```
### R1: Add Book To Catalog

R1-TC-01 Valid add (minimal values)
- Preconditions: Catalog is empty
- Steps:
  1) Open Add Book form
  2) Enter Title="The Pragmatic Programmer", Author="Andrew Hunt",
ISBN="9780135957059", Total copies=1
  3) Submit
- Expected:
  - Success message displayed
  - Redirects to catalog view
  - Catalog shows new row with correct values and "Available 1 / Total 1"

R1-TC-02 Trim whitespace in fields
- Steps: Title="  Clean Code  ", Author="  Robert C. Martin ",
ISBN="9780132350884", Total copies=2
- Expected: Values are stored/displayed trimmed; success + redirect

R1-TC-03 Title at max length (200)
- Steps: Title length=200, valid author, ISBN=13 digits, copies=3
- Expected: Accepts and stores; catalog renders full title without truncation
artifacts

R1-TC-04 Title exceeding max (201)
- Steps: Title length=201, other fields valid
- Expected: Validation error indicating Title too long; no record created

R1-TC-05 Author at max length (100)
- Steps: Author length=100; others valid
- Expected: Accepts; displays correctly

R1-TC-06 Author exceeding max (101)
- Steps: Author length=101; others valid
- Expected: Validation error; no record created

R1-TC-07 ISBN not 13 digits (12 digits)
- Steps: ISBN="123456789012"; others valid
- Expected: Validation error; no record created

R1-TC-08 ISBN contains non-digits
- Steps: ISBN="978013235088X"; others valid
- Expected: Validation error; no record created

R1-TC-09 Total copies zero
```

~~~
- Steps: Total copies=0; others valid
- Expected: Validation error for positive integer required; no record created

R1-TC-10 Total copies negative
- Steps: Total copies=-5; others valid
- Expected: Validation error; no record created

R1-TC-11 Total copies non-integer
- Steps: Total copies="two"; others valid
- Expected: Validation error; no record created

R1-TC-12 Duplicate ISBN add attempt
- Preconditions: A book with ISBN="9780132350884" exists
- Steps: Add a different title/author but same ISBN
- Expected: System rejects duplicate ISBN to maintain data integrity (clear error message)

R1-TC-13 XSS/HTML in Title
- Steps: Title="<script>alert(1)</script>", others valid
- Expected: Input is escaped/sanitized on render; no script executes; add is either rejected or stored safely escaped

Python test functions (unit tests against `library_service.add_book_to_catalog`):

```python
import pytest
import library_service
from library_service import add_book_to_catalog


def test_r1_add_valid_minimal(monkeypatch):
    # Arrange: no duplicate ISBN, insert succeeds
    monkeypatch.setattr(library_service, "get_book_by_isbn", lambda isbn: None)
    monkeypatch.setattr(library_service, "insert_book", lambda *args, **kwargs: True)

    ok, msg = add_book_to_catalog(
        "The Pragmatic Programmer", "Andrew Hunt", "9780135957059", 1
    )
    assert ok is True
    assert "successfully added" in msg.lower()


def test_r1_add_trims_and_allows_boundary_lengths(monkeypatch):
    title = "X" * 200
~~~

~~~
    author = "Y" * 100
    monkeypatch.setattr(library_service, "get_book_by_isbn", lambda isbn: None)
    monkeypatch.setattr(library_service, "insert_book", lambda *args, **kwargs:
True)

    ok, _ = add_book_to_catalog(f"  {title}  ", f"  {author}  ", "1234567890123",
2)
    assert ok is True


def test_r1_add_rejects_title_over_200(monkeypatch):
    monkeypatch.setattr(library_service, "get_book_by_isbn", lambda isbn: None)
    monkeypatch.setattr(library_service, "insert_book", lambda *args, **kwargs:
True)

    title = "Z" * 201
    ok, msg = add_book_to_catalog(title, "A", "1234567890123", 1)
    assert ok is False
    assert "less than 200" in msg.lower()


def test_r1_add_rejects_invalid_isbn_length(monkeypatch):
    ok, msg = add_book_to_catalog("T", "A", "123456789012", 1)
    assert ok is False
    assert "exactly 13 digits" in msg.lower()


def test_r1_add_rejects_nonpositive_copies():
    ok, msg = add_book_to_catalog("T", "A", "1234567890123", 0)
    assert ok is False
    assert "positive integer" in msg.lower()


def test_r1_add_duplicate_isbn(monkeypatch):
    # Arrange: simulate existing book for this ISBN
    monkeypatch.setattr(library_service, "get_book_by_isbn", lambda isbn: {"id":
1})

    ok, msg = add_book_to_catalog("T", "A", "1234567890123", 1)
    assert ok is False
    assert "already exists" in msg.lower()
```


---

### R2: Book Catalog Display
~~~

~~~
R2-TC-01 Empty catalog UI
- Preconditions: No books
- Steps: Visit catalog page
- Expected: Shows empty-state message (no rows), no errors

R2-TC-02 Table columns and formatting
- Preconditions: At least one book exists
- Steps: Visit catalog
- Expected: Columns include ID, Title, Author, ISBN, Available/Total, Actions;
Available format "A / T"

R2-TC-03 Borrow button visibility
- Preconditions: Book A with Available>0, Book B with Available=0
- Steps: Visit catalog
- Expected: Borrow button shown only for Book A; absent/disabled for Book B

R2-TC-04 Long strings render correctly
- Preconditions: Title=200 chars, Author=100 chars book exists
- Steps: Visit catalog
- Expected: Layout remains intact on mobile and desktop; no overflow breaking;
content escaped

R2-TC-05 Availability updates after borrow/return
- Preconditions: Book with Total=2, Available=2
- Steps: Borrow once -> Available=1; Return -> Available=2
- Expected: Catalog reflects updated counts after each action without stale
values

Python test functions (Flask route tests against `GET /catalog`):

```python
import pytest
from typing import List, Dict
from flask import template_rendered
from contextlib import contextmanager


@pytest.fixture(scope="session")
def app():
    from app import create_app
    return create_app()


@pytest.fixture()
~~~

```python
def client(app):
    return app.test_client()


@contextmanager
def captured_templates(app):
    recorded = []
    def record(sender, template, context, **extra):
        recorded.append((template, context))
    template_rendered.connect(record, app)
    try:
        yield recorded
    finally:
        template_rendered.disconnect(record, app)


def _book(id, title, author, isbn, total, available) -> Dict:
    return {
        "id": id,
        "title": title,
        "author": author,
        "isbn": isbn,
        "total_copies": total,
        "available_copies": available,
    }


def test_r2_catalog_renders_list_and_fields(app, client, monkeypatch):
    from routes import catalog_routes
    sample = [
        _book(1, "Clean Code", "Robert C. Martin", "9780132350884", 5, 5),
        _book(2, "1984", "George Orwell", "9780451524935", 2, 0),
    ]
    monkeypatch.setattr(catalog_routes, "get_all_books", lambda: sample)

    with captured_templates(app) as tmps:
        resp = client.get("/catalog")
        assert resp.status_code == 200

    assert len(tmps) == 1
    template, ctx = tmps[0]
    assert template.name == "catalog.html"
    assert ctx["books"] == sample
    assert ctx["books"][0]["available_copies"] == 5
    assert ctx["books"][1]["available_copies"] == 0
```

```
def test_r2_catalog_empty_state(app, client, monkeypatch):
    from routes import catalog_routes
    monkeypatch.setattr(catalog_routes, "get_all_books", lambda: [])

    with captured_templates(app) as tmps:
        resp = client.get("/catalog")
        assert resp.status_code == 200
    _, ctx = tmps[0]
    assert ctx["books"] == []
```

### R3: Book Borrowing Interface

R3-TC-01 Valid borrow
- Preconditions: Patron "012345" exists; Book with Available>=1
- Steps: Enter Patron ID=012345, Book ID=<book id>; Submit
- Expected: Success message; Available decremented by 1; borrow record created with borrow date

R3-TC-02 Patron ID invalid (length != 6)
- Steps: Patron ID="12345"; valid Book ID
- Expected: Validation error for patron id format; no changes to availability

R3-TC-03 Patron ID invalid (non-digits)
- Steps: Patron ID="12AB56"; valid Book ID
- Expected: Validation error; no changes

R3-TC-04 Book unavailable
- Preconditions: Book Available=0
- Steps: Attempt borrow with valid Patron ID
- Expected: Error indicating book not available; no record created

R3-TC-05 Patron limit exactly 5 allowed, 6th denied
- Preconditions: Patron has 4 active borrows; Book has Available>=2
- Steps: Borrow 5th book -> success; attempt 6th -> error indicating limit reached
- Expected: Max 5 enforced

R3-TC-06 Non-existent Book ID
- Steps: Patron valid, Book ID=999999 (absent)
- Expected: Clear error; no changes

~~~
R3-TC-07 Non-existent Patron ID
- Steps: Patron ID not present; valid Book ID
- Expected: Clear error; no changes

R3-TC-08 Concurrent borrow contention
- Preconditions: Book Available=1; Two users attempt borrow nearly simultaneously
- Steps: Submit two borrow requests in rapid succession
- Expected: Exactly one succeeds; atomic decrement; no negative availability

R3-TC-09 Re-borrow after return frees slot
- Preconditions: Patron currently at 5-book limit
- Steps: Return 1 book; borrow another
- Expected: Borrow succeeds; count stays within limit

Python test functions (unit tests against
`library_service.borrow_book_by_patron`):

```python
import pytest
import library_service
from library_service import borrow_book_by_patron


def _fake_book(book_id=1, available=1):
    return {
        "id": book_id,
        "title": "Any",
        "author": "X",
        "isbn": "1234567890123",
        "total_copies": 1,
        "available_copies": available,
    }


def test_r3_borrow_success(monkeypatch):
    monkeypatch.setattr(library_service, "get_patron_borrow_count", lambda pid: 4)
    monkeypatch.setattr(library_service, "get_book_by_id", lambda bid: _fake_book(bid, available=1))
    monkeypatch.setattr(library_service, "insert_borrow_record", lambda *a, **k: True)
    monkeypatch.setattr(library_service, "update_book_availability", lambda *a, **k: True)
~~~

~~~~
    ok, msg = borrow_book_by_patron("123456", 1)
    assert ok is True
    assert "successfully borrowed" in msg.lower()


def test_r3_borrow_invalid_patron_id(monkeypatch):
    monkeypatch.setattr(library_service, "get_book_by_id", lambda bid:
_fake_book(bid, available=1))
    ok, msg = borrow_book_by_patron("12345a", 1)
    assert ok is False and "6 digits" in msg


def test_r3_borrow_book_unavailable(monkeypatch):
    monkeypatch.setattr(library_service, "get_patron_borrow_count", lambda pid:
0)
    monkeypatch.setattr(library_service, "get_book_by_id", lambda bid:
_fake_book(bid, available=0))
    ok, msg = borrow_book_by_patron("123456", 1)
    assert ok is False and "not available" in msg.lower()


def test_r3_borrow_book_not_found(monkeypatch):
    monkeypatch.setattr(library_service, "get_book_by_id", lambda bid: None)
    ok, msg = borrow_book_by_patron("123456", 999)
    assert ok is False and "not found" in msg.lower()


def test_r3_current_impl_allows_6th_due_to_gt_5_check(monkeypatch):
    # Note: implementation denies only when count > 5; 5 active borrows still
allows another
    monkeypatch.setattr(library_service, "get_patron_borrow_count", lambda pid:
5)
    monkeypatch.setattr(library_service, "get_book_by_id", lambda bid:
_fake_book(bid, available=1))
    monkeypatch.setattr(library_service, "insert_borrow_record", lambda *a, **k:
True)
    monkeypatch.setattr(library_service, "update_book_availability", lambda *a,
**k: True)

    ok, _ = borrow_book_by_patron("123456", 1)
    assert ok is True  # reflects current code behavior
```

---

### R4: Book Return Processing
~~~~

~~~
R4-TC-01 Valid on-time return
- Preconditions: Borrowed book due in future
- Steps: Submit return with correct Patron ID & Book ID
- Expected: Success message; Available incremented; return date recorded; fee
$0.00

R4-TC-02 Return by wrong patron
- Preconditions: Book borrowed by Patron A; Patron B attempts return
- Steps: Submit B with Book ID
- Expected: Error indicating mismatch; no changes

R4-TC-03 Return for non-borrowed book
- Steps: Submit valid Patron/Book with no active borrow
- Expected: Error; no changes

R4-TC-04 Late return 1 day overdue
- Preconditions: Borrow recorded 15 days ago
- Steps: Return now
- Expected: Fee $0.50 displayed and recorded; 1 day overdue

R4-TC-05 Late return 8 days overdue (tiered rate)
- Preconditions: Borrow recorded 22 days ago (8 overdue)
- Steps: Return now
- Expected: Fee $4.50 (0.50×7 + 1.00×1); 8 overdue days

R4-TC-06 Very late return fee capped
- Preconditions: Borrow recorded long enough to exceed $15 cap (e.g., 40+ overdue
days)
- Steps: Return now
- Expected: Fee $15.00 (capped)

R4-TC-07 Double submit return
- Preconditions: Borrow just returned
- Steps: Submit return again for same Patron/Book
- Expected: Error indicating already returned; no duplicate records

R4-TC-08 Money formatting
- Steps: Trigger any non-zero fee scenario
- Expected: Fee displays with exactly 2 decimals (e.g., 12.50 not 12.5)

Python test functions (unit tests against
`library_service.return_book_by_patron`):

```python
~~~

```python
import pytest
from datetime import datetime, timedelta
import library_service
from library_service import return_book_by_patron


def _loan(book_id=1, overdue_days=8):
    now = datetime.now()
    return [{
        "book_id": book_id,
        "title": "Clean Code",
        "author": "Robert C. Martin",
        "borrow_date": now - timedelta(days=14 + max(0, overdue_days)),
        "due_date": now - timedelta(days=max(0, overdue_days)),
        "is_overdue": overdue_days > 0,
    }]


def test_r4_return_success_reports_fee(monkeypatch):
    monkeypatch.setattr(library_service, "get_patron_borrowed_books", lambda pid: _loan(1, 8))
    monkeypatch.setattr(library_service, "get_book_by_id", lambda bid: {"id": bid, "title": "Clean Code"})
    monkeypatch.setattr(library_service, "update_borrow_record_return_date", lambda *a, **k: True)
    monkeypatch.setattr(library_service, "update_book_availability", lambda *a, **k: True)
    monkeypatch.setattr(library_service, "calculate_late_fee_for_book",
                        lambda pid, bid: {"fee_amount": 4.50, "days_overdue": 8, "status": "ok"})

    ok, msg = return_book_by_patron("123456", 1)
    assert ok is True
    assert "Returned \"Clean Code\"" in msg
    assert "$4.50" in msg and "8 days" in msg


def test_r4_return_wrong_patron_or_not_borrowed(monkeypatch):
    monkeypatch.setattr(library_service, "get_patron_borrowed_books", lambda pid: [])
    monkeypatch.setattr(library_service, "get_book_by_id", lambda bid: {"id": bid, "title": "X"})
    ok, msg = return_book_by_patron("123456", 1)
    assert ok is False and ("not currently borrowed" in msg.lower())
```

~~~
def test_r4_return_book_not_found(monkeypatch):
    monkeypatch.setattr(library_service, "get_book_by_id", lambda bid: None)
    ok, msg = return_book_by_patron("123456", 999)
    assert ok is False and "book not found" in msg.lower()


def test_r4_return_db_error_paths(monkeypatch):
    monkeypatch.setattr(library_service, "get_patron_borrowed_books", lambda pid: _loan(1, 1))
    monkeypatch.setattr(library_service, "get_book_by_id", lambda bid: {"id": bid, "title": "X"})
    monkeypatch.setattr(library_service, "update_borrow_record_return_date", lambda *a, **k: False)
    monkeypatch.setattr(library_service, "update_book_availability", lambda *a, **k: True)

    ok, msg = return_book_by_patron("123456", 1)
    assert ok is False and ("return date" in msg.lower() or "updating" in msg.lower())
```

---

### R5: Late Fee Calculation API (GET `/api/late_fee/<patron_id>/<book_id>`)

R5-TC-01 On-time (≤ 14 days) -> $0.00
- Preconditions: Borrow date today
- Steps: Call endpoint
- Expected: 200 JSON `{ fee: 0.00, days_overdue: 0 }`

R5-TC-02 Exactly 14 days -> $0.00
- Preconditions: Borrow date exactly 14 days ago
- Steps: Call endpoint
- Expected: 200 JSON `{ fee: 0.00, days_overdue: 0 }`

R5-TC-03 Seven days overdue (21 since borrow)
- Preconditions: Borrow date 21 days ago
- Steps: Call endpoint
- Expected: 200 JSON `{ fee: 3.50, days_overdue: 7 }`

R5-TC-04 Sixteen days overdue (30 since borrow)
- Preconditions: Borrow date 30 days ago
- Steps: Call endpoint
- Expected: 200 JSON `{ fee: 12.50, days_overdue: 16 }`
~~~

~~~
R5-TC-05 Over cap
- Preconditions: Borrow far in past so computed fee > 15
- Steps: Call endpoint
- Expected: 200 JSON `{ fee: 15.00, days_overdue: <>=something positive }` with cap applied

R5-TC-06 Non-existent borrow
- Steps: Use valid formats but no matching active borrow
- Expected: 404 or 400 JSON error with explanatory message; no server error

R5-TC-07 Invalid path params
- Steps: Patron ID="ABC123", Book ID="xyz"
- Expected: 400 JSON validation error; no crash

R5-TC-08 Rounding and formatting
- Preconditions: Choose dates that produce fractional totals
- Steps: Call endpoint
- Expected: Fee rounded/represented with exactly 2 decimals

R5-TC-09 Idempotent calls
- Steps: Call the endpoint multiple times without state changes
- Expected: Same response each time

Python test functions (Flask route tests against `/api/late_fee/<patron>/<book>`):

```python
import pytest
import library_service


@pytest.fixture(scope="session")
def app():
    from app import create_app
    return create_app()


@pytest.fixture()
def client(app):
    return app.test_client()


def test_r5_late_fee_api_happy_path(client, monkeypatch):
    monkeypatch.setattr(library_service, "calculate_late_fee_for_book",
~~~

~~~~
                        lambda pid, bid: {"fee_amount": 12.50, "days_overdue": 16, "status": "ok"})
    resp = client.get("/api/late_fee/123456/1")
    assert resp.status_code == 200
    data = resp.get_json()
    assert data["fee_amount"] == 12.50
    assert data["days_overdue"] == 16


def test_r5_late_fee_api_invalid_patron_id_returns_error_status(client, monkeypatch):
    # Service will return status 'invalid_patron_id'; API currently forwards as 200 JSON
    monkeypatch.setattr(library_service, "calculate_late_fee_for_book",
                        lambda pid, bid: {"fee_amount": 0.0, "days_overdue": 0, "status": "invalid_patron_id"})
    resp = client.get("/api/late_fee/abc123/1")
    assert resp.status_code == 200
    assert resp.get_json().get("status") == "invalid_patron_id"
```

---

### R6: Book Search Functionality

R6-TC-01 Title partial match (case-insensitive)
- Preconditions: Titles: "Clean Code", "CLEAN Architecture"
- Steps: Search `q=clean`, `type=title`
- Expected: Returns both; not other unrelated titles

R6-TC-02 Author partial match (case-insensitive)
- Preconditions: Authors: "Robert C. Martin", "Martin Fowler"
- Steps: Search `q=martin`, `type=author`
- Expected: Returns both; partial match OK

R6-TC-03 ISBN exact match only
- Preconditions: Book with ISBN=9780132350884 exists
- Steps: Search `q=9780132350884`, `type=isbn`
- Expected: Returns the single exact match

R6-TC-04 ISBN with hyphens should not match
- Steps: Search `q=978-0-13-235088-4`, `type=isbn`
- Expected: No results (exact 13 digits required)

R6-TC-05 Trimmed input
~~~~

~~~
- Steps: Search `q="  code  "`, `type=title`
- Expected: Treated as `"code"`; results returned accordingly

R6-TC-06 No results UX
- Steps: Query that matches nothing
- Expected: Empty-state messaging; same table structure; no errors

R6-TC-07 Invalid type handling
- Steps: `type=category`
- Expected: 400 error or clear validation feedback; no crash

R6-TC-08 HTML/SQL injection in query
- Steps: `q="<b>code"` and `q="' OR 1=1 --"` for title/author
- Expected: Inputs escaped; no injection; only relevant results

Python test functions (unit tests against
`library_service.search_books_in_catalog`):

```python
import pytest
import library_service
from library_service import search_books_in_catalog


def _row(id, title, author, isbn):
    return {"id": id, "title": title, "author": author, "isbn": isbn,
            "total_copies": 1, "available_copies": 1}


def test_r6_title_partial_case_insensitive(monkeypatch):
    sample = [
        _row(1, "Clean Code", "Robert C. Martin", "9780132350884"),
        _row(2, "CLEAN Architecture", "Robert C. Martin", "9780134494166"),
        _row(3, "Domain-Driven Design", "Eric Evans", "9780321125217"),
    ]
    monkeypatch.setattr(library_service, "get_all_books", lambda: sample)
    results = search_books_in_catalog("clean", "title")
    titles = {r["title"] for r in results}
    assert "Clean Code" in titles and "CLEAN Architecture" in titles
    assert "Domain-Driven Design" not in titles


def test_r6_author_partial_case_insensitive(monkeypatch):
    sample = [
        _row(1, "A", "Harper Lee", "9780061120084"),
~~~

~~~
        _row(2, "B", "Lee Child", "9780440243694"),
        _row(3, "C", "George Orwell", "9780451524935"),
    ]
    monkeypatch.setattr(library_service, "get_all_books", lambda: sample)
    results = search_books_in_catalog("lEe", "author")
    authors = {r["author"] for r in results}
    assert authors == {"Harper Lee", "Lee Child"}


def test_r6_isbn_exact_match(monkeypatch):
    exact = "9780451524935"
    monkeypatch.setattr(library_service, "get_book_by_isbn",
                        lambda q: _row(3, "1984", "George Orwell", exact) if q ==
exact else None)
    assert search_books_in_catalog(exact, "isbn")[0]["isbn"] == exact
    assert search_books_in_catalog(exact[:6], "isbn") == []


def test_r6_invalid_type_raises():
    with pytest.raises(ValueError):
        search_books_in_catalog("anything", "publisher")


def test_r6_empty_query_returns_empty_list(monkeypatch):
    monkeypatch.setattr(library_service, "get_all_books", lambda: [])
    assert search_books_in_catalog("", "title") == []
```

---

### R7: Patron Status Report

R7-TC-01 Displays current borrows with due dates
- Preconditions: Patron with 2 active borrows
- Steps: Open Patron Status page for that patron
- Expected: Lists both items with accurate due dates (borrow date + 14 days)

R7-TC-02 Total late fees owed aggregates correctly
- Preconditions: Patron with two overdue items (1 and 8 days overdue)
- Steps: View status
- Expected: Shows sum of individual fees (e.g., 0.50 + 4.50 = 5.00)

R7-TC-03 Number of books currently borrowed
- Preconditions: Patron with 3 active borrows
- Steps: View status
~~~

~~~
- Expected: Shows count=3

R7-TC-04 Borrowing history shows returned items
- Preconditions: Patron has past returned items in history
- Steps: View status
- Expected: History section lists returned items with borrow/return dates

R7-TC-05 Patron with no history
- Preconditions: New patron
- Steps: View status
- Expected: Clear empty-state messaging for all sections; no errors

R7-TC-06 Money formatting and caps
- Preconditions: Include overdue items that hit the cap
- Steps: View status
- Expected: Each fee and total display with 2 decimals; capped at $15.00 per book

R7-TC-07 Navigation from main menu
- Steps: From main interface, use menu option to open Patron Status
- Expected: Menu entry exists and navigates correctly

R7-TC-08 Data freshness after actions
- Preconditions: Patron page open in one tab
- Steps: Return a book in another tab; refresh status
- Expected: Counts, current borrows, and late fee totals update accurately

Python test functions (unit tests against
`library_service.get_patron_status_report`):

```python
import pytest
from datetime import datetime, timedelta
import library_service
from library_service import get_patron_status_report


def _borrowed(book_id, title, overdue_days):
    now = datetime.now()
    return {
        "book_id": book_id,
        "title": title,
        "author": "A",
        "borrow_date": now - timedelta(days=14 + max(0, overdue_days)),
        "due_date": now - timedelta(days=max(0, overdue_days)),
        "is_overdue": overdue_days > 0,
~~~

~~~
    }


def test_r7_report_happy_path(monkeypatch):
    current = [
        _borrowed(1, "Clean Code", 8),
        _borrowed(2, "Dune", 0),
    ]
    monkeypatch.setattr(library_service, "get_patron_borrowed_books", lambda pid:
current)
    monkeypatch.setattr(library_service, "get_patron_borrow_count", lambda pid:
2)

    def fee_for(pid, bid):
        return {"fee_amount": 4.50, "days_overdue": 8, "status": "ok"} if bid ==
1 else \
               {"fee_amount": 0.00, "days_overdue": 0, "status": "ok"}
    monkeypatch.setattr(library_service, "calculate_late_fee_for_book", fee_for)

    report = get_patron_status_report("123456")
    assert report["current_borrow_count"] == 2
    assert round(float(report["total_late_fees"]), 2) == 4.50
    assert len(report["current_borrows"]) == 2
    assert isinstance(report.get("history"), list)


def test_r7_invalid_patron_id(monkeypatch):
    report = get_patron_status_report("12345")
    assert isinstance(report, dict) and report.get("status") ==
"invalid_patron_id"
```
~~~