What additional checks would you incorporate into the current tests?

Handling Pop-Ups, Alerts, and iFrames

I didn’t face many issues while executing the behavior, except for one Shadow DOM element. However, it's always a good idea to have a set of pre-defined utility functions to reduce boilerplate code and improve efficiency.

Reporting

Second, it will be essential to have good reporting. It will help other team members track test results efficiently and make it easier to share insights with management.

Performance & Stability Checks

It is crucial to monitor the execution time, as test suites may take longer to run as they grow. So its important to maintain efficiency, and ensure faster feedback for developers.

Retry Mechanism for Flaky Tests

In automated testing, flaky test can occur due to network delays, slow page loads, or temporary issues. Instead of failing a test immediately, a retry mechanism helps rerun the test a set number of times before marking it as failed.

Parallel Execution & CI/CD Integration

Parallel Execution can be really helpful as it will decrease the execution time as well as increase Resource Utilization in a better way.

What new tests would you recommend implementing next?

Module	High Level Scenario
Onboarding /Registration	Invalid login. Password reset functionality.
Core Functionalities	Search bar. Footer menu.
UI/UX & Navigation	Error messages & validation.
Security & Permissions	Direct URL access restriction.
Performance	Load testing.
	Create an end-to-end (E2E) test that also validates the API response.

Does the on-boarding and document upload processes exhibit any non-automatable properties? If so, how would you approach testing these properties?

1. Email Verification via Link (Difficult to Automate)

Why is it difficult?

- The verification link is sent to an external email system, requiring access to an email client or API.
- Clicking the link redirects the user to a browser, making UI automation tricky.

Approach to Testing:

- **Workaround**: Use temporary email services like Mailinator, Yopmail, or a test SMTP server to automate fetching emails via API.

- **Alternative**: Mock the backend response to bypass email verification for automated testing.
- **Manual Testing**: Final verification should still involve a human tester for real-world accuracy.

2. Document Preview for Prime Users (Partially Automatable)

Challenges in Automation:

- Prime users can directly access the document.
- Non-prime users must watch an ad before gaining access.
- Ads may include videos, captchas, or unpredictable interactions, making automation unreliable.

Approach to Testing:

- **Mock API Responses**: Instead of relying on UI, stub API calls to return prime or non-prime user responses.
- **Headless Testing for Ads**:
    - Use **AdBlock extensions** in Selenium to bypass ad-watching.
    - Monitor ad completion events before proceeding with the test.
- **Manual Testing**:
    - Since ads can change frequently, a final manual test should ensure expected behavior.

3. Premium Subscription for Document Download (Can Be Automated)

Testing Non-Prime vs. Prime Users:

- Non-prime users **can only download their own uploads**.
- Prime users **can download any document**.

Approach to Testing:

- **Test for Non-Prime Users:**
    - Upload a document.
    - Attempt to download another user's document (**should be blocked**).
    - Attempt to download the uploaded document (**should be allowed**).
- **Test for Prime Users:**

    - Ensure they can download any document.
- **API Testing Alternative:**
    - Call the document download API directly with different user roles to verify permissions.

4. Additional Non-Automatable Challenges

Dynamic Ads & A/B Testing Features

- Ads and promotional banners might change frequently, causing UI automation to break.
- A/B testing dynamically serves different versions of the same page to users, making it difficult to predict the exact UI elements.
- Elements may have different locators or may not be present at all, depending on the test variation.
- Certain A/B test groups may require specific user attributes or cookies, making it hard to control which version is displayed.
- Solution:
    - Use feature flags to enable/disable certain flows in a test environment.
    - Rely on backend verification rather than UI automation.

Third-Party Payment Integrations (if applicable to subscription purchase)

- If subscription upgrades require a real payment, automating this is difficult.
- Solution:
    - Use test credit card numbers provided by payment gateways
    - Verify API responses for successful transactions instead of relying on UI-based payment confirmation.