Challenge 1: Automation Testing (Test Strategy & High-Level Scenarios)

Test Strategy for Mentoring Feature

Types of Testing Needed:

- **Unit Testing**: For individual functions like schedule validation, availability checks.
- **Integration Testing**: Ensuring APIs work across modules like mentor profiles, booking, notifications.
- **UI/End-to-End Testing**: Verifying user flows (search mentor, book session, etc.).
- **Regression Testing**: Ensure new feature doesn't break login, profile, etc.
- **Performance Testing**: Confirm the feature performs under load (many bookings/searches).
- **Security Testing**: Check permission roles (e.g., users can't edit other people's sessions).
- **Usability Testing**: Ensure intuitive UI/UX (filters, calendar picker).

Key Areas of Focus:

- Mentor/mentee matching logic.
- Scheduling rules (timezones, double-booking prevention).
- Booking/rescheduling/cancel workflows.
- Notifications (email, in-app).
- Edge cases (no mentors available, overlapping time slots).

Test Environments:

- **Staging**: Full test with production-like data.
- **Local/Dev**: For fast iteration and debugging.
- **Mobile + Web**: Responsive UI checks.

Assumptions:

- Feature is accessible after login.
- Both mentors and mentees are verified users.
- Session bookings rely on internal API calls.

High-Level Test Scenarios:

1. **User Registration and Login**:
    - New user registers.
    - Existing user logs in.
2. **Mentor Discovery/Search**:
    - Search by skill/industry/availability.
    - Pagination and filters.
    - No results scenario.
3. **View Mentor Profile**:

    - Open mentor detail view.
    - See availability.
4. **Schedule Session**:
    - Choose date/time, confirm booking.
    - Prevent double-booking.
    - Timezone handling.
5. **Edit/Reschedule Booking**:
    - Reschedule session within limits.
6. **Cancel Booking**:
    - Cancel before session, verify refund/cancellation flow.
7. **Notification System**:
    - Email/in-app notification for booking, changes, reminders.
8. **Edge Cases & Error Handling**:
    - API failure during booking.
    - Schedule in the past.
    - Overlapping session attempts.

Challenge 3: Exploration & Bug

Investigation Steps:

1. **Reproduce in Staging**:
    - Try different browsers, roles (mentor/mentee), and times.
    - Collect HAR logs, dev tools console.
2. **Check Logs**:
    - Backend logs (API failures, exceptions).
    - Frontend logs (network, JS errors).
3. **Compare Requests**:
    - Working vs failing session bookings.

Information Gathering:

- User session tokens, roles, and booking times.
- Logs: API response payloads.
- Feature flags or A/B tests affecting flows.

Potential Root Causes:

- **Frontend**:
    - JavaScript error blocking UI.
    - Bad input validation or timezone parsing.
- **Backend**:
    - Race condition when checking availability.
    - Caching delays on availability.
- **Database**:
    - Stale records due to eventual consistency.

- **Infrastructure**:
    - Load balancer not syncing requests.

Reproduction Strategy:

- Simulate high-load booking environment.
- Automate booking at edge times (e.g., midnight).
- Introduce artificial delay to mimic real user behavior.