Framework Report

for

Hotel Booking Website Project

Version 1.0

Prepared by

Group 6

Vu Nguyet Hang	**22028079**	**22028079@vnu.edu.vn**
Duong Quoc Khanh	**22028090**	**22028090@vnu.edu.vn**
Nguyen Thi Thu Trang	**22028254**	**22028254@vnu.edu.vn**
Nguyen Van Son	**22028020**	**22028020@vnu.edu.vn**
Nguyen Truong Giang	**19021259**	**19021259@vnu.edu.vn**

Instructor: **Assoc. Prof. Dr. Dang Duc Hanh**

Course: **INT2208E_23**

Teaching Assistant: **Kieu Van Tuyen**

Date: **29/03/2004**

CONTENTS

REVISIONS

Version	Primary Author(s)	Description of Version	Date Completed
#1	Team 6	Initial Version	28/03/24

1. React

1.1 Introduction

React is a JavaScript library for building user interfaces. It is used in the booking app to create a dynamic, interactive, and modular UI, improve performance, and simplify state management, resulting in an engaging and user-friendly booking experience.

1.2 Components

Children Props

- Sometimes it becomes necessary to render components within a parent component.
- We can pass functions as children props which get called components render function.
- Children props are used to display whatever you include between the opening and closing tags when you invoke a component.

Higher Order Components

- It is an advanced technique in React which allows reusing component logic.
- A higher order component can be used to transform one component into another component.
- Concretely, a higher-order component is a function that takes a component and returns a new component.

Decompose into Small Components

- Try to decompose large components to small components such that each component performs one function as much as possible.
- It becomes easier to manage, test and reuse small components.

Functional Components with Hooks

- After the release of React v16.08, it is possible to develop function components with state with the new feature 'React Hooks'.

- It reduces complexity of managing states in Class components. So always prefer to use functional components with React Hooks like useEffect(), useState() etc.

Use of Functional & Class Components

- If you need to show Ul without performing any logic or state change, use functional components in place of class components as functional components are more efficient in this case.
- React lifecycle methods like componentDidMount(), componentDidUpdate() etc. cannot be used with functional components, but can be used with Class components.
- While using functional components, you lose control over the render process. It means with the small change in component, the functional component always re-renders.

1.3 Usage

Using React Components in a Hotel Booking Web Project:

Authentication Components:
- Login Form: Responsible for user login.
- Registration Form: Handles new user registration.
- User Profile: Displays user details after successful login.

Listing and Booking Components:
- Listing Cards: Render information about available accommodations.
- Booking Form: Allows users to book a listing.
- Booking Confirmation: Displays booking details after successful reservation.

Context Provider and Consumer Components:
- State Management: React's context API is used to manage global state (e.g., user authentication status, booking data).

Navigation Components:
- Navbar: Provides navigation links (e.g., Home, Listings, Profile).
- Protected Routes: Ensures that certain routes are accessible only when authenticated.

React Functions and Hooks:
- useState: Manages local component state (e.g., form input values).
- useEffect: Handles side effects (e.g., fetching data from the server).
- useContext: Facilitates global state management.
- useHistory: Navigates between different routes programmatically.

Integration with Backend (Node.js and Express):
- React components interact with the backend API (built using Node.js and Express) to fetch data (e.g., listings, bookings).
- Axios or Fetch API is likely used for making HTTP requests.

Styling with CSS (and Possibly Tailwind CSS):
- While the project primarily focuses on React, it may also incorporate CSS or other styling frameworks like Tailwind CSS for a clean and responsive design.

1.4 Benefits

- **React Framework Advantages:**

Performance: React's Virtual DOM and efficient rendering process result in faster loading times and smoother user experiences, crucial for a hotel booking app.

Reusability: React's component-based architecture encourages the creation of reusable and modular components, simplifying development and maintenance.

Declarative Programming: React's declarative paradigm allows developers to describe the desired UI state, making code more readable and maintainable.

SEO-friendly: React applications can be easily optimized for search engines, making them more discoverable and accessible to potential customers.

Flexibility: React's flexibility allows for building a wide range of web applications, from simple websites to complex enterprise applications.

Easy to Learn: React's syntax is relatively simple and straightforward, making it easier for developers to learn and adopt.

- **Comparison with other frameworks:**

Angular: Offers built-in features like routing, forms, and HTTP client, but can be less flexible than React and require more customization.

Vue.js: Provides a balance between simplicity and functionality but might not be suitable for complex web applications like the MERN Booking App project.

Svelte: Delivers high performance and small bundle size, but may have a steeper learning curve and fewer supporting resources than React.

1.5 Limitations

State Management Complexity: Managing state in large applications can become complex, especially when dealing with nested components and multiple data sources.

Lack of Built-in Solutions: React does not provide built-in solutions for routing, state management, and internationalization, requiring the use of third-party libraries.

Performance Bottlenecks: Large and complex React applications can face performance bottlenecks, requiring optimization techniques and careful component design.

Over-reliance on Third-Party Libraries: Depending on numerous third-party libraries can introduce compatibility issues, increase bundle size, and make code maintenance more challenging.

Rapid Ecosystem Changes: React's fast-paced development can lead to frequent API changes and breaking updates, requiring developers to constantly adapt and learn new techniques.

SEO Challenges: React applications can face challenges with SEO due to their dynamic nature, requiring additional effort for proper search engine indexing and crawling.

Debugging Difficulties: Debugging React applications can be complex, especially when dealing with state and component interactions, requiring specialized tools and knowledge.

1.6 Conclusion

React framework is integral to developing a hotel booking web project, offering a wealth of components like authentication forms and listing cards to streamline user interaction. Its benefits include superior performance, reusability, and scalability, supported by a vast community. However, challenges like state management complexity exist, debugging difficulties. Despite this, React's flexibility and robustness make it a preferred choice for building dynamic and efficient web applications.

2. Tailwind css

2.1 Introduction

Tailwind CSS is a utility-first CSS framework designed to enable users to create applications faster and easier. You can use utility classes to control the layout, color, spacing, typography, shadows, and more to create a completely custom component design — without leaving your HTML or writing a single line of

custom CSS. It is used in the booking app to rapidly build and maintain a consistent UI, while also promoting responsive design for different screen sizes and devices.

2.2 Features

Utility-First Approach: Tailwind CSS is based on a utility-first approach, providing a vast set of utility classes that directly apply styling to HTML elements. This approach offers flexibility and enables developers to quickly prototype and style their applications without writing custom CSS.

Responsive Design: Tailwind CSS includes built-in support for responsive design, allowing developers to easily create layouts that adapt to different screen sizes and devices. Responsive utility classes can be used to control the appearance of elements based on breakpoints.

Customization: One of the standout features of Tailwind CSS is its customization capabilities. Developers can easily customize the default theme and configure their own design system by editing the configuration file. This allows for consistency in design and branding across projects.

Modular Scale: Tailwind CSS provides a modular scale for typography, enabling developers to create harmonious and visually appealing text styles by applying predefined classes that scale proportionally.

Flexbox and Grid Utilities: Tailwind CSS includes utility classes for working with Flexbox and CSS Grid, making it easy to create flexible and responsive layouts. Developers can utilize classes such as flex, justify, align, and grid to control the layout and alignment of elements.

Component Compositions: While Tailwind CSS primarily focuses on utility classes, developers can leverage component compositions to create reusable UI components. These compositions are reusable patterns of utility classes that encapsulate common design patterns and components.

Dark Mode Support: Tailwind CSS provides built-in support for implementing dark mode in web applications. Developers can easily toggle between light and dark mode by applying utility classes or using JavaScript to update the theme dynamically.

Theming: Tailwind CSS supports theming through its configuration file, allowing developers to define custom color palettes, typography scales, spacing scales, and more. This makes it easy to maintain a consistent design system and adapt the framework to different project requirements.

Plugin Ecosystem: Tailwind CSS has a vibrant plugin ecosystem that extends its functionality beyond the core framework. Developers can find plugins for adding additional utilities, integrating with popular UI frameworks, optimizing the build process, and more.

2.3 Usage

Using Tailwind CSS Components in a Hotel Booking Web Project:

User Authentication:

- The web provides secure login functionality.
- Tailwind CSS styles the login form, buttons, and other UI elements related to authentication.

Listing and Booking Functionality:

- Customers can find and book accommodations.
- Tailwind CSS styles the listing cards, buttons, and other UI components related to listings and bookings.

Integration with MongoDB:

- Data related to listings, bookings, and user authentication is stored and retrieved using MongoDB.
- Tailwind CSS ensures consistent styling across these data-driven components.

State Management with useContext:

- Tailwind CSS styles the context provider and consumer components.
- These components manage global state using React's useContext hook.

2.4 Benefits

- **Tailwind Css Framework Advantages:**

Easy customization: Tailwind CSS offers high customizability, allowing you to easily tailor the UI to the project's specific requirements.This empowers you to create a unique UI that aligns with the project's branding.

Utility-First Approach: Tailwind CSS adopts a utility-first approach, providing a vast array of utility classes that directly apply styling to HTML elements. This approach streamlines the styling process, reduces the need for writing custom CSS, and makes it easier to prototype and iterate on designs quickly.

Minimal CSS File Size: Unlike traditional CSS frameworks that may include a large number of predefined styles, Tailwind CSS generates minimal CSS file sizes by only including the utility classes that are used in the project. This results in faster loading times and improved performance for web applications.

Rapid Development: Tailwind CSS accelerates the development process by offering a comprehensive set of utility classes for common styling tasks, such as layout, typography, spacing, and colors. Developers can quickly prototype and style their applications without needing to write extensive CSS code, resulting in faster development cycles.

Consistent Design Language: Tailwind CSS promotes consistency in design by providing a predefined set of utility classes that adhere to a consistent design language. This consistency ensures that UI elements

maintain a cohesive look and feel across different parts of the application, enhancing the overall user experience.

- **Tailwind Css vs. Other Popular Frameworks:**

Bootstrap: Provides pre-built components but can be bulky and challenging to customize.

Material UI: Offers Material Design aesthetics but might not be suitable for all projects.

Ant Design: Furnishes a vast array of components and features but can be difficult to learn and use.

2.5 Limitations

Steeper Learning Curve: Tailwind CSS requires a different mindset and way of thinking compared to traditional CSS frameworks. Its utility-first approach can be initially overwhelming, necessitating time and practice to master its extensive class names and syntax.

Debugging Difficulties: Since Tailwind classes are directly applied to HTML elements, debugging styling issues can be more challenging compared to frameworks with encapsulated components. Identifying the specific class responsible for an unintended style can be time-consuming and requires a thorough understanding of Tailwind's class naming conventions.

Potential for Overly Complex Class Names: The flexibility of Tailwind CSS can lead to the creation of overly complex and lengthy class names, especially when combining multiple utility classes. This can make code difficult to read, maintain, and understand, potentially hindering collaboration and code reuse.

Dependency on Third-Party Plugins: Tailwind CSS lacks built-in support for certain UI elements and functionalities, necessitating the use of third-party plugins. This can introduce additional dependencies and potential compatibility issues, increasing the complexity of the project.

Balancing Customization and Consistency: Tailwind CSS's flexibility empowers extensive customization, but this freedom can lead to a lack of design consistency if not carefully managed. Maintaining a consistent design language across the application requires discipline and a well-defined style guide.

Maintenance Overhead: As projects grow in size and complexity, maintaining consistency and managing changes to the design system may become challenging, requiring careful planning and documentation to ensure scalability and long-term maintainability.

2.6 Conclusion

Tailwind CSS offers rapid development and flexibility for web projects like hotel booking platforms. Its utility-first approach simplifies styling and ensures consistency in design. Despite challenges, Tailwind CSS enables developers to create visually appealing and efficient interfaces tailored to project needs, making it an ideal choice for modern web development.

3. Express

3.1 Introduction

Express is a JavaScript framework for building web applications. In the booking app, it serves as the backend foundation, enabling the creation of a robust and scalable API for managing bookings and providing a seamless user experience. Express is used to define API endpoints, handle user requests, implement middleware functions, and manage errors.

3.2 Features

Routing: Express.js offers a powerful and flexible routing system that enables developers to map HTTP requests(GET, POST, PUT, DELETE, etc.) to specific functions or controllers. This allows for a clear and organized structure in handling different routes and their corresponding actions within the application.

Middleware: Express.js supports middleware, which are functions that process HTTP requests before they reach the route handler. Middleware can be used for various purposes, such as authentication, logging, error handling, and request parsing. This modular approach enhances the flexibility and reusability of code in Express.js applications.

HTTP Server: Express simplifies the process of creating HTTP servers in Node.js, providing an easy-to-use API for creating and configuring server instances, listening for incoming requests, and responding to them accordingly.

Templating: Express.js integrates with various templating engines, such as Pug (formerly Jade), EJS, and Handlebars, allowing developers to dynamically generate HTML pages based on data and variables. This

separation of concerns between logic and presentation simplifies the development and maintenance of web applications.

Error Handling: Express.js provides built-in error handling mechanisms to manage and respond to errors that may occur during the request-response cycle. This helps developers to gracefully handle errors and provide informative feedback to users, enhancing the overall user experience.

Security: Express.js offers various security features, such as CSRF protection, XSS prevention, and helmet integration, to help developers build secure web applications. These features mitigate common security vulnerabilities and protect against malicious attacks, ensuring the safety and reliability of the application.

Debuggability: Express.js provides excellent debuggability features, such as detailed error messages and stack traces, which help developers identify and fix issues quickly and efficiently. This reduces development time and improves the overall quality of the application.

Static File Serving: Express allows developers to serve static files such as HTML, CSS, JavaScript, images, and other assets from a specified directory, making it easy to include client-side resources in web applications.

Middleware Integration: Express seamlessly integrates with a wide range of third-party middleware and libraries, allowing developers to extend its functionality by adding features such as session management, authentication, authorization, and more.

HTTP Request and Response: Express provides a convenient API for working with HTTP request and response objects, including methods for accessing request parameters, headers, cookies, and body data, as well as sending responses with status codes, headers, and content.

Routing Middleware: Express allows developers to create modular and reusable route handlers using routing middleware, enabling better organization and encapsulation of application logic.

3.3 Usage

Using Express Components in a Hotel Booking Web Project:

Express Routes:

- The project defines various routes using Express to handle different functionalities.
- These routes map URLs to server-side functions (controllers).

Middleware:

- Middleware functions are used in the Express app to process requests before they reach the route handlers.
- Examples of middleware include:
- Body-parser: Parses incoming request bodies (e.g., form data, JSON) and makes it accessible in the request object.
- Authentication Middleware: Verifies user authentication before allowing access to certain routes.

Controllers:

- Controllers handle the business logic for specific routes.
- For instance:
- The authentication controller manages user login, registration, and token generation.
- The listing controller handles CRUD operations related to accommodations.

Routing:

- Express provides a clean way to define routes and their corresponding handlers.
- For example:
- GET /api/listings: Retrieves a list of available accommodations.
- POST /api/bookings: Creates a new booking.

Error Handling:

- Express middleware can handle errors (e.g., invalid routes, server errors).
- Custom error handlers can be defined to send appropriate responses.

Database Interaction:

- Express routes interact with the MongoDB database using Mongoose (an object modeling tool).
- Mongoose models and schemas define how data is stored and retrieved.

Integration with Other Components:

- It communicates with the React front end via API endpoints.
- It interacts with the MongoDB database to fetch and store data.

3.4 Benefits

- **Express Framework Advantages:**

Simplicity and Flexibility: Express is designed to be simple and flexible, allowing developers to customize their applications according to specific needs without encountering constraints.

High Performance: Express is a lightweight and optimized framework with high performance, enabling applications to run fast and efficiently.

Scalability: Express provides a range of middleware and tools that facilitate easy expansion of application functionality, including integration with various database systems.

Ease of Learning and Use: Express has a user-friendly syntax, making it easy for developers to understand and start building applications quickly.

Routing Flexibility: Express allows flexible routing, enabling efficient management of application routes and easy adjustments when needed.

Wide Range of Middleware Support: Express offers a variety of built-in and third-party middleware support, aiding in request and response handling in a flexible and powerful manner.

- **Express vs. Other Popular Frameworks:**

Koa: Provides concise and efficient syntax, but can be more challenging to learn compared to Express.

Hapi: Offers a rich set of features and structures, but can be more complex than Express.

NestJS: Provides object-oriented architecture and advanced features, but can be heavier than Express for smaller projects.

3.5 Limitations

Manual Configuration: Express requires manual configuration for common features like routing, middleware, and error handling, which can be time-consuming and error-prone, especially for beginners.

Middleware Management: Managing middleware in Express can become cumbersome as the application grows, leading to issues such as middleware duplication, order dependency, and difficulty in debugging middleware chains.

Error Handling Complexity: While Express.js provides error handling mechanisms, managing and debugging errors can be complex, especially in larger and more complex applications. Identifying the source of errors and implementing effective error handling strategies can be challenging.

Security Concerns: While Express.js offers built-in security features, developers still need to be vigilant about security best practices and implement additional security measures to protect their applications from potential vulnerabilities and attacks.

Lack of Built-in Features: Express.js is a minimalist framework and does not provide built-in support for certain functionalities, such as database connectivity, authentication, and user management. Developers need to rely on third-party libraries and modules to implement these features, which can add complexity and maintenance overhead.

Dependency Management: Using Express.js involves managing dependencies, such as third-party libraries and modules. This can add complexity to the project and requires careful consideration of compatibility and version updates.

3.6 Conclusion

Express JS framework provides a straightforward and adaptable solution for handling routes, middleware, and controllers in the hotel booking web project. Its simplicity, high performance, and scalability make it easy to learn and efficient for managing HTTP requests and database interactions. Despite some complexities in manual configuration and middleware management, Express remains a popular choice due to its flexibility and extensive community support, making it an essential component in the project's development. delivering a seamless user experience in a hotel booking web project.