3/18/2024

Project Rubrics

Web Development Course

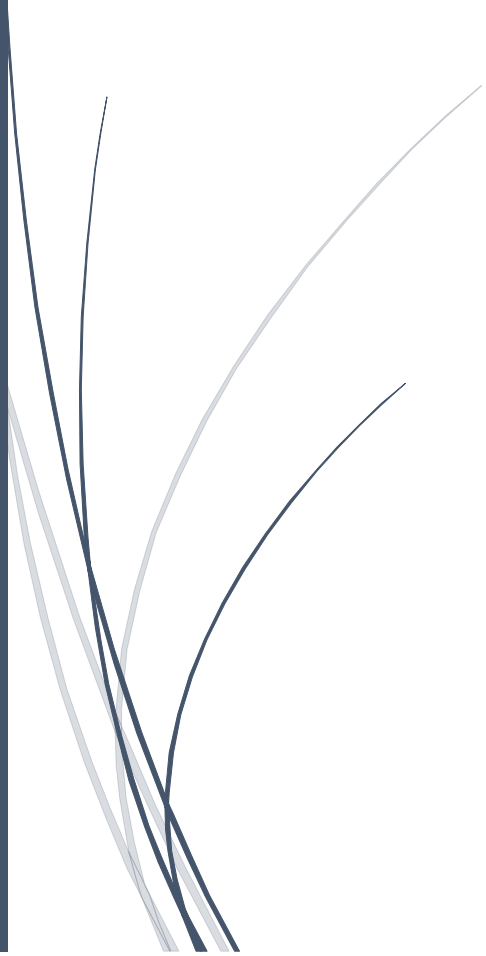

Table of Contents

Documentation

Documentation is crucial component to ensure clear communication, facilitate development, support maintenance, and guide users. Here are some essential documentation requirements:

Project Overview Document

- Introduction to the project, its objectives, and scope.
- Stakeholders involved and their roles.
- Timeline and milestones.

Functional Requirements Document (FRD)

- Detailed description of the system's functionalities from a user perspective. (Two functions)
- Requirements prioritization and dependencies.

Technical Requirements Document (TRD)

- Detailed technical specifications covering architecture, components, and integrations.
- Infrastructure requirements, including server configurations, databases, and third-party services.
- Development tools, frameworks, and libraries used.
- APIs and data formats for integration with external systems.

Database Schema Documentation

- Description of the database schema, including tables, columns, relationships, and constraints.

API Documentation

- Documentation for internal APIs used within the application, including endpoints, request/response formats, and authentication mechanisms.
- Sample requests and responses for common API operations.
- Usage guidelines and best practices for developers consuming the APIs.

Business Requirements

- User Registration and Authentication: Provide users with the ability to create accounts and log in securely.
- Product Catalog Management: Ability to add, update, and remove products with details like name, description, price, and images.
- Shopping Cart: Allow users to add items to their cart, view the cart, and proceed to checkout.
- Checkout Process: Smooth and secure checkout process with various payment options.
- Order Management: Ability to view and manage orders, update order status, send order confirmations, and issue refunds if necessary.
- Inventory Management: Track product availability, manage stock levels, and provide alerts for low stock.
- Shipping and Tax Calculation: Provide accurate shipping costs and tax calculations based on the user's location and the items in their cart.
- Responsive Design: Ensure the website is optimized for various devices (desktop, mobile, tablet) to provide a seamless user experience.

- Search and Filter Functionality: Allow users to search for products easily and apply filters to narrow down their choices.
- Customer Reviews and Ratings: Enable customers to leave reviews and ratings for products, helping other users make informed purchase decisions.
- Promotions and Discounts: Ability to run promotional campaigns, apply discounts, offer coupons, and track their effectiveness.
- Analytics and Reporting: Implement tools to track key metrics like sales, conversion rates, popular products, and customer behaviors to make data-driven decisions.

Technical Requirements

Common

Setup and Architecture

- Set up a project structure that promotes scalability; to move to an enterprise-level solution in the future.
    - All tests should be contained in their own folder.
    - Separate modules are created for any processing.
- Set up a npm project.
    - package.json should contain both devDependencies, and dependencies.
    - Scripts should be created for testing, linting/prettier and starting the server.
    - Build script should run without error.
- Version Control and Collaboration:
    - Manage project using a version control system like Git.
    - Organize commits frequent, descriptive, and properly.
    - Share project with teammates and manage contributions using branching and pull requests.

Functionality

- Set up JWT tokens in your API using modern authentication methods.
    - JWTs should be:
        - Part of the HTTP response.
        - Validated on requests requiring JWT (user secure routes).
        - Generated for each user.

README & Requirements Documentation

- Create a README.md file covering installation instructions, usage guidelines, and any other relevant information for developers and users.
- Create a documentation containing API route information.
    - Correct RESTful routes for the required endpoints.
    - Each RESTful route should be associated with the correct HTTP verb.
    - Request parameters and required headers, and response sample.

Backend

Functionality

- Add and use Express to a node.js project.
    - Start script should run without error.

    - Provided endpoint should open in the browser with status 200.
- Configure middlewares.
- Database
    - Create a database and connect to it.
    - Secure important information by adding salt to user passwords.
        - Encrypt the password field on the user table using the **bcrypt** library.
    - Create CRUD endpoints for models in the application.
        - Split the routes into grouped handler files for better code organization.

Front-end

Functionality

- Dashboard:
    - Create a dashboard page where authenticated users can view basic information or perform actions related to the app's purpose.
- Profile Management:
    - Allow users to view and edit their profile information.
    - Include basic fields like name, email, and profile picture.
- Component Architecture:
    - Project must follow a component-based architecture.
    - Components are appropriately modularized and reusable.
    - Components must be structured logically, with clear separation of concerns.
- Responsive Design:
    - Ensure the app layout is responsive and works well on various screen sizes, including mobile devices.
    - Use CSS frameworks like Material-UI or Bootstrap to facilitate responsiveness.
- Routing
    - Implement client-side routing by building a SPA with React Router.
    - Routes are defined logically, with appropriate handling of nested routes and redirects.
- Forms Handling
    - Work with Forms and user inputs with **Formik**.
    - Validate form’s inputs properly, with error messages displayed as needed.
- State Management:
    - Utilize React's built-in state management for managing local component state.
- Error Handling:
    - Implement basic error handling to display informative error messages to users when something goes wrong.
    - Handle common error scenarios like network failures, invalid inputs, or server errors gracefully.

[Bonuses]

Backend

Functionality

- Database
  - Secure database access info with environment variables
    - You can use **dotenv** to create environment variables.

Code Quality

- Write relevant unit tests to improve code quality and refactoring.
  - Test script runs and all tests created pass.
  - There is at least 1 test per endpoint and at least one test for image processing.

Front-end

Code Quality

- Write basic unit tests for critical components or functionalities using a testing library like Jest and React Testing Library.