Classroom Attention Tracking

Software Engineering

Dr. Andrew Allen

CSCI 5530-A

Team 4

Adam Abdellaoui

Tony Andrews

Philip Fernandez

Austin Lambeth

Bronson Lane

Anthony Shedlowski

10/8/2018

Abstract

When a teacher is giving a lecture or trying to get a point across, one of the key aspects that will make them more successful is knowing if the students are engaged and paying attention. That’s where this system comes into play. It will track the students eyes and other information such as what tabs students have open in order to give the teacher a better idea of how well students in the class are paying attention. This system will give teachers a way of filtering the data to better understand how students are engaging with their class. It will also allow the teacher to retrospectively view specific lectures and see where the attention of the class, or a particular student, was at any point in time.

Table of Contents

1. Introduction
   1. Purpose of system
   2. Design methodology used
   3. Overview of document
2. Proposed Software Architecture
   1. Overview of the system architecture.
   2. Subsystem details and relations
3. Data Flow and Database Design Design
   1. Data storage
   2. Data flow
   3. Database design and layout
4. User interfaces
   1. Teacher interface
   2. Student interface
   3. Admin interface
   4. Filter API
5. Glossary
   1. Users
   2. Apache Storm
6. Appendix
   1. Appendix A – Use case diagram
   2. Appendix B – Use cases with nonfunctional requirements
   3. Appendix C – Logical Diagram
7. **Introduction**
   1. Purpose of the System

This system has a goal of streaming data from multiple Tobii Eye Trackers in order to track the attentiveness of students in a classroom. This data will be stored data in a database and streamed to a display in a teacher application. This data includes where the student is currently looking and other data such as currently opened windows. This should allow for the teacher to better determine if their students are currently engaged in the lecture or not.

* 1. Designed Methodology Used

The software model used will be Kanban whose purpose is to limit the amount of inventory being used in the system, such as making sure that the data does not overflow and be too much to handle. The models used used are the used case diagram for the actors such as the teacher, the student, and the administrator. The model of the erwin database model is used to map out all the data for each actor, breaking down every piece of data belonging to each table. A logical diagram is used as well giving more details on the actions of the actors and how each state interacts with data going from state to state such as teacher or administrator having credentials sent to login the login accepting the request by verifying the actor or not.

* 1. Overview of Document

The document will give out details of the software design, explaining the system and what models, subsystems, interfaces, and terms used. It will explain the hardware and software to be used in the eye tracking software along with the subsystem decomposition, the layers describing the subsystem and identifying the use cases of each actor with subsystems used with each one. The software architecture splits the system into four parts, details of the subsystem and appendix, the login system which which will go through a process of verification, once verified it will allow the teacher to connect to the server and display the user interface, then the teacher should be able to view data from courses. Then there is the data flow and database design which maps out the table of data for each actor and subsystem created in the system.

The document gives details to the user interface, breaking down the functions for each actors interface. The use of a filter api which will help to filter data. A glossary with information on the actors purpose and terms, the terms and use Apache storm. The document has an appendix which gives examples of the use cases of the actors, it gives details on how each should function and gives a logical diagram that breaks down the process the actor goes through in the system to send and receive data to the server.

1. **Proposed Software Architecture**
   1. Overview

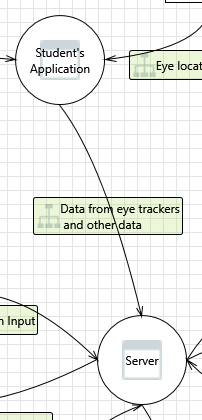
The Classroom Attention Tracker system is split up to 4 parts. These include the server, database, teacher interface, and student data handler.

The server controls the flow of information coming to and from it. It will handle receiving information from students, and transfer it to the correct teacher. The server will also write the data received to a database. It will be using the Apache Storm framework, which is a distributed realtime computation system. This will allow for parallel streams of data to come be sent in realtime and for the data to be sent to the teacher(s) and database simultaneously.

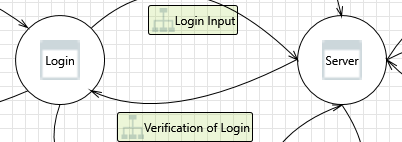
The teacher interface will receive student data from server. It will give the option to stream current data or look at historical data from previous lectures. The interface will use the chosen filter to display the student data. It will display the currently connected students, and where they’re currently looking.

The sub-system on the student’s computer, will start automatically on start-up. It will receive data from the eye tracking hardware and send it to the server. It also sends other information such as username, currently open applications, and etc.

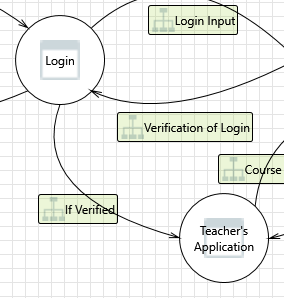
* 1. Subsystem Details and relations (Appendix C)
     1. The subsystem on the student’s side will make an initial connection with the server. This will include their username.It will continuously send eye tracking information to the server to be processed.



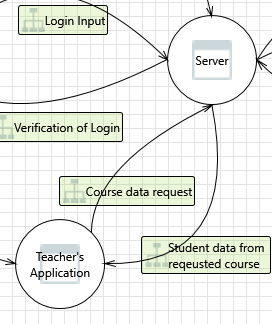
* + 1. The login system will take input from the teacher and pass it to the server. After the server verifies the input, it will tell the login if the input was verified or not.



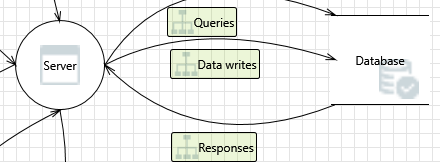
* + 1. If the input from the login was verified, then show the Teacher’s interface.



* + 1. Once the teacher’s interface is shown, the teacher can choose between viewing real-time data from the class, or look at recorded data from previous lectures.



* + 1. The server will read and write data from the database. It will store data as it comes in from students. It will also request data from the server as teachers request previously recorded data from previous lectures.



1. **Data Flow and Database Design Design**
   1. Data Storage
      1. Student data will be stored on a mysql database located on the teacher’s computer.
      2. The data gathered from the students will include eye position, student id, student name, their class, and a timestamp of when the information was gathered.
      3. Data will be stored indefinitely or until the admin decides it is no longer needed.
   2. Data Flow
      1. Student data will be sent from each eye tracker to the server.
      2. The server will then split the data using a bolt from apache storm sending one stream to the database and one to the teacher’s application.
   3. Database Design and Layout
      1. The Database will contain 4 tables student, teacher, data, and course.
      2. Student Table

This table has the student’s Id, first name, and last name. It is connected to the data table by a one to many relationship on the id key because one student may be connected to many pieces of data. It is connected to the course table by a many to many relationship.

* + 1. Teacher Table

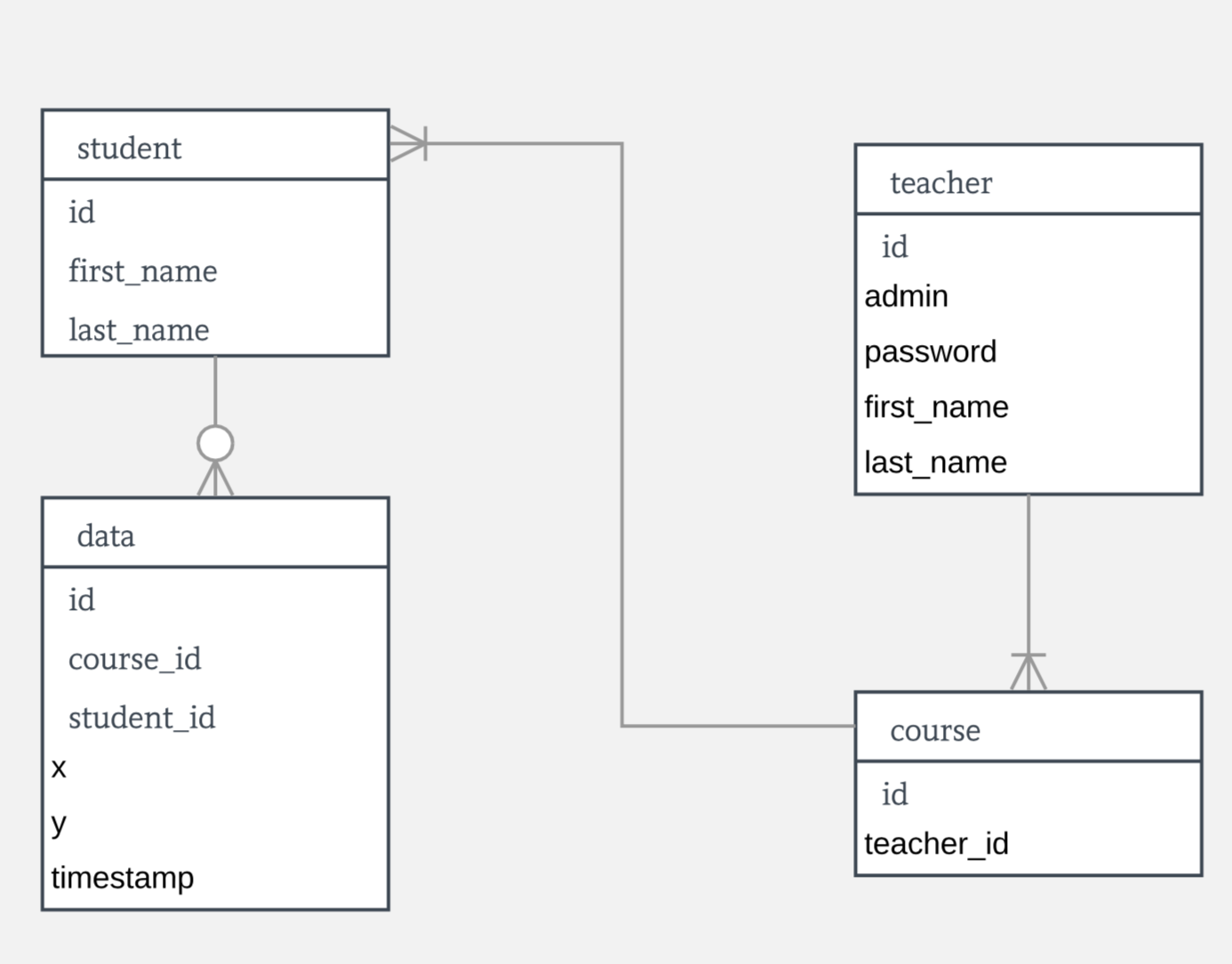
This table has the teacher’s id, a boolean variable named admin, a password used for logging in, first name, and last name. The admin variable is checked on log in to verify if a teacher has administrative privileges or not. The teacher table is connected to the course table by a one to many relationship because one teacher may teach many courses.

* + 1. Data Table

This table has an id, student id, course id, x position of eyes, y position of eyes, and a timestamp of when the data was captured. This table is connected to the student table by a many to one relationship. Each separate piece of data corresponds to one specific student.

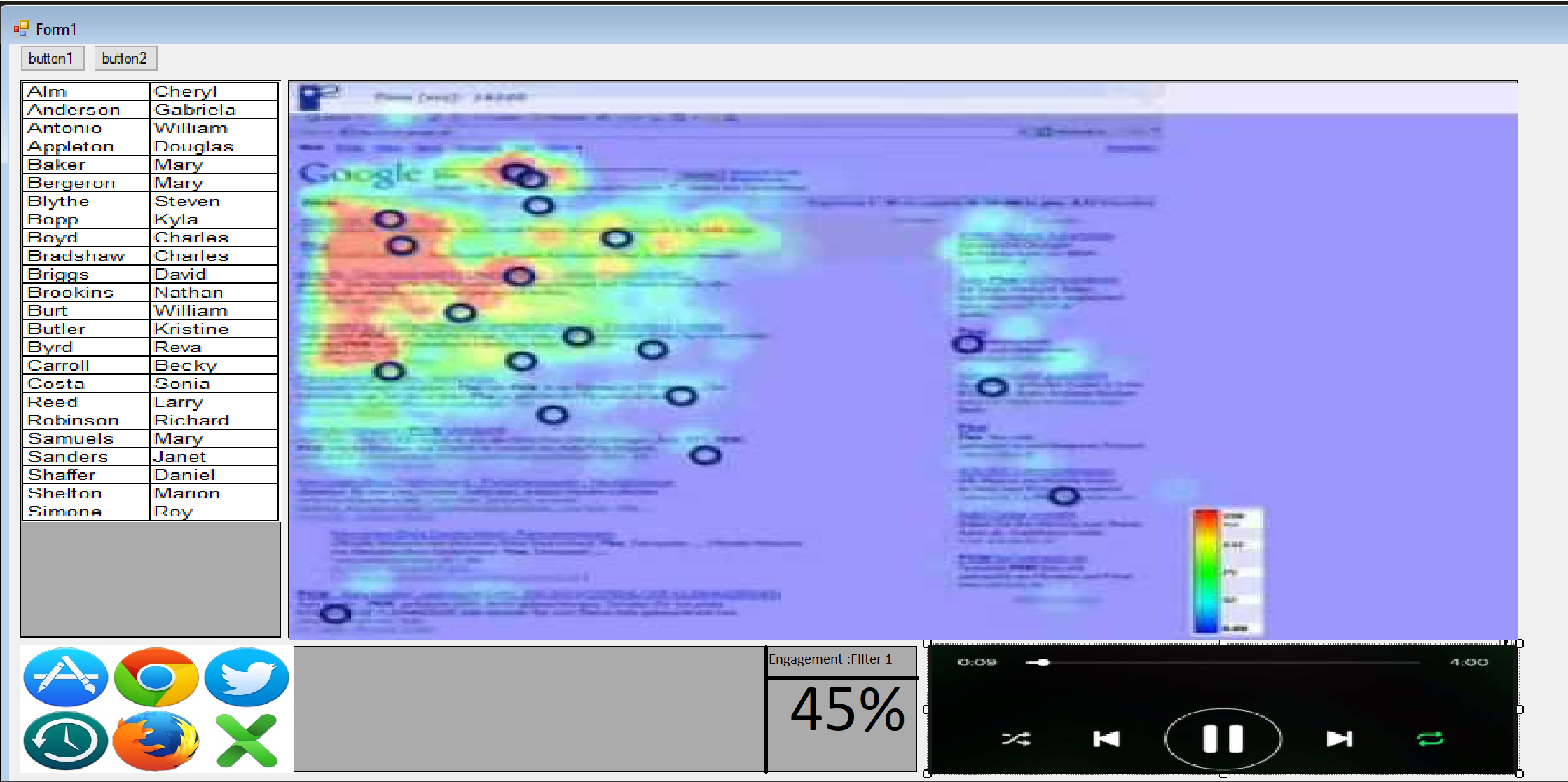
* + 1. Course Table

This table has an id and the teacher id. This table is connected to the teacher table by a many to one relationship because many courses may have the same teacher.



1. **User Interface**
   1. Teacher Interface

The Teacher interface displays the data sent from the collection server in a window. This window will have a filter applied to the data before it is displayed. Filters will be selected by the teacher and can be created through the Filter API. The left most side of the interface will display a list of the current students connected to the server. A list of the open applications on each screen will be available and details of who has what open will be shown on hover.



* 1. Student Interface

The Student will not have a direct interface with the application. All the necessary actions for the student’s side of the application will be handled automatically upon start of and login.

* 1. Admin Interface

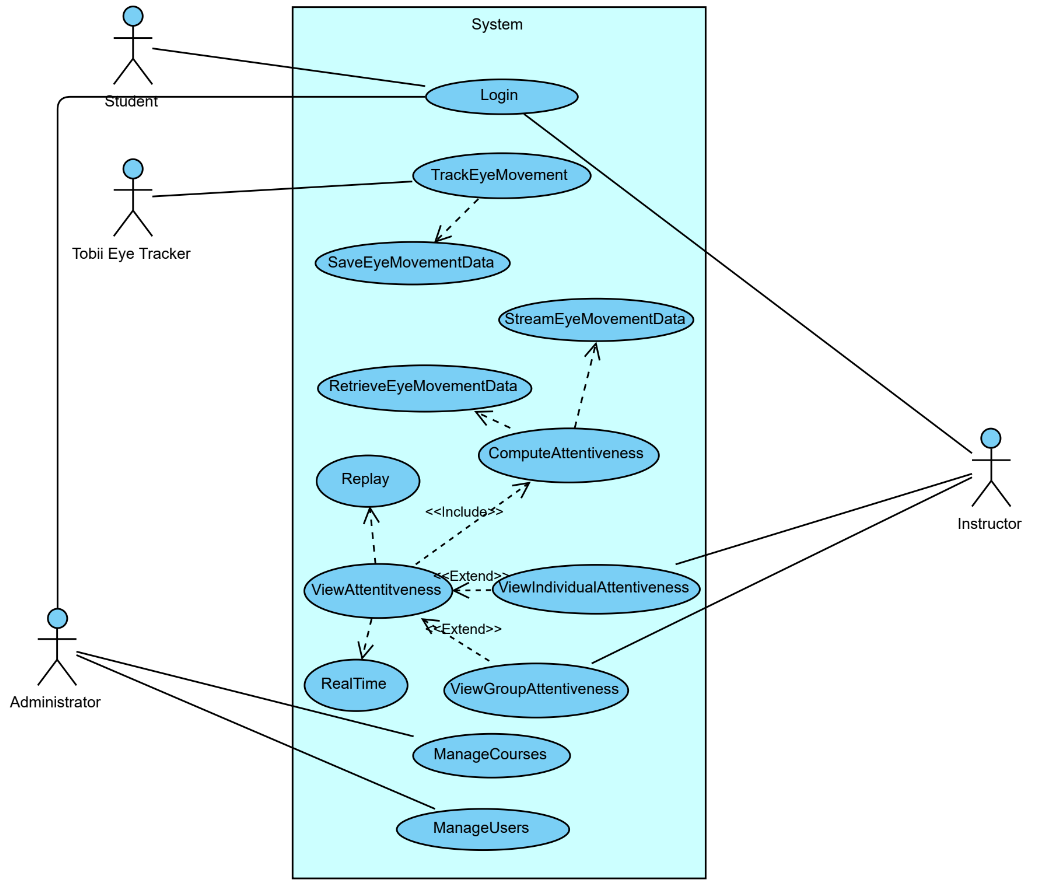
The Administrator will have to be able to configure the system before it is set up and be able to customize the configuration while the system is in use. To do this the administrator window will need to be able to specify where the server, database, and clients are located as well as the ability to set up classes. In setting up a class, the instructor, the time that the class occurs, and the student list will need to be entered. As students/teachers/classes are added and dropped, they will need to change the class setup while the class is still operating without altering the usability of already collected data.

* 1. Filter API

The ability to filter the data as it is sent to the teacher client will be through a filter API. The filter API will take in:student tag, class tag, x,y, timestamp,open applications. The API will export: x, y, intensity value.

1. **Glossary**
   1. Actors
      1. Teacher - person who views and interacts with the student data
      2. Student - person whose data is tracked, stored, and displayed
      3. Administrator (Admin) - person in charge of classes and student lists
   2. Apache Storm
      1. Bolt - it processes any number of input streams and produces any number of new output streams. Most of the logic of a computation goes into bolts, such as functions, filters, streaming joins, streaming aggregations, talking to databases, and so on.
      2. Spout - is a source of streams in a computation. Typically a spout reads from a queueing broker such as Kestrel, RabbitMQ, or Kafka, but a spout can also generate its own stream or read from somewhere
2. **Appendix**

**6.1. Appendix A – Use Case Diagram**

****

**6.2. Appendix B – Use Cases with Non-functional Requirements**

**1. Server**

5.1.1 The server controls the information coming from the students and going to the teachers.

5.1.2 The server receives the tracked eye data and other information from the students.

5.1.3 The server sends all of the data from the students and saves them to a database to be retrieved.

5.1.4 The server sends the data from the students to the the teacher per request.

**2. Student**

5.2.1 The student uses their school login to log in to the computer.

5.2.2 The software will pull the student’s username from the submitted login information.

5.2.3 The software automatically starts and tries to connect to the server.

5.2.4 After connecting, the system will start sending data to the server when requested.

**3. Teacher**

5.3.1 The teacher enters their username and password to log in to the application.

5.3.2 The teacher can select a class to view from a list of the classes they currently teach.

5.3.3 The teacher can choose between recording for a class or reviewing past recordings.

5.3.4 The application will retrieve the requested information from the server.

5.3.5 The teacher can select a filter for the data.

5.3.6 The teacher can select from a list of students to see individual data.

**4. Admin**

5.4.1 The admin enters their username and password to log in to the application.

5.4.2 The admin is authenticated and allowed access to the database.

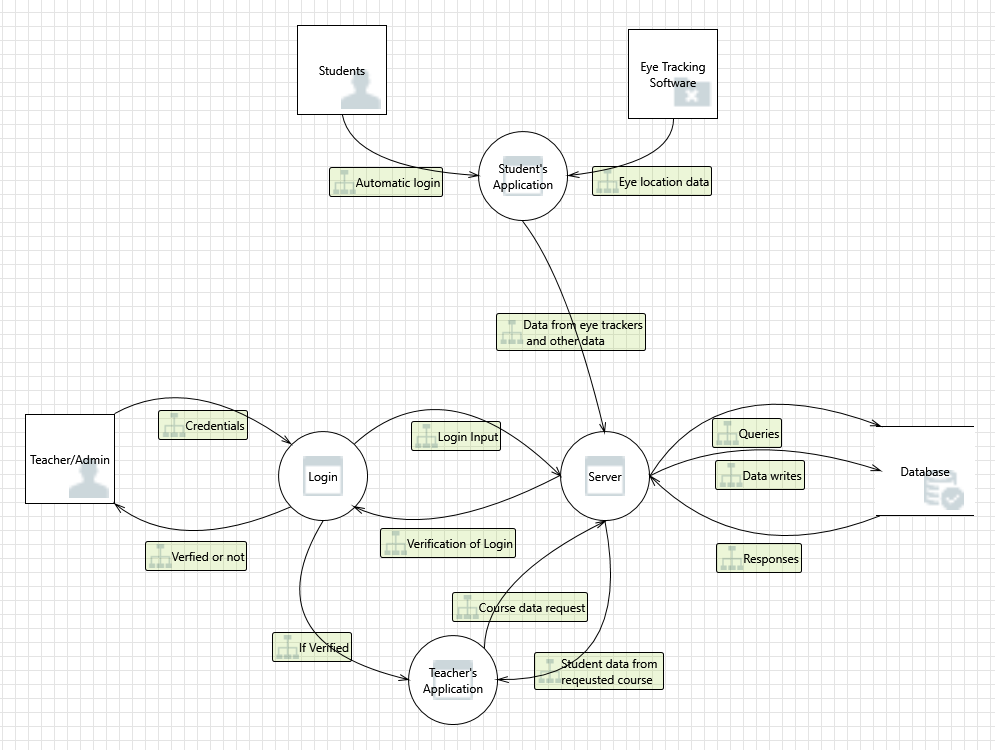
5.4.3 The admin can add or delete courses from a course list.

5.4.4 The admin can add or delete users from a user list.

5.4.5 The admin manages the database connection.

5.4.6 The admin can reset a teacher’s password.

**6.3. Appendix C - Logical Diagram**

****