The Selenium Automation Testing for Spyne.ai Image Upscaler involved setting up the environment, designing and running test cases for file uploads, image upscaling, and UI validation. The project used Java, Selenium, TestNG, Log4j, Extent Report, Page Object Model, and Robot Framework, and included cross-browser testing, headless execution, and CI/CD integration.

1. Environment Setup

=> Actions Taken:-

- Installed Selenium WebDriver: For the chosen programming language (Java), the Selenium WebDriver was installed to enable browser automation.
- Set Up Browser Driver: A browser driver, such as ChromeDriver for Chrome or GeckoDriver for Firefox, was set up to interact with the web browser. This involved downloading the driver executable and configuring it so that Selenium can use it to control the browser.
- Installed Necessary Libraries: Libraries like `TestNG` for Java were installed to facilitate test execution and management.

=> Result:-

- The testing environment was successfully set up, allowing for the automation of web interactions. The browser driver was correctly configured, and the necessary testing libraries were available for use.

2. Test Case Design

=> Actions Taken:-

- Identified Key Functionalities: The core functionalities of the Spyne.ai Image Upscaler were identified, such as navigating to the website, uploading files, and upscaling images.
- Designed Test Cases: Test cases were created to cover these functionalities:
  - Navigation: Ensured that the website loaded correctly and that the URL and main elements were present.
  - File Upload: Checked if different image formats could be successfully uploaded.
  - Image Upscaling: Verified that the image upscaling process was triggered and completed as expected.
  - UI Validation: Ensured that all key user interface elements, like buttons and previews, were functioning properly.
  - Error Handling: Tested how the application handled invalid file uploads.

- Download Functionality: Confirmed that the upscaled image could be downloaded.
- Performance Testing: Measured how long it took to upscale images of various sizes.

=> Result:-
- Comprehensive test cases were designed to cover all important aspects of the Image Upscaler's functionality.

3. Automated Test Script Implementation

=> Actions Taken:-
- Navigation Test: Automated the process of navigating to the Image Upscaler page and verified the page loaded correctly.
- File Upload Test: Automated the upload of valid image files and confirmed the files were accepted.
- Invalid File Upload Test: Attempted to upload invalid file types to ensure the application handled errors correctly.
- Image Upscaling Test: Automated the image upload and verified that the upscaling process was completed successfully.
- UI Validation Test: Checked that all user interface elements, such as buttons and image previews, were present and functional.
- Download Functionality Test: Automated the download of the upscaled image and ensured it was successful.
- Performance Testing (Optional): Measured the time required to upscale images of different sizes to evaluate performance.

=> Result:-
- The automated tests covered all identified functionalities. The scripts executed the intended actions and validated the outcomes, ensuring the application worked as expected.

4. Test Execution

=> Actions Taken:-
- Executed Test Cases: Ran the automated test scripts across different browsers (e.g., Chrome, Firefox) to ensure cross-browser compatibility.
- Validated Outcomes: Compared the actual results of the tests with the expected outcomes to determine if the tests passed or failed.

=> Result:-

- The tests were executed successfully on multiple browsers, with results documented to show which tests passed and which failed. Any discrepancies were noted for further investigation.

5. Reporting

=> Actions Taken:-

- Generated Test Report: Created a summary report that included the results of each test case, such as pass/fail status and any errors encountered.
- Included Screenshots: Captured screenshots for any failed test cases to provide visual evidence of issues.

=> Result:-

- A detailed report was produced, summarizing the results of the tests. The inclusion of screenshots for failed cases helped in diagnosing and addressing issues.

6. Bonus Tasks (Optional)

- Cross-Browser Testing:
  - Implemented tests to run on multiple browsers simultaneously, ensuring the application's functionality across different platforms.
- Headless Testing:
  - Configured tests to run in headless mode (without a GUI) to optimize execution time and efficiency.
- CI/CD Integration:
  - Integrated the tests into a Continuous Integration/Continuous Deployment (CI/CD) pipeline using tools like Jenkins to automate test execution with each code change.

=> Result:-

- The additional tasks enhanced the testing process by improving test coverage, execution efficiency, and integration with development workflows.