**Developers Guide**

1. Introduction

Welcome to the developer guide for the sea-cargo logistics application! This guide is designed to assist developers in understanding the architecture, components, and development process of the application. Whether you're a new developer joining the project or an experienced developer seeking to contribute, this guide will provide you with the necessary information to get started and contribute effectively.

Purpose of the Developer Guide

The purpose of this developer guide is to:

* Provide an overview of the sea-cargo logistics application and its key features.
* Explain the technologies used in the backend and frontend development of the application.
* Guide developers through setting up the development environment and project structure.
* Detail the process of backend and frontend development, including API documentation, database schema, UI components, and state management.
* Serve as a reference for developers working on the sea-cargo logistics application, offering insights and guidance throughout the development lifecycle.

Target Audience

This guide is intended for developers who will be contributing to the development of the sea-cargo logistics application. This includes backend developers, frontend developers, full-stack developers, and anyone involved in the development, deployment, and maintenance of the application.

Feedback and Contributions

We welcome feedback and contributions to this developer guide. If you have suggestions for improvement, encounter any issues, or would like to contribute additional content, please feel free to reach out to the project maintainers.

2. Prerequisites

Before you begin developing for the sea-cargo logistics application, ensure that you have the following prerequisites in place:

Backend Development:

* Node.js: The backend of the application is built using Node.js. Ensure that you have Node.js installed on your development machine. You can download Node.js from [Node.js official website](https://nodejs.org/).
* MongoDB: The application uses MongoDB as its database. Install MongoDB Community Edition on your machine. You can download MongoDB from [MongoDB official website](https://www.mongodb.com/try/download/community).
* Git: Version control for the project is managed using Git. Install Git on your machine if you haven't already. You can download Git from [Git official website](https://git-scm.com/).

Frontend Development:

* HTML: The frontend of the application is built using HTML for markup. Ensure that you have a basic understanding of HTML for frontend development.
* JavaScript: The application uses JavaScript for frontend interactivity. Familiarize yourself with JavaScript for frontend scripting.
* Tailwind CSS: The application uses Tailwind CSS for styling. Get acquainted with Tailwind CSS by visiting [Tailwind CSS official website](https://tailwindcss.com/).

Development Tools:

* Integrated Development Environment (IDE): Choose an IDE of your preference for coding. Popular options include Visual Studio Code, Atom, Sublime Text, etc.
* Postman: Postman is a useful tool for testing APIs. Install Postman on your machine for API testing. You can download Postman from [Postman official website](https://www.postman.com/).

Additional Recommendations:

* Familiarity with RESTful API design principles.
* Knowledge of version control with Git and GitHub.

Ensure that you have all the prerequisites installed and set up correctly before proceeding with the setup of the development environment.

3. Setting Up the Development Environment

Before you can start developing for the sea-cargo logistics application, you need to set up your development environment. Follow these steps to get started:

Cloning the Repository:

1. Open your terminal or command prompt.
2. Navigate to the directory where you want to clone the repository.
3. Run the following command to clone the repository:
4. bash
5. Copy code

-> git clone <repository-url>

1. Replace <repository-url> with the URL of the GitHub repository.

Installing Dependencies:

1. Navigate to the cloned repository directory.
2. For the backend:
   * Navigate to the backend directory.
   * Run npm install to install backend dependencies.
3. For the frontend:
   * Navigate to the frontend directory.
   * Run npm install to install frontend dependencies.

Configuring Environment Variables:

1. Backend:
   * Create a .env file in the backend directory.
   * Define environment variables such as PORT, MONGODB\_URI, and JWT\_SECRET.
   * Refer to the .env.example file for reference.
2. Frontend:
   * Create a .env file in the frontend directory.
   * Define any necessary environment variables.
   * Refer to the .env.example file for reference.

Starting the Development Servers:

1. For the backend:
   * Navigate to the backend directory.
   * Run npm start to start the backend server.
2. For the frontend:
   * Navigate to the frontend directory.
   * Run npm start to start the frontend development server.

Accessing the Application:

Once the development servers are running, you can access the sea-cargo logistics application in your web browser. The frontend development server typically runs on port 3000, while the backend server runs on the port specified in your environment variables.

4. Project Structure

The sea-cargo logistics application follows a structured project layout to maintain code organization and facilitate collaboration among developers. Below is an overview of the project structure for both the backend and frontend components:

Backend Project Structure:

* backend/: Root directory for the backend codebase.
  + src/: Contains the source code for the backend application.
    - config/: Configuration files for the application, including database configuration, middleware setup, and environment variables.
    - controllers/: Controllers responsible for handling HTTP requests and responses.
    - models/: Mongoose models defining the data schema for MongoDB.
    - routes/: Route definitions for the RESTful API endpoints.
    - middleware/: Custom middleware functions used in request processing.
    - services/: Business logic services for handling application logic.
    - utils/: Utility functions and helper modules used throughout the application.
    - app.js: Entry point for the backend application, where middleware is initialized and routes are defined.
  + package.json: NPM package configuration file specifying dependencies and scripts.

Frontend Project Structure:

* frontend/: Root directory for the frontend codebase.
  + index.html: HTML template for the frontend application.
  + css/: Directory for CSS stylesheets.
  + js/: Directory for JavaScript files.
  + pages/: Directory for other HTML pages.
  + assets/: Static assets such as images, fonts, and icons.
  + package.json: NPM package configuration file specifying dependencies and scripts.

5. Backend Development

Backend development for the sea-cargo logistics application involves building the server-side components responsible for handling requests, processing data, and interacting with the database. In this section, we'll cover the following aspects of backend development:

Overview:

* The backend is developed using Node.js and Express.js, providing a robust and scalable foundation for building RESTful APIs.
* The application follows a modular architecture, with separate components for controllers, models, routes, middleware, and services, ensuring maintainability and extensibility.

Technologies Used:

* Node.js: A JavaScript runtime environment that allows developers to build scalable and efficient server-side applications.
* Express.js: A minimal and flexible Node.js web application framework that provides a robust set of features for building APIs and web applications.
* MongoDB: A NoSQL database used for storing and managing data in a flexible and scalable manner.
* Mongoose: An Object Data Modeling (ODM) library for MongoDB and Node.js, providing a straightforward schema-based solution for modeling application data.

API Documentation:

* The backend API endpoints are documented using tools such as Swagger or Postman, providing developers with comprehensive documentation for interacting with the API.
* The documentation includes details about each endpoint, including URL, HTTP method, request and response payloads, authentication requirements, and example usage.

Authentication and Authorization:

* The application implements authentication and authorization mechanisms to secure API endpoints and protect sensitive data.
* Role-based access control (RBAC) is implemented to restrict access to certain endpoints based on user roles and permissions.

Database Schema:

* The database schema is designed using Mongoose schemas, defining the structure of the data stored in MongoDB.
* Each Mongoose schema represents a data model for a specific entity in the application, such as users, orders, shipments, etc.
* Relationships between entities are established using references or embedded documents, depending on the nature of the data.

6. Frontend Development

Frontend development for the sea-cargo logistics application involves building the client-side components responsible for user interaction and presentation of data. In this section, we'll cover the following aspects of frontend development:

Overview:

* The frontend is developed using HTML, JavaScript (Vanilla JS), and Tailwind CSS, providing a lightweight and efficient solution for building responsive and user-friendly interfaces.
* The application follows a component-based architecture, with reusable UI components for common elements such as headers, navigation bars, forms, etc.

Technologies Used:

* HTML: A markup language used for structuring the content and layout of web pages.
* JavaScript (Vanilla JS): The programming language of the web used for adding interactivity and dynamic behavior to web pages.
* Tailwind CSS: A utility-first CSS framework used for styling the frontend components, providing a customizable and scalable approach to CSS development.

Project Structure:

* The frontend project is structured into directories for organizing HTML, CSS, JavaScript, and assets.
* HTML files are organized based on page structure, with separate files for each page of the application.
* CSS stylesheets are modularized and organized into directories based on component styling.
* JavaScript files contain logic for frontend interactivity and functionality, with separate files for different components and modules.

UI Components:

* The application utilizes reusable UI components for common elements such as buttons, input fields, dropdowns, modals, etc.
* Components are designed to be modular, encapsulating both structure and behavior for easy reuse and maintainability.
* Tailwind CSS classes are used to style components, providing a consistent and cohesive design across the application.

State Management:

* Frontend state management is handled using JavaScript, with stateful components and data manipulation functions.
* State is managed at the component level, with parent-child component communication and prop drilling for passing data between components.
* Event handling and DOM manipulation are used to update component state and trigger UI updates in response to user actions.

7. Deployment

Deploying the sea-cargo logistics application involves making the application accessible to users over the internet. This section provides guidance on deploying both the backend and frontend components of the application.

Backend Deployment:

* Choose a hosting provider or platform for deploying the backend of the application. Popular options include AWS, Heroku, Google Cloud Platform, or a self-managed server.
* Ensure that the hosting environment meets the requirements for running a Node.js application and MongoDB database.
* Deploy the backend codebase to the chosen hosting environment using deployment tools such as Git, FTP, or deployment scripts.
* Set up environment variables and configuration files as required for the deployed environment, including database connection strings, API keys, and other sensitive information.
* Test the deployed backend to ensure that it is accessible and functioning correctly.

Frontend Deployment:

* Choose a hosting provider or platform for deploying the frontend of the application. Options include static site hosts like Netlify, Vercel, GitHub Pages, or serving the frontend from the same server as the backend.
* Build the frontend codebase for production using build tools such as Webpack, Parcel, or Create React App.
* Deploy the built frontend assets to the chosen hosting environment using deployment tools or built-in deployment features provided by the hosting provider.
* Configure DNS settings and domain mappings to point to the deployed frontend application.
* Test the deployed frontend to ensure that it is accessible and displays correctly across different devices and browsers.

8. Conclusion

Congratulations on completing the developer guide for the sea-cargo logistics application! Throughout this guide, we've covered various aspects of backend and frontend development, deployment strategies, and best practices for contributing to the project.

As developers, you play a crucial role in building and maintaining the sea-cargo logistics application. Your expertise and dedication ensure the reliability, scalability, and usability of the application, ultimately delivering value to our users and stakeholders.

By following the guidelines and recommendations outlined in this guide, you can contribute effectively to the project, collaborate with team members, and deliver high-quality code that meets the needs of our users.

Remember that software development is an iterative process, and continuous improvement is key to success. Stay curious, keep learning, and don't hesitate to experiment with new technologies and techniques to enhance the sea-cargo logistics application further.