CASE STUDY

Restful-Booker — API Testing with Postman

API testing · Postman scripted assertions · REST convention validation

Overview

Functional API testing of the public Restful-Booker service, covering the full booking lifecycle: health check, authentication, create, read, update, patch, and delete. Built in Postman with scripted assertions (pm.test) rather than manual response inspection, so results are repeatable and not dependent on me eyeballing each response.

My Role

Sole tester. I designed the 13-case suite to walk the booking resource through its full lifecycle in sequence — create, verify, update twice (PUT and PATCH), delete, then verify the delete and re-query the collection — rather than testing each endpoint in isolation, since that sequencing is closer to how the API is actually used.

Results at a Glance

Metric	Result
Test cases executed	13
Pass rate	84.6% (11 passed)
Defects logged	2 — both Open
Severity split	1 Medium, 1 Low

The Findings

Both defects came from checking the response against a stated expectation, not just checking for a 2xx status:

- BUG-01 (Low) — DELETE /booking/{id} returns 201 Created instead of the conventional 204 No Content. The delete itself works; only the status code deviates from REST convention. Low functional risk, but it will silently break any automated suite that asserts strictly on 204.
- BUG-02 (Medium) — GET /booking with firstname/lastname filters can return a booking ID that doesn't actually match the filter. This is a real server-side logic issue, not a cosmetic one: any consumer trusting the filtered result set gets wrong data.

The point of ranking BUG-02 above BUG-01, even though BUG-01 was caught first, is the underlying judgment call: a non-standard status code is an inconvenience for downstream automation, but a wrong filtered result is a data-correctness problem that can propagate into whatever system consumes it.

Test Design Approach

- Full resource lifecycle coverage in sequence (create → read → update → patch → delete → verify), not just isolated endpoint checks.
- Positive and negative paths on authentication before touching protected endpoints.
- Status-code assertions written against documented REST conventions, not just “did it return something in the 200s.”
- Query-parameter filtering tested against actual returned record data, not just response shape — which is what surfaced BUG-02.

What I'd Do Differently Next Time

- Add a broader set of boundary and negative test cases around the query-filtering functionality specifically, since BUG-02 suggests the filter logic may not be reliable across other input combinations either.
- Wire the collection into Newman + a CI pipeline (GitHub Actions) so this becomes a regression suite, not a one-time report.
- Add schema validation (not just field-level assertions) to catch structural response changes automatically.

Skills Demonstrated

- API testing in Postman with scripted pm.test assertions
- Full resource-lifecycle test design (CRUD sequencing, not isolated endpoint checks)
- REST convention and status-code validation
- Defect severity reasoning based on downstream impact, not just discovery order

Full report: Restful-Booker Bug Summary Report (Word) — available in the project repository.