**Term Project: *Chatterbox***

Test Plan Document

**Table of Contents**

[Purpose and Scope](#_unoidi8dkabd) 2

[Target Audience](#_rn2tykxfoaxd) 2

[Terms and Definitions](#_b1jwe8mf918x) 2

[**Test Plan Description**](#_cfttai1sf11a) **2**

[Scope of Testing](#_vvjzliusqvvy) 3

[Testing Schedule](#_w4hfmna0954j) 3

[Release Criteria](#_kalnxnceu8ci) 3

[**Unit Testing**](#_xcdlztkcawhp) **3**

[Client](#_kyq8tnhal1g1) 4

[Client Initializer](#_dx28m3y0ctpx) 4

[Client Handler](#_hg7vjook0dxm) 4

[Server](#_k01umip560mr) 4

[Server Initializer](#_7hbf1ho6kmc0) 4

[Server Handler](#_ke1qla1ru6pw) 4

[API/Chat history](#_7wv9tebpu9i9) 4

[Login](#_kjgdplnetixt) 5

[**Integration Testing**](#_gfrp9xy6r0cj) **5**

[User Interface](#_mel0wvl5fsb4) 5

[Conversation Functionality](#_5sk5ezuyj66c) 5

[Chat history](#_u32ob812jzni) 5

**Introduction**

Hello and welcome to the Chatterbox Testplan. This is a document to outline what tests we will perform on the Chatterbox app to make sure it doesn’t break. It’s essential to test and have a test plan for a variety of reasons. We need to test individual pieces of code to prove each of them run on their own, and also test to see how all of pieces interact in a variety of scenarios across the application.

## Purpose and Scope

The Purpose of this document is to inform the reader of the test document all of the tests we will perform on the product to ensure it is up to standards. We will discuss the different types of testing I will perform, as well as lay out the individual tests the program needs to pass.

## Target Audience

The Target audience for this document are the stakeholders: The Professor, the Grader and I. If this were not a school project, a quality assurance team and project manager might also be a relevant audience for this document.

## Terms and Definitions

Unit Testing tests individual methods. Validation Testing tests integrated/system wide behaviors.

# Test Plan Description

The test plan description outlines the big picture aspects of testing, such as what is in and out of scope, what pace I will be testing, the schedule of tests I will be using, what errors are so egregious that I can not ship with them in, and what requirements are so essential, I can not ship without them.

## Scope of Testing

Before I define what is in scope, I want to take a moment to talk about what is out of scope. I will not be testing or troubleshooting Netty, Protobuf or any other 3rd party tool. I will be using the most stable versions and testing to see if the implementation works, but I won’t be contributing to the functionality of those products. I will assume anything in the java standard library probably works.

The area’s of the project I will be testing are ones which check core functionality based on code I have written. I plan to prioritize tests on the most essential aspects (the ability to send and receive messages) which will get many test and test features not necessary for core functionality (saving chat history) using integration testing as they are added.

## Testing Schedule

I plan to test everything before I turn in the final project.

## Release Criteria

Maximum fault tolerance: As this program doesn’t have lives on the line, I will not need to mathematically prove each piece of code. I am also willing to ship with some features not fully implemented (better to have a minimum viable product then none at all.) If some auxiliary feature is not fully implemented, or fully tested at launch (such as chat history), that is a fault I’m willing to tolerate. Before being deployed, my goal is for Chatterbox to pass all of the unit and integration tests.

# Unit Testing

In this section, I will be laying out specific methods in my system which require testing and how I will know if the test is successful.

## Client

This is the core functionality of the program and so testing all of the individual functions that make this happen is key

#### Client Initializer

Both the client and server initializer throw errors when they do not work. However, there are a few specific cases which I want to test to make sure they work well

* Check to see if you connect to server.
* Check to see if chat history loads.

#### Client Handler

* When a user types something to the public chat, everyone receives a message, other than the user who sent it.
* Test 10, 100, 1000 and 10,000 character blocks to stress test.
* Test public chat with 10 or 100 people.

## Server

#### Server Initializer

* Server runs with just one client attached
  + Person can chat in public chat, and it’s just them

#### Server Handler

* Server runs with 10 clients attached
* Server runs with 0 clients attached

## API/Chat history

* Ensure that chat history can be stored
* Ensure that chat history can be retrieved
* Ensure that chat history matches what you stored when you retrieve

## Login

* Check that logging in allows the user to access the client
* Check that login data pulls up the correct search history.
* Check that an incorrect password has an error message.

# Integration Testing

This is about taking features which are added later in the process and seeing how they work well with the rest of the code. This describes system wide behaviours and interactions

## User Interface

* Make sure the button to switch from one on one chat to public chat switches chat
* Make sure the send button sends
* Make sure the login screen takes the username and the password in the correct boxes.
* Make sure the users can’t send nothing (enter key doesn’t send unless there is at least a space in the box)
* Ensure the users can’t overflow the input buffer
* Ensure that the ability to switch from one private message to another private message with a different user works.

## Conversation Functionality

* Stress test many (10, 100) users, chatting with many characters (100, 1000).
* Try a sample conversation.
* Switch between several private conversations and public chat.
* Login, chat and logout.

## Chat history

* Ensure a user can see the most recent message they sent.
* Ensure that up to the last 10 messages in the conversation on either side is stored.
* Ensure that the last 10 messages of group chat show up.