Repository Management Web Application

1. Test Plan & Strategy

Scope

1. **Repository Management**
    - Creating, editing, deleting repositories
    - Verifying repository visibility settings (public/private)
2. **Issue Tracking**
    - Creating, editing, deleting issues
    - Assigning issues to users
    - Adding comments and labels to issues
3. **Pull Requests**
    - Creating pull requests
    - Reviewing and merging pull requests
4. **User Management**
    - Adding, removing, and updating user roles
    - Verifying permissions (admin, contributor, viewer)

Test Types

1. **Functional Testing** - Ensure core features work as expected.
2. **Regression Testing** - Verify that new updates do not break existing functionality.
3. **Integration Testing** - Check interactions between different parts of the application.
4. **UI/UX Testing** - Ensure a smooth user experience.
5. **Security Testing** - Identify issues such as authentication, unauthorized access to repos etc.
6. **Performance Testing** – Measure response times under load.

Test Environment

- **Development Environment**: For unit testing and early bug detection.
- **Staging Environment**: A replica of production used for E2E testing.
- **Production Environment**: Live environment.

Test Data Requirements:

- Sample users with different roles (admin, contributor, viewer)
- Pre-existing repos with various configurations
- Sample issues and pull requests

2. Test Case

See link to Test Case here Test Case Document

3. Automation Approach

Tests to Automate

1. Repository creation, issue creation, pull request creation
2. User permissions, pull request workflows
3. Form submissions for repository creation and issue tracking

Sample Automated Test (Selenium with Java & TestNG)

To view test script, refer to the file named RepositoryCreationTest.java or click the link to view Automation Test Script

Choice of Framework

- **Selenium with TestNG**: Supports cross-browser testing and integrates well with CI/CD.
- **Familiar Language**: I chose Java as it's my preferred language to use for Selenium.

- **Maintainability Considerations**: Use Page Object Model (POM) for better structure.

4. **Bug Reporting & Tracking**

1. Identify and reproduce the bug.
2. Log the issue into a tracking application.
3. Assign priority (Critical, High, Medium, Low).
4. Attach screenshots, screen recordings, logs, steps to reproduce for better understanding.

Bug Report Template

Bug ID	**BUG_001**
Title	Repository creation fails with special characters
Description	When a user enters special characters in the repository name, the creation process fails.
Steps to Reproduce	1. Login 2. Create a repository with the name “Test@Repo” 3. Click “Create”
Expected Result	The repository should be created, or a specific error message should be displayed.
Actual Result	The repo creation fails
Severity	High
Status	Open
Assigned To	Gospel Chukwuemeka

5. **Additional Considerations**

CI/CD Integration

- **Build & Test**
    - Code is pushed to GitHub.
    - CI pipeline (Jenkins, GitHub Actions) compiles and runs unit tests in dev environment.
- Deployment & Validation
    - Deploy to staging for further testing (automated/manual QA).
    - Post-deployment smoke tests ensure application stability.
    - If tests pass, deploy to production.

Managing Tests for New Features and Regression

- New feature tests are prioritized before release.
- Regression tests run automatically before each deployment.

Performance & Security Testing Strategies

- **Load Testing**: Use JMeter to simulate high traffic.