Test Plan & Strategy for Repository Management Web Application

1. Scope

This document outlines the testing approach for the Repository Management Web Application. The key functionalities covered include:

- **Repository Management**: Creation, editing, and deletion of repositories.
- **Issue Tracking**: Creating issues, commenting, assigning labels, and closing issues.
- **Pull Request Handling**: Opening, reviewing, merging, and closing pull requests.
- **User Management**: Adding and removing users, role-based access control, authentication, and authorization.

2. Types of Testing

To ensure a high-quality application, the following test types will be conducted:

- **Functional Testing**: Validating that each feature works as intended.
- **API Testing**: Ensuring backend APIs return correct responses and handle edge cases.
- **Integration Testing**: Verifying the interaction between different modules and components.
- **Regression Testing**: Ensuring that new changes do not impact existing functionality.
- **UI/UX Testing**: Evaluating the user interface for usability, consistency, and accessibility.
- **Security Testing**: Assessing authentication, authorization, and protection against vulnerabilities.
- **Performance Testing**: Measuring response times, scalability, and system behavior under load.

3. Test Environment

Testing will be conducted across multiple environments:

- **Development Environment**: Used for initial testing during feature development.

- **Staging Environment**: A replica of production where comprehensive testing occurs before deployment.
- **Production Environment**: Monitored post-deployment for any live issues.

Test Data Requirements

- Sample user accounts with different access levels (Admin, Contributor, Viewer).
- Test repositories with various configurations.
- Example pull requests and issues to validate workflows.
- Simulated high user activity for performance evaluation.