IMMANUEL YABES

**Challenge 1: Automation Testing (Technical Skills)**

**Objective:** Evaluate your ability to devise a comprehensive test strategy and design high-level test scenarios for a common application feature.

**Tasks:**

1. **Outline a Test Strategy:** Describe your overall approach to testing this feature. Consider:
   * Types of testing needed
   * Key areas of focus.
   * Environments (where would you test?).
   * Any assumptions you're making.
2. **Identify High-Level Test Scenarios:** List the main scenarios you would test. Don't write detailed steps, but cover the different paths and conditions. Examples:
   * Register.
   * Login.
   * Search for a mentor.
   * …
   * Make a schedule

Answer :

**1. Test Strategy**

**Types of Testing Needed :**

* **Functional Testing** – Validate all core functionalities like registration, scheduling, and messaging.
* **UI/UX Testing** – Ensure responsive, user-friendly interfaces across devices.
* **API Testing** – Test mentor-related endpoints for data accuracy and performance.
* **Security Testing** – Protect user data and enforce role-based access controls.
* **Performance Testing** – Assess system behavior during peak loads (e.g., multiple simultaneous bookings).
* **Automation Testing** – Automate repeatable tests like login, search, and booking flows.
* **Regression Testing** – Ensure the mentoring feature doesn’t break existing features.

**Key Areas of Focus :**

* Mentor and mentee onboarding.
* Real-time search and filtering.
* Availability management and scheduling.
* Booking/rescheduling flows.
* Notifications and reminders.
* Authorization and access restrictions.

**Test Environments :**

* **Staging/Test Environment:** Primary location for test execution using test data.
* **Production Environment (Monitoring):** Observe user behavior post-deployment via logs and analytics.

**Assumptions :**

* Users can be mentors, mentees, or both.
* Time slots are managed in real-time and avoid overlaps.
* Backend APIs are documented and accessible.
* Messaging is available post-booking only.

**2. High-Level Test Scenarios**

**Registration and Profile Management**

* Register as a mentor with valid data.
* Register with missing/invalid data.
* Register using an existing email.
* Edit mentor/mentee profile (bio, photo, expertise).
* View another user’s profile (restricted fields).

**Search and Filter Mentors**

* Search mentors by keyword.
* Filter by availability, industry, rating, experience.
* Sort search results.
* No results found edge case.

**Scheduling & Availability**

* View mentor’s calendar.
* Book a session at an available time.
* Attempt to double-book or book unavailable times.
* Reschedule or cancel a session.
* Conflict handling for overlapping sessions.

**Notifications**

* Booking confirmation sent to both parties.
* Session reminders sent prior to the meeting.
* Notifications on rescheduling or cancellation.
* Edge case: Email delivery failure handling.

**Communication (if applicable)**

* Initiate a message after a session is booked.
* Attempt to message without a valid session (blocked).
* Blocked users or disabled messaging test.

**Edge Cases & Validation**

* Network failure during session booking.
* Scheduling a session in the past.
* Input field validation (e.g., max characters in bio).
* Accessing mentor features as a mentee (restricted access).
* Unauthorized API access test

**Challenge 3: Exploration & Bug**

**Objective:** Evaluate your analytical thinking, problem-solving approach, and understanding of potential failure points in [dealls](https://dev.dealls.com/).

**Tasks:**

1. **Investigation Steps:** Describe the step-by-step process you would follow to investigate this intermittent bug.
2. **Information Gathering:** What specific information would you try to gather to help diagnose the problem?
3. **Potential Root Causes:** List potential technical reasons across different system layers.
4. **Reproduction Strategy:** How would you attempt to reliably reproduce this issue in a test environment to confirm the cause and verify a fix?

Answers :

**Hypothetical Bug:**

Intermittent failure when a mentee tries to book a session — the confirmation screen doesn't load, and no session is recorded.

**1. Investigation Steps**

1. **Reproduce the Bug**
   * Attempt to recreate the issue in the staging/test environment using the exact steps provided (e.g., user role, mentor selected, time slot).
   * Run the test across different browsers and devices to observe variations.
2. **Check Logs**
   * Review frontend console logs for any JavaScript or network errors.
   * Review backend server logs (API errors, database write failures, timeouts).
3. **Network Analysis**
   * Use browser DevTools to inspect API request/response details (status codes, payloads, time taken).
   * Monitor if the booking request is sent and how the server responds.
4. **Examine Edge Cases**
   * Try booking close to the session start time.
   * Test multiple bookings at the same time to simulate concurrency.
5. **Collaborate with Developers**
   * Sync with developers for deeper inspection into service and database layers.
   * Validate if there’s retry logic, transaction rollback, or race conditions.

**2. Information Gathering**

* **Exact Steps to Reproduce (from bug reporter)**
* **User Data & Session ID** (if available)
* **Timestamp of occurrence**
* **Browser, OS, and device info**
* **API request/response logs**
* **Error stack traces (frontend/backend)**
* **Booking database state before and after the attempt**
* **Mentor’s availability state at the time of booking**

**3. Potential Root Causes**

| **Layer** | **Possible Issues** |
| --- | --- |
| **Frontend** | UI not waiting for async booking response; button double-clicks; local caching issues |
| **API/Backend** | Race condition in booking logic; validation failure; service timeout; bad error handling |
| **Database** | Deadlock on booking row; write failure; transaction rollback |
| **Network** | Intermittent latency or timeout between frontend and API |
| **Infrastructure** | Load balancer misrouting or inconsistent state across replicated services |

**4. Reproduction Strategy**

* Use a test user to **replay exact booking steps** in a controlled environment.
* Simulate **various network speeds and latencies** using browser DevTools or tools like tc (Linux).
* Set up **logging hooks** or use a proxy like **Charles or Fiddler** to monitor live requests.
* Use **test automation scripts** (e.g., Cypress/Playwright) to execute the scenario multiple times and detect failure patterns.
* Add **temporary instrumentation logs** on the backend to trace session booking flow.