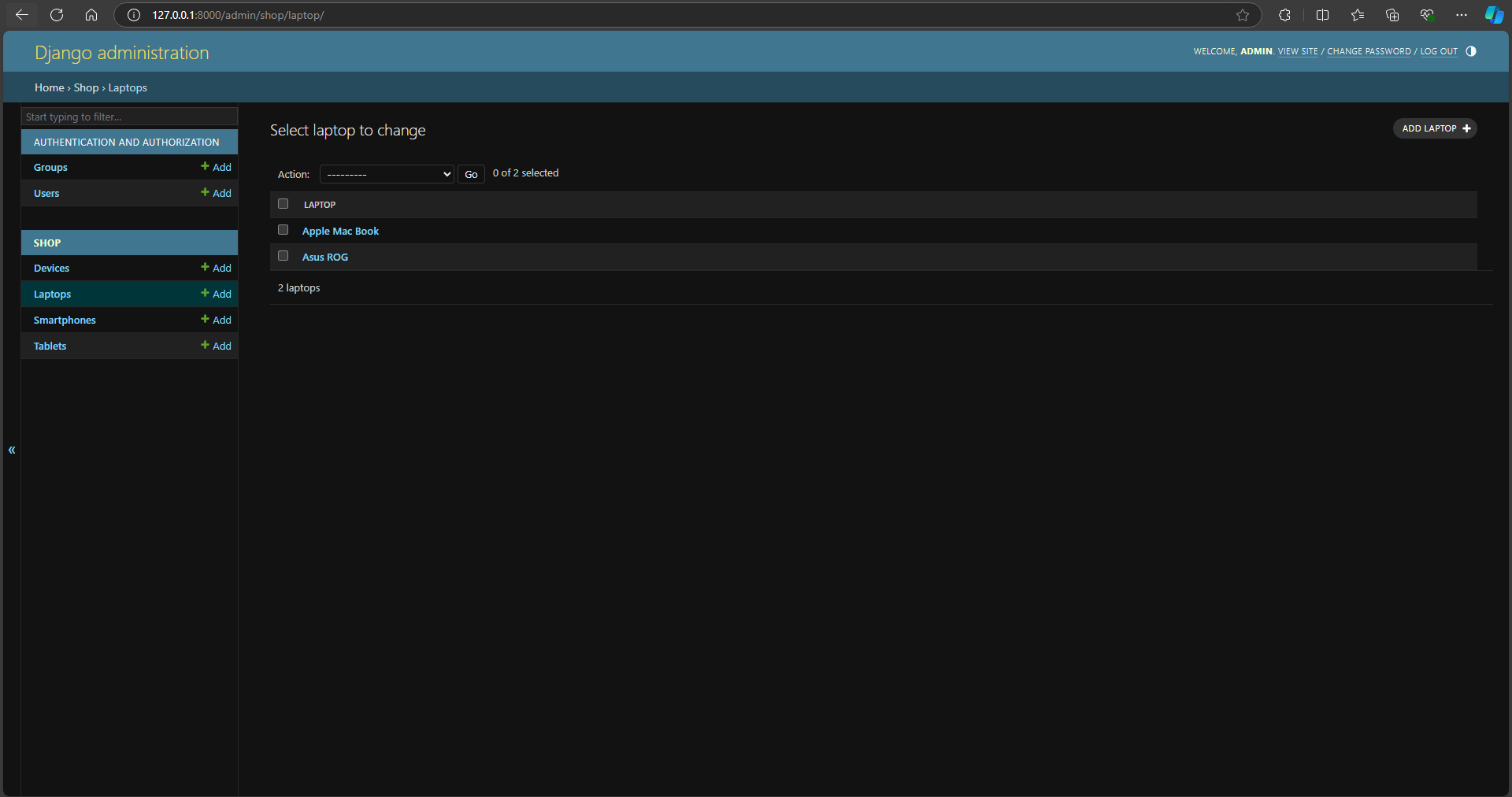
This Django-based project showcases the implementation of polymorphism and encapsulation in an online shop for electronic devices. The project allows users to browse, search, and view details of different device categories, including smartphones and laptops, with a user-friendly interface. Polymorphism is demonstrated through dynamic behavior in device types, while encapsulation ensures secure manipulation of device attributes, such as stock. The admin panel provides control over device management, and users can also add new devices via a pop-up form. This project effectively integrates key object-oriented programming principles into a web application.

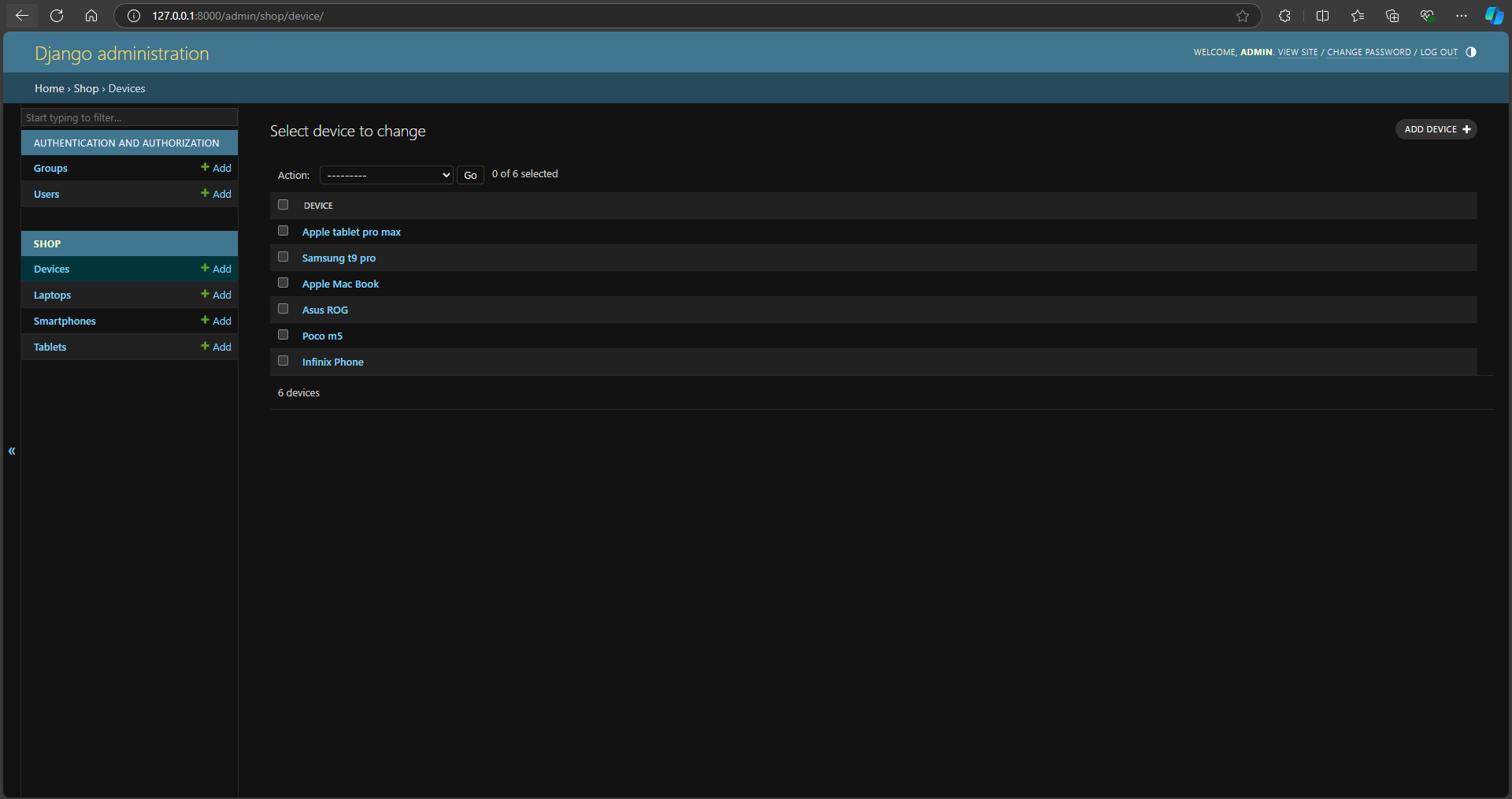
1. **Task Output Screenshot**

****









## Project Setup

### Starting the Project-

### The project begins by setting up a new Django environment. Django is selected for its ease of use in handling both the backend and frontend components. Initial setup involves configuring necessary settings, installing dependencies, and initializing the database using migrate commands.

### Creating the Model-

The core of the project revolves around defining the Device model and its subclasses, such as Smartphone and Laptop, which represent different types of electronic devices. Each device has essential fields like name, price, description, stock, category, and manufacturer. Encapsulation is enforced via getter and setter methods for private fields like stock.

## User Interface Development

### Data Insertion Form-

### A pop-up form allows users to add new devices dynamically. The form collects input such as device name, base price, stock, category, and manufacturer, ensuring all data is captured and stored in the database. The form is integrated into the homepage and enhances user interaction.

### Displaying Inserted Data-

### The devices uploaded by users or added via the admin panel are displayed on the homepage in a styled box. Each device includes its name and a link to view more details, such as description, category, and stock. This user interface ensures easy navigation and device exploration.

## Functionality Improvements

### Categorizing Inserted Data-

Devices are dynamically categorized based on their type (e.g., Smartphone, Laptop). This categorization demonstrates polymorphism, as each device can be of a different class but is handled uniformly in the interface. This helps keep everything organized.

### Admin Interface-

### The admin panel is enhanced to display device names instead of generic object names, facilitating easier management. The Django admin interface allows authorized users to add, modify, and delete devices efficiently.

## Conclusion

This project highlights the integration of object-oriented programming concepts—specifically polymorphism and encapsulation—within a Django web application. With a user-friendly interface, dynamic device management, and well-organized admin tools, the project provides an efficient solution for an online electronic shop system. It balances frontend aesthetics with backend functionality, ensuring a smooth experience for both users and administrators.