Software Testing Document

for Student Attention Tracking

Software Engineering

Fall 2018

Adam Abdellaoui

Tony Andrews  
Philip Fernandez

Austin Lambeth

Bronson Lane

Anthony Shedlowski

1. Introduction……………………………………………………………………………2
   1. Background…………………………………………………………………...2
   2. Objectives……………………………………………………………………..2
2. Features to be Tested………………………………………………………………..3
3. Test Approach………………………………………………………………………...4
   1. Environment Requirements/Needs…………………………………………4
   2. Approach………………………………………………………………………4
4. Test Cases………………………………………………………………………….....5
   1. Unit/Input Testing……………………………………………………………..5
   2. Integration/Server Testing…………………………………………………...8
5. Summary……………………………………………………………………………...11
6. References/Definitions………………………………………………………………12
7. **Introduction**
   1. **Introduction:**

The purpose of the tobii eye tracker is to track eye movement from students in a classroom setting. We want to be able to allow a teacher to track eye movement throughout a class in real time, but also give them the ability to go back and look at past lectures. Teachers will also be able to see which applications students have opened during class as well. This document applies to the testing of the attention tracking subsystems and the system as a whole.

* 1. **Objectives:**

The attention tracking system is comprised of two main parts, the transfer of data from students to teachers via the backend server and the teacher’s interface. We will be testing the subsystems.

* Testing that the server receives data as well as transmits it.
* Test the subsystem’s connections and access to the database
* Test the functionality of the UI for the teacher and have it pull data from from the database for verification.
* Test that the Admin can update the database correctly
* Test that the the teacher UI is receiving data from the eye trackers via the server

1. **Features to be tested:**

|  |  |
| --- | --- |
| **Features:** | **Expected Results:** |
| GUI | Allows end user to manipulate and view data. |
| Database | Stores and receives data of students and teachers usernames and passwords, requiring verification to authenticate user. |
| System Server | Handles connection of Teacher and Student applications. |
| Teacher Interface | Selects desired course. Sees currently connected students, a heatmap of where students attention is fixated, currently opened applications. |
| Student Application | Connects and sends data to the server |
| Administrator Controls | Creates courses, teachers, and students and manages entries in database. |
| Login | Allows teachers and administrators to sign into the application using username and password for verification. |

1. **Test Approach:**

* 1. **Environment Requirements/Needs**

The system is comprised of at least three instances, students, teachers, and the server. The instances need to be on seperate machines as they each have a specific port assigned to them. This requires separate machines as the port can only be bound to one process/machine at a time. These machines are also required to be on the same local network so that they can properly communicate and connect. The server is written in Java, so the instance running the server will need a JRE version 8 or newer . The students and teacher instances are written in C# and Windows forms, and they will require Windows 7 or newer.

* 1. **Approach**

Our approach to testing this software will include integration testing, and full system testing.

The unit testing will include testing the teacher interface inputs. This will include button testing, and valid input. This will also test SQL injections, as the interface subsystem has direct access to the database.

Integration testing will test the integration of the subsystems with the server and the communication with one another. If the database can be accessed and updated with new information. These test will check if the server is receiving the information in the correct format as well as sending in the correct format.

The full system test sees if the system works as intended. This will test the overall goal of the system, which is for the system to transfer data in real time to the correct teacher. The system correctly connect students to the correct teacher and connect/disconnect students when lectures are over and new ones begin.

1. **Test Cases**
   1. **Unit/Input Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Scenario** | **Test Steps** | **Test Data** | **Expected Results** | **Pass/**  **Fail** |
| Check Teacher Login | 1.Open app  2.Enter username  3.Enter password  4.Click submit | Username  Password | Teacher goes to the course selection screen |  |
| Check Teacher Login Invalid data | 1.Open app  2.Enter invalid username/password or both  4.Click submit | Invalid Username/  Password | Teacher will not be logged in and will receive an alert |  |
| Check Login for SQL injection | 1.Open app  2.Enter SQL injection string into the text box  4.Click submit | SQL injection string | The SQL injection will be denied and the teacher will not be logged in |  |
| Check Admin Login | 1.Open app  2.Enter username  3.Enter password  4.Click submit | Username  Password | Admin is logged into the Admin screen |  |
| Check Admin Login Invalid data | 1.Open app  2.Enter invalid username/password or both  4.Click submit | Invalid Username/  Password | Admin will not be logged in and will receive an alert |  |
| Teacher Chooses Class | 1.Teacher clicks the class they want | courseName | Teacher is logged in to the class |  |
| Teacher stops and starts recording data | 1. When teacher logs in it starts recording  2. When teacher exits the UI it stops recording | Press start/stop button | 1. Teacher starts receiving live data from the the eye tracker  2. The eye tracker stops sending data |  |
| Teacher views data from a specific student | 1. Teacher selects the student that they wanna see | studentName | Teacher sees data only from that student |  |
| Admin adds a teacher to a class | 1.Admin enters Teacher’s name  2.Admin enters the class name  3.Admin selects add | teacherName  className | Teacher is added to the selected class |  |
| Admin adds a student to a class | 1.Admin enters Student’s name  2.Admin enters the class name  3.Admin selects add | studentName  className | Student is added to the correct course in the database |  |
| Admin removes a student from a class | 1.Selects student from class, edits by clicking on and removing. | 1.Click remove 2.Remove student from Class A | Student is removed from the course in the database |  |
| Admin removes a teacher from a class | 1.Selects teacher from class.  2. Removes teacher from class | teacherName  className | Teacher is removed from the course in the database |  |
| Admin changes a teacher’s class | 1.Admin selects teacher  2.Admin edits course for teacher. | teacherName  className | The teacher is removed from one course in the database and added to a new course in the database |  |
| Admin changes a student’s class | 1.Admin selects student  2.Admin edits course for student. | Click student  Change Class A  to Class B | The student is removed from one course in the database and added to a new course in the database |  |

* 1. **Integration/Server Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Scenario** | **Test Steps** | **Test Data** | **Expected Results** | **Pass/**  **Fail** |
| Server receives a request for a connection from a student | 1.While server is running, a student tries to connect  2.Student application sends username | Student’s username and IP address | The server accepts the connection and creates a new thread to handle the connection |  |
| Server receives a request for a connection from a teacher | 1.While server is running, a student tries to connect  2.Teacher sends username and course number | Teacher username and course number | The server accepts the connection and creates a new thread to handle connection |  |
| Server doesn't receive anything from a client and their ‘heartbeat’ counter equals 0 | A client, teacher or student, disconnect from the server, and their ‘heartbeat’ counter reaches 0 | No transmission ‘heartbeat’ equals 0 | The server stops the thread and connection to that client, and removes them from the list of connected clients |  |
| New lecture is streaming on the server, after a teacher connects. | A teacher is connected and wants to stream a lecture | Teacher course number and the list of connected students | The server determines a port to stream on and sends a command to all involved (students and teacher) in the lecture to change to the selected port |  |
| Teacher stops streaming and disconnects from the server | 1. The teacher decides to stop streaming and either disconnects or closes their interface  2.The server ‘senses’ the disconnect | the disconnect | The server will send a command to all students currently in the lecture and sends them a command to change to the default port |  |
| A student connects after the teacher is already streaming | 1.Student connects to the server and the server has already issued a command to all students to change their port | New student connection | The server figures out what their current or upcoming class is and matches them to a current streaming lecture and issues them a command with the corresponding port |  |
| Teacher views historical data | 1.Teacher is able to view record data of students time within a class once connected to eye tracker. | Historical data | 1. Teacher can go to a previous class time and be able to see all the data from that class |  |
| Server receives Information from student | 1. Student is set to default UDP port  2. If Teacher is logged in student info is sent to correct UDP port | Student connection | Student’s UDP port is changed to the correct port value and sent to the server |  |
| Server receives malformed information | 1.The server doesn’t fully receive a transmission or packet. | Only part of a data transmission | The server should drop the current information and ask the client for a retransmission |  |
| Server detects Teacher login | 1.Server receives login information for the teacher  2.Server receives the teacher’s course | Login info  Teacher course | All students in that class are updated to the correct UDP port |  |
| Server detects teacher disconnect | 1. Server receives disconnect from Teacher | Teacher disconnect | All students connected to the class are updated to the default port |  |

1. **Summary**

This document lays out our expectations and goals in regards to how the student attention tracking software will work and how it is to be tested. It explains the requirements for the system to successfully run and track data and our approach to testing the system as a whole.

The test cases listed above cover both the inputs received by the system from the teacher and the admin as well as the integration of the server to the different subsystems. They represent the items that we think need to be tested at this point. If during the testing we see other items that need to be covered they will be added to the test cases.

1. **References/Definitions**

Server - Java package that contains the classes necessary to run the ‘Server’. It is also the backend. It that handles the data transfer from students to teachers

Teacher (Teacher interface) - A teacher that wants to stream a current lecture or view historically data from a previous class

Student (Student Application) - eye tracking software running on the student computer that sends data to the Teacher interface

Admin - An elevated teacher login, that allows for database management

Default port - I port that the server doesn't listen on that helps identity which students are not currently streaming for a lecture, but also doesn’t stop them from streaming data.

Class (Lecture) - a class that is currently in session that could be streamed to a teacher’s interface.