VIETNAM NATIONAL UNIVERSITY, HANOI

UNIVERSITY OF ENGINEERING AND TECHNOLOGY

FRAMEWORK REPORT FOR ONLINE ECOMMERCE WESITE

SEMESTER 232

GROUP 10

Group members:	**Phan Anh Tú**	**Student ID: 22028238**
	Mai Anh Tuấn	**Student ID: 22028144**
	Nguyễn Ngọc Hưng	**Student ID: 22028142**
	Nguyễn Đức Phát	**Student ID: 22028298**
	Nguyễn Nhật Phong	**Student ID: 22028272**

Course: Software Engineering

Course ID: INT2208E 23

Instructor: **Assoc. Prof. Đặng Đức Hạnh**

HÀ NỘI – 2024

Date	Document version	Description	Author
27th, March, 2024	1.0	**Section details:** Chapter I. Backend 1. Introduction 2. Components 2.1. Routers 2.2. Model (M) 2.3. View (V) 2.4. Controller (C) 3. Usage 3.1. Routes 3.2. Resource 3.3. Public 3.4. Config.db 3.5. Apps Chapter II. Frontend 1. Introduction 2. Components 2.1. Redux 2.2. Virtual Dom 2.3. JSX (JavaScript XML) 3. Usage 4. Benefits 5. Limitations 6. Conclusion	 Nguyễn Ngọc Hưng Nguyễn Ngọc Hưng Mai Anh Tuấn Mai Anh Tuấn Mai Anh Tuấn Nguyễn Ngọc Hưng Phan Anh Tú Phan Anh Tú Mai Anh Tuấn Nguyễn Ngọc Hưng Nguyễn Đức Phát Nguyễn Đức Phát Nguyễn Nhật Phong Nguyễn Đức Phát Nguyễn Đức Phát Nguyễn Nhật Phong Nguyễn Nhật Phong Nguyễn Đức Phát

Contents

Table of Figures

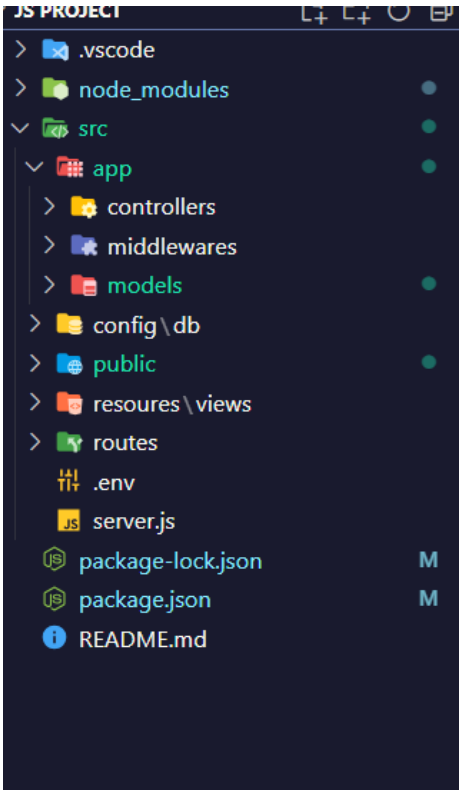
Chapter I. Backend

1. Introduction

In this project, we have chosen Node.js as the backend platform for our web application, specifically utilizing Express.js. Express.js is a minimalist web application framework for Node.js, designed to build web applications and APIs quickly and with minimal effort. It provides a robust set of features to develop web and mobile applications. Additionally, Express.js offers a clear routing system and a range of middleware, which is essential for us to create a website following the MVC (Model-View-Controller) architecture.

For a web-based e-commerce project like ours, accuracy and speed are paramount. We need a framework that allows for swift and precise interactions between sellers and their customers in real-time. In this regard, selecting Express.js for the backend is our top choice.

2. Components



Now I will describe some of the components of the project. As mentioned earlier, this project is supported by Express.js following the MVC (Model - View - Controller) pattern, so the components in the project are supported accordingly:

2.1. Routers:

- Express.js provides a robust routing system that allows developers to define routes for handling various HTTP requests (GET, POST, PUT, DELETE, etc.). Routes can be defined to match specific URL patterns and are associated with functions (handlers) that execute when a matching request is received.

2.2. Model(M):

- Represents data and business logic.

2.3. View (V):

- Presents data to users and handles user interactions.

2.4. Controller (C):

- Intermediary between model and view, handles user input, processes data, and updates the view accordingly.

3. Usage

3.1. Routes

Figure 1. routes detail

- The router plays a pivotal role in defining the routes for the website. It handles the - - configuration of endpoints, which are crucial for delivering specific functionalities to users.
- This component orchestrates the flow of requests within the website, determining which endpoints utilize middlewares or JWT (JSON Web Tokens) for user authentication.
- Moreover, it serves as an interface for frontend developers to collaborate, allowing them to integrate user interface elements seamlessly.

3.2. Resource

Figure 2. resources section detail

- The resource section serves as the initial workspace for frontend development. It encompasses files responsible for rendering interface elements such as headers, bodies, and footers.
- Before advancing to more sophisticated frameworks like React.js, developers utilize this section to optimize user interaction by preloading necessary resources and enhancing frontend-backend connectivity.

3.3. Public

Figure 3. public directory detail

- In the public directory, essential resources are made readily available to users. This includes static files like logos and backgrounds, which are integral for creating an immersive user experience.
- Express.js configures and routes this directory, ensuring quick access to resources without the need for fetching data from external APIs.

3.4. Config/db

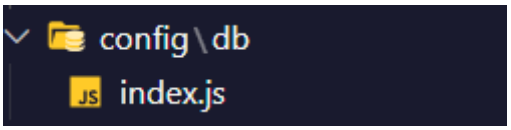


Figure 4. config.db detail

- This section is dedicated to providing support for database operations. It enables seamless establishment of connections to preferred data platforms, facilitating efficient data management and retrieval.

3.5. Apps

Figure 5. app detail

3.5.1.Controllers

- Controllers serve as the backbone of the website, acting as intermediaries between routes and models. They encapsulate the logic responsible for executing actions corresponding to specific routes.
- Configured in alignment with route names, controllers leverage asynchronous JavaScript functions for tasks such as data retrieval and route redirection, ensuring smooth user interactions.

3.5.2.Middlewares:

- Middlewares are instrumental in defining route protections and enforcing user authentication mechanisms. By leveraging user cookies and other authentication strategies, they bolster the website's security posture.
- Serving as a critical component of Authentication and Authorization, middlewares play a pivotal role in safeguarding sensitive data and functionalities.

3.5.3.Model

- Models define the data schema and structure, providing a blueprint for database tables related to users and other entities.
- By ensuring data integrity and coherence, models contribute to a well-organized backend architecture and facilitate seamless data management processes

4. Benefits

- **Speed and Scalability:** Node.js's non-blocking, event-driven architecture ensures fast and efficient handling of concurrent connections, making it ideal for handling high traffic loads in e-commerce scenarios.
- **Rapid Development:** Express.js's minimalist framework allows for quick feature development with minimal boilerplate code, enabling businesses to iterate and deploy new features rapidly.

- **Middleware Support:** Express.js provides a rich set of middleware for common tasks like request parsing, authentication, and error handling, reducing development time and ensuring consistent behavior across the application.
- **Flexible Routing:** Express.js offers a powerful routing system for defining complex routing logic, essential for dynamic e-commerce applications with multiple endpoints.
- **Microservices Architecture:** Node.js and Express.js support a microservices-based architecture, allowing for better scalability, fault isolation, and maintainability of the e-commerce platform.
- **Rich Ecosystem:** Node.js's vast ecosystem of third-party modules and libraries via npm allows developers to leverage existing solutions for functionalities like payment processing, data analytics, and CRM integration.
- **Real-time Updates:** Node.js enables real-time communication between server and clients, facilitating features such as live product updates, chat support, and real-time order tracking, enhancing the shopping experience.

5. Limitations

Express.js has a few limitations:

- **Minimalism:** While its minimalist design is advantageous for simplicity, it may lack certain features found in more feature-rich frameworks.
- **Scalability:** Although Express.js is capable of handling moderate to high traffic loads, it may require additional effort to optimize for extremely large-scale applications.
- **Learning Curve:** While easy to start with, mastering advanced concepts and best practices in Express.js may take time and effort.

6. Conclusion

In conclusion, Express.js significantly contributes to our e-commerce website project by providing a modular program structure, facilitating the implementation of the MVC architecture, and enhancing performance through its non-blocking I/O capabilities. Its flexibility, clear routing system, and middleware support enable us to build a well-organized, scalable, and high-performing web application that meets the demands of modern e-commerce environments.

Chapter II. Frontend

1. Introduction

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library. The main purpose of ReactJS in a project context is to provide a reactive approach and easily manage the user interface components of web applications.

2. Components

2.1. Redux

Redux is an extremely important part of ReactJS and is commonly used. In a ReactJS there are no dedicated modules to process data, so it is set up independently by dividing the view into different components to help them connect better together. The connection and relationships between components in ReactJS need special attention because there is only one data flow, which is the data flow from parent to child. Using this one-way data flow is somewhat difficult for those new to using and applying it to projects. Besides the limitations, ReactJS can promote all its functions and roles while using this one-way mechanism. Because the functions of the view become much more complex.

2.2. Virtual Dom

Virtual Dom is an important part that almost every framework uses it as ReactJS. Users do not need to operate directly on the Dom but can still see the view and changes. Because Virtual Dom acts as a model and also as a view, a change in one of the two factors will cause the other factors to change. On the contrary, if you do not manipulate Dom molecules directly, you can still implement Data Binding mechanisms.

2.3. JSX (JavaScript XML)

ReactJS uses JSX, a programming language that combines JavaScript and HTML in a way that is easier to read and understand.

3. Usage

In our project, we use ReactJS to develop the user interface of the web application. Each interface element is built as React components, making it easy to split and reuse code. JSX allows us to write encoded HTML inside JavaScript, creating code that is easy to read and maintain.

4. Benefits

- **High performance**: ReactJS uses Virtual DOM to optimize application performance. Virtual DOM allows ReactJS to update changes on a web page faster and more efficiently than the traditional way, increasing application speed and performance.
- **Reuse**: ReactJS allows reuse of UI components, helping to reduce development time and costs. UI elements can be reused in many different parts of the application, increasing the flexibility and scalability of the application.

5. Limitations

- React.js is only a view layer.
- Integrating React.js into a traditional MVC framework such as rails would require some configuration.
- There is a learning curve for beginners who are new to web development.
- Difficult to reach for beginners / beginners in web programming

6. Conclusion

ReactJS is a powerful tool for developing the user interface in our project. With features like Virtual DOM and JSX, ReactJS helps optimize performance and increase flexibility in web application development

References

1. React - A JavaScript library for building user interfaces
2. ExpressJS Documentation
3. NodeJS Documentation