Approach and Challenges

The project is divided into two main parts: the frontend, which is built with React, and the backend, which is powered by Ruby on Rails.

Frontend –

1. Setting Up React: Started by setting up a new React project using create-react-app . This gave a basic project structure to work with.

1. Component Structure: Organized an application into reusable components. Common components include a PostList for displaying blog posts, a PostDetail for viewing a single post, and a PostForm for creating or editing posts.
2. Routing: I implemented routing using a React Router library . This allows to navigate between different views, such as the list of blog posts and the post detail page.
3. API Requests: To interact with the backend, I made API requests using the Axios library and sent requests to create, read, update, and delete blog posts.

Backend –

1. Setting Up Rails: I started by creating a new Rails application using the rails new command. This generated the basic structure for the backend.
2. Database Models: I defined database models using Rails' ActiveRecord. For a blog site, I created a Post model with attributes - title and description.
3. Controllers: I created POST controller to handle CRUD operations for your Post model.
4. Database Migration: I used Rails migrations to create the database tables and define their structure.
5. API Endpoints: I implemented API endpoints that accepted and returned data in JSON format. These endpoints were used by your React frontend to communicate with the backend.

Challenges –

1. CORS (Cross-Origin Resource Sharing): I have encountered issues with when making API requests from React frontend to the Rails backend on a different domain. To solve this, I configured CORS settings in Rails application.