Blog App Documentation

Summary:

This is my first project with Ruby on Rails. While I have worked on several backend projects using PHP and Laravel, this is my initial experience with Ruby. I have implemented authentication with JWT and performed CRUD operations on posts. I am excited to dive deeper into Ruby and Ruby on Rails for this project.

What I have done

1- Build an authentication system

Build an authentication system in Ruby on Rails project with the following features:

1. **Token Generation and Validation:**
   - Integrate JWT to generate and validate tokens.
   - Create JsonWebToken service class to handle token generation and validation. *“app/services/json_web_token.rb”*.
2. **Authentication Methods:**
   - Implement an authenticate_user method in ApplicationController to ensure authentication for requests.
   - Implement method to get current authenticated user and access it over controllers using *current_user.*
3. **Usage:**
   - Use *before_action :authenticate_user* in each controller to ensure all actions require authentication, and *current_user* to access current authenticated user.

Here is an Api endpoints:

1. /register [POST]

Here is an example for successful request:

2. /login [POST]

Here is an example for successful login:

You can run *rails test* to run tests on authentication endpoints

2- Post Management Operations:

Create operations for a Post with Comments:

- Complete CRUD operations on Posts
- A Post has many Comments and belongs to a User.
- A Comment is a separate entity and belongs to both a User and a Post.
- Tags can be added to a Post and accepted in the request as an array.
- The tags are then joined by a comma , and saved in the database as a string.

Implement Policies:

- The Post's author can only delete or edit their own Post.

- The Post's author can delete any Comment associated with their Post.

- A Comment's author can edit their own Comment.

- A Comment can be deleted by its author or the Post's author.

Additional Features:

- Integration with **Sidekiq** for background tasks (e.g., deleting posts after 24 hours).

- Posts are automatically deleted 24 hours after their creation. This could be implemented as a background job using **Sidekiq** that checks for posts older than 24 hours and deletes them.

Here is all APIs Docs => PostmanDocs