**Moving Target**

**Test Plan Document**

**Version 1.0**

Moving Target

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 24/03/2012 | 1.0 | First version of test plan | Filipe |

Table of Contents

[1. Introduction 4](#_Toc320380426)

[1.1 Purpose 4](#_Toc320380427)

[1.2 Scope 4](#_Toc320380428)

[1.3 Intended Audience 4](#_Toc320380429)

[2. Test Plan Identifier 5](#_Toc320380430)

[3. Integration Testing 6](#_Toc320380431)

# Introduction

We are developing a messaging dropping system (temp) following the agile iterative methodology. The same procedure was followed for our testing process. In this document, we’ll outline how the testing process was performed iteratively.

## 1.1 Purpose

The purpose of this document is to provide a detailed description of the testing process that we performed while developing our project. We will list all relevant test cases designed to assess the validity our system’s functionality and requirements. These test cases are derived from the user stories described in our Vision Document.

## 1.2 Scope

Unit tests are performed on individual components of our system. Components that provide single functions are combined in our integration and acceptance tests.

## 1.3 Intended Audience

This document is intended mainly for the software developers so that they can verify that the application is behaving as expected. It is designed as a reference for the development team to verify the status of all the tests, and known defects with their fixes. It is also intended for our client (s). They can verify the coverage and status of the tests as well as the current state of the system.

# Test Plan Identifier

IT: Integration testing

UT: Unite testing

# Test Motivation

As this was our first project with a client outside of Concordia and our final Engineering project, it was imperative that we could ensure that the application we end up submitting behaves as intended. This was also the first time that we worked on a project, where scale was a real issue and posed a great risk to the functionality of the application. Our goal is to mitigate these risks by monitoring and documenting the behaviour of the different modules that make up our system. Having a proper test process ensures this.

# Test Items

Every feature and high risk component of the application have been tested. A description of these features is available in the Vision Document.

# Features to be tested

The tested features are the following:

Note: These are all commands.

1. Creating a message

* Stores an audio message with its coordinates and speed in the database using mappers.

1. Creating a user

* Stores a user with a valid and unique email address in the database using mappers.

1. Deleting a user

* Deletes an existing user from the database using its user id and version.

1. Deleting a message

* Deletes an existing message from the database using its message id

1. Downvoting a message

* Decrements the rating of a valid message using mappers.

1. Upvoting a message

* Increments the rating of a valid message using mappers.

1. Getting message ids

* Returns a list of message ids in the proximity of the specified longitude and latitude. The speed is used to determine the range.

1. Getting server parameters

* Returns the list of server parameters that are stored in the database. Used for JSPs.

1. Pinging a server

* Verifies the status of a server by verifying it’s active or inactive depending on the response received.

1. Reading a message

* Returns the corresponding message of each valid and existing message id sent.

1. Reading a user

* Returns the user corresponding to the user id that was sent if it is valid and exists I n the database.

1. Updating a user

* Allows the information of a user to be modified. This increments the user’s version as well.

# Features not to be tested

?

# Testing Approach

Different testing approached were employed when testing the different functionalities of the system. These include unit and integration testing, black box and white box testing and acceptance tests.

For unit and white box testing, the tests were implemented using JUnit. These tests were designed alongside the code and test specific functions. Each of them have a set of inputs values and outputs. The outputs are compared to expected actuals to see if they match. If they do, then the test passes.

For integration and black box testing, the components tested in the unit tests were combined. An HTTP request was sent for each of the commands, and the response sent by the server was read and compared to our expected response.

# Item Pass/Fail Criteria

# Integration Testing

|  |  |
| --- | --- |
| Test Case ID | IT-0001 |
| Related User Story |  |
| Tested Item | CreateUserCommand with valid parameters |
| Preconditions | Database must exist |
| Input | Email: testing@test.com  Password: capstone  User Type: USER\_NORMAL  Response Type: bin |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The user is added to the User table. The response returns the User id and contains a status code equal to 200 |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testCreateUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0002 |
| Related User Story |  |
| Tested Item | CreateUserCommand with an invalid email |
| Preconditions | Database must exist |
| Input | Email: test  Password: capstone  User Type: USER\_NORMAL  Response Type: bin |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The user is not added to the User table. A ParameterException is thrown and the response contains a status code equal to 400 |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testCreateUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0003 |
| Related User Story |  |
| Tested Item | CreateUserCommand with an invalid password |
| Preconditions | Database must exist |
| Input | Email: testing2@test.com  Password: ““ (Empty String)  User Type: USER\_NORMAL  Response Type: bin |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The user is not added to the User table. A ParameterException is thrown and the response contains a status code equal to 400 |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testCreateUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0004 |
| Related User Story |  |
| Tested Item | CreateUserCommand with an invalid user type |
| Preconditions | Database must exist |
| Input | Email:testing3@test.com  Password: capstone  User Type: test  Response Type:bin |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The user is not added to the User table. A ParameterException is thrown and response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testCreateUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0005 |
| Related User Story |  |
| Tested Item | CreateUserCommand with an invalid response type |
| Preconditions | Database must exist |
| Input | Email: testing4@test.com  Password: capstone  User Type: USER\_NORMAL  Response Type: test |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The user is not added to the User table. A ParameterException is thrown and the response contains a status code equal to 400 |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testCreateUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0006 |
| Related User Story |  |
| Tested Item | CreateMessageCommand with valid parameters |
| Preconditions | Database must exist as well as at least on user in the User Table |
| Input | File: A valid amr audio file  Latitude: -73  Longitude: 45  Speed: 50  Email: testing@test.com |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message is added to the Message table. The response returns the Message id and contains a status code equal to 200. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testCreateMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0007 |
| Related User Story |  |
| Tested Item | CreateMessageCommand with an invalid latitude |
| Preconditions | Database must exist as well as at least on user in the User Table |
| Input | File: A valid amr audio file  Latitude: test  Longitude: 45  Speed: 50  Email: testing@test.com |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message is not added to the Message table. A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testCreateMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0008 |
| Related User Story |  |
| Tested Item | CreateMessageCommand with an invalid longitude |
| Preconditions | Database must exist as well as at least on user in the User Table |
| Input | File: A valid amr audio file  Latitude: -73  Longitude: test  Speed: 50  Email: testing@test.com |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message is not added to the Message table. A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testCreateMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0009 |
| Related User Story |  |
| Tested Item | CreateMessageCommand with an invalid speed |
| Preconditions | Database must exist as well as at least on user in the User Table |
| Input | File: A valid amr audio file  Latitude: -73  Longitude: 45  Speed: test  Email: testing@test.com |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message is not added to the Message table. A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testCreateMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0010 |
| Related User Story |  |
| Tested Item | CreateMessageCommand with an invalid email |
| Preconditions | Database must exist as well as at least on user in the User Table |
| Input | File: A valid amr audio file  Latitude: -73  Longitude: 45  Speed: 50  Email: test |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message is not added to the Message table. A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testCreateMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0011 |
| Related User Story |  |
| Tested Item | CreateMessageCommand with a non-existing email (user) |
| Preconditions | Database must exist as well as at least on user in the User Table |
| Input | File: A valid amr audio file  Latitude: -73  Longitude: 45  Speed: 50  Email: abc@abc.com |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message is not added to the Message table. A MapperException is thrown and the response contains a status code equal to 500. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testCreateMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0012 SPEED NEEDS TO BE VERY HIGH TO FIND MESSAGES |
| Related User Story |  |
| Tested Item | GetMessageIds with valid parameters |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Latitude: -73  Longitude: 45  Speed: 100000  Response type: bin  Sort type: user\_rating |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The response returns a list of message ids and contains a status code of 200 |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testgetMessageIdsCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0013 |
| Related User Story |  |
| Tested Item | GetMessageIds with an invalid longitude |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Latitude: -73  Longitude: test  Speed: 100000  Response type: bin  Sort type: user\_rating |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testgetMessageIdsCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0014 |
| Related User Story |  |
| Tested Item | GetMessageIds with an invalid latitude |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Latitude: test  Longitude: 45  Speed: 100000  Response type: bin  Sort type: user\_rating |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testgetMessageIdsCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0015 SQL EXCEPTION IS THROWN. VALIDADE SORT IN COMMAND |
| Related User Story |  |
| Tested Item | GetMessageIds with an invalid sort type |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Latitude: -73  Longitude: 45  Speed: 100000  Response type: bin  Sort type: test |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. (Or maybe it’s 500) |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testgetMessageIdsCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0016 |
| Related User Story |  |
| Tested Item | GetMessageIds with an invalid speed |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Latitude: -73  Longitude: 45  Speed: test  Response type: bin  Sort type: user\_rating |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testgetMessageIdsCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0017 |
| Related User Story |  |
| Tested Item | ReadMessageCommand with valid parameters |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Message id: 1920723613361488800001389104779468219  Response type: bin |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The response contains the requested message (s) and contains a status code of 200. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testReadMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0018 |
| Related User Story |  |
| Tested Item | ReadMessageCommand with an invalid message id |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Message id: test  Response type: bin |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testReadMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0019 |
| Related User Story |  |
| Tested Item | ReadMessageCommand with an non-existing message id |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Message id: 1  Response type: bin |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The response will not contain a message but contain a status code of 200. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testReadMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0020 |
| Related User Story |  |
| Tested Item | ReadMessageCommand with an invalid response type |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Message id: 1920723613361488800001389104779468219  Response type: test |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testReadMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0021 |
| Related User Story |  |
| Tested Item | UpVoteMessage with a valid message id |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Message Id: 1920723613361488800001389104779468219 |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message’s user rating is incremented and the response contains a status code equal to 200. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testReadMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0022 |
| Related User Story |  |
| Tested Item | UpVoteMessage with an invalid message id |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Message Id: test |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message’s user rating is not incremented. A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testUpvoteMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0023 |
| Related User Story |  |
| Tested Item | UpVoteMessage with a non-existing message id |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Message Id: test |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message’s user rating is not incremented. A MapperException is thrown and the response contains a status code equal to 500. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testUpvoteMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0024 |
| Related User Story |  |
| Tested Item | DownVoteMessage with a valid message id |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Message Id: 1920723613361488800001389104779468219 |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message’s user rating is decremented and the response contains a status code equal to 200. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testDownvoteMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0025 |
| Related User Story |  |
| Tested Item | DownVoteMessage with an invalid message id |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Message Id: test |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message’s user rating is not decremented. A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testDownvoteMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0026 |
| Related User Story |  |
| Tested Item | DownVoteMessage with a non-existing message id |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Message Id: 1 |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message’s user rating is not decremented. A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testDownvoteMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0027 |
| Related User Story |  |
| Tested Item | DeleteMessageCommand with a valid message id |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Message Id: 1920723613361488800001389104779468219 |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP DELETE request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message is removed from the Message Table and the response contains a status code equal to 200. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testDeleteMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0028 |
| Related User Story |  |
| Tested Item | DeleteMessageCommand with an invalid message id |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Message Id: test |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP DELETE request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message is not removed from the Message Table. A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testDeleteMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0029 |
| Related User Story |  |
| Tested Item | DeleteMessageCommand with a non-existing message id |
| Preconditions | Database must exist. The Message and User table must contain at least one entry |
| Input | Message Id: 1 |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP DELETE request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The message is not removed from the Message Table. A MapperException is thrown and the response contains a status code equal to 500. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testDeleteMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0030 |
| Related User Story |  |
| Tested Item | ReadUserCommand with valid parameters |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: 10047449289939392790145175573910595744  Response type: bin |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The response returns a user and contains a status code of 200. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testDeleteMessageCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0031 |
| Related User Story |  |
| Tested Item | ReadUserCommand with an invalid user id |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: test  Response type: bin |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testReadUserCommand |
| Test Case ID | IT-0032 |
| Related User Story |  |
| Tested Item | ReadUserCommand with an invalid response type |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: 10047449289939392790145175573910595744  Response type: test |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testReadUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0033 |
| Related User Story |  |
| Tested Item | ReadUserCommand with a non-existing user id |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: 1  Response type: test |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A MapperException is thrown and the response contains a status code equal to 500. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testReadUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0034 |
| Related User Story |  |
| Tested Item | UpdateUserCommand with valid parameters |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: 10047449289939392790145175573910595744  Password: newPassword  User type: USER\_NORMAL  Response type: bin  Version: 1 |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The corresponding user in the User Table is updated. The response contains a status code of 200. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testUpdateUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0035 |
| Related User Story |  |
| Tested Item | UpdateUserCommand with an invalid user id |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: test  Password: newPassword  User type: USER\_NORMAL  Response type: bin  Version: 1 |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testUpdateUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0036 |
| Related User Story |  |
| Tested Item | UpdateUserCommand with an invalid password |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: 10047449289939392790145175573910595744  Password: test  User type: USER\_NORMAL  Response type: bin  Version: 1 |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testUpdateUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0037 |
| Related User Story |  |
| Tested Item | UpdateUserCommand with an invalid use type |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: 10047449289939392790145175573910595744  Password: newPassword  User type: test  Response type: bin  Version: 1 |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testUpdateUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0038 |
| Related User Story |  |
| Tested Item | UpdateUserCommand with an invalid response type |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: 10047449289939392790145175573910595744  Password: newPassword  User type: USER\_NORMAL  Response type: test  Version: 1 |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testUpdateUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0039 |
| Related User Story |  |
| Tested Item | UpdateUserCommand with an invalid version |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: 10047449289939392790145175573910595744  Password: newPassword  User type: USER\_NORMAL  Response type: bin  Version: test |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testUpdateUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0040 |
| Related User Story |  |
| Tested Item | UpdateUserCommand with a non-existing user id |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: 1  Password: newPassword  User type: USER\_NORMAL  Response type: bin  Version: 1 |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP POST request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A MapperException is thrown and the response contains a status code equal to 500. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testUpdateUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0041 |
| Related User Story |  |
| Tested Item | DeleteUserCommand with valid parameters |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: 10047449289939392790145175573910595744  Version: 2  Response type: bin |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP DELETE request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The corresponding user is deleted from the User Table. The response contains a status code of 200. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testDeleteUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0042 |
| Related User Story |  |
| Tested Item | DeleteUserCommand with an invalid user id |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: test  Version: 2  Response type: bin |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP DELETE request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testDeleteUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0043 |
| Related User Story |  |
| Tested Item | DeleteUserCommand with a non-existing user id |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: 1  Version: 2  Response type: bin |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP DELETE request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A MapperException is thrown and the response contains a status code equal to 500. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testDeleteUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0044 |
| Related User Story |  |
| Tested Item | DeleteUserCommand with an invalid version |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id: 10047449289939392790145175573910595744  Version: test  Response type: bin |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP DELETE request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testDeleteUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0045 |
| Related User Story |  |
| Tested Item | DeleteUserCommand with an invalid response type |
| Preconditions | Database must exist. The User table must contain at least one user |
| Input | User id:10047449289939392790145175573910595744  Version: 2  Response type:test |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP DELETE request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | A ParameterException is thrown and the response contains a status code equal to 400. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testDeleteUserCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0046 |
| Related User Story |  |
| Tested Item | GetServerParametersCommand |
| Preconditions | Database must exist. The ServerParameters table contains tuples |
| Input | None |
| Steps | 1. Create a uri. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The response returns the server parameters and contains a status code of 200. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testGetServerParametersCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0047 |
| Related User Story |  |
| Tested Item | PingCommand |
| Preconditions | There’s at least one server available. |
| Input | None |
| Steps | 1. Create a uri. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The response contains a status code of 204. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testPingCommand |

|  |  |
| --- | --- |
| Test Case ID | IT-0048 |
| Related User Story |  |
| Tested Item | UnsupportedCommand |
| Preconditions | There’s at least one server available. |
| Input | Command: test |
| Steps | 1. Create a uri with the parameters. 2. Perform a HTTP GET request with the uri. 3. Read the response and compare it to the excepted outcome. |
| Expected Outcome | The response contains a status code of 501. |
| Code | https://github.com/BENGMN/soen490/blob/master/DemoClient/src/Tests/CommandTests.java  Function testUnsupportedCommand |