CS3505 - Multi-User Spreadsheet Desktop Application

Richard Timpson Jabrail Ahmed Tyler Brewster Conner Grimes
Zack Muhlestein Peter Forsling

March 6, 2019

Contents

1 Introduction

The following document outlines a communications protocol for a client-server multi-user spreadsheet desktop application. The following protocol gives no information for either technologies used to implement the application on the client or server side, nor any details as to how the applications should be implemented. The protocol is split into two aspects, logging into the server and choosing a spreadsheet to edit, and editing a spreadsheet. All of the data transfer between client and server will happen using TCP and Sockets, and the data sent back and forth will be JSON.

1.1 Overview and Technologies

Put some more specific information here about the technologies

1.2 Connection Overview

Give an overview about the connection

1.3 Editing Overview

Give an overview about editing

2 Connection

2.1 UML Diagram

Figure 1 shows what the initial connection of the client to the server looks like as a process. The initial connection is broken up into two different phases; logging into the server and choosing a spreadsheet to edit.

2.2 Socket Connection

Before a user logs into the server, there must be an initial connection from the client to server so that they can begin communicating back and forth with each other. The connection will be made using standard networking practices with a socket connection built over TCP/IP. On the initial start of the server, it should begin listening for socket connections at whatever IP address and port are sufficient. It is important that the client knows what specific IP address and port number to connect to. With this information, the client should complete the socket connection with the server. If there are any errors with the connection, TCP should report them and both the client and server can handle the exceptions accordingly. If the connection is successful, the socket should stay connected to begin the transfer of data.

2.3 Loggin In

Once the connection has been made, the server will expect the client, at some point in time, to send the information for either logging in, or creating a new user. This data will be sent as a JSON string terminated by a “n” character, as was discussed in the introduction (need to make sure to talk about our data representation in the beginning of the document). Because the user can either log in or

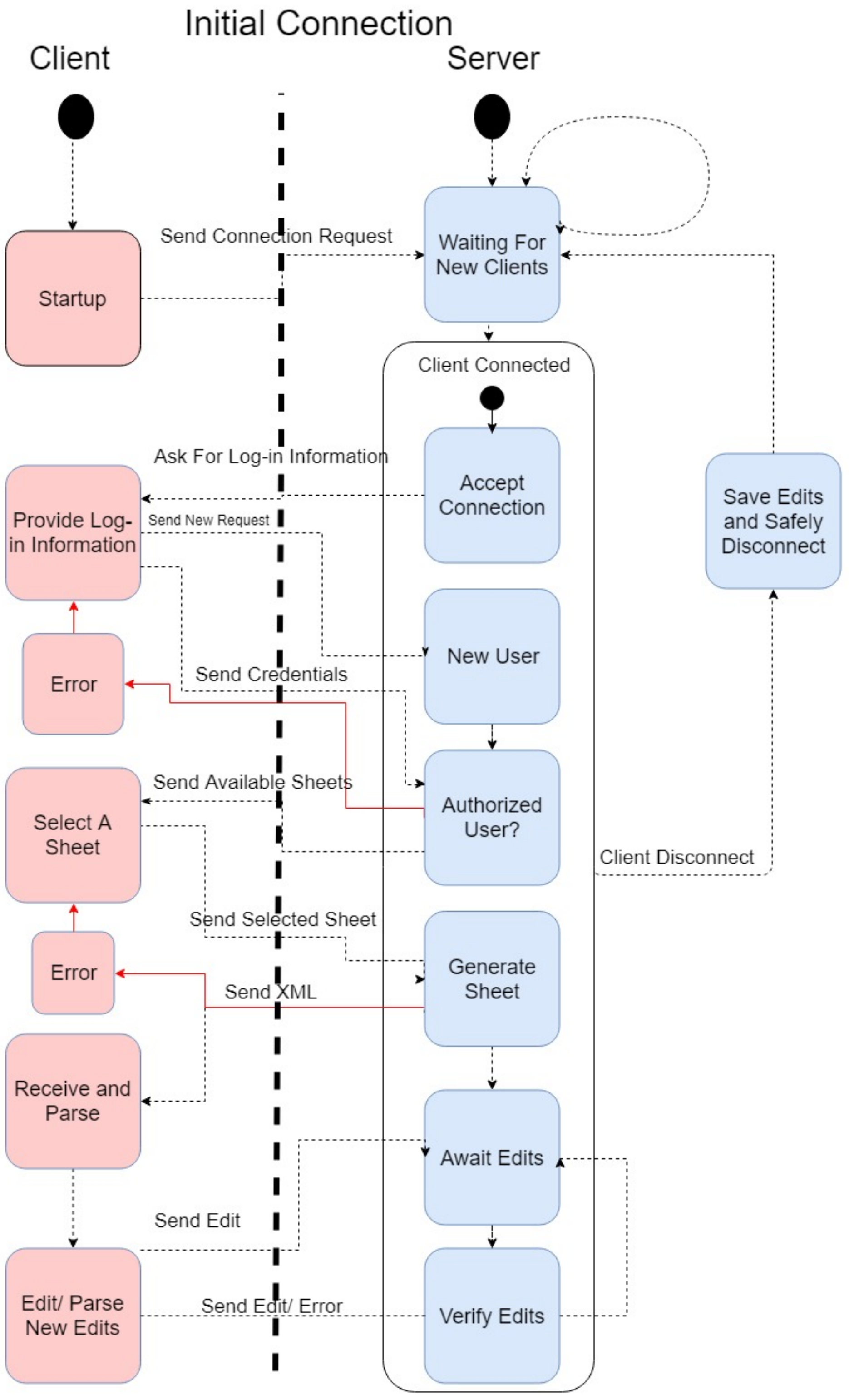

Figure 1: The UML diagram for the initial connection

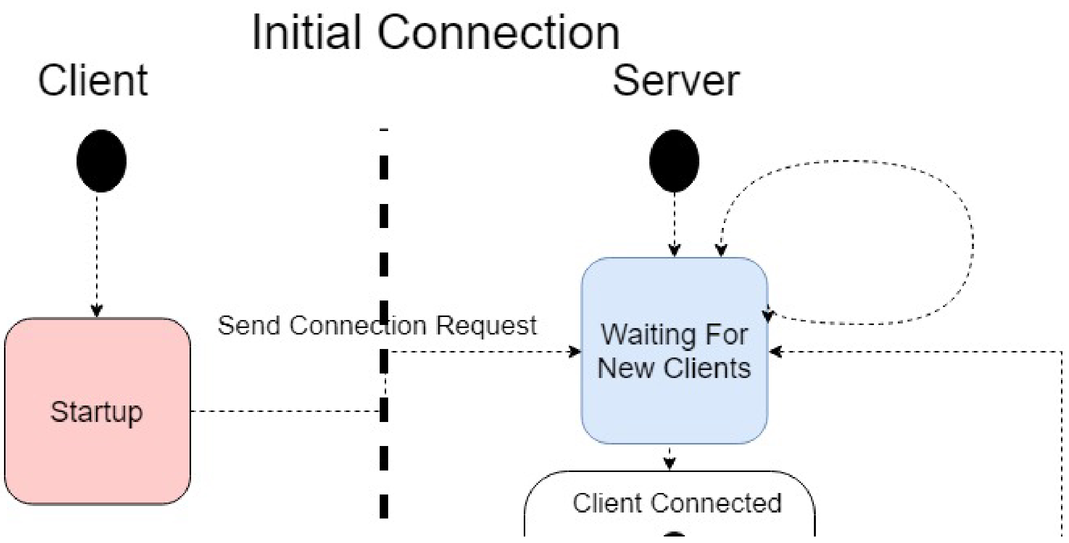

Figure 2: The socket connection in 1

create a new user, there will be two different types of strings. For logging in, it will be formatted like so.

```
{
    "name": "the users name",
    "password": "the users password",
}
```

If the client wants to create a new user, the string will formatted as such.

```
{
    "new_user": "true",
    "name": "the users name",
    "password": "the users password",
}
```

It is the responsibility of the server to parse either string and perform the correct functionality based on that string, such as saving or checking the username and password. The server then needs to send a response back to the client, letting them know if the client successfully logged in or created a new user. If for any reason the server could not validate the credentials, it should send an error message back to the client so that it knows that it failed to connect. The format for the success message is as follows...

```
{
```

```
            "success": "a boolean value that is true if successful, false if not",
            "message": "a message to the client letting them know what the error was
                if there was one, or if the user was successfully verified or created"
        }
```

If the client failed to log in, the server will expect another login message from the client, and will continue in this state until it receives valid login credentials.

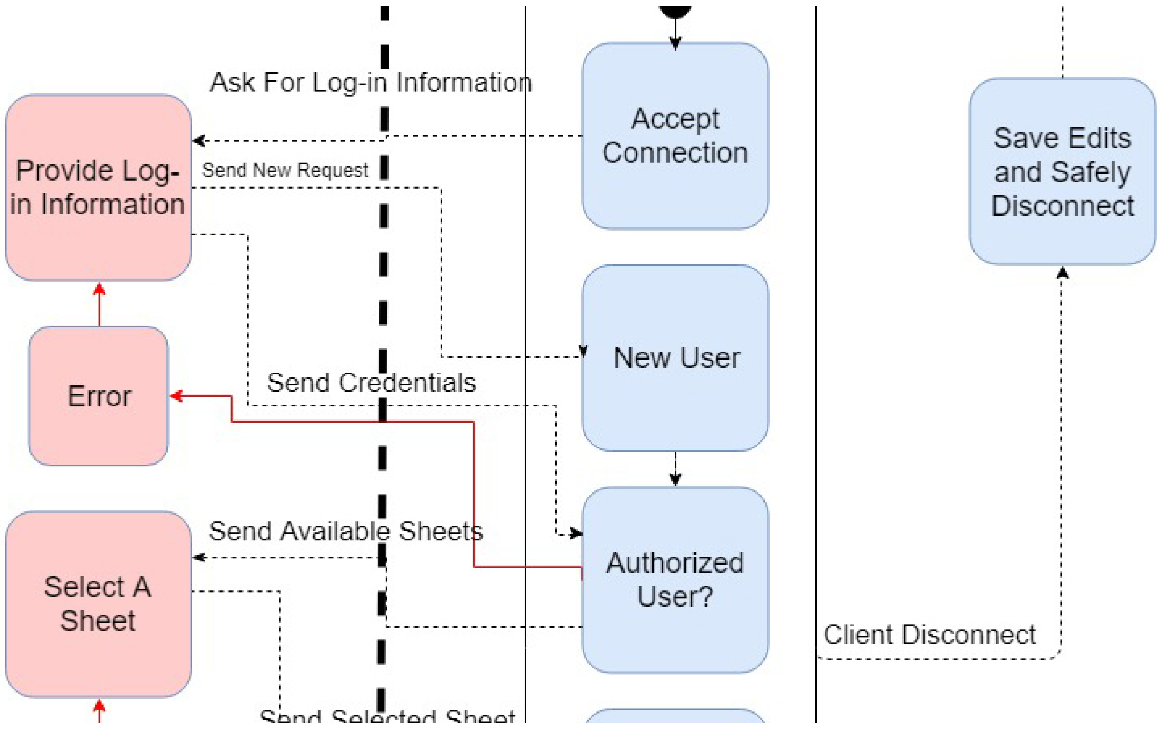

Figure 3: Logging in in figure 1

2.4 Picking a Spreadsheet

If the client did send valid login credentials, immediately after sending a success message, the server will send a JSON string containing an array of objects that contain information of the spreadsheets on that server that the user can access. The following string represents one valid spreadsheet object.

```
        {
            "name": "the spreadsheets name",
            "id": "the spreadsheet id",
            "last_edit_date": "a date time  string in the format dd-mm-yyyy hh:mm:ss",
        }
```

The client then needs to send back to the server information about the spreadsheet it wants to select. As with logging in, the client can send two options for spreadsheet selection; either picking an existing spreadsheet or creating a new one. If the client wants to create a new spreadsheet, it need only send the name of the spreadsheet that it wants to create, like so.

```
        {
            "name": "name of the spreadsheet",
        }
```

If it wants to pick an existing spreadsheet, it should send the id of the spreadsheet that it would like.

```
        {
            "name": "name of the spreadsheet",
        }
```

Again the server needs to send a message back to the client letting it know if it was successfully able to validate a spreadsheet for the client to edit. The JSON string should be the same as it is with logging in.

```
        {
            "success": "a boolean value that is true if successful, false if not",
            "message": "a message to the client letting them know what the error was
                if there was one, or if the user was successfully verified or created"
        }
```

Once the client has successfully chosen a spreadsheet to edit, the editing process will begin with the server sending all of the cells containing values as edits.

3 Editing

3.1 Generating a saved spreadsheet

The server first sends the function code 0 followed by a JSON array containing all of the active users. Next, the server sends all of the non-empty cells as basic edits, following which it will send the locations of all other users for the client to display.

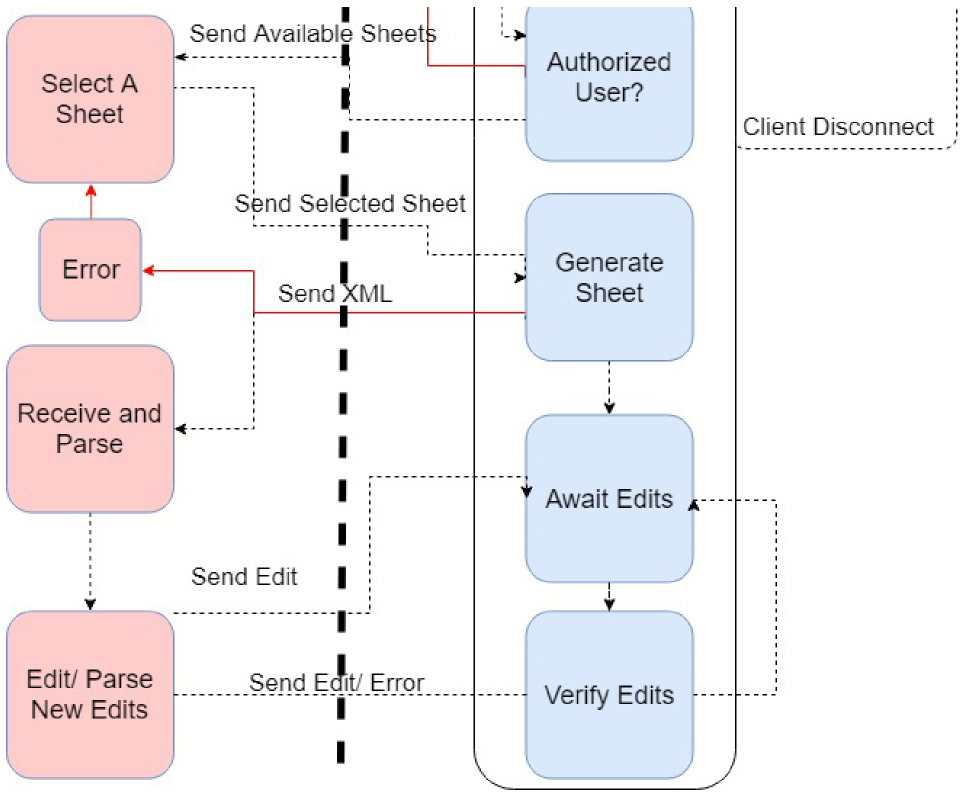

Figure 4: Logging in in figure 1

This is an example of the start up message from the server to the client:

```
{
    "cell": "two characters i.e a1",
    "users_active": ["userID1", "userID2", etc.],
}
```

4 Conclusion