Written Evaluation

1. Vision for automating change detection for multiple landing pages

My approach to change detection is based on monitoring updates in the build repository and triggering tests only when changes are introduced. When a commit or pull request is created, the CI pipeline identifies which files or components were modified and determines the impacted areas of the landing pages.

Automated tests are grouped and tagged by functionality, such as content, layout, or forms. Based on the detected changes, the CI system runs only the relevant tagged tests instead of the full test suite.

This ensures faster feedback, reduces unnecessary test execution, and keeps the automation scalable as the number of landing pages grows.

2. Processes and tools I would use (including AI)

I would use a clear and stable automation process supported by modern, reliable tools. UI automation would be implemented using a framework such as Playwright, integrated into the CI pipeline to run tests automatically on each relevant change.

Tests would be organised by purpose and tagged to support selective execution. Visual checks would be used for key landing page sections, while functional tests would cover critical user interactions such as forms and call-to-action buttons.

AI would be used as a supporting tool to improve efficiency, for example by analysing test failures, highlighting recurring issues, and assisting with test maintenance when page structure changes.

The overall focus would be on fast feedback, test reliability, and easy maintenance.

3. Most challenging situation in test automation & how I resolved it

In my current role, when I joined the team, there was no standard for test element IDs. Because of this, automated tests were using unstable selectors and often failed whenever the UI changed, which caused flaky tests and unreliable CI results.

To fix this, I created a simple and clear standard for test element IDs and presented it to the team. The team agreed with the approach, but due to workload and priorities, no developer was available to implement the changes at that time.

To speed things up and reduce the flakiness as soon as possible, I volunteered to implement the changes myself, working closely with the developers. After applying the standard and updating the tests, the test flakiness was greatly reduced, the CI pipeline became stable, and build validation became more reliable.

This also helped improve confidence in the automation and made the tests easier to maintain in the long term.