QA Automation Assignment — HelloBooks (Login + Onboarding/Signup)

Product

HelloBooks — finance/accounting SaaS
Target under test (UAT): https://dev.hellobooks.ai/

Goal: Assess your ability to analyze user flows, design robust test cases, and implement a maintainable UI automation suite for **Login** and **Onboarding/Signup**.

You will:

1. map the flows,
2. write test cases,
3. automate high-priority scenarios,
4. produce reports & documentation.

In-scope user journeys

1. **Signup / Onboarding**
    a. New user registration (happy path)
    b. Field validations (required, format, length)
    c. Email/phone/OTP verification (mock/simulate if real OTP is blocked)
    d. Password policy & masking
    e. Duplicate email handling
    f. Post-signup onboarding steps (e.g., organization/entity setup if prompted)
    g. Redirects & first-login state
2. **Login**
    a. Valid login
    b. Invalid login (wrong password, unregistered email)
    c. Locked/disabled user handling (if applicable; otherwise propose how you'd test)
    d. Remember me (if present)
    e. Logout and session invalidation
    f. Forgot password link presence (automation optional)

What to Deliver

1. **Test Case Document** (Excel/Google Sheets)
    a. Columns: `ID | Suite (Login/Signup) | Title | Pre-reqs/Test Data | Steps | Expected Result | Priority (P1/P2/P3) | Type (Positive/Negative) | Status (for your run)`
    b. Provide at least **25 test cases** (balanced across Login & Signup).
2. **Automation Suite (5–10 tests min)**
    a. Selenium + TestNG / JUnit, or Playwright for Java .
    b. Use Page Object Model (or Screenplay) and sensible waits (no brittle sleeps).
    c. Include retry/flake strategy where needed.
    d. Record artifacts (screenshots/video on failure).
3. **Execution Report**
    a. HTML/JSON/JUnit output + brief summary (pass/fail counts, notable defects).
    b. Link to run video (if your framework supports it).
4. **README**
    a. Setup (prereqs, node/java versions), install, run, config
    b. How to switch env/credentials
    c. How OTP is mocked/simulated (see "Test Data & Assumptions")
5. **Defect Log** (if you find issues)
    a. `ID | Title | Steps to Reproduce | Actual | Expected | Evidence (screenshot/video) | Severity`

Environment, Test Data & Assumptions

- **Env:** https://dev.hellobooks.ai/
- **Accounts:** Create your own test account(s). If OTP/email verification is required and blocked:
    - Propose & implement one of:
        - Use a disposable inbox (e.g., MailSlurp/Mailinator) and pull OTP via UI/API; or
        - Add an **OTP seam**: stub/mocking layer in your test to bypass OTP UI and hit the post-verification step (explain how in README); or
        - Parameterize OTP from environment variable and manually input during run for one case; automate others by mocking.
- **Data:** Generate unique emails: `qa.automation+<timestamp>@example.com`

- **Browsers:** Run on **Chromium**; bonus for adding **Firefox/WebKit** (Playwright) or **Chrome/Edge** matrix.

High-Priority Scenarios to Automate (suggested)

Signup / Onboarding

1. P1 — Happy path signup → verification (real or mocked) → first login → onboarding page reached.
2. P1 — Required fields show validation when empty; inline messages are readable.
3. P1 — Invalid email format rejected; helpful error shown.
4. P1 — Weak password rejected per policy; error shown.
5. P2 — Duplicate email during signup blocked; user sees proper message.
6. P2 — Post-signup redirect correctness (dashboard/onboarding wizard).

Login

7. P1 — Valid login → lands on authenticated start page; logout works; session ends.
8. P1 — Invalid password shows error; no session created.
9. P1 — Unregistered email shows error; no session created.
10. P2 — "Remember me" persists session (if present) across reload; logout clears it.

Sample Test Cases (starter set)

Use/extend these in your sheet.

1. **Signup – Happy Path** (P1, Positive)
   Steps: open signup → fill valid data → submit → complete OTP/verification (mock if needed) → first login → verify onboarding screen.
   Expected: account created; onboarding visible.
2. **Signup – Required Field Validation** (P1, Negative)
   Steps: leave email/password blank → submit.
   Expected: field-level errors; submit blocked.
3. **Signup – Email Format Validation** (P1, Negative)
   Steps: email `foo@bar` (no TLD) → submit.
   Expected: inline "invalid email" error.

4. **Signup – Weak Password** (P1, Negative)
   Steps: `abc123` → submit.
   Expected: password policy message; submit blocked.
5. **Signup – Duplicate Email** (P2, Negative)
   Pre-req: existing account.
   Steps: sign up with same email.
   Expected: "email already registered".
6. **Signup – XSS in Name Field** (P2, Negative, Security-lite)
   Steps: name `<script>alert(1)</script>` → submit.
   Expected: sanitized; no script executes; safe error or proper save.
7. **Signup – Oversized Input** (P2, Negative)
   Steps: very long name (256+ chars).
   Expected: client-side limit or server validation; graceful message.
8. **Login – Happy Path** (P1, Positive)
   Steps: enter valid creds → submit.
   Expected: redirected to authenticated landing; profile visible.
9. **Login – Wrong Password** (P1, Negative)
   Steps: valid email + wrong password.
   Expected: "invalid credentials" error; no session.
10. **Login – Unregistered Email** (P1, Negative)
    Steps: random email + any password.
    Expected: user-friendly error; no session.
11. **Login – Remember Me** (P2, Positive)
    Steps: check "remember me" → login → reload/close-reopen.
    Expected: still signed in (within session TTL); on logout, fully cleared.
12. **Login – Rate Limiting/Lockout (Exploratory)** (P3, Negative)
    Steps: N failed attempts quickly.
    Expected: throttle/lockout message or captcha; document observed behavior.

Add additional onboarding steps if a wizard appears (company name, currency, timezone, etc.): required fields, validation, navigation between steps, back/forward behavior, save & resume (if present).

Automation Requirements

- **Structure:** Page Object Model; clear locators (data-testid preferred if available).
- **Stability:** Explicit waits for visible/enabled; avoid fixed sleeps.
- **Config:** `.env` (e.g., baseURL, headless, email seed, OTP mode).

- **Idempotence:** Tests should create and clean up test data where feasible (or namespace test users).
- **Reporting:** HTML or Allure/Playwright/Cypress default reports + screenshots on failure.
- **CI-readiness (bonus):** Provide a simple `npm run test:ci` (or Maven/Gradle for Java) and note parallelism.

Basic Security/Privacy Checks (lightweight)

- Password input is masked; clipboard copy prevented if policy requires.
- Sensitive errors not leaked in UI (no stack traces).
- Cookies: `HttpOnly`/`Secure` set on auth cookies (document observation via DevTools; automation check optional).

Submit: repo link + report artifacts + test case sheet.