Project Overview

A blog platform backend built with Spring Boot will provide the core functionalities for a blogging system. This includes user management, post creation, editing, and publishing, as well as comment management and other features. The backend will expose RESTful APIs for frontend consumption.

Core Functionalities

User Management

- User registration, login, and logout
- Profile management (username, email, password, bio, avatar)
- Role-based access control (admin, author, reader)
- Password reset and recovery.

Post Management

- Post creation, editing, and deletion.
- Post publishing and unpublishing.
- Post categorization and tagging.
- Post drafts and scheduling.
- Featured posts and recommendations.

Comment Management

- Comment creation, editing, and deletion.
- Comment approval and moderation.
- Nested comments (replies)

Additional Features (Optional)

- Search functionality for posts and users.
- Notification system (email, push notifications)
- Analytics and reporting
- Integration with social media platforms
- Content management system (CMS) for managing blog appearance and layout.

Technical Specifications

Backend Technologies

- **Spring Boot:** Framework for building the backend application.
- **Spring Data JPA:** Data access layer for interacting with the database.
- **Spring Security:** Authentication and authorization.
- **Spring Web:** Building RESTful APIs
- **Database:** MySQL, PostgreSQL, or MongoDB (depending on project requirements)
- **Build tool:** Maven or Gradle
- **Testing framework:** JUnit or TestNG

API Endpoints

- User endpoints (registration, login, profile, etc.)
- Post endpoints (create, read, update, delete, publish, etc.)
- Comment endpoints (create, read, update, delete, approve, etc.)
- Authentication endpoints (token-based authentication)

Data Model

- User (id, username, email, password, role, created_at, updated_at)
- Post (id, title, content, author_id, category_id, created_at, updated_at, published_at)
- Comment (id, content, author_id, post_id, parent_id, created_at, updated_at)
- Category (id, name)
- Tag (id, name)

Security Considerations

- User authentication and authorization
- Data encryption (passwords, sensitive information)
- Input validation and sanitization.
- Protection against common vulnerabilities (SQL injection, XSS, CSRF)

Error Handling

- Proper error handling and reporting
- Meaningful error messages

Data Validation

- Input validation for all API requests.
- Constraints and validation rules for data models

Testing

- Unit testing for backend components
- Integration testing for API endpoints
- Security testing

Project Structure

- src/main/java
    - com/yourcompany/blog
        - controllers
        - services
        - repositories
        - models
        - config
        - security
- src/main/resources
    - application.properties

    - database scripts

Deployment

- Consider deployment options like Docker, Kubernetes, or cloud platforms (AWS, GCP, Azure)

Additional Considerations

- **Performance:** Optimize database queries, caching, and load balancing for high performance.
- **Scalability:** Design the application to handle increasing traffic and data volume.
- **Maintainability:** Write clean, well-structured, and documented code.
- **API Documentation:** Generate clear and comprehensive API documentation using tools like Swagger or OpenAPI.

Remember to adapt this specification based on your specific project requirements and constraints.