AMD project: Waste sorting apps description

Author: Prof. Nguyen Vu Anh Trung
Campus: Greenwich University Saigon Vietnam

Date: 16 July 2024

A backend architecture for a Waste Management app for teaching young people to do waste sorting, using Node.js, Express, and MongoDB:

Backend Architecture

1. API Server

The API server will be built using Node.js and Express. It will handle all the incoming requests from the frontend application and interact with the database to perform CRUD (Create, Read, Update, Delete) operations.

The API server will consist of the following components:

1. **Routes**: These will define the various endpoints for the application, such as:
    - `/users`: For user-related operations (registration, login, profile management, etc.)
    - `/waste-categories`: For managing waste categories (e.g., organic, recyclable, hazardous)
    - `/waste-items`: For managing individual waste items and their sorting information
    - `/challenges`: For managing educational challenges related to waste sorting
2. **Controllers**: These will handle the logic for each endpoint, such as:
    - Validating and processing incoming data
    - Interacting with the database to perform CRUD operations
    - Returning appropriate responses to the client

3. **Services**: These will encapsulate the business logic of the application, such as:
    - Calculating waste sorting scores
    - Generating educational content and challenges
    - Providing recommendations for proper waste disposal
4. **Middleware**: This will handle cross-cutting concerns, such as:
    - Authentication and authorization
    - Error handling
    - Request logging

2. Database

The database for the Waste Management app will be MongoDB, a popular NoSQL database. The database will consist of the following collections:

1. **Users**: Storing user information, such as name, email, password, and waste sorting scores.
2. **WasteCategories**: Storing information about different waste categories, such as name, description, and disposal guidelines.
3. **WasteItems**: Storing information about individual waste items, such as name, category, and sorting instructions.
4. **Challenges**: Storing information about educational challenges related to waste sorting, such as description, difficulty level, and scoring criteria.

3. Authentication and Authorization

The Waste Management app will implement a secure authentication and authorization system using JSON Web Tokens (JWT). This will ensure that only authorized users can access and perform actions within the application.

4. Error Handling and Logging

The API server will have a robust error handling mechanism to provide meaningful error messages to the client and log any issues for debugging purposes. This can be achieved using a combination of middleware and custom error handling functions.

5. Scalability and Performance

As the user base and data volume grow, the backend architecture should be designed to handle increased traffic and data storage requirements. This can be achieved through techniques such as:

- Implementing caching mechanisms for frequently accessed data
- Scaling the database horizontally by adding more MongoDB instances
- Utilizing load balancing to distribute the API server load across multiple instances

6. Documentation and Testing

The backend API should be thoroughly documented, using tools like Swagger or Postman, to ensure easy integration with the frontend application and facilitate future maintenance and development. Additionally, comprehensive unit and integration tests should be implemented to ensure the reliability and stability of the backend system.

This proposed backend architecture provides a solid foundation for the Waste Management app, allowing for efficient data management, secure user authentication, and scalable performance. As the project evolves, you can further refine and expand this architecture to meet the specific requirements of your application.