CST438 Assignment 9

Design unit tests for the Rest controller in assignment 8. You are not required to code the tests although you might it helpful to do so. The set of tests should be designed to provide “C0” (Statement) level coverage -- that is every code block is executed by at least one test in the test suite. Recall that unit tests should use the FIRST principle (Fast, Independent, Repeatable, Self-checking, Timely).

Complete the following table which describes each test. Submit this completed document for credit.

The first few tests are done for you as an example.

Test	Description
T1	Checkout a book URL tested: **PUT** /book/{book_id}/checkout/{patron_id} Mock: BookRepository.findByBookId returns a Book entity Mock: PatronRepository.findByPatronId returns a Patron entity with no fines Expected HTTP Response Code: **200 Ok** Expected Return Value: **0** Verify: BookRepository.save is called
T2	Attempt to checkout a book when patron has fines URL tested: **PUT** /book/{book_id}/checkout/{patron_id} Mock: BookRepository.findByBookId returns a Book entity Mock: PatronRepository.findByPatronId, returns a Patron entity with fines Expected HTTP Response Code: **200 Ok** Expected Return Value: **-3** Verify: BookRepository.save is not called
T3	Attempt to checkout a book with an invalid patron_id URL tested: **PUT** /book/{book_id}/checkout/{patron_id} Mock: BookRepository.findByBookId returns a Book entity Mock: PatronRepository.findByPatronId returns null Expected HTTP Response Code: **404 Not Found** Expected Return Value: message: “Patron with ID <some ID> does not exist!” Verify: BookRepository.save is not called
T4	Attempt to checkout a book with an invalid book_id URL tested: **PUT** /book/{book_id}/checkout/{patron_id} Mock: BookRepository.findByBookId returns null Expected Http Response Code: **404 Not Found** Expected Return Value: message: “Book with ID <some ID> does not exist!” Verify: BookRepository.save is not called
T5	Return a book URL tested: **PUT** /book/{book_id}/return Mock: BookRepository.findByBookId returns a Book entity Expected HTTP Response Code: **200 Ok** Expected Return Value: **0** Verify: BookRepository.save is called

T6	Attempt to return a book with an invalid book_id URL tested: **PUT** /book/{book_id}/return Mock: BookRepository.findByBookId returns null Expected HTTP Response Code: **404 Not Found** Expected Return Value: message: “Book with ID <some ID> does not exist!” Verify: BookRepository.save is not called
T7	List a patron's checkouts URL tested: **GET** /patron/{patron_id}/checkouts Mock: PatronRepository.findByPatronId returns a Patron entity Mock: BookRepository.findByCheckoutPatronId returns a List of Book entities Expected HTTP Response Code: **200 Ok** Expected Return Value: A list of BookDTO objects, each representing a Book checked out by the patron Verify: BookRepository.findByCheckoutPatronId is called
T8	Attempt to list a patron's checkouts with an invalid patron_id URL tested: **GET** /patron/{patron_id}/checkouts Mock: PatronRepository.findByPatronId returns null Expected HTTP Response Code: **404 Not Found** Expected Return Value: message: “Patron with ID <some ID> does not exist!” Verify: BookRepository.findByCheckoutPatronId is not called
T9	Display a patron's information URL tested: **GET** /patron/{patron_id} Mock: PatronRepository.findByPatronId returns a Patron entity Expected HTTP Response Code: **200 Ok** Expected Return Value: A PatronDTO object, representing a Patron object with all their information (ID, name, fees) Verify: HTTP Response Code is **200 Ok**
T10	Attempt to display a patron's information with an invalid patron_id URL tested: **GET** /patron/{patron_id} Mock: PatronRepository.findByPatronId returns null Expected HTTP Response Code: **404 Not Found** Expected Return Value: message: “Patron with ID <some ID> does not exist!” Verify: HTTP Response Code is **404 Not Found**