TEST SUMMARY REPORT · TEST CYCLE · API · EXECUTED

API testing of the Toolshop

4 high-severity authorisation defects in a teaching e-commerce API, found in a full, reproducible run.

I wrote and ran a Postman and Newman suite against the Toolshop API, from account setup to a database reset. The run is reproducible from a clean database. It found defects in authorisation and input validation; in the worst of them, customers can change other customers' invoices and messages, and anyone can change a product's price.

Application Practice Software Testing (Toolshop), sprint5	**Commit** c752a70	**API version** 5.0
Environment Local Docker · localhost:8091	**Tools** Postman, Newman	**Period** 14–17 October 2026
Author Pelin Dragos		

355
REQUESTS
in one Newman run, 2 min 40 s

582
ASSERTIONS
35 failed, each explained

19
DEFECTS
4 of them high severity

118
TEST CASES
written before the run, in 17 folders

1. Results of the final run

355 requests, 582 assertions, 35 failed, in 2 min 40 s. No request failed to complete.

Folder	Requests	Failed assertions	Defects shown
00 Setup	13	0	—
01 Environment	1	0	—
02 Catalog	34	0	—
03 Registration	26	1	DEF-003
04 Authentication	27	0	—
05 Profile	6	0	—
06 Cart and payment	55	4	DEF-009
07 Orders and invoices	32	6	DEF-010, DEF-011, DEF-013
08 Favorites	13	0	—
09 Messages	21	3	DEF-012
10 Data isolation	27	6	DEF-015, DEF-016

Folder	Requests	Failed assertions	Defects shown
11 Administration	29	8	DEF-017, DEF-018
12 Reports	24	0	—
13 Dedicated users	23	3	DEF-005, DEF-006
14 Contract	21	1	DEF-007
15 Time dependent	1	2	—
16 Database reset	2	1	DEF-019

2. Defects

19 defects: 4 high, 13 medium, 2 low. Severity is the author's decision, based on impact. None has been retested, because no fix has been applied.

ID	Severity	Defect	Tests
DEF-001	▲ **Medium**	Product search misses products when a 4+ character term appears inside a word	T-C05, T-C07
DEF-002	○ **Low**	Postcode lookup returns different localities for the same postcode, depending on spacing (to be confirmed)	T-E01, T-E03
DEF-003	▲ **Medium**	Registration accepts an email address without a valid format	T-B04
DEF-004	▲ **Medium**	Change-password validation returns a redirect (302) instead of 422 for clients that do not send Accept: application/json	T-A14
DEF-005	▲ **Medium**	Forgot-password reveals whether an email has an account	T-A11
DEF-006	○ **Low**	/totp/login/totp route points to a method that does not exist (500)	T-A15
DEF-007	▲ **Medium**	Product list with an unknown sort column returns 500	T-K06
DEF-008	▲ **Medium**	Server errors expose the full stack trace (APP_DEBUG=true) and a page beyond the range takes 30+ seconds	T-K06, T-K09
DEF-009	▲ **Medium**	Payment check accepts an unknown payment method	T-E10
DEF-010	▲ **Medium**	An empty cart can be turned into an invoice (201)	T-E09
DEF-011	◆ **High**	Any logged-in user can change the status of another user's invoice	T-F04
DEF-012	◆ **High**	Any logged-in user can change the status of another user's contact message	T-H07
DEF-013	▲ **Medium**	Invoice PDF generation fails for new invoices; download returns 404	T-F02
DEF-014	◆ **High**	Any logged-in user can download another customer's invoice PDF	T-J03
DEF-015	▲ **Medium**	Favorites and message replies by id skip the owner check	T-J06, T-J07
DEF-016	▲ **Medium**	Carts can be read, changed and deleted by anyone who knows the cart id	T-J01
DEF-017	◆ **High**	Anyone (even without a token) can create products and change their price	T-I03
DEF-018	▲ **Medium**	Brands, categories and product specifications can be created without admin role	T-I06, T-C16
DEF-019	▲ **Medium**	The /refresh rate limit is cleared by the same request that triggers it	T-K10

High-severity defects in detail

DEF-011 · Any logged-in user can change the status of another user's invoice

The endpoint that changes an invoice's status checks neither the role nor the owner. A customer can mark another customer's order as completed.

1. Customer A places an order and receives the invoice ID.
2. Customer B, with a different account, sends a status change for that invoice.
3. The invoice is read back as an admin.

Expected	The change is refused with 403 or 404, and the status stays the same.
Observed	200, and the status becomes COMPLETED.
Impact	Anyone with an account can falsify order and billing data for other customers.
Probable cause	The status update filters neither by owner nor by role. The other invoice actions do filter by owner; this route is the exception.
Suggested fix	Restrict the route to the admin role. If customers may cancel orders, allow only that, and only on their own.

DEF-012 · Any logged-in user can change the status of another user's contact message

The same flaw as DEF-011, on contact messages: a customer can change the status of another customer's message.

1. Customer A sends a contact message.
2. Customer B sends a status change for that message.
3. The message is read back as an admin.

Expected	The change is refused with 403 or 404, and the status stays the same.
Observed	200, and the status changes.
Impact	Support can lose track of real requests. The content of the messages is not exposed.
Probable cause	The status route checks the allowed values only, with no role or owner check.
Suggested fix	Restrict the route to the admin role, as the reports already are.

DEF-014 · Any logged-in user can download another customer's invoice PDF
The invoice PDF download looks the file up by invoice number only. Invoice numbers are sequential, so another customer's PDF is easy to fetch.

1. An invoice that belongs to customer A is chosen.
2. Customer B, with no link to A, requests that invoice's PDF with their own token.

> Confirmed in the Postman runner and by direct reproduction with an invoice whose PDF existed. In the final run no new PDF was generated (DEF-013), so the download returned 404 and the test passed.

Expected	403 or 404, with no file.
Observed	200 with application/pdf, 11,376 bytes, containing customer A's billing data.
Impact	The name, address, products and total of any customer can be downloaded by anyone with an account.
Probable cause	The download searches by invoice number and does not check the owner, unlike the single-invoice read.
Suggested fix	Look the invoice up by number and owner (except for the admin), and return 404 otherwise.

DEF-017 · Anyone (even without a token) can create products and change their price
The create and update routes for products have no authentication and no role check. A visitor can create products, and anyone can change a price.

1. A price change is sent to an existing product, with no token.
2. The same change is sent with a customer's token.
3. The product is read back.

Expected	401 without a token, 403 for a customer. The price does not change.
Observed	Both changes return 200. The price changes from 14.15 to 0.01.
Impact	Anyone can change catalogue prices or add false products.
Probable cause	The role check applies to the delete action only. Create, update and patch have no authentication.
Suggested fix	Require authentication and the admin role on every write action.

Evidence notes for the other defects

- **DEF-001:** Found by comparing searches. The suite records the results but does not assert on this difference; the request below shows the “ham” search.
- **DEF-002:** Closed as a limitation of the test postcode driver, still marked to be confirmed. The setup uses one fixed format, so the final run does not exercise it.
- **DEF-004:** Seen in the earlier Postman runs. In the final run the collection sends Accept: application/json, so this test returns 422 and passes.
- **DEF-008:** Seen in the earlier runs. In the final run the same request returned 200, which points to the cache write in the test environment. The debug-mode exposure stands.
- **DEF-016:** Treated as a defect for changing and deleting. Reading a cart by its hard-to-guess ID is documented as the anonymous cart model.

3. Scope

In scope

- Authentication, sessions, password change and reset
- Registration and profile
- Catalogue: search, filters, product details, invalid input
- Cart, checkout and payment
- Orders and invoices, including the invoice PDF
- Favourites and contact messages
- Data isolation between two customers and a visitor
- Administration and roles
- Reports, validation and error handling
- Database reset and rate limiting

Out of scope

- The user interface, end to end
- SQL and database checks
- Accessibility
- Performance and load
- Social login
- The server-sent events endpoint
- The deliberately vulnerable variant of the application

118 test cases were planned across 12 areas (A–L) and built as 17 folders of requests.

4. Method

1. **Plan.** Areas, risks and the order of the run, with the evidence each test needs.
2. **Write the cases.** Each case states why it matters, the expected result and the pass and fail criteria, before it is run.
3. **Build the collection.** One Postman request per check, with assertions and the data set up in the first folder.
4. **Run folder by folder.** Tokens expire in five minutes, so each folder starts by refreshing them.
5. **Keep the evidence.** Responses and test results saved per test, with tokens and secrets masked.
6. **Report the defects.** Steps, expected and observed results, impact, probable cause and a fix, for each one.
7. **Run it all again.** One full Newman run from a reset database; every failure explained by a defect or a stated limit.

5. Limits

- The target is a local copy of a teaching application. The defects are real for this version; the conclusions do not describe a live service.
- Three defects are not visible in the final run (DEF-004, DEF-008, DEF-014). Each one says where its evidence comes from.
- One test, the expired token, cannot pass in a full run, because the token is still valid. It needs its own run after five minutes, and is reported as a limit of the test design.
- One result is an observation, not a check: another user’s message returns 200 with an empty body. No content is exposed.

- Folder-level evidence shows that a test passed in a run, not that every part of its response was checked.
- Newman counts 582 assertions; the per-request export adds up to 600. The 18-assertion difference has not been fully traced, so the summary figure is used.
- No fix has been applied, so nothing has been retested.

6. Contribution

My work

- Choosing the scope, the areas and the severity of each defect.
- Reviewing every test, correcting the wrong ones, and running the folders in Postman.
- Saving and masking the evidence, and checking results against the saved responses.
- Deciding what counts as a defect and writing the defect reports.

Assisted by AI

- Drafting the test cases and the Postman scripts, which I reviewed and corrected.
- Running the final collection through Newman and building the report from the run files.
- Writing the English text of this page and of the downloads, from the project files.

The Postman collection, environment, full Newman report, request catalogue and defect register are published with this report. Tokens, secrets and generated test passwords are masked in every published file.