Logan Miller

Term Project: *LCAP*

Test Plan Document

Table of Contents

Introduction

This document prescribes the testing that will be conducted on LCAP in order to ensure it satisfactorily meets the necessary requirements and that the modules are able to run bug free. Within this document, readers will find a overarching description of the testing to take place as well as a robust description of the unit and integration testing to take place.

Purpose and Scope

The purpose of this document is to describe a systematic plan for both individual functionality testing and for integrated testing for the LCAP. This document will identify the modules to be tested, the order in which the testing will conducted, and how all testing will be conducted. The core modules of the program, the server, the client, the message manager, and user manager, will all be tested.

Target Audience

This document is prescribed for managers and software engineers who will use the document as a blueprint for testing the software's capabilities in order to ensure it meets the minimum requirements put forth, as well as for the stakeholders, Professor Xie and Bin Lin, in order to ensure the software is being tested to the full extent.

Terms and Definitions

- Client (User) - Any person who creates an account on the application in order to communicate with other users
- Global Chat - The stream of inputs that all online users are able to view
- LCAP - Logan's Chat Application
- Private Chat - The stream of inputs between two distinct users
- Receiver - A client that receives a message through either the public or private chat
- Sender - A client that sends a message through either the public or private chat
- Server - The actor that will manage message traffic as well as user privileges

- UI - The user interface, the interface the user will use in order to engage with LCAP

Test Plan Description

The purpose of this section is to prescribe the scope of the testing that will take place as well as to define the schedule in which the order of testing will be prescribed.

Scope of Testing

Testing of the program will be conducted in two parts. The first of which will be a black-box style approach in testing individual modules. The second will be for testing the aggregate parts of the software in order to ensure it all functionality is present and bug free when working in unison. The server, the client, the message manager and the user manager will all be tested.

Testing Schedule

The unit testing will be conducted daily, as well as upon the completion of any new functions. Integration testing will be conducted on 6/4/2017, and then daily, in order to identify problems requiring attention and re-coding or redesign.

Release Criteria

The program should be able to meet the minimum system requirements prescribed by the stakeholders. The maximum fault tolerance allowed will be 90%. Modules must be able to function without crashing the program before LCAP can be deployed.

Unit Testing

The purpose of this section is to identify individual artifacts for testing and prescribe how they they will be tested. These tests will identify any faults in the module's functionality and establish a baseline for debugging or redesign.

Server Module

The server class handles connections between users as well as fetching and supplying requested data. For this reason no functions will be be given in depth testing. The functionality of the server is purely for establishing connections and shifting data. All function will be tested individually as implemented and then the server will be tested thoroughly during integration testing. All checks for improper data inputs will be done at the client and manager levels. The minimum criteria for release is the server establishing a connection and relaying data between users.

Client Module

The client class acts as the sole interface through which a user interacts with the server and other clients. Functions that will not be tested thoroughly: getChatHistory, getGlobalHistory, disconnect, and displayMessage. These functions require single input and only allow for one interaction with the user and will therefore be tested during implementation. Functions to test thoroughly are the login and createAccount functions as they have potential for multiple errors. the minimum criteria for release is the user being able to create an account, log in and initiate chat with other users without disconnecting and with all message being displayed clearly.

Test Case 1: login

User attempts to login with a given username and password. If username longer than 20 characters is supplied an error message is displayed.

Test Case 1: login to an already online account

User attempts to login to an account with an account already online. No connection should be established, an error message is displayed to the user.

Test Case 3: Attempt to login with no input

User attempts to login without giving any input. Client should display proper error message.

Test Case 4: Attempt to create an account

User attempts to create an account with a given username and password as input. If username or password are longer than 20 characters an error message is displayed.

Test Case 5: Attempt to create an account already in existence

User attempts to create an account that is already in existence. An error message is displayed.

Test Case 6: Attempt to create an account with special characters

The user attempts to create an account using special characters used for parsing file names. An error message is displayed stating those characters may not be used.

Message Manager Module

The messageManager class acts as the custodian of all user's message histories. All communication with the messageManager is done through the server. Functions to be tested are saveToFile, checkForFile, and displayConversation. displayGlobal and createFile is omitted from testing as it has one input and one output and can be tested during implementation. The minimum criteria for release is the message manager correctly saving and creating user messages in the corresponding files.

Test Case 1: saveToFile

The messageManager attempts to save a message to a file given two inputs, the name of both users and the message content.

Test Case 2: saveToFile with empty strings

The messageManager receives a message with no input, the messageManager will halt the saving of data, empty strings are not allowed to be saved.

Test Case 3: checkForFile

The messageManager checks for a file given two user names, should valid input be given, it finds the correct file with no errors.

Test Case 4: checkForFile with empty usernames

The messageManager is prompted to check for a file without being given either username or one username is omitted from the input. The function immediately returns an error message to the server.

Test Case 5: displayConversation given the same username twice

The function receives the same username twice, implying the server wishes to fetch a conversation history between a user and himself. No history is saved, nor is the user allowed to have a conversation with himself. The function returns an error statement.

Client Manager Module

The userManager class acts of as the custodian of all user's account information. All communication with the messageManager is done through the server. Functions to be tested are: saveToFile, checkUsername, and checkPassword. setStatus and getStatus are omitted from testing as they require only one input and supply one output therefore they will be tested during implementation. The minimum criteria for release is the client manager correctly saving and creating user account files.

Test Case 1: saveToFile with empty username

The functions receives no username when prompted to save a message. An error message is sent back to the server stating no username given.

Test Case 2: checkUsername with empty username

The function receives a request to check for a username, but no username given as input. An error message is sent back to the server stating no username given.

Test Case 3: checkPassword with empty password

The function receives a request to check a password, but no password is given. An error message is sent back to the server stating no username given.

Integration Testing

Describe the purpose of this section and outline its contents. Only a few sentences are expected here. The purpose of this section is to outline the testing of aggregate modules in how they will interact with each other. This round of testing will determine the modules ability to communicate with each other and identify any deficits requiring debugging or redesigning.

Normal Operation

The first test case describes the scenario in which the server is started, a user logs in to an account, a user sends a message, the user logs out, and the server is shut down. This will ensure all modules are able to communicate with one another in a simulation of how LCAP will run.

Unexpected Server Disconnect

The second test case describes the scenario in which the server is shut down unexpectedly while users are communicating. During this test, two users will be connected and be communicating. The server will be manually shut down to sever the user's connection. A error message of lost connection will be displayed to the users.

Unexpected User Disconnect

The third test case describes the scenario in which a user disconnects while in a private chat with another user. Two users will be connected to each other. One user will be manually disconnected during the conversation. The private chat will be closed with the user who has disconnect and an error message will be displayed to the user still online.