# Testing Approaches

For this test, I applied following approach:

* Cover core functionalities of Core API
  + CreateQueue
  + SendMessage
  + ReceiveMessage
  + DeleteMessage
* Extend to extended functionality
* Extend further tests with
  + Negative cases
  + All attributes
  + All methods

However, due to time and knowledge limit, I wasn’t able to cover all cases. I have listed all the tests that I would like to develop for testing in each class. They don’t have actual code, just to throw exception if they are invoked.

Regarding the structure of the test project, I applied as following

* Developed based on Maven + TestNG
* A BaseTest class developed with 2 methods for @BeforeTest and @AfterTest annotations
  + @BeforeTest will run before each test. It will prepare the connection, loading parameters’ value, preparing conections, etc.
  + @AfterTest will run at the end of each test. It will clean up the data created during test execution and close the connection.
* Common functions are stored in BaseTest.
* Each API is developed in a class, extends from BaseTest.
* All tests of that API are stored in that class.

The reason for using that structure is

* Connection info, configuration, reusable functions are store in a central place.
* Easy to locate to each API and the tests developed for that API.

# What are not good with the submission?

* Too much repetitive code. I was exploring the functionality, and the quickest way was to copy code.
* The coverage may not as extensive as an API test suite should have.

# Future development:

* Reduce redundant code: develop more library for common functions, like send message, receive message, delete queue, etc.
* Develop a skeleton of test, which other similar APIs can reuse. Just need to pass different set of parameters.
* Load parameters’ value from external file. Currently, I’m using hardcode value and random string, which is not ideal for automation testing.
* Create different test groups for existing tests.
* Include ElasticMQ call in BaseTest.

In summary, I have spent more than 4 and less than 5 hours for this task. It was due to I haven’t worked much with API testing, and first attempted to start ElasticMQ before each test run wasn’t success. The setup of AWS SDK and the start of stand-alone ElasticMQ didn’t take much time. I was taking time to understand the APIs and implement the tests for them. For a candidate which have more experience in API testing, they would take less than 4 hours to complete this.