Project: Creating a blogging API

Opened: Monday, 7 October 2024, 8:58 AM

The Blogify API

In groups of two, you will be tasked with creating the following scenario. This intermediate project will be evaluated and will count towards your course grade.

Company Mission:

Blogify aims to provide a seamless blogging experience, enabling users to create, publish, and manage blog posts, while fostering user interaction via comments and likes. The company's vision is to make information sharing dynamic, collaborative, and community-driven. Blogify needs a powerful backend to ensure the best possible user experience.

The Challenge:

As part of Blogify's backend team, you are tasked with developing the **Blogify API** that will serve as the foundation for the platform. The API will manage everything from user registration and authentication to blog posts, comments, and user engagement (likes).

Core Features:

1. User Management:

- **Account Creation**: Users should be able to register by providing an email, username, and password.
  - **Route**: `POST /users/register`
  - **Validations**: Ensure unique email and username.
- **Login**: Users can log in with their email and password.
  - **Route**: `POST /users/login`
  - **Optional**: JWT authentication token generation for session management.
- **CRUD Operations for Users**:
  - `GET /users/:id`: Retrieve a specific user's profile.
  - `PUT /users/:id`: Update user profile.
  - `DELETE /users/:id`: Delete a user account (soft delete with status).

2. Blog Post Management:

- **Creating Posts**: Users can create blog posts on various topics.
  - **Fields**: Title, Content, Author (user reference), Tags, Timestamps (created_at, updated_at).
  - **Route**: `POST /posts`
- **Reading Posts**: Allow retrieval of all posts, or specific ones by ID.
  - **Route**: `GET /posts` (with optional filters like tag, author, date).
  - `GET /posts/:id`: Retrieve a specific blog post.
- **Editing and Deleting Posts**:
  - `PUT /posts/:id`: Update an existing blog post.
  - `DELETE /posts/:id`: Soft delete or permanently remove a blog post.

3. Commenting System:

- **Adding Comments**: Users can comment on any blog post.
  - **Fields**: Comment Text, Author, Post ID, Timestamp.
  - **Route**: `POST /posts/:id/comments`
- **Managing Comments**:

  - `PUT /comments/:id`: Edit a comment.
  - `DELETE /comments/:id`: Delete a comment.

Technical Requirements:

The API should follow **RESTful principles**, with clear and consistent endpoint naming, proper HTTP verbs (GET, POST, PUT, DELETE), and status codes.

Use **Express.js** to manage routing and request handling, creating a scalable, modular architecture.
Implement **JWT-based authentication** (optional but encouraged) for securely logging in users and performing actions on their account or posts.

MongoDB will be used to store and manage the data. You will use the following C**ollections**:

- **Users**: Stores user information (username, email, password hash).
- **Posts**: Stores blog posts with references to authors and timestamps.
- **Comments**: Stores comments on posts, referencing users and posts.
- **Likes**: Tracks user engagement with posts.

Set up routes for each feature (user, post, comment, like) following REST standards.

Document your API using **OpenAPI** to provide an interactive interface for testing and API exploration.

This structured approach will ensure the API is functional, secure, and easy to scale, providing Blogify with a solid backend foundation for its blogging platform.

Grading summary

Visible groups Cohorte Caen B.Eng.3 2024-2025

Hidden from students	Yes
Participants	18
Submitted	0
Needs grading	0