**✅ List of Problems You Faced and Solved**

**1. Incorrect Payload Structure**

* **Issue:** You tried to parse a payload assuming it was a dictionary, but it was a list.
* **Error Message:** 'list' object has no attribute 'get'
* **Fix:** Added logic to handle both single and batch webhook events:
* events = payload if isinstance(payload, list) else [payload]

**2. Improper Response Format**

* **Issue:** Your Flask webhook was always returning a list of results, even when the test expected a single dict.
* **Fix:** You added a conditional return:
* if len(results) == 1:
* return jsonify(results[0]), 200
* else:
* return jsonify(results), 200

**3. Indentation Bug in except Block**

* **Issue:** Python syntax error due to improperly indented except block.
* **Fix:** Properly aligned the block:
* except Exception as e:
* logger.error(...)
* return jsonify(...), 500

**4. Test Failures Due to Duplicate Event IDs**

* **Issue:** Tests like test\_valid\_webhook() failed because they reused the same evt\_test\_001 ID.
* **Fix:** You used dynamic or unique IDs via helper functions like get\_unique\_payload().

**5. Mismatch in Expected vs Actual Response Format in Tests**

* **Issue:** Tests expected:
* response.json()["status"]

But received:

[{'event\_id': ..., 'status': ...}]

* **Fix:** You adjusted the response structure based on number of events processed.

**6. Hardcoding Event IDs Across Multiple Tests**

* **Issue:** Tests with fixed event IDs (like evt\_test\_001) would be treated as duplicates.
* **Fix:** Introduced a helper to generate **unique event IDs** dynamically.

**7. Tests Failing Without Clear Reset of DB State**

* **Issue:** Old test data remained in DB, causing unexpected results (e.g., duplicates).
* **Fix:** You added (or planned) a clear\_payment\_events() helper for test isolation (either via API or DB reset).

**8. Signature Validation Issues**

* **Issue:** Invalid or missing X-Razorpay-Signature headers caused 403 errors.
* **Fix:** Ensured correct HMAC-SHA256 signature was generated using your shared secret.

**9. Handling Invalid JSON Format**

* **Issue:** If malformed JSON was sent, it raised an unhandled exception.
* **Fix:** You added a proper error handler:
* try:
* payload = request.get\_json(force=True)
* except Exception:
* return jsonify({'error': 'Invalid JSON format'}), 400

**10. No test\_helpers.py Initially**

* **Issue:** You reused event payloads directly in tests which caused duplication issues.
* **Fix:** You introduced test\_helpers.py to generate unique event IDs and optionally reset DB state.

**🎯 Summary of What You Learned**

| **Area** | **What You Learned** |
| --- | --- |
| Payload Handling | How to support both single and batch webhook payloads |
| Webhook Security | How to validate HMAC-SHA256 signatures |
| Error Handling | Graceful handling of malformed requests |
| Response Formatting | Tailoring API response structure based on test needs |
| Test Automation | Importance of unique event IDs and test isolation |
| Logging & Debugging | Reading logs for payload parsing and server errors |