Tester Practical Screening Assignment

Purpose of This Assignment

This assignment is designed to evaluate whether you can operate as a **product-minded quality tester** in a startup environment.

Our primary focus is on **how you think as a user and product tester**, not just on the number of bugs you find or your familiarity with a tool.

In particular, we want to assess whether you can:

- Understand a product logically and holistically
- Identify broken flows, logical gaps, and usability issues
- Anticipate real user behavior and edge cases
- Demonstrate strong **problem-finding and reasoning skills**
- Communicate observations clearly and structurally

Automation is important, but it is **secondary** to product-level thinking in this assignment.

Important Context

- Testing here is **not limited to predefined test cases**
- You are expected to explore, question assumptions, and break flows
- Reasoning, clarity, and ownership matter more than raw bug count
- The tools and frameworks used **do not reflect our internal tech stack**
- This assignment is intentionally **stack-agnostic**, focusing on:
    - Testing fundamentals
    - Product understanding
    - Logical reasoning
    - Automation thinking

These skills apply equally to frontend, backend, and AI/ML systems.

Submission Requirements (Strict)

- Create **ONE Google Spreadsheet**
- The spreadsheet must contain **TWO SEPARATE SHEETS**:
    1. **Product-Level Testing (Manual & Exploratory) → Sheet 1 (Main Priority)**
    2. **Automation Testing → Sheet 2 (Secondary Priority)**
- You are free to design the structure of each sheet
  (Sheet design and clarity are part of the evaluation.)
- Content must be clear enough for a **non-technical HR reviewer**
- Share the spreadsheet with **View access**

SHEET 1: PRODUCT-LEVEL TESTING

(Manual Exploratory Testing – Core Evaluation)

Objective

This sheet carries the **highest weight** in evaluation.

We want to assess your ability to:

- Validate whether the product is **logically working as expected**
- Identify issues in:
    - Core logic and workflows
    - Functional behavior
    - Error handling and edge cases
    - UI and UX (as part of the overall experience)
- Think like a real user, not just a tester, following the steps
- Demonstrate strong **problem-finding skills**

This sheet answers the key question:
“Is this candidate good at understanding a product and finding meaningful problems?”

SaaS Product to Test

Use the public demo of OrangeHRM:
👉 https://opensource-demo.orangehrmlive.com/

Login Credentials

- Username: `Admin`
- Password: `admin123`

Treat this as a **real client-facing SaaS product**, not a demo.

What You Must Do

- Perform **manual exploratory testing**
- Navigate across multiple modules
- Test realistic user flows and actions
- Focus first on:
    1. **Logical correctness**
    2. **Functional behavior**
    3. **Error handling and edge cases**
    4. **UI clarity**
    5. **UX flow and usability**

UI and UX issues should be evaluated **in the context of functionality and logic**, not in isolation.

Key Areas to Explore (Indicative, Not Exhaustive)

You are expected to explore multiple areas, including but not limited to:

- Login & authentication behavior
- Dashboard usefulness and clarity
- Navigation and menu structure
- Forms and data entry flows
- Validation messages and error handling
- Tables, filters, pagination, and views
- Flow between actions (create → edit → view → delete)
- Consistency and predictability of behavior

What to Capture in the Spreadsheet (Sheet 1)

For each issue, observation, or suggestion, document:

- Module/Page name
- Issue type
  (Logical Issue / Functional Bug / UX Issue / UI Issue / Improvement / Observation)
- Clear description of the issue or behavior
- Steps or context in which it occurs
- User impact
  (confusion, wrong data, blocked flow, inefficiency, etc.)
- Severity
  (Low / Medium / High)
- Suggested improvement or fix (if applicable)

You may also note **positive observations** where things work well.

What We Are Evaluating in Sheet 1

- Ability to understand the product deeply
- Logical reasoning and flow validation
- Quality of issues found (not cosmetic nitpicking)
- User-centric and business-aware thinking
- Severity prioritization
- Depth, curiosity, and ownership

SHEET 2: AUTOMATION TESTING

(Code-Based Automation – Mandatory)

Objective

This sheet evaluates your ability to:

- Understand an existing automation framework
- Implement and execute automated tests
- Identify regression-prone and high-risk flows
- Communicate automation results clearly

Automation is important, but it is **secondary** to product-level testing in this assignment.

Application Under Test

👉 https://www.saucedemo.com/

This is a demo e-commerce application used to simulate real-world testing scenarios.

Automation Framework (Mandatory Source)

Use the official Sauce Labs ImageRunner example repository:
👉 https://github.com/saucelabs/saucectl-imagerunner-example

Choose **ONE** of the following sub-projects:

- `java-junit-selenium`
- `java-testng-selenium`
- `playwright` (JavaScript / TypeScript)
- `webdriverio-desktop-browser`
- `testcafe` (optional; may be unstable)
- `java-junit-appium` (only if comfortable with mobile automation)

⚠️ **Mandatory**

- You must write and execute automation scripts
- Conceptual or theoretical test cases alone are **not sufficient**

What You Must Do

- Clone the repository
- Set up the selected framework locally
- Run at least one existing sample test
- Write your own automated test scripts
- Execute the tests
- Observe failures, gaps, and unexpected behavior
- Document all findings in the spreadsheet

Mandatory Scenarios to Automate

Automate **at least 5–7 scenarios**, including:

- Log in with valid credentials
- Login with invalid credentials
- Product listing validation
- Add product(s) to cart
- Cart content validation
- Checkout flow (basic happy path)
- Logout (optional but recommended)

What to Capture in the Spreadsheet (Sheet 2)

For each automated test case, document:

- Test case name
- User flow/feature covered
- Tool & framework used
- Test type (Smoke / Functional / Regression)
- Execution status (Pass / Fail)
- Observed behavior
- Bugs or inconsistencies found
- Risks of missing coverage
- Suggestions for further automation

You are **not required to submit your code repository**; however, the documentation must clearly indicate that tests were implemented and executed.

What We Are Evaluating in Sheet 2

- Evidence of real automation work
- Logical scenario selection
- Understanding of regression risks
- Clarity of reporting
- Ownership mindset (what else should be automated and why)

Marking Criteria (Weightage)

Area of Evaluation	Weight
Product-Level Testing (Sheet 1)	**65%**
Logical & functional coverage	25%
Problem-finding & reasoning skills	20%
User-centric thinking (UX in context)	10%
Clarity, structure, and severity judgment	10%
Automation Testing (Sheet 2)	**35%**
Correct setup & execution	10%
Scenario relevance & coverage	10%
Observations & risk identification	10%
Documentation clarity	5%

General Expectations (Both Sheets)

- Clear, structured, and readable documentation
- No generic or copy-pasted content
- Honest observations (working features can be noted)
- Quality of thinking over quantity of findings

Important Notes

- This is **not a timed assignment**—quality > speed
- You are encouraged to explore edge cases
- You are not expected to find everything
- We are evaluating **how you think and communicate**, not perfection