**CS673 Software Engineering** 

**Team 2 - Project Name**

**Software Test Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Dipayan Mazumder | Security Lead | *DM* | 05-10-2024 |
| Pranjal Ekhande | Requirement Lead | *PE* | 05-10-2024 |
| Aman Jain | Configuration lead | *AJ* | 05-10-2024 |
| Mukul Jangid | Design and Implementation lead | *MJ* | 05-10-2024 |
| Praveen Singh | Team Lead | *PS* | 05-10-2024 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **1.0** | **Dipayan**  **Mazumder, Pranjal Ekhande, Aman Jain, Praveen Singh** | **05-10-2024** | **First Draft** |
| **2.0** | **Mukul Jangid** | **05-28-2024** | **Second Draft** |
| **3.0** | **Mukul Jangid** | **06-10-2024** | **Third Draft** |

[Testing Summary](#_heading=h.gjdgxs)

[Manuel Tests Reports](#_heading=h.30j0zll)

[Automated Testing Reports](#_heading=h.1fob9te)

[Testing Metrics](#_heading=h.3znysh7)

[References](#_heading=h.2et92p0)

[Glossary](#_heading=h.tyjcwt)

# Testing Plan

# **Test Objectives**

* Verify that all essential functionalities are implemented correctly.
* Ensure integration with other services is seamless.
* Validate the security features of the platform.

**Test Criteria**

* All test cases must pass without errors.
* No critical or high-severity defects should remain unresolved.

# Testing Summary

In this section, you will summarize what was tested, who is involved in testing, testing techniques used, and testing result. You may have the following tests

* + Unit Testing
  + Integration testing
  + System Testing
  + Acceptance Testing
  + Regression Testing

# Manual Testing Report

**Test Case 1: Retrieve All Projects**

* Test Case ID: TC001
* Test Name: Test Retrieve All Projects
* New or Old: New
* Test Items: ***ProjectController.getAllProjects()***
* Test Priority: High
* Dependencies: None
* Preconditions: The database must contain project data.
* Input Data: JSON payload with project details.
* Test Steps:

1. Send a GET request to ***/apiv1/project/getallprojects***.

* Postconditions: The response should contain a list of projects.
* Expected Output: A JSON array of projects.
* Actual Output: None
* Pass or Fail: Pass
* Bug ID/Link: None
* Additional Notes: None

**Test Case 2: Add Project**

* Test Case ID: TC002
* Test Name: Test Add Project
* New or Old: New
* Test Items: ***ProjectController.addProject()***
* Test Priority: High
* Dependencies: None
* Preconditions: Valid project data must be available.
* Input Data: None
* Test Steps:

1. Prepare input data:

*{*

*"projects": [*

*{*

*"projectid": "project003",*

*"projectname": "W1A206-8286-42A-SN2109F9W",*

*"userid": "user1",*

*"taskid": 24,*

*"description": "test 2",*

*"created\_on": 1622548800,*

*"updated\_on": 1622548800,*

*"state": "operating",*

*"type": "development"*

*},*

*{*

*"projectid": "project005",*

*"projectname": "W1A206-8286-42A-SN2109F9W",*

*"userid": "user1",*

*"taskid": 24,*

*"description": "test 2",*

*"created\_on": 1622548800,*

*"updated\_on": 1622548800,*

*"state": "operating",*

*"type": "development"*

*}*

*]*

*}*

1. Send a POST request to ***/apiv1/project/addprojects*** with the above data.

* Postconditions: The projects should be added to the database.
* Expected Output: A success message with the response code 200.
* Actual Output: None
* Pass or Fail: Pass
* Bug ID/Link: None
* Additional Notes: None

**Test Case 3: Edit Project**

* Test Case ID: TC003
* Test Name: Test Edit Project
* New or Old: New
* Test Items: ***ProjectController.editProject()***
* Test Priority: High
* Dependencies: None
* Preconditions: The project must exist in the database.
* Input Data: JSON payload with updated project details.
* Test Steps:

1. Prepare input data:

*{*

*"projectid": "project005",*

*"projectname": "W1A206-8286-42A-SN2109F9W",*

*"userid": "user1",*

*"taskid": 24,*

*"description": "new updated data",*

*"updated\_on": 1622549800,*

*"state": "operating",*

*"type": "development"*

*}*

1. Send a POST request to ***/apiv1/project/editProject*** with the above data.

* Postconditions: The project details should be updated in the database.
* Expected Output: A success message with the response code 200.
* Actual Output: None
* Pass or Fail: Pass
* Bug ID/Link: None
* Additional Notes: None

**Test Case 4: Add Tasks to a Project**

* Test Case ID: TC004
* Test Name: Test Add Tasks to a Project
* New or Old: New
* Test Items: ***TaskController.addTasks()***
* Test Priority: High
* Dependencies: None
* Preconditions: Valid task data must be available.
* Input Data: JSON payload with task details.
* Test Steps:

1. Prepare input data:

*{*

*"tasks": [*

*{*

*"task\_id": "task\_7",*

*"project\_id": "proj\_001",*

*"task\_name": "Task 7 Test",*

*"description": "Task 3 Test",*

*"status": "To Do",*

*"priority": "Low",*

*"assigned\_user\_id": 1,*

*"due\_date": 1749477368,*

*"created\_on": 1717941368,*

*"updated\_on": 1717941368*

*}*

*]*

*}*

1. Send a POST request to ***/apiv1/project/addtasks*** with the above data.

* Postconditions: The tasks should be added to the project in the database.
* Expected Output: A success message with the response code 200.
* Actual Output: None
* Pass or Fail: Pass
* Bug ID/Link: None
* Additional Notes: None

# Automated Testing Report

**Frameworks Used**: JUnit, Mockito, Spring Boot Test

**Location of Test Code**: The test code resides in the ***src/test/java/com/cs673olsum24/promanager/controller*** directory.

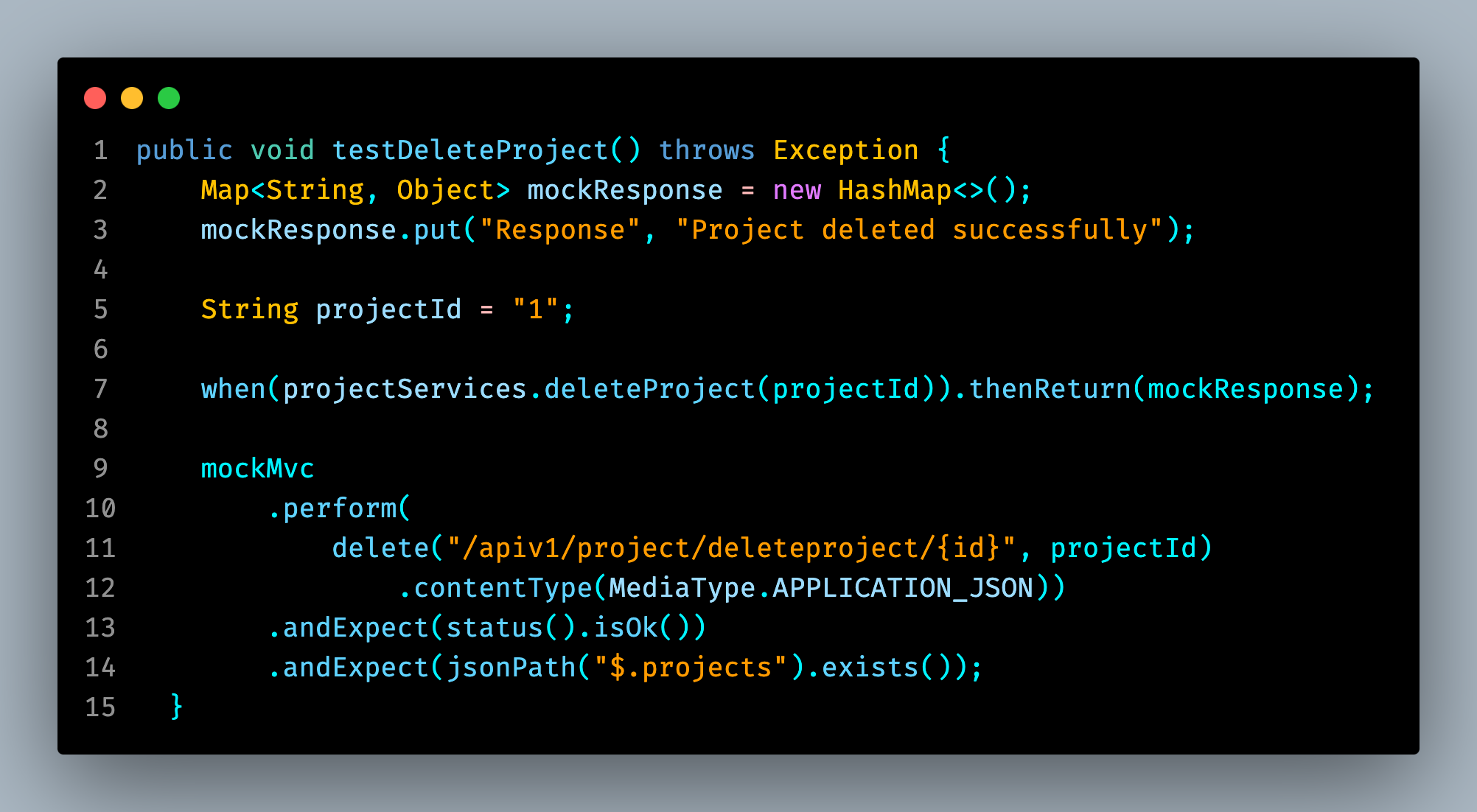
**Example Test Class**: ***ProjectControllerTest.***

#### **Test Case 1: Delete Project**

**Test Case ID**: ATC001  
**Test Name**: Test Delete Project  
**Test Description**: This test verifies the deletion of a project using the deleteProject endpoint.  
**Test Steps**:

* Mock the response for the deleteProject service.
* Send a DELETE request to ***/apiv1/project/deleteproject/{id}*** with a project ID.
* Verify that the response status is OK and the response contains the success message.

**Test Code**:



#### **Test Case 2: Add Project**

**Test Case ID**: ATC002  
**Test Name**: Test Add Project  
**Test Description**: This test verifies the addition of a new project using the addProject endpoint.  
**Test Steps**:

* Mock the response for the addProject service.
* Prepare the project data payload.
* Send a POST request to ***/apiv1/project/addprojects*** with the project data.
* Verify that the response status is OK and the response contains the success message.

**Test Code**:



#### **Test Case 3: Edit Project**

**Test Case ID**: ATC003  
**Test Name**: Test Edit Project  
**Test Description**: This test verifies the editing of a project using the editProject endpoint.  
**Test Steps**:

* Mock the response for the editProject service.
* Prepare the updated project data payload.
* Send a POST request to ***/apiv1/project/editProject*** with the data.
* Verify that the response status is OK and the response contains the success message.

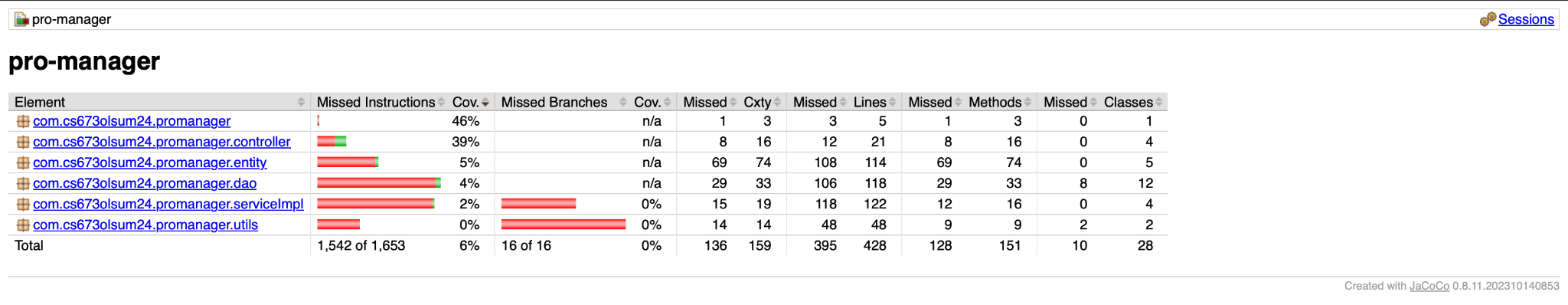
**Test Code**:



# Testing Metrics

We decided on these metrics

1. Number of Test Cases: **3 (Automated) + 3 (Manual)** [More will be added].
2. Test Coverage: 6%.
3. Defect Rate: Number of defects found per test case.
4. Test Execution Time: 7.259 seconds.
5. Cyclomatic Complexity: Measure of code complexity.



# References

1. Project Repository
2. Spring Boot Documentation
3. JUnit Documentation
4. Mockito Documentation

# Glossary

1. **API**: Application Programming Interface
2. **JUnit**: A unit testing framework for Java
3. **Mockito**: A mocking framework for unit tests in Java
4. **STD**: Software Test Document