Senior SQA Engineer Take-Home Test

Objective:

Evaluate the candidate's skills in developing automated test scripts, designing detailed manual test cases, and analyzing test coverage, especially for complex scenarios.

Task 1: Automated Testing

Write automated test scripts using a testing framework of your choice (e.g., Cypress, Playwright or webdriver io with javascript). The test cases should cover the following scenarios for a web application, with consideration for handling ReCaptcha in the login flow.

Instructions:

1. **Login Feature (ask.permission.io):**
    a. Implement test scripts to:
        i. Starting the automation on the **www.permission.io** page.
        ii. Validate a successful login with valid credentials.
        iii. Test login failure with incorrect credentials.
        iv. Test account lockout after multiple failed login attempts.
    b. **ReCaptcha Handling:**
        i. **Option 1:** Use manual intervention of script
        ii. **Option 2 (Bonus Points):** Design and or implement automated ReCaptcha handling
2. **AI Agent Feature (https://ask.permission.io/ai)**
    a. **Basic Chat Functionality:**
        i. **Pre-Chat Validation:**
            1. AI page loads correctly and chat interface is visible
            2. Input field and send button are functional
            3. Placeholder text and UI elements render properly
    b. **Chat Interaction Tests:**
        i. **Message Handling:**
            1. Send simple text messages and validate responses
            2. Verify message history persistence during session
            3. Validate timestamps on messages
3. **README File Requirement:**
    a. **Create a README.md file** that includes:
        i. Detailed instructions for setting up the required technologies to run the test scripts (e.g., how to install dependencies and any necessary configurations).

ii. Steps to execute each test path, including running the automation and validating the desired outcomes.
iii. Additional notes on handling ReCaptcha, if applicable, and any special considerations for running the tests.

Guidance:

Form Submission:

- Verify that required fields trigger appropriate validation messages.
- Ensure the form can be submitted successfully when all fields are filled correctly.

UI Element Checks:

- Ensure all buttons and links are working as expected.
- Test for the correct display of dynamic content (e.g., popups or modals).

Deliverables:

- Automated test scripts (with logic for handling ReCaptcha as described above).
- A README.md file as outlined above.
- A test report indicating passed/failed tests.

Task 2: Test Case Design and Coverage Analysis

Objective: Design comprehensive manual test cases for the **"Daily Earn"** feature on the Permission platform, where users can search for brands at https://ask.permission.io/earn. Test cases should address **functional correctness, edge cases, performance, and user experience**. Additionally, specify validation steps both from the **user's perspective** and through **system/database verification**.

Requirements:

1. **User Actions:**
   - User initiates a search for a brand.
   - User selects a brand from search results.
   - User completes the Brand "Daily Earn" action to receive rewards.
   - User views brand in “My Brands”
2. **Expected Outcomes:**
   - Successful search and selection process.
   - Proper allocation of rewards upon completion.
   - Accurate and consistent user feedback (notifications, UI updates).

***Optional Bug Reporting:**

- While creating test cases, you may optionally document **1 or 2 bugs** encountered in the application. For each identified bug:
    i. **UX Bug**: Highlight issues affecting usability or intuitiveness.
    ii. **Broken Functionality**: Document any feature that does not work as expected.
    iii. Provide a **brief description of the bug** in the test case document, including:
        1. **Bug Title**: Short description.
        2. **Steps to Reproduce**: Instructions to replicate.
        3. **Expected vs. Actual Results**.

Deliverables:

1. **Detailed Test Cases Document**:
    - List manual test cases with **clear steps, expected results, and priority**.
    - Consider **edge cases** (e.g., no results, timeout scenarios, invalid characters in search).
    - User experience (e.g., intuitive navigation, accessible feedback messages).
2. **Success Validation Strategy**:
    - **User Perspective**:
        - Check for Confirmation Messages or Indicators
        - Observe Real-Time Updates
    - **System/Database Validation**:
        - Provide a list of expected validation checks for the backend systems and database. Although you won't have access, specify what fields, data types, and records would be necessary to verify successful completion.
3. **Comprehensive Coverage Report**:
    - Outline how test cases cover functionality, and user experience.
    - Indicate any testing limitations, potential risks, or areas where additional testing is recommended.

Task 3: Exploratory Testing

Objective: Conduct unscripted exploratory testing to discover potential issues, usability problems, and edge cases that may not be covered by formal test cases. This task evaluates the candidate's ability to think critically and test beyond predefined scenarios.

Requirements:

Target Areas for Exploration:

- **Daily Earn Feature** (https://ask.permission.io/earn)
- **AI Agent Feature** (https://ask.permission.io/ai)

- **Register Flow + Chrome Extension Install**
- **Navigation and User Journey**

Exploratory Testing Approach:

Examples:

1. **Session-Based Testing**: Conduct focused 15-30 minute exploratory sessions for each target area
2. **Charter-Driven Exploration**: Define brief testing charters such as:
    - "Explore the search functionality for brands to discover usability issues and unexpected behaviors"
    - "Investigate the AI chat feature's response handling under various input conditions"
    - "Test the user journey from login to completing a daily earn task"

Documentation Requirements:

For each exploratory session, document:

- **Session Charter**: Brief description of the exploration focus
- **Time Spent**: Duration of the exploratory session
- **Areas Covered**: Specific features/workflows explored
- **Findings**: Issues, observations, or potential improvements discovered
- **Questions Raised**: Any uncertainties or areas requiring clarification
- **Risks Identified**: Potential problem areas that may need additional testing

Testing Techniques to Apply:

- **Boundary Testing**: Test limits of input fields, search queries, etc.
- **Error Handling**: Attempt to break the application through unexpected inputs
- **User Workflow Variations**: Test alternative paths through the application
- **Cross-Browser/Device Testing**: Test on different browsers or devices if possible
- **Performance Observation**: Note any slow loading times or unresponsive elements

Deliverables:

- **Exploratory Testing Report**: Summary document containing:
    - Session notes for each exploratory testing session
    - List of issues/bugs discovered (prioritized by severity)
    - Usability observations and recommendations
    - Areas recommended for additional testing
    - Overall risk assessment based on exploration findings

General Guidelines:

- **Time Limit:** none, but don’t spend too much time. We want to see how much you can do in a timebox. Limit yourself to x hours.
- **Evaluation Criteria:**
    - Code quality and maintainability (for automation).
    - Thoroughness of test cases and coverage.
    - Communication clarity in writing, documentation, test strategie,s and optionally bugs reported.
    - Problem-solving.