Introduction

An Automated Test Plan is a document that explains how software testing will be performed automatically using tools such as Selenium, Playwright, or Cypress, instead of manual testing.

This project focuses on designing and executing an end-to-end automated testing plan for the GrabDocs web application using Selenium WebDriver and Playwright. The goal is to verify that major user workflows, such as login, document upload, search, and file export, function correctly across different browsers.

Both tools are used to demonstrate their unique strengths:

- Selenium: Best for broad cross-browser automation and traditional UI workflows.
- Playwright: Best for modern, dynamic web pages with faster and more reliable automation.

The test scripts are written in Python, stored in GitHub, and executed to validate end-to-end user interactions. This ensures the GrabDocs platform delivers a stable and consistent user experience.

System Under Test (SUT)

- Application: GrabDocs Web Application, SauceDemo.com
- Tools: Selenium (Python) & Playwright (Python)
- Testing Type: Automated, Black-box, End-to-End
- Scope: Web UI only (no API, backend, or database testing)

Tool Division

Selenium

Used for:

- Login
- Upload
- Export/Download
  (Traditional UI, form-heavy, cross-browser)

Playwright

Used for:

- Search
- Document preview

- Logout
  (Faster automation, dynamic content behavior)

Test Objectives

This automated test plan aims to:

- Verify that the most important GrabDocs workflows function correctly from the user's point of view.
- Use Selenium and Playwright to automate browser actions such as login, upload, search, and document viewing.
- Ensure these workflows behave consistently across different browsers (Chrome, Edge).
- Demonstrate understanding of automation tools by writing clean, organized, and maintainable test scripts.
- Increase testing accuracy and reduce manual repetition through automation.
- Verify that key SauceDemo e-commerce workflows function correctly from a user's perspective.
- Use Selenium to automate browser interactions such as login, add-to-cart, checkout, and logout.
- Ensure that the workflows perform consistently in Chrome and Edge.
- Demonstrate understanding of end-to-end web testing automation.
- Improve testing efficiency and reduce manual test repetition through automation.

Test scope

The test scope outlines what is included and excluded in the automated testing of the GrabDocs web application. This project focuses only on web-based user interactions to ensure that the core features function correctly from the end user's perspective. All tests are executed through the browser using Selenium and Playwright, without any access to backend systems or internal APIs.

The in Scope includes:

- User login and invalid login attempts
- Document upload for supported and unsupported file types
- Searching for documents by name
- Opening and previewing uploaded documents
- Exporting or downloading documents
- Logging out of the application
- General UI behavior and navigation
- Both positive and negative test scenarios

The out Scope include:

- Backend or API-level testing
  Database verification or data integrity checks
- Evaluation of AI-generated responses or model accuracy
- Mobile or tablet platform testing
- Server performance, load testing, or infrastructure validation

Testing Levels

- **Functional Testing** – test each feature (login, upload, search, export).
- **Regression Testing** – re-test all features after updates.
- **Negative Testing** – test invalid actions (wrong password, unsupported file).
- **End-to-End Testing** – test the complete workflow (login → upload → search → logout).

Key Test Scenarios

This section describes the major test scenarios selected for automation, focusing on the most important functionalities of the GrabDocs platform:

Login Scenarios

- Login with valid credentials - user enters dashboard.
- Login with invalid password - error message displayed.
- Login with empty fields - validation message.

Document Upload Scenarios

- Upload a valid PDF file - upload completes and confirmation appears.
- Upload a DOCX file - file is accepted and processed.
- Upload an unsupported file type (e.g., ZIP) - system shows “file type not supported.”
- Attempt to upload a very large file - system should show a warning or take longer.
- Cancel an upload halfway - upload stops and the UI resets properly.

Search Scenarios

- Search for an existing uploaded document - document appears in search results.
- Search for a name that does not exist - “no results found” is shown.
- Search immediately after a new upload - newly uploaded document appears.

Document Preview Scenarios

- Open a successfully uploaded document - preview loads.
- Try to open a document before upload completes - system should show a processing message.

- Refresh the page while viewing a document - preview should reload or session should update.

Export / Download Scenarios

- Export a document - file downloads successfully.
- Cancel a download - download is stopped.
- Attempt export on a corrupted file - system should show an error.

Logout Scenarios

- Click logout from the dashboard - user returns to login page.
- Attempt to go back using the browser “Back” button - user should stay logged out.

Negative / Error Handling Scenarios

- Upload a corrupted file - error message shown.
- Perform an action with a poor network (turn Wi-Fi off during upload) - clear failure or retry message appears.
- Enter invalid characters in search - the system still handles input gracefully.

Performance / Responsiveness

- Measure how long it takes to upload a 5–10MB file.
- Measure the time for a search result to appear.
- Check if the UI responds correctly when switching between pages.

Accessibility Checks

- Verify that buttons and links are keyboard-navigable.
  Check if main UI elements have proper labels.

Test Environment

The automated tests will be executed in a controlled environment to ensure consistent and reliable results. All testing will be performed on the GrabDocs web application, accessed through modern web browsers. The primary testing machine will run Windows 11, and the tests will be executed using both Selenium WebDriver and Playwright, each configured to automate browser actions through Python scripts.

The tests will run on the latest versions of Google Chrome and Microsoft Edge, ensuring compatibility across widely used browsers. Stable internet connectivity is required for accessing the GrabDocs platform and for uploading documents during testing.

The test environment will use synthetic test data, including sample PDF and DOCX files, and dedicated test user accounts to avoid interfering with real user data. All automated test scripts, configurations, and resources will be stored and version-controlled in GitHub.

Selenium and Playwright Script Implementation

Execution Screenshots

Conclusion

This automated test plan provides a structured approach to validating the GrabDocs web application using Selenium and Playwright. By focusing on key functionalities and end-to-end workflows, this plan ensures consistent, repeatable, and reliable automated testing that supports the stability and quality of the GrabDocs user experience.