System Requirements Specification

Index

For

Volunteer Platform

Version 1.0

TABLE OF CONTENTS

VOLUNTEER PLATFORM APPLICATION

System Requirements Specification

BACKEND-SPRING BOOT RESTFUL APPLICATION

1 Project Abstract

The **Volunteer Platform Application** is implemented using Spring Boot with a MySQL database. The application aims to provide a comprehensive platform for managing and registering different types of volunteers for different types of programs.

Following is the requirement specifications:

	Volunteer Platform Application
Modules	
1	Event
2	User
Event Module Functionalities	
1	Get all upcoming events
2	Create a new event
3	Update an event by id
4	Cancel event

User Module Functionalities	
1	Register an user
2	Login user
3	Login out user
4	Enroll for an event

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 an Event not found with ID: {id}.
- When updating an event, if the event ID does not exist, the operation should throw an Event not found with ID: {id}.

2.2 USER CONSTRAINTS

- When enrolling for an event by ID, if the event ID does not exist, the operation should throw an User not found with ID: {id}.

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 - User

- Name should not be empty.
- Password should not be empty.
- Email should not be null.
- Description should not be empty.

4 BUSINESS VALIDATIONS - Event

- Name should not be empty.
- Date should not be null.
- Time should not be null.
- Description should be empty.

5 REST ENDPOINTS

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

5.1 EVENTCONTROLLER

URL Exposed		Purpose
1. /api/events		Fetches all the events
Http Method	GET	
Parameter	-	
Return	List<Event>	
2. /api/events		Creates a new event
Http Method	POST	
Parameter 1	-	
Return	Event	
3. /api/events/{id}		Updates an event by id
Http Method	PUT	
Parameter	id	
Return	Event	
4. /api/events/cancel/{eventId}		Enrolling for an event
Http Method	POST	
Parameter 1	eventId	
Return	-	

5.2 USERCONTROLLER

URL Exposed		Purpose
1. /api/users		Creates an user
Http Method	POST	
Parameter	-	
Return	User	
2. /api/users/login		Logins the user
Http Method	POST	
Parameter 1	-	
Return	User	
3. /api/users/logout		Logout the user
Http Method	POST	
Parameter	-	
Return	User	

4. /api/users/events/{eventId}/enroll		
Http Method	POST	Enrolls for an event
Parameter	eventId	
Return	-	

6 TEMPLATE CODE STRUCTURE

6.1 PACKAGE: COM.LOANAPPLICATION

Resources

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

6.2 PACKAGE: COM.VOLUNTEERPLATFORM.REPOSITORY

Resources

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

6.3 PACKAGE: COM.VOLUNTEERPLATFORM.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.
UserService (interface)	● Interface to expose method signatures for user related functionality. ● Do not modify, add or delete any method.	Already implemented.

6.4 PACKAGE: COM.VOLUNTEERPLATFORM.SERVICE.IMPL

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.

UserServiceImpl (class)	● Implements UserService. ● Contains template method implementation. ● Need to provide implementation for user related functionalities. ● Do not modify, add or delete any method signature	To be implemented.

6.5 PACKAGE: COM.VOLUNTEERPLATFORM.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
UserController (Class)	● Controller class to expose all rest-endpoints for user related activities. ● May also contain local exception handler methods	To be implemented

6.6 PACKAGE: COM.VOLUNTEERPLATFORM.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.
UserDTO (Class)	Use appropriate annotations from the Java Bean Validation API for validating attributes of this class.	Partially implemented.

6.7 PACKAGE: COM.VOLUNTEERPLATFORM.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 **id** as primary key. ● Map this class with an **event table**. ● Generate the **id** using the IDENTITY strategy	Partially implemented.

User (Class)	• This class is partially implemented. • Annotate this class with proper annotation to declare it as an entity class with **id** as primary key. • Map this class with a **user table**. • Generate the **id** using the IDENTITY strategy	Partially implemented.

6.8 PACKAGE: COM.VOLUNTEERPLATFORM.EXCEPTION

Resources

Class/Interface	**Description**	**Status**
NotFoundException (Class)	• Custom Exception to be thrown when trying to fetch or delete the entity info which does not exist. • Need to create Exception Handler for same wherever needed (local or global)	Already implemented.

1 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

    NOTE: After typing the second sql command (sudo systemctl start mysql), you may encounter a warning message like :

    ```
    System has not been booted with systemd as init system (PID 1). Can't operate.
    Failed to connect to bus: Host is down
    ```

    >> Please note that this warning is expected and can be disregarded. Proceed to the next step.

    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.**