## ****1. test\_validate\_error\_handling (GET /booking with invalid params)****

**Failures Observed:**

* Status 500 returned for invalid date formats ('checkin': '13-03-2014', 'checkout': '21-05-2014').
* Status 500 returned for non-date strings ('checkin': 'sally', 'checkout': 'kelly').
* Status 200 returned for unexpected types ('firstname': 123456, 'firstname': '!@#$%^&\*()').
* Status 200 returned for unknown query parameters ('unknown': 'Unknown parameter').

**Justification:**

* The API is **not handling invalid input gracefully**. Instead of returning a 400 (Bad Request), the server returns 500 (Internal Server Error) or 200 for invalid data.
* This indicates a lack of **input validation** and **error handling** on the server side.

**Recommendations:**

1. Implement **input validation** for all query parameters.
2. Return appropriate HTTP status codes:
   * 400 Bad Request for invalid formats or types.
   * 422 Unprocessable Entity for unsupported or invalid values.
3. Update tests to match **expected API contract**, once server-side validation is implemented.
4. Consider adding **boundary and negative test cases** for dates, strings, and unknown fields.

## ****2. test\_by\_applying\_single/multiple\_filters (GET /booking with filters)****

**Failures Observed:**

* Booking ID not returned for valid filters like {'checkin': '2025-09-20'} or multiple filters ({'lastname': 'TestAutomation', 'checkin': '2025-09-10', 'checkout': '2025-09-15'}).

**Justification:**

* Likely caused by **filter mismatch**: the test creates a booking with checkin='2025-09-20', but multiple filter test queries 'checkin': '2025-09-10'.
* The API’s filter logic may **not support multiple criteria combined** or has a delay in indexing newly created bookings.

**Recommendations:**

1. Use **dynamic and consistent filter values** matching the created booking.
2. Consider **simplifying filters** for E2E tests (e.g., single-field filter like firstname).
3. Implement **wait/polling logic** if the system is eventually consistent.

## ****3. test\_validate\_error\_handling (PATCH /booking with invalid data)****

**Failures Observed:**

* Status 200 returned instead of expected 400 when sending invalid payload ('totalprice': 'not-a-number').
* Field totalprice is set to null instead of failing.

**Justification:**

* The API does **not enforce data type validation** for updates.
* Invalid payloads are silently accepted, which can lead to **data corruption**.

**Recommendations:**

1. Enforce **server-side type validation** for all fields.
2. Return **400 Bad Request** when input data types are invalid.
3. Update test assertions after the API is fixed.

## ****4. test\_delete\_with\_invalid\_auth / test\_concurrent\_delete****

**Failures Observed:**

* Status 201 returned instead of expected 401/403/405 when deleting with invalid credentials.
* Concurrent DELETE returned [201, 405] instead of [201, 404].

**Justification:**

* API **does not enforce authentication** for DELETE operation.
* Concurrent deletion behavior differs from expected contract.

**Recommendations:**

1. Enforce **authentication and authorization** checks.
2. Define clear **expected behavior for concurrent deletions** (e.g., first delete 201, second delete 404).
3. Update tests to match **API contract once fixed**.

## ✅ ****Overall Observations****

* Most failures stem from **insufficient server-side validation**, **inconsistent filtering**, and **authentication gaps**.
* Your tests are correctly detecting these **critical gaps**, highlighting the value of automated E2E and negative testing.

## ****General Recommendations****

1. **Server-Side Improvements:**
   * Input validation, error handling, and consistent HTTP status codes.
2. **Test Enhancements:**
   * Ensure filters match created booking data.
   * Use polling/waiting where eventual consistency may affect retrieval.
   * Extend negative tests for malformed inputs, invalid types, and unauthorized access.
3. **CI Pipeline:**
   * Failures like 500 or unexpected 200 should **trigger alerts** to prevent deployment of unstable API.