# Asg3 ReadMe

The source code you require for the assignment can be found in the ‘src’ folder inside the ‘BRLS’ directory of this resource bundle.

Please Note: there are defects in the code according to the specifications provided. Do NOT fix the code, merely report on any defects detected. For the purposes of this assignment, a defect is where the code does not adhere to the specification. Nothing else.

For Part A of Assignment 3 you are required to develop a Master Test Plan

Guidance:

1. Leave the master test plan until we have covered integration, system, and user acceptance testing in lectures.
2. Base your master test plan on an evaluation of the Project Vision, the NFR Analysis, the Architecture Notebook, and the design documents and diagrams provided. Consider ALL test approaches in terms of their effectiveness in detecting the defects in the codebase.
3. You may use your imagination and initiative to a limited extent in determining your overall test strategy. Try to keep it reasonable in terms of the needs of a small country town library and council.

Part B is more prescriptive. The ‘master test plan’ is as follows:

The objective is to test the implementation of the ‘Borrow Book’ use case as thoroughly as possible.

As a minimum, you are required to perform Unit and Low Level integration tests for the following methods in the specified classes:

Library:

patronCanBorrow

\*issueLoan (optional)

commitLoan

Loan:

commit

\*checkOverDueDate (optional)

Patron:

hasOverDueLoans

takeOutLoan

Book

isAvailable

borrowFromLibrary

If you are attempting to achieve higher grades (DI and HD), you should test the methods identified as optional above and also unit and integration test the control class methods identified below.

BorrowBookControl:

cardSwiped

bookScanned

commitLoans

As a final stage of integration testing, you might consider carrying out ‘scenario testing’ in which you automate the sequence of control class calls that would be used in a use case scenario.

You are not required to test the UI class.

System or Functional Acceptance Tests (FAT)should be carried out for as many use case scenarios as seems appropriate to you. All FAT should be grouped under the test scenario ‘Borrow Book’. There is a Test Scenario Template provided in the resource bundle to support you with this.

Guidance

1. Pick one version of JUnit and stick to it.
2. You might find it easiest to perform low level integration testing first. This type of testing does not require you to use test substitutes.
3. Functional Acceptance tests are also very easy – they require no coding, just careful thought as to the initial conditions, and close attention to the expected results for each step of a use case scenario.
4. Watch the HowTos and go through some internet tutorials before getting started with Mockito (or the test substitute framework of your choice)