**TEST FILE FOR ONLINE LIBRARY MANAGEMENT**

In the assignment given I used google test framework to cover various scenarios for both the “book” and “library” classes

The steps were as follows:

**Step 1:setting up**

Just trying and ensure the directory of the project has the following structure

“my\_project/

├── src/

│ └── book.h

│ └── book.cpp

│ └── library.h

│ └── library.cpp

└── tests/

└── book\_test.cpp

└── library\_test.cpp”

**Step 2: writing of classes**

*The book class* :

“// src/book.h

#ifndef BOOK\_H

#define BOOK\_H

#include <string>

class Book {

private:

std::string title;

std::string author;

bool available;

int rating;

public:

Book(std::string title, std::string author);

std::string getTitle() const;

std::string getAuthor() const;

bool isAvailable() const;

void setAvailable(bool availability)

int getRating() const;

void setRating(int newRating);

};

#endif // BOOK\_H”

*The library class :*

“// src/book.cpp

#include "book.h"

Book::Book(std::string title, std::string author) : title(title), author(author), available(true), rating(0) {}

std::string Book::getTitle() const {

return title;

}

std::string Book::getAuthor() const {

return author;

}

bool Book::isAvailable() const {

return available;

}

void Book::setAvailable(bool availability) {

available = availability;

}

int Book::getRating() const {

return rating;

}

void Book::setRating(int newRating) {

rating = newRating;

}”

**step 3: writing of the test cases file**

since we are using google test framework we create test files using google test

**Book Test**

Here goes the code for the book test file;

“// tests/book\_test.cpp

#include "gtest/gtest.h"

#include "book.h"

TEST(BookTest, Constructor) {

Book book("Test Title", "Test Author");

EXPECT\_EQ(book.getTitle(), "Test Title");

EXPECT\_EQ(book.getAuthor(), "Test Author");

EXPECT\_TRUE(book.isAvailable());

EXPECT\_EQ(book.getRating(), 0);

}

TEST(BookTest, SetAvailable) {

Book book("Test Title", "Test Author");

book.setAvailable(false);

EXPECT\_FALSE(book.isAvailable());

}

TEST(BookTest, SetRating) {

Book book("Test Title", "Test Author");

book.setRating(5);

EXPECT\_EQ(book.getRating(), 5);

}

int main(int argc, char \*\*argv) {

::testing::InitGoogleTest(&argc, argv);

return RUN\_ALL\_TESTS();

}”

**Library Test**

Here goes a file for testing the library class;

“// tests/library\_test.cpp

#include "gtest/gtest.h"

#include "library.h"

TEST(LibraryTest, AddBook) {

Library library;

Book book("Test Title", "Test Author");

library.addBook(book);

std::vector<Book> books = library.searchBooksByTitle("Test Title");

ASSERT\_EQ(books.size(), 1);

EXPECT\_EQ(books[0].getTitle(), "Test Title");

EXPECT\_EQ(books[0].getAuthor(), "Test Author");

}

TEST(LibraryTest, SearchBooksByTitle) {

Library library;

Book book1("Test Title", "Test Author");

Book book2("Another Title", "Another Author");

library.addBook(book1);

library.addBook(book2);

std::vector<Book> books = library.searchBooksByTitle("Test Title");

ASSERT\_EQ(books.size(), 1);

EXPECT\_EQ(books[0].getTitle(), "Test Title");

}

TEST(LibraryTest, BorrowBookAvailable) {

Library library;

Book book("Test Title", "Test Author");

library.addBook(book);

std::vector<Book> books = library.searchBooksByTitle("Test Title");

ASSERT\_FALSE(books.empty());

Book& foundBook = books[0];

library.borrowBook(foundBook);

EXPECT\_FALSE(foundBook.isAvailable());

}

TEST(LibraryTest, BorrowBookNotAvailable) {

Library library;

Book book("Test Title", "Test Author");

library.addBook(book);

std::vector<Book> books = library.searchBooksByTitle("Test Title");

ASSERT\_FALSE(books.empty());

Book& foundBook = books[0];

foundBook.setAvailable(false);

testing::internal::CaptureStdout();

library.borrowBook(foundBook);

std::string output = testing::internal::GetCapturedStdout();

EXPECT\_EQ(output, "Book \"Test Title\" by Test Author is not available for borrowing.\n");

}

TEST(LibraryTest, ListAvailableBooks) {

Library library;

Book book1("Test Title 1", "Test Author 1");

Book book2("Test Title 2", "Test Author 2");

library.addBook(book1);

library.addBook(book2);

book2.setAvailable(false);

testing::internal::CaptureStdout();

library.listAvailableBooks();

std::string output = testing::internal::GetCapturedStdout();

EXPECT\_NE(output.find("Test Title 1"), std::string::npos);

EXPECT\_EQ(output.find("Test Title 2"), std::string::npos);

}

int main(int argc, char \*\*argv) {

::testing::InitGoogleTest(&argc, argv);

return RUN\_ALL\_TESTS();

}”

**Step 4: compiling and running the tests**

After installing and configuring the google test we now compile and run the tests

Within the code there are some comments to easen understanding and know what each line does

“# Navigate to the project root directory

cd my\_project

# Create a build directory

mkdir build

cd build

# Run CMake to generate build files

cmake ..

# Build the tests

make

# Run the Book tests

./tests/book\_test

# Run the Library tests

./tests/library\_test”

Now let’s break it down in such a way that we will understand it

The test covers almost all scenarios including:

1. Constructing “book” object and trying to to validate its initial state
2. Trying to modify availability and rating of the books in the library
3. Enebling the owner to addd books to the library and allowing end user to search for them by title
4. Enebles end uer to borrow a book and it checks its availability
5. It handles the scenario whereby a book might already be borrowed
6. It lists the books available after the rating