System Requirements Specification Index

For

REST API for Blog Application

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Table of Contents

1 Project Abstract

Blog Application is Spring boot RESTful application with MySQL, where it allows to manage the blogs and can post comments on the blog.
Following is the requirement specifications:

	Blog Application
Modules	
1	Blogs
2	Comments
Blog Module Functionalities	
1	Create a Blog
2	Update a Blog
3	Delete a Blog
4	Get the Blog by Id
Comment Module Functionalities	
1	Post a comment on the Blog

2 Assumptions, Dependencies, Risks / Constraints

2.1 Blog Constraints:

- While fetching the Blog by Id, if Id does not exist then operation should throw custom exception.
- While Updating the Blog by Id, if Id does not exist then operation should throw custom exception
- While deleting the Blog by Id, if Id does not exist then operation should throw custom exception

2.2 Comment Constraints

- If you want to post a comment on a blog, if blog id does not exists, then operation should throw a custom exception.

Common Constraints

- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
- All the business validations must be implemented in dto classes only.
- All the database operations must be implemented on entity object only
- Do not change, add, remove any existing methods in service layer
- In Repository interfaces, custom methods can be added as per requirements.
- All RestEndpoint methods and Exception Handlers must return data wrapped in **ResponseEntity**

3 Business Validations

- Blog title is not null, min 3 and max 100 characters.
- Blog content is not null, min 3, max 200 characters.
- Comment blog id is not null.
- Comment comment is not null, min 3 and max 200 characters.

4 Rest Endpoints

Rest End-points to be exposed in the controller along with method details for the same to be created

4.1 BooksController

URL Exposed		Purpose
1. /api/blogs		Successful Blog creation Response code: 201 (Created) If title or content is blank then Response code: 400 (Bad request)
Http Method	POST	
Parameter 1	BlogDto	
Return	BlogDto	
/api/blogs/{id}		If blog id is valid then Response code: 200 (OK) If blog id is invalid then Response code: 404 (Not Found)
Http Method	GET	
Parameter 1	Long(id)	
Return	BlogDto	
/api/blogs/{id}		For valid blog id Response code: 200 (OK) For invalid blog id: 404 (Not Found)
Http Method	PUT	
Parameter 1	Long(Id)	
Parameter 2	BlogDto	
Return	BlogDto	
/api/blogs/{id}		For valid blog id Response code: 200 (OK)

Http Method DELETE; Parameter 1 Long(Id); Return Boolean	For invalid blog id: 404 (Not Found)
/api/blogs/comment Http Method POST; Parameter 1 CommentDto; Return CommentDto	For valid blog id Response code: 200 (OK) For invalid blog id: 404 (Not Found)

5 Template Code Structure

5.1 Package: com.iiht.training.blogs

Resources

SpringbootBlogsServiceApplication (Class)	This is the Spring Boot starter class of the application.	Already Implemented. **You are free to add any bean in this class.**

5.2 Package: com.iiht.training.blogs.entity

Resources

Class/Interface	Description	Status
BlogEntity (class)	● Annotate this class with proper annotation to declare it as an entity. class with **id** as primary key. ● You can use javax package for annotations. ● Map this class with **blogs** table. ● Generate the **id** using **IDENTITY** strategy	Partially implemented.
CommentEntity(class)	● This class is partially implemented. ● Annotate this class with proper annotation to declare it as an entity class with **id** as primary key. ● You can use javax package for annotations. ● Map this class with **comments** table. ● Generate the **id** using the **IDENTITY** strategy	Partially implemented.

5.3 Package: com.iiht.training.blogs.dto

Resources

Class/Interface	Description	Status
BlogDto (class)	Use appropriate annotations from the **Java Bean Validation API** for validating attribute of this class. (Refer **Business Validation** section for validation rules).	Partially implemented.
CommentDto (class)	Use appropriate annotations from the **Java Bean Validation API** for validating attribute of this class. (Refer **Business Validation** section for validation rules).	Partially implemented.

5.4 Package: com.iiht.training.blogs.repository

Resources

Class/Interface	Description	Status
BlogRepository (interface)	1. Repository interface exposing CRUD functionality for **Blog** Entity. 2. You can go ahead and add any custom methods as per requirements	Already implemented
CommentRepository (interface)	1. Repository interface exposing CRUD functionality for **Comment** Entity.	Already implemented

	2. You can go ahead and add any custom methods as per requirements	

5.5 PACKAGE: COM.IIHT.TRAINING.BLOGS.SERVICE

Resources

Class/Interface	Description	Status
BlogService (interface)	Interface to expose method signatures for Blog related functionality. Do not modify, add or delete any method	Already implemented.
CommentService (interface)	Interface to expose method signatures for Comments related functionality. Do not modify, add or delete any method	Already implemented.

5.6 PACKAGE: COM.IIHT.TRAINING.BLOGS.SERVICE.IMPL

Resources

Class/Interface	Description	Status
BlogServiceImpl (class)	● Implements **BlogService**. Contains template method implementation. ● Need to provide implementation for Blog related functionalities ● Add required repository dependency	To be implemented.

	● Do not modify, add or delete any method signature	
CommentServiceImpl (class)	● Implements **CommentService.** Contains template method implementation. ● Need to provide implementation for student related functionalities ● Add required repository dependency ● Do not modify, add or delete any method signature	To be implemented.

5.7 PACKAGE: COM.IIHT.TRAINING.BLOGS.EXCEPTION

Resources

Class/Interface	**Description**	**Status**
GlobalHandler (class)	● RestControllerAdvice Class for defining global exception handlers. ● Contains Exception Handler for **InvalidDataException** class. ● Use this as a reference for creating exception handler for other custom exception classes	Partially implemented.
ExceptionResponse (class)	● Object of this class is supposed to be returned in case of exception through exception handlers	Already implemented.

Class/Interface	Description	Status
BlogNotFoundException (Class)	● Custom Exception to be thrown when trying to get the blog details. ● Need to create Exception Handler for same wherever needed (local or global)	Already created.

5.8 PACKAGE: COM.IIHT.TRAINING.BLOGS.CONTROLLER

Resources

Class/Interface	Description	Status
BlogController (Class)	● Controller class to expose all rest-endpoints for Blog and Comment related activities. ● May also contain local exception handler methods	To be implemented

6 Execution Steps to Follow

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal
3. To build your project use command:

   `mvn clean package -Dmaven.test.skip`
4. To launch your application, move into the target folder (cd target). Run the following command to run the application:

   `java -jar springboot-blogs-service-0.0.1-SNAPSHOT.jar`

5.This editor Auto Saves the code

6. If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.

7. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.

8. To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.

9. This is a web-based application, to run the application on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.

   Note: The application will not run in the local browser

10. Default credentials for MySQL:
    a. Username: root
    b. Password: pass@word1
11. To login to mysql instance: Open new terminal and use following command:
    a. `sudo systemctl enable mysql`
    b. `sudo systemctl start mysql`
    c. `mysql -u root -p`

    The last command will ask for password which is ‘pass@word1’
12. Mandatory: Before final submission run the following command:

    `mvn test`

13. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**