**REST API's DESIGN FOR BLOG APPLICATION**

1. **HIGH LEVEL REQUIREMENTS FOR BLOG APPLICATION:**
2. **Posts Management :**

Build the REST API's for POST resource-

1. Create Post REST API
2. Get Single Post REST API
3. Get All Posts REST API
4. Get All Posts REST API with Pagination and Sorting
5. Update Post REST API
6. Delete Post REST API

**ii. Comments Management for Post:**

Build the REST API for Comment resource(one to many mapping)-

1. Create Comment REST API- /posts/{post-id}/comments
2. Get Single Comment REST API- /posts/{post-id}/comments/{comment-id}
3. Get All Comments REST API- /posts/{post-id}/comments
4. Update Comments REST API- /posts/{post-id}/comments/{comment-id}
5. Delete Comment REST API- /posts/{post-id}/comments/{comment-id}

**iii. Exceptions handling and Validations:**

1. handle the exceptions and errors and return proper error response to the client.
2. Validate the REST API request and send the validation error response to the client.

**iv. Securing REST APIs:**

1. Secure REST APIs using Database Authentication
2. Build Login/Signin REST API
3. Build Register/ SignUp REST API
4. Use JWT(JSon Web Token) token based Authentication to Secure the REST APIs
5. Implement Role-based security- ADMIN and USER roles

**v. Category Management:**

Build the REST APIs for Category resource-

1. Create category REST API
2. Get Single Category REST API
3. Get All Categories REST API
4. Update Category REST API
5. Delete Category REST API
6. Get Posts by Category REST API

**vi. Versioning REST APIs and Deploying on AWS Cloud**

1. Versioning REST APIs using different strategies
2. Deploy Blog App on AWS Cloud

**II. Selecting Technology Stack for Blog App**

* **Java Platform:** Java 17+
* **Java Frameworks**: Spring Framework and it's portfolio projects like Spring Boot, Spring Security and Spring Data JPA
* **Token Based Authentication:** JWT(Json Web Token)
* **Build Tool:** Maven
* **IDE:** IntelliJ IDEA(You can use STS(spring tool suite))
* **Server**: Tomcat embedded server
* **Database:** MySQL database
* **REST Client:** Postman
* **Cloud for Deployment:** AWS Cloud

**III. Identify Resources for Blog App**

Identify the resources for create REST endpoints

1. **Post:** Build CRUD REST APIs and will also support Pagination & Sorting.
2. **Comment:** Build CRUD REST APIs and will establish one to many mapping between post and comment resources.
3. **User:** Create Register and Login REST APIs and will secure the REST APIs using JWT token based authentication.
4. **Category:**  Build CRUD REST APIs and will establish one to many mapping between post and category resources.

**IV. Spring Boot Application Architecture**



* We are using three-layer or three-tier architecture to develop our spring boot application.
* **Controller layer-** We use to develop REST APIs and we can call controller layer as an API layer because we basically create Spring MVC controller and we keep all the rest APIs in controller class.
* **Service layer-** We keep all our business logic.
* **DAO or repository-** We keep all our database related logic or persistence logic. And DAO layer is responsible to communicate with database.
* **Postman Client-** We use Postman Client to call REST APIs or to test REST APIs
* **DTO's-** To transfer data between client and server.
* **JSON-** Used as a mediatype , as a message exchange format between client and server.

**V. REST API Design for Post Resource**

1. **GET -** Get all posts

**URL path:** /api/posts

**Status Code:** 200 (OK)

**2. GET -** Get post by Id

**URL path:** /api/posts/{id}

**Status Code:** 200 (OK)

**3. POST -** Create a new post

**URL path:** /api/posts

**Status Code:** 201 (Created)

**4. PUT -** Update existing post with Id

**URL path:** /api/posts/{id}

**Status Code:** 200 (OK)

**5. DELETE -** Delete post by Id

**URL path:** /api/posts/{id}

**Status Code:** 200 (OK)

**6. GET -** Paginating and sorting posts

**URL path:** /api/posts?pageSize=5&pageNo=1sortBy=firstName

**Status Code:** 200 (OK)

**VI. REST API Design for Comment Resource**

1. **GET -** Get all comments which belong to post with id=postId

**URL path:** /api/posts/{postId}/comments

**Status Code:** 200 (OK)

**2. GET -** Get comment by id if it belongs to post with id=postId

**URL path:** /api/posts/{postId}/comments/{id}

**Status Code:** 200 (OK)

**3. POST -** Create a new comment for post with id=postId

**URL path:** /api/posts/{postId}/comments

**Status Code:** 201 (Created)

**4. PUT -** Update comment by id if it belongs to post with id=postId

**URL path:** /api/posts/{postId}/comments/{id}

**Status Code:** 200 (OK)

**5. DELETE -** Delete comment by id if it belongs to post with id=postId

**URL path:** /api/posts/{postId}/comments/{id}

**Status Code:** 200 (OK)

**VIi. REST API Design for SignUp/Register and Signin/Login**

1. **POST -** Sign Up

**URL path:** /api/auth/signup

**Status Code:** 200 (OK)

**2. POST -** Sign In

**URL path:** /api/auth/signin

**Status Code:** 200 (OK)