Tests to Automate & Why

For this Repository Management Web App, I'd automate tests that are repetitive, time-consuming, and critical for smooth functionality. These include:

- Smoke Tests – Basic tests to ensure key features (like login, repo creation) work after updates.
- Regression Tests – Making sure new changes don't break existing features.
- API Tests – Checking if backend APIs return correct responses and handle edge cases.
- Cross-Browser Tests – Ensuring the UI works consistently on different browsers.

What I Won't Automate

- Exploratory Testing – Some things need a human touch to catch unexpected bugs.
- UI/UX Testing – Visual elements, user experience, and accessibility are best reviewed manually.
- Small, one-off features

Choice of Framework & Maintainability Considerations

Why Selenium?

- Works well for UI automation across different browsers.
- Supports multiple programming languages (Python, Java, JavaScript).
- Can be integrated into CI/CD pipelines for automated execution.

Maintainability Considerations

- Use Page Object Model (POM) to organize locators and actions separately.
- Implement explicit waits instead of time.sleep() for better efficiency.
- Run tests in headless mode for faster execution in CI/CD.