1. Stakeholder Analysis

Key Stakeholders:

- **Development Team**: Requires fast feedback on code changes and test automation for CI/CD pipelines.
- **QA Team**: Needs a reliable framework to create, execute, and manage automated test cases.
- **Project Managers**: Require test reports and insights to track project progress.
- **Business Analysts**: Need validation that business requirements are met.
- **End Users**: Indirectly impacted by the framework's ability to ensure a bug-free experience.
- **IT & Security Teams**: Require compliance with security and performance standards.

Stakeholder Needs:

- **Ease of Use**: The framework should have a user-friendly interface and well-documented processes.
- **Integration**: It should integrate with existing development and CI/CD tools.
- **Scalability**: The framework must support various test types and adapt to project growth.
- **Reliability**: It must ensure consistent and repeatable test execution.
- **Reporting & Monitoring**: Should provide detailed logs and reports for tracking test results.

2. User Stories & Use Cases

User Stories:

1. **As a QA engineer**, I want to create and execute automated test cases so that I can validate application functionality efficiently.
2. **As a developer**, I want the framework to run tests in a CI/CD pipeline so that I can get instant feedback on code changes.
3. **As a project manager**, I want access to test reports so that I can track project quality.
4. **As a business analyst**, I want to ensure that automated tests cover all business requirements.
5. **As an IT administrator**, I want the framework to comply with security and performance standards.

Use Cases:

1. **Automated Test Execution**: The framework runs test cases and logs results.
2. **Continuous Integration (CI/CD) Support**: Tests are triggered automatically after each code commit.
3. **Test Case Management**: QA teams can create, modify, and manage test cases easily.

4. **Multi-Browser and Device Testing**: Ensure the web application works across different environments.
5. **Report Generation**: Detailed test execution reports are generated and shared with stakeholders.
6. **Integration with Issue Tracking Tools**: Automatically create bug reports in JIRA if a test fails.

3. Functional Requirements

1. **Test Automation Support**:
    - Support for functional, regression, and smoke testing.
    - Ability to write and execute test scripts.
2. **Integration Capabilities**:
    - Seamless integration with CI/CD tools (Jenkins, GitHub Actions, etc.).
    - Integration with test management tools (JIRA, TestRail, etc.).
3. **Multi-Browser Testing**:
    - Support for Chrome, Firefox, Edge, and Safari.
4. **Test Execution Management**:
    - Parallel test execution capability.
    - Scheduled test execution.
5. **Test Reporting & Logging**:
    - Generate detailed logs and test reports.
    - Real-time monitoring dashboard.
6. **Data-Driven Testing**:
    - Ability to run tests with multiple sets of input data.
7. **Security & Authentication**:
    - Support for testing authentication mechanisms (OAuth, SSO, etc.).
    - Secure test data storage.
8. **Scalability & Extensibility**:
    - Ability to add new test modules easily.
    - Support for API and database testing.

4. Non-Functional Requirements

1. **Performance**:
    - Test execution should complete within a reasonable timeframe.
    - Framework should efficiently handle concurrent test execution.
2. **Security**:
    - Secure handling of sensitive test data.
    - Restricted access based on user roles.
3. **Usability**:
    - Easy-to-use test script creation and execution interface.
    - Clear and detailed documentation.
4. **Reliability**:
    - Ensures minimal false positives/negatives.
    - High uptime for automated test execution.
5. **Maintainability**:
    - Modular design for easy updates and extensions.
    - Readable and reusable test scripts.

6. **Compatibility**:
    - Works across different operating systems (Windows, macOS, Linux).
    - Compatible with multiple versions of browsers and frameworks.

This structured approach ensures that all key requirements for an automated testing framework are captured effectively, helping to build a robust, scalable, and efficient solution.