# Test Description

Testing was focused on happy path tests for the behavior (requirements) described in the case description. Normally when more time is given, I would also implement better error handling and test erroneous input.

## Testing The Service Layer

I was not sure if I could use third party mocking libraries (such as “Moq” for xUnit) to mock the repository interfaces. So instead, I provided real implementations of the repositories when testing the service layer. In this way, I verified the implementation of the data storage at the same time. Normally I would mock the interfaces to control the responses from them.

I added one happy path test for each service method as a form of sanity check that I implemented all operations specified in the case description.

## Testing The Repository Layer

Usually, this layer is implemented by a third-party solution in which case the actual implementation shall already be unit tested. I did not unit test the data storage separately since it is an implementation detail; and as stated before I would normally mock this layer or provide a fake.

## Test Scenarios

I added some additional test scenarios like the one provided in the case description. I focused on testing the behavioral cases I could elicit from the case description, like state transition and issue filtering.