Group 17

To be able to correctly develop test in a proper manner we started out with drawing up a decision table. In the decision table we state what we want our functions to do and how we want them to respond as well as where it will fail. As follows:

Add Method

Conditions				
C1: Non-empty string	N	Y	Y	Y
C2: Sender telephone number [5-10]	-	N	Y	Y
C3: Receiver telepone number [5-10]	-	-	N	Y
Actions				
A1: Wrong inputs	X	X	X	-
A2: Return unique identifier	-	-	-	X

Delete Method

Conditions			
C1: Unique identifiers exist	N	Y	Y
C2: Not fetched	-	N	Y
Actions			
A1: Return negative integer	X	X	-

A2: Return positive integer	-	-	X

Replace Method

Conditions				
C1: Unique identifier exists	N	Y	Y	Y
C2: Not fetched	-	N	Y	Y
C3: Non-empty string	-	-	N	Y
Actions				
A1: Return negative integer	X	X	X	-
A2: Unique identifier is returned	-	-	-	X

Fetch Method

Conditions			
C1:Have logged on	N	Y	Y
C2: Recipient Identifier exists	-	N	Y
Actions			
A1: Fail	X	X	-

A2: Messages returned	-	-	X

Fetch-complete Method

Conditions			
C1: Messages have been fetched	N	Y	Y
C2: User mark fetched messages	-	N	y
Actions			
A1: Return positive number	-	-	X
A2: Return negative number	X	X	-

Then we made a TDD template so that we knew how we were going to approach our test development.

We started out with writing tests for the add function. We don't want that function to take empty messages or invalid phone numbers. So we test whether the function notices this and writes correct error messages if it receives faulty input. So when it came to implementation, this was a priority to achieve. If the add function doesn't notice a faulty input the test will fail.

In order for the delete message to be tested, we first need to create an add object. Since otherwise there will be no object to delete. Then the tests we wrote checks whether or not the message can be deleted, according to our decision table. We then wrote the function as such that the system can only remove a message if the message haven't been fetched and that there is a message ID.

Our expectations on the replace function according to the decision board is quite similar to the delete function. So there are a couple of simmular tests in the replace test. We added that the test will not accept an empty entry when replacing a message string. We take care of this in

the implementation and make sure in our function that the message haven’t been fetched nor that we are trying to enter an empty string.

The fetch and ”fetch complete” was quite a struggle. We actuely had to come back and re evaluate our decision board and our test cases during our implementation because we discovered we were testing our function on the wrong basis. Our re evaluation then resulted in that we check if the fetch function has anything to gather from a hashmap, if there are non, the function fails the test. Fetch complete should remove any messages that already has been delivered to the user. Fetch calls fetch complete when successfully sent the messages to the client and if the function successfully fills all the requirements from the design board. If the function manages to fulfill all it’s duties it will return a positive one. We also want the fetch complete to fail if the messages haven’t been received successfully. So we also like to test the function to fail under faulty circumstances.