System Requirements Specification

Index

For

Event Management System

Version 1.0

Table of Contents

EVENT MANAGEMENT SYSTEM

System Requirements Specification

You need to consume APIs exposed by Backend application in Angular to make application work as FULLSTACK

BACKEND-SPRING BOOT RESTFUL APPLICATION

1 PROJECT ABSTRACT

The **Event Management System** is a FullStack Application with a backend implemented using Spring Boot with a MySQL database and a frontend developed using Angular. It enables users to manage various aspects of event planning and organization.

Following is the requirement specifications:

	Event Management System
Modules	
1	Event
Event Module Functionalities	
1	Create an Event
2	Update the existing Event details
3	Get the Event by Id
4	Get all Events
5	Delete an Event
6	Search for Events by Name
7	Search for Events by Status

2 ASSUMPTIONS, DEPENDENCIES, RISKS / CONSTRAINTS

2.1 EVENT CONSTRAINTS

- When fetching an Event by ID, if the event ID does not exist, the operation should throw a custom exception.
- When updating an Event, if the event ID does not exist, the operation should throw a custom exception.
- When removing an Event, if the event ID does not exist, the 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

- Name of the event is not null, min 3 and max 20 characters.
- Description of the event is not null and should be between 5 and 200 characters in length.
- Status of the event is not null.

4 REST ENDPOINTS

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

4.1 EVENTCONTROLLER

URL Exposed			Purpose
1. /events	Http Method	GET	Fetches all the events
	Parameter	-	
	Return	List<Events>	
2. /events	Http Method	POST	Add a new event
	Parameter 1	Event	
	Return	Event	
3. /events/{id}	Http Method	DELETE	Delete events with given events id
	Parameter 1	Long (id)	
	Return	-	
4. /events/{id}	Http Method	GET	Fetches the event with the given id
	Parameter 1	Long (id)	
	Return	Event	
5. /events/{id}	Http Method	PUT	Updates existing event
	Parameter 1	Long (id)	
	Parameter 2	Event	
	Return	Event	
6. /events/searchByName?name={name}	Http Method	GET	Fetches the event with the given name
	Parameter 1	String (name)	
	Return	List<Events>	
7. /events/searchByStatus?status={status}	Http Method	GET	Fetches the event with the given status
	Parameter 1	String (status)	
	Return	List<Events>	

5 Template Code Structure

5.1 PACKAGE: COM.EVENTMANAGEMENT

Resources

EventManagementApplication (Class)	This is the Spring Boot starter class of the application.	Already Implemented

5.2 PACKAGE: COM.EVENTMANAGEMENT.REPOSITORY

Resources

Class/Interface	**Description**	**Status**
EventDAO (interface)	● Repository interface exposing CRUD functionality for Event Entity. ● You can go ahead and add any custom methods as per requirements.	Partially implemented.

5.3 PACKAGE: COM.EVENTMANAGEMENT.SERVICE

Resources

Class/Interface	**Description**	**Status**
EventService (interface)	● Interface to expose method signatures for event related functionality. ● Do not modify, add or delete any method.	Already implemented.

5.4 PACKAGE: COM.EVENTMANAGEMENT.SERVICE.IMPL

Resources

Class/Interface	Description	Status
EventServiceImpl (class)	● Implements EventService. ● Contains template method implementation. ● Need to provide implementation for event related functionalities. ● Do not modify, add or delete any method signature	To be implemented.

5.5 PACKAGE: COM.EVENTMANAGEMENT.CONTROLLER

Resources

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

5.6 PACKAGE: COM.EVENTMANAGEMENT.DTO

Resources

Class/Interface	**Description**	**Status**
EventDTO (Class)	Use appropriate annotations from the Java Bean Validation API for validating attributes of this class.	Partially implemented.

5.7 PACKAGE: COM.EVENTMANAGEMENT.ENTITY

Resources

Class/Interface	**Description**	**Status**
Event (Class)	● This class is partially implemented. ● Annotate this class with proper annotation to declare it as an entity class with **eventId** as primary key. ● Map this class with an **event table**. ● Generate the **eventId** using the IDENTITY strategy	Partially implemented.

5.8 PACKAGE: COM.EVENTMANAGEMENT.EXCEPTION

Resources

Class/Interface	**Description**	**Status**
InvalidDataException (Class)	● Object of this class is supposed to be returned in case of exception through exception handlers	Already implemented.
EventNotFoundException (Class)	● Custom Exception to be thrown when trying to fetch or delete the event info which does not exist. ● Need to create Exception Handler for same wherever needed (local or global)	Already implemented.
GlobalExceptionHandler (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.	Already implemented.

5.9 PACKAGE: COM.EVENTMANAGEMENT.MODEL.EXCEPTION

Class/Interface	Description	Status
ExceptionResponse (Class)	● Object of this class is supposed to be returned in case of exception through exception handlers	Already implemented.

6 CONSIDERATIONS

A. There is no roles in this application
B. You can perform the following possible action

Event

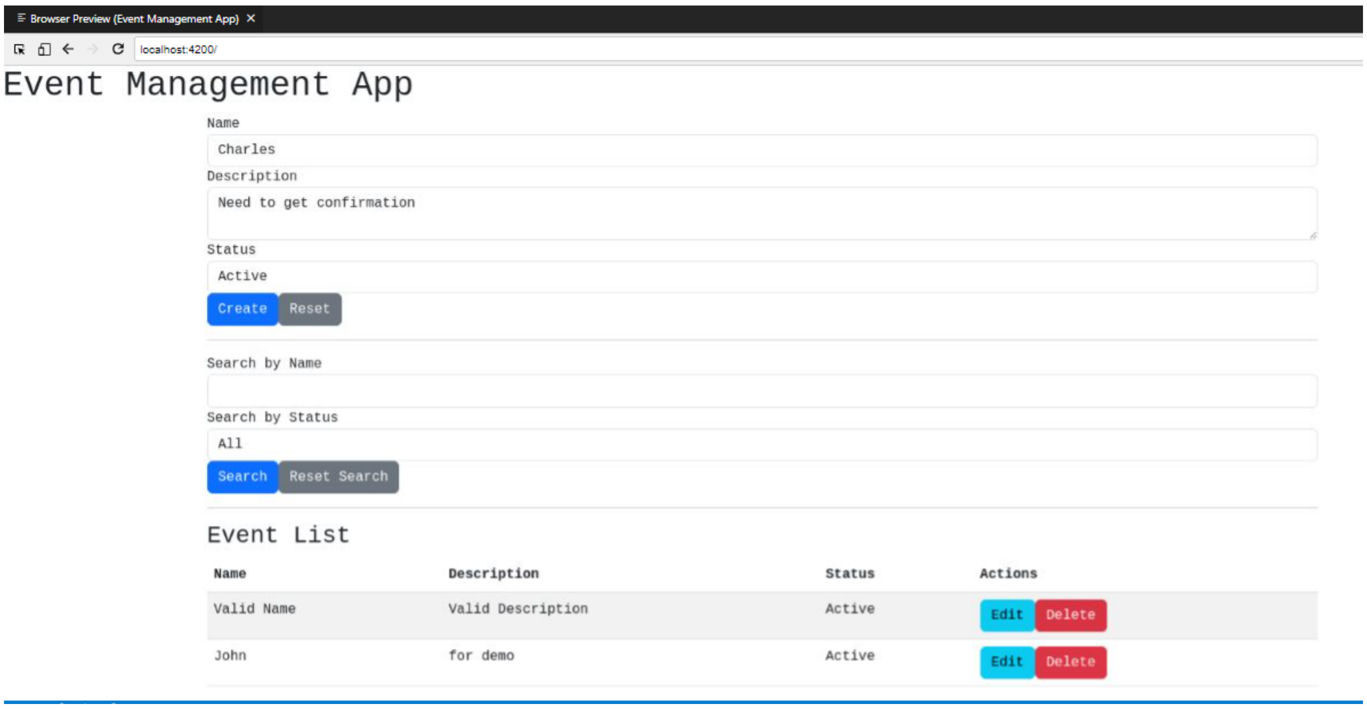
1 PROBLEM STATEMENT

Event Management System is SPA (Single Page Application), It enables users to manage various aspects of event planning and organization like it allows to add event, update event, delete event, get event by id, get all events, search event by name, and search event by status.

2 PROPOSED EVENT MANAGEMENT SYSTEM WIREFRAME

UI needs improvisation and modification as per given use case and to make test cases passed.

2.1 HOME PAGE



Event Management App

Name: Charles

Description: Need to get confirmation

Status: Active

Search by Name:

Search by Status: All

Event List

Name	Description	Status	Actions
Valid Name	Valid Description	Active	Edit Delete
John	for demo	Active	Edit Delete

3 BUSINESS-REQUIREMENT:

As an application developer, develop the Social Networking App (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Home Page	As a user I should be able to visit the Home page as the default page.
US_01	Home Page	As a user I should be able to see the homepage and perform all operations: Acceptance criteria: 1. As a user I should be able to furnish the following details at the time of creating an event. 1.1 Name 1.2 Description 1.3 Status 1.4 Search by Name 1.5 Search by Status 2. The Update button should be disabled by default, and should be enabled when you click on the Edit button. 3. Name field min length is 3 and max length 50. 4. Description field min length is 5 and max length 200. 5. Status should be not null. 6. Name, description, status fields are mandatory. If any constraint is not satisfied, a validation message must be shown.

4 EXECUTION STEPS TO FOLLOW FOR BACKEND

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 the Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal.**
3. **cd into your backend project folder**
4. **To build your project use command:**

   ```
   mvn clean package -Dmaven.test.skip
   ```

5. **To launch your application, move into the target folder (cd target). Run the following command to run the application:**

   ```
   java -jar <your application jar file name>
   ```

6. **This editor Auto Saves the code.**
7. **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.**
8. **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.**
9. **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.**
10. **To test any UI based application the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**
11. **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.**

1 Execution Steps to Follow for Frontend

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. **This is a web-based application, to run the application on a browser, use the internal browser in the environment.**
4. **You can follow series of command to setup Angular environment once you are in your project-name folder:**
   a. **npm install -> Will install all dependencies -> takes 10 to 15 min**
   b. **npm run start -> To compile and deploy the project in browser. You can press <Ctrl> key while clicking on localhost:4200 to open project in browser -> takes 2 to 3 min**
   c. **npm run test -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min**
5. **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.**