

WEB APPLICATION DEVELOPMENT USING NODEJS - 502070

LAB SESSION 3-4

By Mai Van Manh

OBJECTIVES

1. Develop a website using Node.js with the [Express](#) framework.
2. Explore the concept of view engines and implement them in the project, such as [EJS](#).
3. Utilize [npm](#) tool to install external modules for enhanced functionality.
4. Investigate the [middleware](#) concept within Express JS.
5. Familiarize yourself with modules that assist in processing HTML forms, such as [body-parser](#), [multer](#), [cookie-parser](#), [express-session](#), and more.
6. Handle [HTML form](#) data sent in various formats like [x-www-form-urlencoded](#), [multipart/form-data](#), and [JSON](#).
7. Pass data to the web application from [environment variables](#).
8. Synthesize the acquired knowledge to create a simple product management exercise.

EXERCISE. Building a website to perform basic product management functions as follows:

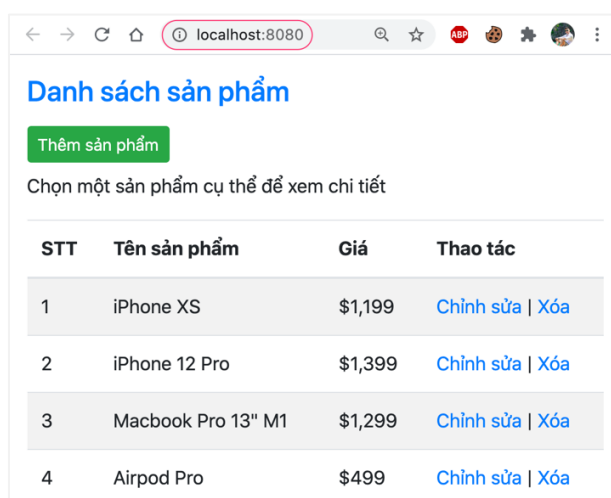
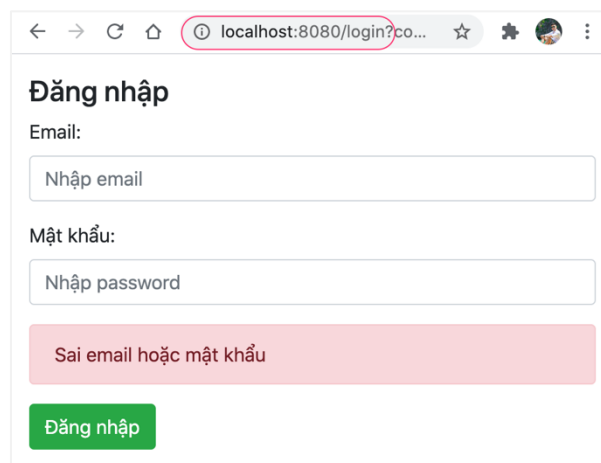


Figure 1. Homepage interface displaying the list of products

- **Login Page (/login)**: Users are required to log in before accessing any content on the website. If accessing any content (as described below) without logging in, they will be redirected to the login page.
 - o **GET**: Display the login interface.
 - o **POST**: Handle login and display errors if any. Account information needs to be read from **environment variables** (with username and password created by the student). If login is successful, redirect to the homepage.

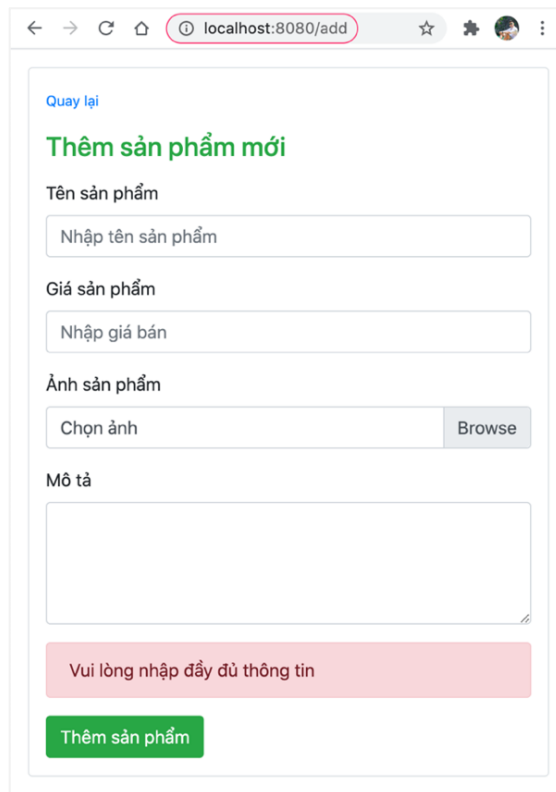


The screenshot shows a web browser window with the address bar displaying 'localhost:8080/login?co...'. The page title is 'Đăng nhập'. It contains two input fields: 'Email:' with a placeholder 'Nhập email' and 'Mật khẩu:' with a placeholder 'Nhập password'. Below these fields is a red error message box that says 'Sai email hoặc mật khẩu'. At the bottom is a green button labeled 'Đăng nhập'.

Figure 2. The login interface

- **Home Page**: Display a table containing the list of products. If the user is not logged in, automatically redirect to the login page.
- **Product Detail Page (/id)**: Display detailed information about a product. A product includes information such as product code, product name, selling price, illustrative image, and description. If the **product's id** is invalid, either redirect or display an appropriate error message.
- **Add Product Page (/add)**:
 - o **GET**: Display an HTML form interface to input a new product. It should include the ability to upload an illustrative product image.

- **POST:** Handle the functionality of adding a product and display errors if any. If successful, redirect to the homepage and display a **toast message** indicating the successful addition of the product.



Quay lại

Thêm sản phẩm mới

Tên sản phẩm

Giá sản phẩm

Ảnh sản phẩm

Mô tả

Vui lòng nhập đầy đủ thông tin

Figure 3. Interface for adding a new product

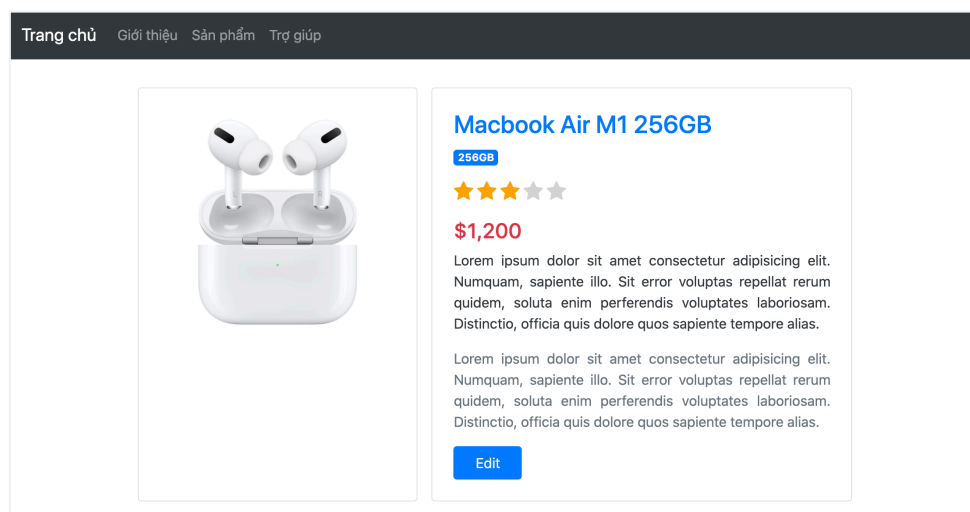


Figure 4. Interface for the detail page of a product

- **Edit Product Page (/edit/{id}):**
 - o **GET:** Display the information of the product with the corresponding **id** for adjustment/updating of necessary details. If the **product's id** is invalid, either redirect or display an appropriate error message.
 - o **POST:** Handle updating the product and display errors if any. If the update is successful, redirect to the homepage along with a toast message to indicate the successful update of the product.
- **Delete Product Page (/delete):**
 - o **POST:** Delete a product based on its **id**. Before deletion, display a **modal dialog** to confirm the action. After deletion, display a toast message to inform the user of the outcome (success/failure). Utilize ajax or fetch API to delete the product without the need to reload the webpage.



Figure 5. Confirm dialog displayed before deleting a product

Additional Requirements:

- Implement **rate limiting** feature to prevent DDOS attacks.
- If users access any path not described above, redirect them to the path **/error** and display a 404 not found error message interface.
- Utilize **Express.js** to create the web application.
- Choose a view engine to create the interface for the website, **EJS** is recommended.
- All form data should be validated using middleware such as **express-form**, **express-validator**, etc.
- Data only needs to be stored in memory; no need to employ storage techniques like files or databases.

- Configure login accounts through [environment variables](#).
- Limit file formats and sizes during uploads.
- The prompt provides a few sample interfaces; students should design the remaining interfaces based on requirements. Utilize [Bootstrap](#) or similar libraries to create visually appealing and responsive interfaces.