Software System for Data Collection of EEG Input and Response Times During Mathematics Tutoring

A Manuscript Submitted to the

Graduate Faculty of the

Mathematical, Computing, and Information Sciences Department

Of Jacksonville State University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

With a Major in Computer Systems and Software Design

By

JASON EDWARD LEVI

Jacksonville, Alabama

April 7, 2014

ACKNOWLEDGMENTS

I would like to thank my wife, Dr. Tamara Levi for supporting me through my Master’s degree program. I would also like to thank my studio advisor, Dr. Plotnick, for her help and direction in creating this project. I would also like to thank the other members of my studio committee, Dr. Thornton and Dr. Ford. Lastly, I would like to thank Ms. Cynthia Jensen, the Graduate Assistant Supervisor for the opportunities and guidance she provided.

Contents

[1.0 Overview 3](#_Toc384136686)

[1.1 Purpose 3](#_Toc384136687)

[1.2 Scope 4](#_Toc384136688)

[1.3 Objective 4](#_Toc384136689)

[1.4 Assumptions and Constraints 4](#_Toc384136690)

[1.5 Project Deliverables 5](#_Toc384136691)

[1.0.6 Schedule and Budget Summary 5](#_Toc384136692)

[2.0 Literature Review 7](#_Toc384136693)

[2.1 EEG 7](#_Toc384136694)

[2.1.1 Attention Deficit Disorder/ Attention Deficit Hyperactivity Disorder 8](#_Toc384136695)

[2.1.2 EEG Research 8](#_Toc384136696)

[2.2 NeuroSky 9](#_Toc384136697)

[2.2.1 NeuroSky Research 9](#_Toc384136698)

[2.2.2 EEG Competitors 11](#_Toc384136699)

[2.3 Attention and Learning 11](#_Toc384136700)

[2.3.1 Research in selective attention 12](#_Toc384136701)

[2.3.1 Refocusing Student Attention 12](#_Toc384136702)

[2.4 Relaxation, Frustration and Learning 13](#_Toc384136703)

[2.4.1 Managing Frustration 14](#_Toc384136704)

[2.4.2 Mindfulness Training Research 14](#_Toc384136705)

[2.5 Tutoring and Learning 15](#_Toc384136706)

[2.5.1 Possible issues with tutoring 15](#_Toc384136707)

[2.5.2 Learning from human tutoring 17](#_Toc384136708)

[2.6 Conclusion 18](#_Toc384136709)

[3.0 References 77](#_Toc384136710)

[4.0 Software Requirement Specification 19](#_Toc384136711)

[4.1 Functional Requirements 19](#_Toc384136712)

[4.1.1 Client/Student Functions 19](#_Toc384136713)

[4.1.2 Tutor/Server Functions 20](#_Toc384136714)

[4.2 Use Case Diagram 21](#_Toc384136715)

[4.3 Use Cases 22](#_Toc384136716)

[4.3.1 Student/Client Use Cases 22](#_Toc384136717)

[4.3.2 Tutor/Server Use Cases 25](#_Toc384136718)

[4.4 Activity Diagrams 29](#_Toc384136719)

[4.4.1 Activity Diagram Student 29](#_Toc384136720)

[4.4.2 Activity Diagram Tutor 30](#_Toc384136721)

[4.5 Class Diagram 31](#_Toc384136722)

[4.6 Deployment diagram 32](#_Toc384136723)

[4.7 External Interface 33](#_Toc384136724)

[4.7.1 Splash Screen 33](#_Toc384136725)

[4.7.2 Student/Client Interfaces 33](#_Toc384136726)

[4.8 Software and Hardware Requirement 37](#_Toc384136727)

[5.0 Non-Functional Requirements 38](#_Toc384136728)

[5.1 Usability 38](#_Toc384136729)

[5.2 Performance 38](#_Toc384136730)

[5.3 Supportability 38](#_Toc384136731)

[5.4 Scalability 38](#_Toc384136732)

[6.0 System Documentation 39](#_Toc384136733)

[6.1 Student/User program 39](#_Toc384136734)

[6.1.1 studProj 39](#_Toc384136735)

[6.1.2 brain 40](#_Toc384136736)

[6.1.3 chart 43](#_Toc384136737)

[6.1.4 loginGui 44](#_Toc384136738)

[6.1.5 mathPort 45](#_Toc384136739)

[6.1.6 startGui 50](#_Toc384136740)

[6.1.7 studCom 50](#_Toc384136741)

[6.1.8 studData 52](#_Toc384136742)

[6.2 Tutor/Server Program 58](#_Toc384136743)

[7.0 Concurrent and Future Research 74](#_Toc384136744)

[7.1 Grant 74](#_Toc384136745)

[7.1.1 Scope of Grant 75](#_Toc384136746)

[8.0 Future Research 76](#_Toc384136747)

# 1.0 Overview

In my experience, individuals who are struggling in mathematics often make the statement “I am just not good at math.” The statement seems to be a product of frustration at not understanding the steps and underlying meaning of mathematics principles being studied. These feelings are compounded as the individual continues to try to learn, but seems to not be able to advance. By the time an individual has entered college he or she may well have given up attempting to learn mathematics and does not put forth the attention to the subject needed to advance his or her knowledge. A system is needed that allows an individual to understand and modify the inner processes that can hold them back so that they can productively practice basic mathematics. By getting feedback on when he or she is experiencing a lack of attention and when frustration is building while attempting to complete mathematics problems, the individual will be able to learn how to focus his or her attention and relax helping build confidence and retain information easier. This may mitigate the effects of prior failures. The system I propose will provide a graphical display that contains mathematics problems with a dynamic graph of the user’s attention to help the individual monitor focus and the effects of frustration.

## 1.1 Purpose

The purpose of this studio is to create mathematics tutoring software that collects data on the time it takes for students to answer certain problems, gauge the accuracy of the answers, and record the brainwaves that are produced by the student. The EEG data that will be focused on is generated by relaxation and attention. Attention has been shown to correlate with the beta brainwaves, alpha waves with relaxation, and theta waves are associated with frustration (Gittis, 2005).

## 1.2 Scope

The objective of this Studio project is to create software that can identify weak areas in the student’s mathematics knowledge through the measure of time to answer, accuracy of the answers, and teach/train the student to work at an optimal level using the EEG data as described above. The EEG data will be collected through an instrument called MindWave® which is sold by the company NeuroSky. The MindWave is a single channel EEG that does not need specialized training to be administered. As discussed, frustration from not being good at math negatively affects attention and causes tension which has been shown to negatively affect learning (Napoli, 2005, pg. 100).

## 1.3 Objective

The objective of this software is to help the individual and/or tutor correctly identify the problem areas that the student is having. In the future, the software could be used to gather data on teaching strategies or directed learning and compare the efficacy of using brainwave feedback against other pedagogical methods, but that is outside of the scope of this studio.

## 1.4 Assumptions and Constraints

The general assumptions, based on prior research, are that focused practice, attention, and relaxation training will lead to better overall performance in mathematics. In addition, use of the software will be in controlled environments where external distractions can be limited. If the user is constantly distracted by external stimuli the purpose of the system and reliability of the results would be compromised. In addition, with the relatively inexpensive price of the NeuroSky MindWave, this system could be rolled out on a larger scale at a future time.

I will be unable to complete full scale testing due to time and money for the EEG system.

Completing the software system will allow future testing. I will also be constrained by the SDK provided by NeuroSky. In regards to this, other open source projects show that completing the software will be possible. Such projects are; MindBlaster (NeuroSky, 2010, Coding4Fun), MindStream (Blue, 2011, pg. 1) and Puzzlebox Orbit (NeuroSky, 2010, PuzzleBox).

## 1.5 Project Deliverables

* Software Project Management Plan
* Software Requirements Specifications
* Software Design Document
* Software Quality Assurance Plan including a Software Verification and Validation Plan and the Test Design Document
* User Manual
* Final Studio Document

## 1.0.6 Schedule and Budget Summary

Equipment has already been purchased so there is not a budget requirement.

The schedule includes the following tentative milestones:

|  |  |
| --- | --- |
| **Item** | **Due date** |
| Proposal (this document) | Completed |
| Proposal Presentation | Completed |
| Software Project Management Plan | Completed |
| Software Requirements Specifications | Completed |
| Software Design Document | Completed |
| Software Quality Assurance Plan | Completed |
| Studio I Presentation | Completed |
| Front end/Database | Completed |
| MindWave Software | Completed |
| Software Integration | Completed |
| Final Studio Paper | Completed |
| Studio II Presentation | April 9, 2014 |

# 2.0 Literature Review

My literature review began trying to answer five questions. These questions are the foundation of my studio. The questions are:

1. Will collecting EEG data be helpful for individuals to view internal processes?
2. Can the MindWave sold by NeuroSky measure attention and relaxation instantaneously, and unobtrusively collect accurate EEG data that can be displayed to the user in an easy to understand way?
3. Does a person’s level of attention have an effect on his or her learning and understanding of a subject?
4. Does a person’s level of relaxation have an effect on his or her learning and understanding of a subject?
5. What is the effect of tutoring on a person’s learning and understanding of a subject?

If each question can be answered positively, then creating software to record EEG signals from a person could be used to assist in his or her learning. In addition, collecting and analyzing the data provided by the software could help future development of new techniques in tutoring and education.

## 2.1 EEG

Electrical signals from the nervous system were discovered around 1848 (Evans, Abaranel, 1999). The first observation related to peripheral nervous system that sends signals to and from the brain to the rest of the body. This was not until 1929 where Hans Berger described “oscillating electrical activity recorded from the human scalp” (Evans, Abaranel, 1999, pg. 4). During the 1930’s to 40’s it became clear that EEGs could be used to diagnosis epilepsy and brain trauma, but any further advancements would require more precise ways to study the information provided. With the advent of the digital computer it became possible to study outputs in enough detail to start seeing patterns. Only recently has it become possible with the release of the MindWave from Neurosky, to capture EEG readings at home or in other non-lab environments.

### 2.1.1 Attention Deficit Disorder/ Attention Deficit Hyperactivity Disorder

Attention Deficit Disorder (ADD) and Attention Deficit Hyperactivity Disorder (ADHD) affected 10% of the population in 1999 (Evans, et al., 1999). Those that suffer from ADD/ADHD are more likely to suffer from other learning disorders (LD) and conduct disorders (CD). (Evans, et al., pg. 1). One of the main symptoms of ADD/ADHD is difficulty in task completion, particularly if the task is perceived by the individual to be irrelevant. In general, those suffering from ADD/ADHD have similar areas of the brain which show abnormalities (Evans, et al., 1999). At this time, diagnosis of these disorders is made by using EEG, Positron Emission Tomography (PET), and there is also a survey for parents and teachers which called the Conner scale. In addition, EEG feedback has been used to treat ADD/ADHD successfully since the mid-1970s. Two factors that interfere with EEG treatment of ADD/ADHD are depression and home stressors. If the individual is suffering from either situation, EEG feedback treatment will not be sufficient for treating ADD/ADHD. These conditions could also affect the effectiveness of the tutoring system under development.

### 2.1.2 EEG Research

To measure the effectiveness of EEGs to predict if children were paying attention, Lutsyuk, Eismont, and Pavlenko studied EEG potentials while conducting psychological tests (2006, pg. 209). The test looked at measuring the child’s working efficiency, productivity of attention and accuracy of attention (Lutsyuk, et al. 2006, pg. 210). The research found correlations between the coefficient of reactivity (CR) which is computed by comparing alpha waves while eyes are closed to when eyes are opened. The presence of alpha waves indicates that an individual is in a state of rest. This difference indicates that the individual is moving from a state of rest to a more active state. In addition, the accuracy of the attention index positively correlated to the ratio of beta rhythm to theta rhythms. In all, the research found that it is possible to measure the amount that a child is paying attention, which should be applicable to all age groups.

## 2.2 NeuroSky

Incorporated in 2004 by Stanley Yang, NeuroSky has partnered with MIT, Carnegie Mellon, Stanford University, and many others to create and refine the MindWave. This has allowed EEG research and the development of brain computer interfaces (BCI) that would not have been possible previously due to the cost of previously available equipment. Released as an open source platform with a software development kit (SDK), the MindWave has opened up the world of EEG to researchers and the general public. So far the product has been used for games such as the Star Wars Force Trainer, helping those with Lou Gehrig’s disease communicate, and controls for aerial drones. At the time of writing this article two products are offered by NeuroSky. The MindWave, which is a less expensive system that can be used with computers and is designed for comfort and ease of use, and the MindSet which has headphones and will work with android and Ios devices. Both devices use the same hardware and software for EEG functions (Milo, 2013, pg.1-3).

### 2.2.1 NeuroSky Research

In the article, *Evaluating a Brain-Computer Interface to Categories Human Emotional Response* (2010), Katie Crowley found that the MindWave was suitable to measure relaxation and attention (pg. 278). The research used two tests: the Stroup Color-Word Interference Test and the Towers of Hanoi. The Stoup test requires the subject to name the color of the text while ignoring what the text actually says. For instance, the subject would be shown the word “red” (written in green) and the subject would have to say the word green. Our mind first interprets the word as red and it takes selective attention to ignore the meaning of the word and say the actual color. The Stroup test is an established test for inducing stress and testing attention (Crowley, 2010, pg. 278). The second test was measuring frustration of subjects trying to complete the Tower of Hanoi. Subjects were shown to have higher levels of stress until they figured out how to break the problem into smaller steps to solve it. Both tests “clearly demonstrate NeuroSky’s suitability as a minimally invasive means of measuring attention and meditation level of a subject” (Crowley, 2010, pg. 278).

In the article, *Toward Exploiting EEG Input in a Reading Tutor* (2011), Jack Mostow, Kai-min Chang, and Jessica Nelson used the Neurosky MindSet to test the ability of the system to identify when a student is reading a difficult sentence. The researchers found that they could tell when the individual read a difficult sentence. The participants had to read three easy sentences at the K-1 level and difficult passages from the Graduate Record Exam (GRE) and ACE GED test, both of which they read aloud. They were then asked to answer multiple-choice close questions to verify they were reading for meaning (Mostow, et al., 2011, pg. 231). The process was repeated except the participants read to themselves. The research used six adult readers and nine children nine to ten year olds. “In summary, this pilot study gives hope that a school-deployable EEG device can capture tutorially relevant information” (Mostow, et al., 2011, pg. 236).

### 2.2.2 EEG Competitors

Other companies that manufacture EEGs have products that range from large multichannel machines used in research and the medical field to at home units like the MindWave. One comparable unit to the Mind Wave is the Emotiv Insight. This is a multichannel device with high resolution, but uses a more expensive headset and developer package. The Emotive Insight costs $750 dollars and the developer and research SDK would cost $1,250. At this cost a medical grade EEG could be purchased starting at $1,108 dollars. In addition, the sensors use saline pads that have to be replaced and add cost. The use of saline pads would make the device uncomfortable for some users. (emotiv, pg. 1) By comparison, the NeuroSky MindWave cost is $79.99 and sits comfortably on the user’s head without the need for saline pads or other connective devices.

## 2.3 Attention and Learning

The role of attention in memory has not always been a focus of research in psychology. The Gestalt and Behaviorist schools in psychology related the input (stimulus) we receive from our environment and our resulting output (behavior) on conditioning or isomorphism. These schools of thought claimed that all behavior can be tied to a specific stimulus, and memory is what ties the stimulus to the behavior. So learning is presumed to be the act of a stimulus on our brain which records that stimulus and maps it to a behavior. These theories did not account for the ability of individuals to pay attention to one stimulus above others. The ability to block out extraneous information and focus attention is called selective attention and is what I will be measuring (Kahneman, 1973, pg. 1-3).

### 2.3.1 Research in selective attention

Selective attention allows us to focus and learn material and has been shown to develop as we age. This was shown in Miller’s work in *Children’s Attention Allocation, Understanding of Attention, and Performance on Incidental Learning Task* (1981). Children were shown 12 drawings in pairs of two. Each pair contained an animal and a household item. Half the children were told to remember the animal and the other half were told to remember the household item. To test accidental learning the children were asked to remember what drawings were paired with the drawings they were asked to remember. For example, if the children were told to learn the animals they would be tested on what animals were shown, but then asked to remember what household item was paired with that animal. The number of animals remembered corresponds to focused learning and the number of household items to accidental learning. Children in grade level 2 scored 5% for both learned and accidentally learned drawings. Children in grades 5-8 showed improved scores for both learned and accidentally learned drawings to 53% for learned and 11% for accidental learning (Miller, pg. 1187). The central drawings asked to be remembered were highest for grade 8 at 63% learned and 16% accidental. This research showed that as children get older, their ability to tune out extraneous information increases and added attention allows for improvement in all areas of learning. Anecdotally, this phenomenon has been noticed by educators who note that when students pay attention, usually, they do better in school.

### 2.3.1 Refocusing Student Attention

Being able to see inside a student’s brain to know when they are paying attention would be a powerful tool. Research is being conducted that uses the NeuroSky MindWave to alert a robot when individuals are not paying attention while the robot tells a story. The amount of attention being paid by the participant is represented by the formula below:

Equation 1:

(Szafir, Mutlu, 2012, pg. 13)

In Equation 1, is the beta wave, is the alpha waves, and is the theta wave of the participant. The attention level of the individual is represented by. When attention level of the individuals drops below a preset threshold immediacy gestures from the robot are attended to refocus their attention. Three test cases were used, and in the first test case robot had no interventions and attention slowly fell over time. The second test case had random interventions and attention plateaued with less variation. Lastly, the robot would provide immediacy cues every time attention levels dropped. The findings showed that immediacy cues prompted by an EEG allowed participants to remember more of the story being told (Szafir, Mutlu, 2012, pg.17-18). The participants were asked fourteen questions with the average of correct answer being “6.30 (SD=3.40), 7.44(SD=1.94), and 9.00 (SD = 1.76) in the low immediacy, random immediacy, and adaptive immediacy conditions, respectively” (Szafir, et al., 2012, pg. 17-18). These results are also interesting since the standard deviation dropped as the attention of the participants were more efficiently bolstered by the use of the MindWave Mindset.

## 2.4 Relaxation, Frustration and Learning

As with attention, it seems like common sense that if an individual is frustrated or stressed he or she will not be able to learn effectively. Anecdotally though, students refute this by believing in the effectiveness of cram sessions. Their belief is that the effects of the pressure help them to complete tasks and study effectively. In this literature review, the frustration that will be discussed is related to the feeling of becoming blocked, not understanding the subject, and anxiety. The environment that an individual works best in is not addressed, but trying to minimize the frustration of not understanding a subject brings, and helping the individual relax enough to refocus attention is. This will be relatively easy to measure since (as can be seen in Equation 1) frustration is shown through theta waves. By including theta waves in information given to the user, he or she will be able to monitor his or her frustration and focus on staying relaxed (Gittis, 2005, pg. 1).

### 2.4.1 Managing Frustration

Peter Haiman in *How Children Manage Frustration Affects Ability to Focus and Learn*: *ADHD* (2000) discusses coping with frustration and anxiety from learning situations. If the child has the correct support structures he or she combats the anxiety and learns that relaxing and focusing helps learning. If these support structures do not exist, children will distract themselves to defend against anxiety. These distractions take the form of silly or aggressive behaviors, excessive talking, acting out, and/or procrastination. Anxiety disturbs attention which in turn disturbs learning (Haiman, 2000, pg. 1). By monitoring theta waves by using Equation 1, we can account for frustration and help the person being tutored to effectively see the effects of frustration of learning and give the tutor an indication that intervention may be needed.

### 2.4.2 Mindfulness Training Research

Mindfulness is the ability to understand internal process without judgment. That is; understanding anxiety, frustration, and having skills to deal with those feelings. In *Mindfulness Training for Elementary School Students: The Attention Academy* (2005); Maria Napoli, Paul Rock Krech, and Lynn C. Holley studied the effects of mindfulness on learning. As with the article discussed in the previous section (Hamian, 2005), this article reviews the effect of stress and anxiety on learning. As a result of negative situations in the child’s life, and the frustration around learning, children’s overall performance at school is negatively affected. The total number of participants in the study was 228 with 120 males and 108 females. The control group had 114 members and did not receive mindfulness training, while the remaining students received the training. The study taught the experimental group to relax and maintain their attention. All students in the experimental group showed improvement in performance with selective attention and reduction of test anxiety. The most improvement was seen in the students with ADHD. Neither group showed improvement in the sustained attention. One limitation noted was that measuring attention of non-ADHD students was difficult since the scales were designed for children with ADHD and might not be appropriate for students without ADHD.

## 2.5 Tutoring and Learning

As with attention and relaxation, it seems like common sense that tutoring would help with student performance. However, the research results have been inconsistent. The outcome of some research finds that the individual’s state of mind effects the degree that tutoring will help, while other research finds tutoring is effective. In general, tutoring is seen to be effective if the tutor and the student are engaged and focused on the topic at hand. The EEG system will be able to gauge the attention and relaxation of the student to keep them on task. Overall, a monitoring system would help to track the performance and focus tutoring on the areas that need attention.

### 2.5.1 Possible issues with tutoring

Mark Bray in *The shadow education system: private tutoring and its implications for planners* (2007); analyzed private tutoring to identify the impact of tutoring on academic performance. Three studies from around the world were reviewed and the meta review found that the effects of private tutoring are not always positive. In a study conducted in Mauritius Greece (1995), sixth grade students were studied for literacy. The items studied were divided into malleable factors (those that a school could affect) and non-malleable factors (those a school could not affect). Tutoring was able to explain more positive variation in test scores then other malleable factors, but not as strongly as the non-malleable factors of “English spoken in the home and the socio-economic level of the home” (Bray, 2007, pg. 47). Research completed in Polydorides, Greece (1986), had similar results. It showed positive correlations between tutoring and academic performance, but the correlations were weak and not consistent although they were statistically significant. Lastly, two studies were reviewed that were conducted in Egypt (1990-1991) that found no significant effect of tutoring on academic performance. To explain these mixed results Bray suggested four items that tutoring depends on to be effective: “

* the content and mode of delivery of the tutoring;
* the motivation of the tutors and tutees;
* the intensity, duration and timing of tutoring; and
* the type of pupils who receive tutoring”

(Bray, 2007, pg. 46-50).

Even though the studies cited were from older research and from very disparate cultural groups, the quality of tutoring was found to be important and there is no evidence to suggest that it is not true today as well. This being said, having an application that tracks performance and intensity of the students work should help to improve the quality of tutoring by providing feedback to the tutor as well as giving indications of when tutoring is effective..

### 2.5.2 Learning from human tutoring

In the article *Learning from human tutoring*; findings of the studies suggest that tutoring is effective at increasing academic performance and hypothesis for why tutoring is effective are given. (Chi, Siler, Jeong, Yamuchi, Hausmann, 2001, pg. 471-472) The three hypotheses are: The tutor-centered pedagogical hypothesis, student-centered constructive hypothesis, and the interactive coordination hypothesis. The tutor-centered pedagogical hypothesis focuses on the tutor’s dialogue with student. This is called the tutoring frame and is listed below. “

1. Tutor asks an initiating question;
2. Student provides a preliminary answer;
3. Tutor gives (confirmatory or negative) feedback on whether the answer is correct or not
4. Tutor scaffolds to improve or elaborate the student’s answer in a successive series of exchanges (taking 5-10 turns)
5. Tutor gauges student’s understanding of the answer”

(Chi, et al., 2001, pg. 472-473).

Classroom dialog generally ends at step three, but in the tutoring frame the tutor is able to build on the question that the student gives and gauge their understanding. “The tutoring frame suggests that tutors basically dominate the dialogue, dictate the agenda, craft the appropriate next question, select the next example and problem to be solved, pose the analogy and counter example, give the feedback, and so forth” (Chi, et al. 2001, pg. 473). This helps the student build on previous knowledge and take small steps to provide scaffolding. These positive effects can be undermined by the tutor ignoring when a student is confused or given long-winded explanations (Chi, et al., 2001, pg. 476). The student centered constructive hypothesis is a little easier to explain. It focuses on the student perceptive and views the student through active learning. Tutoring assists with learning “because its interactive nature affords greater opportunities for students to engage in more constructive activities, as compared to a classroom” (Chi, et al. 2001, pg. 477). This helps aid in active learning. The interactive coordination hypothesis states that both the tutor and the student interacting together reinforce learning. The article’s research found that the tutor’s skill and students being active participants in the process both lead to learning. Increasing the effectiveness of tutors and helping students become active participants will enhance the amount learned.

## 2.6 Conclusion

From the sections above it can be seen that enhancing student’s attention and helping him or her relax during tutoring will enhance learning. This supports the idea that building a tutoring system that focuses on maintaining student attention while quizzing them in mathematics would be useful. Using an EEG to visualize a student’s attention while she or he completes a task has been shown to be effective in learning environments (Szafir, et al., 2012). In addition, the NeuroSky MindWave has been shown to be effective at capturing and displaying the attention of a user unobtrusively (Crowley, 2010). By completing a tutoring system that uses the MindWave, one can create a system that adapts to the student to enhance learning and increase academic performance.

# 

# 4.0 Software Requirement Specification

## 4.1 Functional Requirements

The Functional Requirements section of this document will outline the features required to accomplish the goals of creating a math tutoring system that uses the NeuroSky MindWave to capture, record and display EEG readings. These requirements are only concerning the studio portion of this project. Future work will include added functionality and research. In general, there will be two software components. One will be the client or student component that provides math problems to solve and the EEG data for attention and relaxation. The second component will be a tutor dashboard which displays the current student’s progress.

### 4.1.1 Client/Student Functions

Essential

F-00 Student will turn on MindWave and software sync will occur.

F-01 Student will start the software and log in using his or her Jacksonville State University (JSU) username. (The first portion of their email address)

F-02 Start screen of the client software will display student’s overall average for attention, time to answer, and percent correct. There will also be a start button and log out button

F-03 After the student clicks start button math problems will load from a test bank.

F-04 For each question, answer, attention data, and relaxation data will be recorded.

F-05 During quizzing the client interface will display a math problem and client’s current attention data.

F-06 After answering a question the client will receive an indication if he or she answers a question correctly or incorrectly. If the answer was incorrect then the correct answer is shown

F-07 Questions will continue to load until preset amount of math problems is reached.

F-08 After completing twenty questions the start screen will load with updated metrics.

F-09 Student’s number of correct answers, time to answer, and attention levels will be recorded.

Future

F-11 Students will be able to view their historical data.

### 4.1.2 Tutor/Server Functions

Essential

F-12 Tutor/Server start screen will display connected clients and their current rank.

F-13 As client/student log in to system they will be authenticated.

F-14 As students answer question sets (five questions) his or her data will be displayed (e.g. EEG and number correct) on the tutor screen.

Expected

F-17 Server users with appropriate access privileges will be able to view historical data for selected students.

## 4.2 Use Case Diagram



## 4.3 Use Cases

### 4.3.1 Student/Client Use Cases

ID: 1 Authorize Student/Student Login

|  |  |  |
| --- | --- | --- |
| Use-Case Name: Authorize Student/Student Login | ID: 1 | Importance Level: High |
| Primary Actor: Student | Use Case Type: Detail, essential | |
| Stakeholders and Interests:  Student – wants to start software to begin tutoring session and improve mathematics skill  Tutor – wants to see that student is connected to the server and MindWave if applicable  Professor – wants student to do better in class | | |
| Brief Descriptions: Student starts software and validates account with server | | |
| Precondition: Student already registered  Trigger: Opening program  Type: External | | |
| Relationships:  Association:  Include: Operation with MindWave  Extend: Operation without MindWave  Generalization: | | |
| Normal Flow of Events:   1. Student starts software 2. Software syncs with MindWave 3. System requests username 4. Account authenticated with server 5. Starting page shows current rank, earned points, EEG data, and a button to start quizzing 6. Execute Use Case ID: 2 | | |
| Subflows: | | |
| Alternate/Exceptional Flows:  Authentication fails:  3a. Account authentication fails  3b. System request re-enter username and password  3c. If authentication fails repeat Step 3b up to 3 times  3d. If authentication fails system notifies student to contact tutor and system returns to login screen  3e. If authentication succeeds return to Step 4 | | |

ID: 2 Operation With MindWave

|  |  |  |
| --- | --- | --- |
| Use-Case Name: Operation With MindWave | ID: 2 | Importance Level: High |
| Primary Actor: Student | Use Case Type: Detail, essential | |
| Stakeholders and Interests:  Student –wants to begin quiz sections and earning levels to improve his or her ability  Tutor – wants to be able to view the student’s progress during quizzing sessions  Professor – wants student to do better in class | | |
| Brief Descriptions: Student views current rank and then begins mathematics quizzing | | |
| Precondition: Student is logged into system  Trigger: Execution of Use Case 1  Type: External | | |
| Relationships:  Association: Student  Include: Authorize Student/Student Login  Extend: Operate without Mindwave  Generalization: | | |
| Normal Flow of Events:   1. After clicking the button the student gets math question 2. Answers to question is indicated correct or incorrect 3. Steps 2 and 3 are repeated for 20 different math questions at current level 4. Results are displayed to student the start screen 5. Start button takes student back to 1 1 (To be determined when level is increased) 6. Logout button returns to login screen | | |
| Subflows:  4a. Every five questions student data sent to server | | |
| Alternate/Exceptional Flows:  Student logs out  2a. Student wants to quit, execute Step 6.  MindWave Sync fails:  5a. Sync failed message with a retry button and continue button  5b. Retry button pushed go to Step 5  5c. Continue button: Attention data set to 0: Use Case ID: 3  5d. After 5 failed attempts or 1 minute of no action client informed to contact tutor and systems returns to login screen | | |

ID:3 Operation Without MindWave

|  |  |  |
| --- | --- | --- |
| Use-Case Name: Operation Without MindWave | ID: 3 | Importance Level: High |
| Primary Actor: Student | Use Case Type: Detail, essential | |
| Stakeholders and Interests:  Student –wants to begin quiz sections and earning levels to improve his or her ability  Tutor – wants to be able to view the student’s progress during quizzing sessions  Professor – wants student to do better in class | | |
| Brief Descriptions: Student views current rank and then begins mathematics quizzing | | |
| Precondition: Student is logged into system but MindWave sync failed  Trigger: Student clicks start button to begin quizzing  Type: External | | |
| Relationships:  Association: Student  Include: Authorize Student  Extend:  Generalization: | | |
| Normal Flow of Events:   1. After clicking start button student gets math question 2. Answers to question is indicated correct or incorrect 3. Steps 2 and 3 are repeated for 20 different math questions 4. Results are displayed to the student on the start screen through text 5. Start button takes student back to Step 1 (To be determined when level is increased) 6. Logout button returns to login screen | | |
| Subflows:  4a. Every five questions student data sent to server | | |
| Alternate/Exceptional Flows:  2a. Student wants to quit, execute Step 7. | | |

### 4.3.2 Tutor/Server Use Cases

ID: 4 Dynamic Reporting

|  |  |  |
| --- | --- | --- |
| Use-Case Name: Dynamic Reporting | ID: 4 | Importance Level: High |
| Primary Actor: Tutor | Use Case Type: Detail, essential | |
| Stakeholders and Interests:  Tutor – Wants to see ongoing student/client sessions | | |
| Brief Descriptions: Tutor can view the progress of students as they are taking quizzes | | |
| Precondition: Student authenticated  Trigger: Tutor has logged in  Type: External | | |
| Relationships:  Association: Tutor  Include:  Extend:  Generalization: | | |
| Normal Flow of Events:   1. As student takes quiz, data is displayed on tutor screen on a bar graph with green bars being correct and red bars indicating incorrect answers | | |
| Subflows: | | |
| Alternate/Exceptional Flows: | | |

ID: 5 Historical Reporting

|  |  |  |
| --- | --- | --- |
| Use-Case Name: Historical Reporting | ID: 5 | Importance Level: High |
| Primary Actor: Tutor | Use Case Type: Detail, essential | |
| Stakeholders and Interests:  Tutor – Wants to view past student performance  Administrator - Wants to view past student performance | | |
| Brief Descriptions: Server software retrieves historical data and displays it | | |
| Precondition: Tutor is logged in  Trigger: Tutor clicked on historical data button for student  Type: External | | |
| Relationships:  Association: Tutor, Administrator  Include:  Extend:  Generalization: | | |
| Normal Flow of Events:   1. Click history button beside current student data 2. System displays historical data | | |
| Subflows: | | |
| Alternate/Exceptional Flows: | | |

ID: 6 Populates Questions

|  |  |  |
| --- | --- | --- |
| Use-Case Name: Populates Questions | ID: 6 | Importance Level: High |
| Primary Actor: Administrator | Use Case Type: Detail, essential | |
| Stakeholders and Interests:  Administrator - Wants to add questions to system | | |
| Brief Descriptions: Administrator adds questions to system | | |
| Precondition: Administrator is logged in  Trigger: Administrator clicked on enter questions  Type: External | | |
| Relationships: A  Association: Administrator  Include:  Extend:  Generalization: | | |
| Normal Flow of Events:   1. System displays interface for entering questions (TBD) 2. Administrator enters questions with questions levels (TBD) 3. Administrator logs out | | |
| Subflows: | | |
| Alternate/Exceptional Flows: | | |

ID:7 Register Student

TO BE DETERMINED

## 4.4 Activity Diagrams

### 4.4.1 Activity Diagram Student



### 4.4.2 Activity Diagram Tutor



## 4.5 Class Diagram

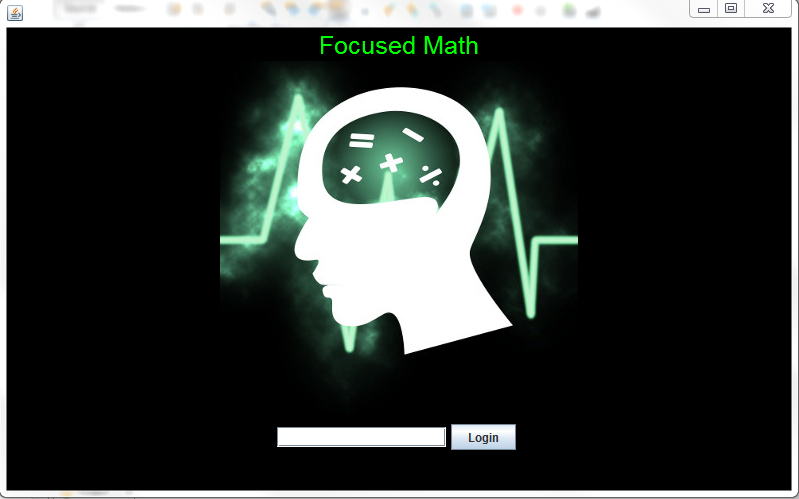


## 4.6 Deployment diagram



## 4.7 External Interface

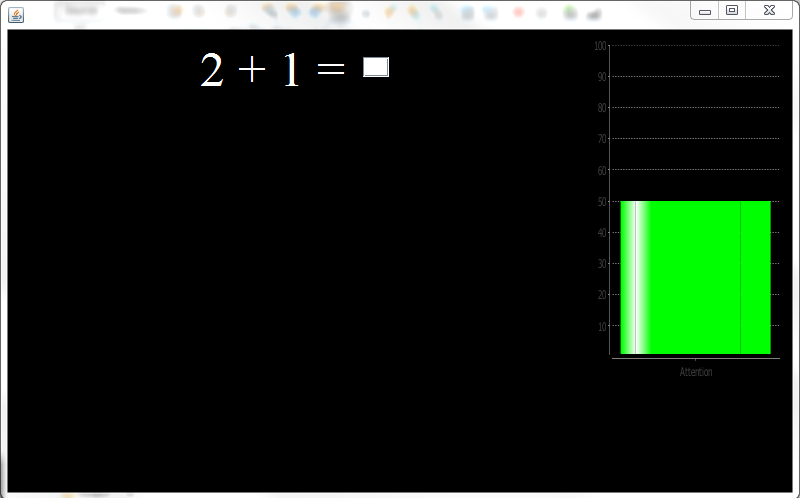
### 4.7.1 Splash Screen



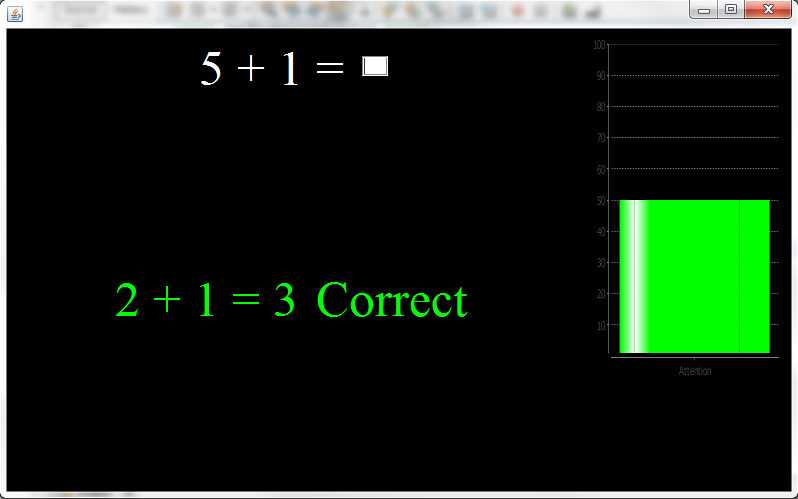
Source: Image by Kristen Cunningham

### 4.7.2 Student/Client Interfaces

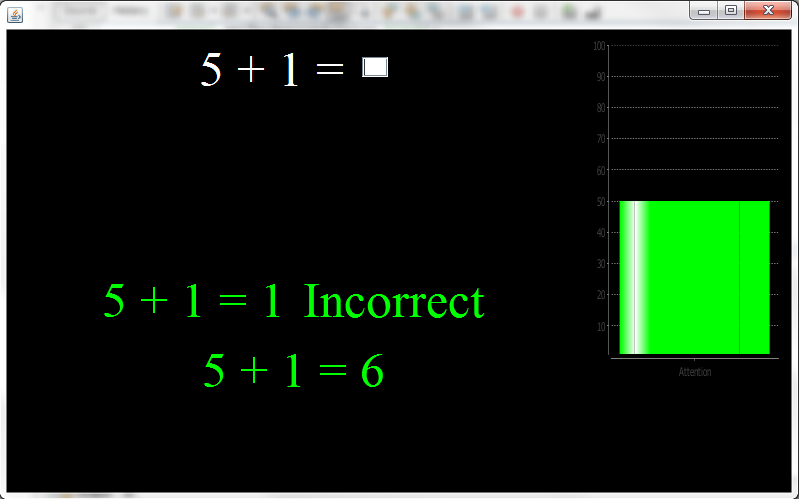
#### 4.7.2.1 Student/Client Interface Waiting for Student Response



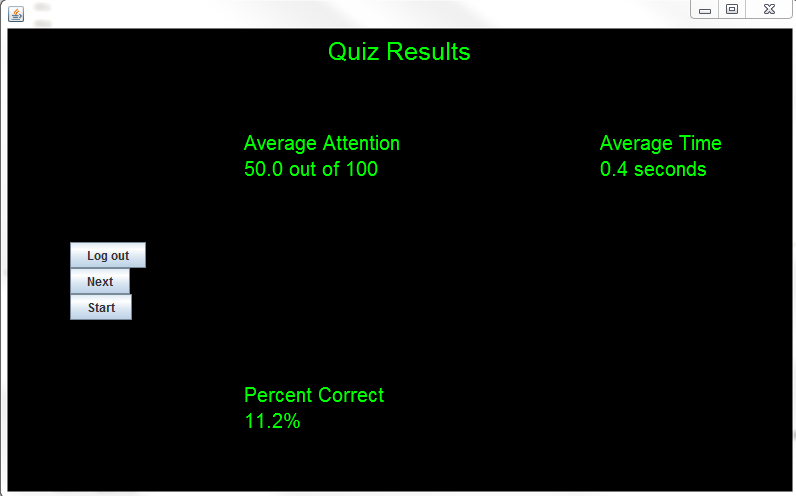
#### 4.7.2.2 Student/Client Interface Correct Answer



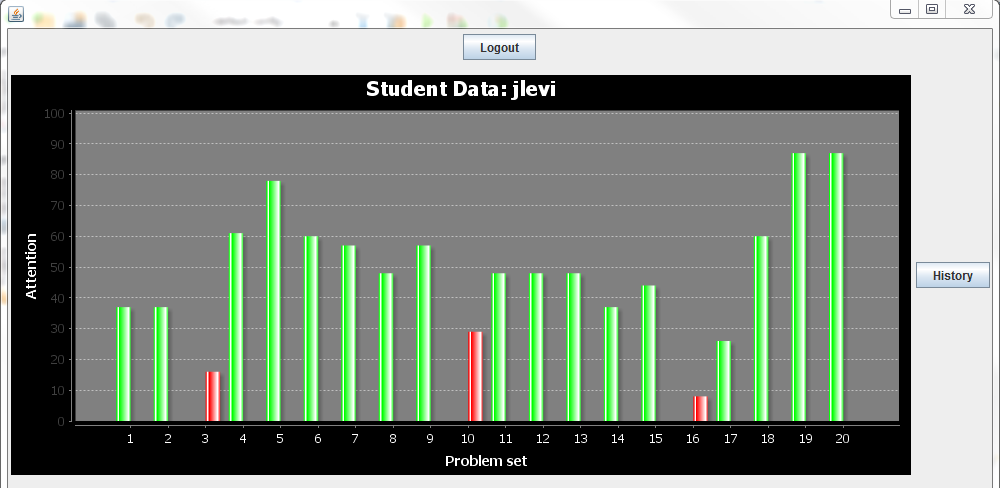
#### 4.7.2.3 Student/Client Interface Incorrect Answer



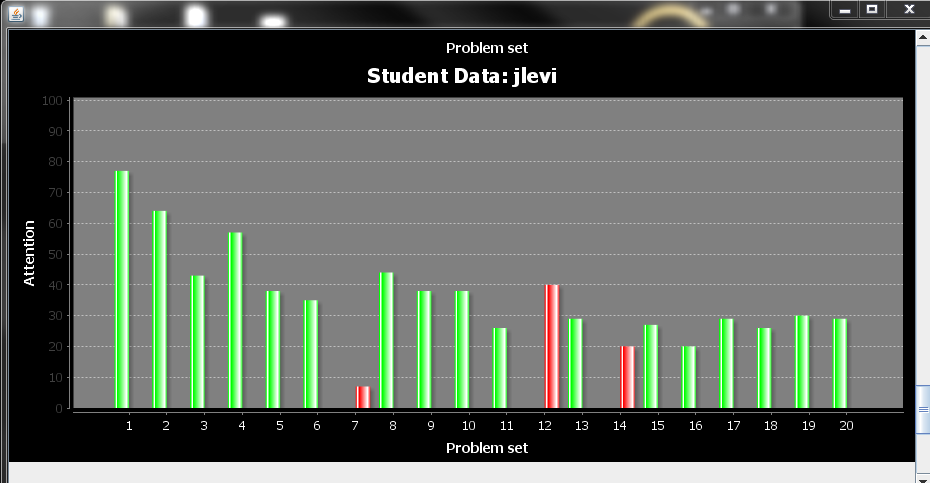
4.7.2.4 Student/Client Interface After Question Set Competed



#### 4.7.2.4 Tutor/Server Client Monitoring



#### 4.7.2.5 Tutor/Server Historical Data For Student



## 4.8 Software and Hardware Requirement

The hardware being used will be the NeuroSky MindWave discussed earlier in the literature review. There will be two MindWaves, one owned by me, that will connect to two Windows PCs provided by Jacksonville State University. The second MindWave has been purchased under grant 10000-3-223-70000-20. A third Windows PC provided by Jacksonville State University will be used to host the server application which will store user accounts, store historical data, and also will serve as tutor interface. In the future, a dedicated server could be used to facilitate more clients accessing the server program simultaneously. The software will be written in Java and designed specifically for a Windows PC. A few modifications would be needed to get the client or server software to work on other operating systems.

# 5.0 Non-Functional Requirements

## 5.1 Usability

The user interface will be designed so that the system will be as intuitive as possible for the student and the tutor. This will reduce stress and time needed for students and tutors to start using the system and allow for a more pleasant experience. In the future, online user documentation will be provided to make accessibility of system documentation easier.

## 5.2 Performance

The goal of the system is to run with no perceivable lag for the two clients using the system simultaneously. In addition, data will be sent to the server for display to the tutor with very little perceivable delay. The student data will be sent to the server in five question groups so the tutor will be able to see the performance of each client. The configuration noted will support up to ten clients. More clients would need to be supported by a dedicated server.

## 5.3 Supportability

Support for the system will be provided during research by Jason Levi. Continued support will be facilitated by system documentation.

## 5.4 Scalability

The system is being designed with two concurrent users in mind but it could support up to ten with no perceivable lag. More users could be supported with dedicated servers. In addition, the system will be primarily used for mathematics tutoring, but with minor modifications will be to support other curriculum domains. To make the changes noted above, future testing will be needed to determine performance degradation and specific solutions to be suggested.

# 6.0 System Documentation

The system is a prototype to support further testing discussed under the Future Research section of the paper. The focus of the design was to be as modular as possible to allow easier enhancements in the future. The section is divided into the Student/User program and Tutor/Server program. Each one is a stand-alone program that provides functionality to the other. For the prototype all data is stored in flat files. In the future the flat files will be replaced by a database. This change will be easy to implement replacing the student and studData classes.

## 6.1 Student/User program

### 6.1.1 studProj

**Name: changePanel(JPanel x)**  
 Arguments: JPanel x  
 Returns: None  
 Pre-condition: Program already running  
 Post-condition: Old panel will be removed and new one added to frame  
 Exceptions: None  
 Flow of Events:

1. The function removes panel main
2. Replaces main with the JPanel from the argument and adds it to the frame
3. Function redraws frame

Exception Handling: None

**Name: windowClosing()**  
 Arguments: WindowEvent e  
 Returns: None  
 Pre-condition: Program already running  
 Post-condition: Closes sockets and terminates program  
 Exceptions: None  
 Flow of Events:

1. Sends server logout message
2. Closes socket
3. Closes frame

Exception Handling: None

**Name: main**  
 Arguments: String args[]  
 Returns: None  
 Pre-condition: Program not running  
 Post-condition: Frame created and running  
 Exceptions: None  
 Flow of Events:

1. Creates StudProj object
2. Creates the frame

Exception Handling: None

### 6.1.2 brain

**Name: disconnect()**  
 Arguments: None  
 Returns: None  
 Pre-condition: NeuroSky is connected  
 Post-condition: NeuroSky socket, input streams, output stream are closed  
 Exceptions: IOException  
 Flow of Events:

1. Function closes NeuroSocket
2. Function closes all associated input/output streams

Exception Handling: No notifications sent and program is closed. This should clear all sockets handling the issue.

**Name: run()**  
 Arguments: None  
 Returns: None  
 Pre-condition: NeuroSky is turned on and ready for connection  
 Post-condition: NeuroSky is connected and ready for use  
 Exceptions: SocketException, IOExcetption, JSONException  
 Flow of Events:

1. Program connects socket to localhost on port 13854
2. Input and output streams are created
3. Inputstream is initialized to a BufferedReader
4. NeuroSky is set to JSON data format
5. Thread is started to monitor and record transmission from NeuroSky

Exception Handling:

* SocketException – Connection is retried 5 times if not successful program flow continues without NeuroSky Mindwave®.
* IOExcecption – Connection is retried 5 times if not successful program flow continues without NeuroSky Mindwave®.
* JSONException – Connection is retried 5 times if not successful program flow continues without NeuroSky Mindwave®.

**Name: interrupt()**  
 Arguments: None  
 Returns: None  
 Pre-condition: Brain objects thread is running  
 Post-condition: Thread is interrupted  
 Exceptions: None  
 Flow of Events:

1. Function interrupts thread

Exception Handling: None

**Name: done()**  
 Arguments: None  
 Returns: None  
 Pre-condition: Brain objects thread is running  
 Post-condition: Thread is stopped  
 Exceptions: None  
 Flow of Events:

1. Function stops thread

Exception Handling: None

**Name: getAttention()**  
 Arguments: None  
 Returns: Integer attention  
 Pre-condition: NeuroSky is updating attention levels  
 Post-condition: Current attention of user is returned  
 Exceptions: None  
 Flow of Events:

1. Function returns current value of users attention

Exception Handling: None

### 6.1.3 chart

**Name: createPanel(String title)**  
 Arguments: String title  
 Returns: JPanel  
 Pre-condition: Program frame is already created  
 Post-condition: Blank chart is created  
 Exceptions: None  
 Flow of Events:

1. Function is called and JPanel is created and a chart is added

Exception Handling: None

**Name: createChart(String title)**  
 Arguments: String title  
 Returns: JFreeChart  
 Pre-condition: Program frame and JPanel already created  
 Post-condition: Chart is initialized and returned  
 Exceptions: None  
 Flow of Events:

1. Function creates JFreeChart object
2. Chart settings are initialized

Exception Handling: None

**Name: createDataset(int y)**  
 Arguments: Integer y  
 Returns: None  
 Pre-condition: JFreeChart is already created  
 Post-condition: Data is added to the data set of the chart  
 Exceptions: None  
 Flow of Events:

1. Series and category for data is initialized
2. Value for y data is set

Exception Handling: None

### 6.1.4 loginGui

**Name: returnLog()**  
 Arguments: None  
 Returns: JPanel  
 Pre-condition: Frame is already created  
 Post-condition: User views login splash  
 Exceptions: None  
 Flow of Events:

1. Function creates and sets up login panel
2. Function returns JPanel

Exception Handling: None

**Name: actionPerformed(ActionEvent event)**  
 Arguments: ActionEvent  
 Returns: None  
 Pre-condition: returnLog panel is already created and user is ready to login  
 Post-condition: User is either logged into the system or gave an invalid login  
 Exceptions: None  
 Flow of Events:

1. User clicks the Login button
2. Request is sent to server to validate login
3. Server responds with ready
4. Login username is sent to server
5. If valid user true is sent back
6. Client software requests quiz questions and averages for user
7. Server sends data
8. Averages are parsed and Start screen is loaded
9. If invalid username, user is notified

Exception Handling: None

### 6.1.5 mathPort

**Name: loadQues()**  
 Arguments: None  
 Returns: None  
 Pre-condition: User has logged in with a valid username  
 Post-condition: Quiz questions are ready for user  
 Exceptions: IOException  
 Flow of Events:

1. Function reads average file from disk
2. File is parsed and placed into an array

Exception Handling: User notified to contact tutor

**Name: test()**  
 Arguments: None  
 Returns: Boolean  
 Pre-condition: User has answered a mathematical question   
 Post-condition: Boolean is set true or false whether the supplied answer is correct  
 Exceptions: None  
 Flow of Events:

1. Function compares user answer to answer set in questions file
2. Returns true is user answer was correct or false if incorrect

Exception Handling: None

**Name: resetForm()**  
 Arguments: None  
 Returns: None  
 Pre-condition: User has started a quiz  
 Post-condition: Next question is set for user  
 Exceptions: None  
 Flow of Events:

1. Functions sets next question from array of questions
2. Increments variable for array index

Exception Handling: None

**Name: setAnswer(String x)**  
 Arguments: String x  
 Returns: None  
 Pre-condition: User has started a quiz  
 Post-condition: Answer is updated to correct answer for the current problem  
 Exceptions: None  
 Flow of Events:

1. Function sets answer to current problem

Exception Handling: None

**Name: resetAnswer()**  
 Arguments: None  
 Returns: None  
 Pre-condition: User has completed quiz  
 Post-condition: Answer variable returned to null  
 Exceptions: None  
 Flow of Events:

1. User has completed quiz
2. Function called to set variable to null

Exception Handling: None

**Name: returnX()**  
 Arguments: None  
 Returns: int  
 Pre-condition: Quiz started  
 Post-condition: First number being operated on is sent  
 Exceptions: None  
 Flow of Events:

1. First number being operated on is returned

Exception Handling: None

**Name: returnY()**  
 Arguments: None  
 Returns: int  
 Pre-condition: Quiz started  
 Post-condition: Second number being operated on is sent  
 Exceptions: None  
 Flow of Events:

1. Second number being operated on is returned

Exception Handling: None

**Name: returnCorrect()**  
 Arguments: None  
 Returns: int  
 Pre-condition: Quiz started  
 Post-condition: Returns correct answer to problem as integer  
 Exceptions: None  
 Flow of Events:

1. Correct answer to question is returned

Exception Handling: None

**Name: updateChart()**  
 Arguments: None  
 Returns: None  
 Pre-condition: Quiz started and NeuroSky is connected  
 Post-condition: Polls NeuroSky and updates users attention  
 Exceptions: None  
 Flow of Events:

1. Thread starts and updates chart with attention levels

Exception Handling: None

**Name: quizPanel()**  
 Arguments: None  
 Returns: JPanel  
 Pre-condition: User has clicked the start button from the start screen  
 Post-condition: Quiz panel is created and displayed for user to take quiz  
 Exceptions: None  
 Flow of Events:

1. JPanel that displays quiz questions is returned

Exception Handling: None

**Name: keyPressed(KeyEvent e)**  
 Arguments: KeyEvent  
 Returns: JPanel  
 Pre-condition: User is currently answering questions from the quiz  
 Post-condition: User input is stored  
 Exceptions: None  
 Flow of Events:

1. User types in answer and input is stored in a string
2. User presses enter and answer is checked
3. System displays information user is correct or incorrect either “Correct” or “Incorrect”. If incorrect, correct answer is displayed.
4. New question is loaded
5. Every five questions are sent to the server
6. When question twenty is reached server notified and new averages received

Exception Handling: None

### 6.1.6 startGui

**Name: returnGui()**  
 Arguments: None  
 Returns: JPanel  
 Pre-condition: User has logged in  
 Post-condition: Start screen will be displayed for user  
 Exceptions: None  
 Flow of Events:

1. JPanel that displays start screen is returned

Exception Handling: None

**Name: actionPerformed(ActionEvent event)**  
 Arguments: ActionEvent  
 Returns: None  
 Pre-condition: User has successfully logged into system  
 Post-condition: Either user will be logged out or start a quiz  
 Exceptions: None  
 Flow of Events:

1. Start button will load quiz for user
2. Logout button returns use to login screen

Exception Handling: None

### 6.1.7 studCom

**Name: recieveFile(String File)**  
 Arguments: String File  
 Returns: None  
 Pre-condition: Successful connection to server has been made  
 Post-condition: File will be sent from server and saved on hard drive  
 Exceptions: None  
 Flow of Events:

1. File received from server

Exception Handling: None

**Name: sendCom(String command)**  
 Arguments: String command  
 Returns: None  
 Pre-condition: Successful connection to server has been made  
 Post-condition: Sends string to server to act as communication  
 Exceptions: None  
 Flow of Events:

1. Function sends string to server

Exception Handling: None

**Name: close(String student)**  
 Arguments: String student  
 Returns: None  
 Pre-condition: Successful connection to server has been made  
 Post-condition: User is logged out and server connection terminated  
 Exceptions: IOException  
 Flow of Events:

1. User click close button in upper right hand corner
2. Chooses to close program
3. Server is sent message to log out user
4. Socket to server is closed

Exception Handling: No notifications sent and program is closed. This should clear all sockets handling the issue.

**Name: logOut(String student)**  
 Arguments: String student  
 Returns: None  
 Pre-condition: Successful connection to server has been made  
 Post-condition: User clicked logout button  
 Exceptions: None  
 Flow of Events:

1. User click logout button
2. User is logged out of server

Exception Handling: None

### 6.1.8 studData

**Name: setUser(String x)**  
 Arguments: String x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Username is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores username

Exception Handling: None

**Name: setAttention(int x)**  
 Arguments: Integer x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Attention data is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores attention level

Exception Handling: None

**Name: setFrustration(int x)**  
 Arguments: Integer x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Frustration is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores frustration level

Exception Handling: None

**Name: setCorret(boolean x)**  
 Arguments: Boolean x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: If question answer correctly is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores if the user answered question correctly

Exception Handling: None

**Name: setStartTime(long x)**  
 Arguments: long x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Time question is started is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores start time

Exception Handling: None

**Name: setStopTime(long x)**  
 Arguments: long x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Time question is finished is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores time question answered

Exception Handling: None

**Name: setProblem(String x)**  
 Arguments: String x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Question being answered is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores question

Exception Handling: None

**Name: setAnswer(String x)**  
 Arguments: String x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Answer is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores the answer the user inputted

Exception Handling: None

**Name: setToWrite(studData[] x)**  
 Arguments: studData[] x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: studData array is stored  
 Exceptions: None  
 Flow of Events:

1. Function stores studData array

Exception Handling: None

**Name: getAttention()**  
 Arguments: None  
 Returns: String  
 Pre-condition: User is logged in and data is being retrieved about him/her  
 Post-condition: Attention level is returned  
 Exceptions: None  
 Flow of Events:

1. Function returns attention level

Exception Handling: None

**Name: getStartTime()**  
 Arguments: None  
 Returns: Long  
 Pre-condition: User is logged in and data is being retrieved about him/her  
 Post-condition: Start time is returned  
 Exceptions: None  
 Flow of Events:

1. Function returns start time for question

Exception Handling: None

**Name: getStopTime()**  
 Arguments: None  
 Returns: Long  
 Pre-condition: User is logged in and data is being retrieved about him/her  
 Post-condition: Stop time is returned  
 Exceptions: None  
 Flow of Events:

1. Function returns time question was finished

Exception Handling: None

**Name: getCorrect()**  
 Arguments: None  
 Returns: Boolean  
 Pre-condition: User is logged in and data is being retrieved about him/her  
 Post-condition: Whether the question is answered correctly is returned  
 Exceptions: None  
 Flow of Events:

1. Function returns whether question is answered correctly

Exception Handling: None

**Name: getProb()**  
 Arguments: None  
 Returns: String  
 Pre-condition: User is logged in and data is being retrieved about him/her  
 Post-condition: Question user was asked is returned  
 Exceptions: None  
 Flow of Events:

1. Function returns question

Exception Handling: None

## 6.2 Tutor/Server Program

### 6.2.1 TutorProj

**Name: loadUser()**  
 Arguments: None  
 Returns: None  
 Pre-condition: Server just started and loading valid accounts  
 Post-condition: An array of registered users will be created  
 Exceptions: None  
 Flow of Events:

1. File of valid user accounts and profess mappings loaded
2. Accounts are parsed into Student objects
3. Student objects are stored in array

Exception Handling: None

**Name: retLogins()**  
 Arguments: None  
 Returns: Student Array  
 Pre-condition: Accounts have already been loaded from file  
 Post-condition: An array of registered users will be returned  
 Exceptions: None  
 Flow of Events:

1. Returns an array of users

Exception Handling: None

**Name: retCount()**  
 Arguments: None  
 Returns: Integer   
 Pre-condition: Accounts have already been loaded from file  
 Post-condition: Count of users will be returned  
 Exceptions: None  
 Flow of Events:

1. An Integer is returned that is the number of allowed users

Exception Handling: None

**Name: changePanel(JPanel x, String user)**  
 Arguments: JPanel x, String User  
 Returns: None  
 Pre-condition: Server already running  
 Post-condition: Adds panel for user who logged in  
 Exceptions: None  
 Flow of Events:

1. The function adds a JPanel for user that logged in
2. Function redraws frame

Exception Handling: None

**Name: deletePanel(String user)**  
 Arguments: String User  
 Returns: None  
 Pre-condition: Server already running  
 Post-condition: Removes panel for user who logged out  
 Exceptions: None  
 Flow of Events:

1. The function removes a JPanel for user who logged out
2. Function redraws frame

Exception Handling: None

**Name: actionPerformed(ActionEvent)**  
 Arguments: Action Event  
 Returns: None  
 Pre-condition: Server already running  
 Post-condition: Starts server, options are to display history of user logged in, or logged out  
 Exceptions: None  
 Flow of Events:

1. User clicked Begin Tutoring button to start server
2. User clicked history button to look through students past quizzes
3. User click logout and server stopped and program terminated

Exception Handling: None

**Name: windowClosing(WindowEvent e)**  
 Arguments: WindowEvent  
 Returns: None  
 Pre-condition: History information has been loaded  
 Post-condition: History frame will be closed  
 Exceptions: None  
 Flow of Events:

1. User clicked x in upper right corner of history window
2. History window closes

Exception Handling: None

### 6.2.2 chart

**Name: createPanel(String title)**  
 Arguments: String title  
 Returns: JPanel  
 Pre-condition: Program frame is already created  
 Post-condition: Blank chart is created  
 Exceptions: None  
 Flow of Events:

1. Function is called and JPanel is created and a chart is added

Exception Handling: None

**Name: createChart(Category Dataset, String title)**  
 Arguments: Category Dataset, String title  
 Returns: JFreeChart  
 Pre-condition: Program frame and JPanel already created  
 Post-condition: Chart is initialized and returned  
 Exceptions: None  
 Flow of Events:

1. Function creates JFreeChart object
2. Chart settings are initialized

Exception Handling: None

**Name: createDataset(int att, String corr)**  
 Arguments: Integer att, String Corr  
 Returns: CategoryDataset  
 Pre-condition: JFreeChart is already created  
 Post-condition: Student’s attention level and correctness is added to the data set of the chart  
 Exceptions: None  
 Flow of Events:

1. Series and category for data is initialized
2. Category data is set by String corr which represents if the question was answer correctly
3. String att is the attention level of the student
4. Value for y data is set

Exception Handling: None

**Name: createHistoryDataset(String[] data)**  
 Arguments: String[] data  
 Returns: CategoryDataset  
 Pre-condition: JFreeChart is already created  
 Post-condition: Historical data is added to the data set of the chart  
 Exceptions: None  
 Flow of Events:

1. Series and category for data is initialized
2. Category data is set by the zero position of the array which represents if the question was answer correctly
3. The one position is the attention level of the student
4. Tokens are used to parse the data

Exception Handling: None

### 6.2.3 dataBreak

**Name: returnVaue()**  
 Arguments: None  
 Returns: String Array  
 Pre-condition: Student has completed quiz sets  
 Post-condition: An array of quiz results  
 Exceptions: None  
 Flow of Events:

1. Function loads file and parses the data into tokens
2. String array returned

Exception Handling: None

**Name: returnAverageVaue(String y)**  
 Arguments: String y  
 Returns: String Array  
 Pre-condition: Student has completed quiz sets  
 Post-condition: An array of the student’s average results  
 Exceptions: None  
 Flow of Events:

1. Function loads file and parses the data into tokens
2. String array returned

Exception Handling: None

### 6.2.4 serverListen

**Name: startServer()**  
 Arguments: None  
 Returns: None  
 Pre-condition: Server not currently running  
 Post-condition: Creates a ExecutorService and attaches listening socket to it  
 Exceptions: None  
 Flow of Events:

1. Executor created with a limit of two threads (Only two connections)
2. Socket created and submitted to Executor
3. Thread started

Exception Handling: None

**Name: startData(student[] x, int y, TutorProj z)**  
 Arguments: Student Array, Integer y, TutorProj z  
 Returns: None  
 Pre-condition: Objects already created  
 Post-condition: Data set in method variables  
 Exceptions: None  
 Flow of Events:

1. Data set to method variables

Exception Handling: None

**Name: sendFile(String nameFile, Socket clientSocket)**  
 Arguments: String nameFile, Socket clientSocket  
 Returns: None  
 Pre-condition: Client software requests file  
 Post-condition: File sent to client  
 Exceptions: None  
 Flow of Events:

1. Request for file sent through serverListen
2. Function sends file over provided socket

Exception Handling: None

**Name: ClientTask(Socket clientSocket)**  
 Arguments: Socket clientSocket  
 Returns: None  
 Pre-condition: Server started and ready for connection  
 Post-condition: Client session placed in thread  
 Exceptions: None  
 Flow of Events:

1. Server starts and listening port placed in a thread
2. Thread handles all communication between client and server

Exception Handling: None

### 6.2.5 stuFileList

**Name: retFile()**  
 Arguments: String x  
 Returns: File Array  
 Pre-condition: Student has completed quizzes before  
 Post-condition: File Array of the students historical data  
 Exceptions: None  
 Flow of Events:

1. Returns a file array of a student’s historical data

Exception Handling: None

### 6.2.6 studData

**Name: setUser(String x)**  
 Arguments: String x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Username is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores username in method level variable

Exception Handling: None

**Name: setAttention(int x)**  
 Arguments: Integer x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Attention data is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores attention level

Exception Handling: None

**Name: setFrustration (int x)**  
 Arguments: Integer x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Frustration is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores frustration level

Exception Handling: None

**Name: setCorret (boolean x)**  
 Arguments: Boolean x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: If question answer correctly is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores if the user answered question correctly (true = correct)

Exception Handling: None

**Name: setStartTime (long x)**  
 Arguments: long x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Time question is started is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores start time

Exception Handling: None

**Name: setStopTime (long x)**  
 Arguments: long x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Time question is finished is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores time question answered

Exception Handling: None

**Name: setProblem (String x)**  
 Arguments: String x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Question being answered is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores question

Exception Handling: None

**Name: setAnswer (String x)**  
 Arguments: String x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Answer is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores the answer the user inputted

Exception Handling: None

**Name: setToWrite(studData[] x)**  
 Arguments: studData[] x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about them  
 Post-condition: studData array is stored  
 Exceptions: None  
 Flow of Events:

1. Function stores studData array

Exception Handling: None

**Name: getAttention()**  
 Arguments: None  
 Returns: String  
 Pre-condition: User is logged in and data is being retrieved about him/her  
 Post-condition: Attention level is returned  
 Exceptions: None  
 Flow of Events:

1. Function returns attention level

Exception Handling: None

**Name: getStartTime()**  
 Arguments: None  
 Returns: Long  
 Pre-condition: User is logged in and data is being retrieved about him/her  
 Post-condition: Start time is returned  
 Exceptions: None  
 Flow of Events:

1. Function returns start time for question

Exception Handling: None

**Name: getStopTime()**  
 Arguments: None  
 Returns: Long  
 Pre-condition: User is logged in and data is being retrieved about him/her  
 Post-condition: Stop time is returned  
 Exceptions: None  
 Flow of Events:

1. Function returns time question was finished

Exception Handling: None

**Name: getCorrect()**  
 Arguments: None  
 Returns: Boolean  
 Pre-condition: User is logged in and data is being retrieved about him/her  
 Post-condition: Whether the question is answered correctly is returned  
 Exceptions: None  
 Flow of Events:

1. Function returns whether question is answered correctly

Exception Handling: None

**Name: getProb ()**  
 Arguments: None  
 Returns: String  
 Pre-condition: User is logged in and data is being retrieved about them  
 Post-condition: Question user was asked is returned with correct answer  
 Exceptions: None  
 Flow of Events:

1. Function returns question

Exception Handling: None

**Name: writeData(studData [] data)**  
 Arguments: studData Array  
 Returns: None  
 Pre-condition: Student has completed a quiz  
 Post-condition: Student data written to file  
 Exceptions: None  
 Flow of Events:

1. Function creates file
2. Function writes student data to file

Exception Handling: None

**Name: readData(String x)**  
 Arguments: String x  
 Returns: String array  
 Pre-condition: Student has completed a quiz  
 Post-condition: Student historical data read from file and returned in array  
 Exceptions: ExceptionIO  
 Flow of Events:

1. Function reads file
2. Function parses data from file into String array
3. Function returns String array

Exception Handling: User is notified

**Name: averageData()**  
 Arguments: None  
 Returns: String array  
 Pre-condition: Student has completed a quiz  
 Post-condition: Student has completed at least one quiz  
 Exceptions: ExceptionIO  
 Flow of Events:

1. Data is received from client program with student data
2. Student data is averaged with historical data if any exists
3. Data is written to a file

Exception Handling: None

### 6.2.7 student

**Name: setUser(String x)**  
 Arguments: String x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Username is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores username

Exception Handling: None

**Name: setProf(String x)**  
 Arguments: String x  
 Returns: None  
 Pre-condition: User is logged in and data is being stored about him/her  
 Post-condition: Student’s professor is recorded  
 Exceptions: None  
 Flow of Events:

1. Function stores student’s professor

Exception Handling: None

**Name: getUser()**  
 Arguments: None  
 Returns: String x  
 Pre-condition: User is logged in and his/her username is being returned  
 Post-condition: Username is returned  
 Exceptions: None  
 Flow of Events:

1. Function returns their username

Exception Handling: None

**Name: getProf()**  
 Arguments: None  
 Returns: String x  
 Pre-condition: User is logged in and his/her professor’s name is being returned  
 Post-condition: Professor’s name is returned  
 Exceptions: None  
 Flow of Events:

1. Function returns their professor’s name

Exception Handling: None

# 7.0 Concurrent and Future Research

## 7.1 Grant

To continue and expand on the studio the JSU Faculty Research Grant 10000-30223-70000-20 was rewarded by Jacksonville State University for research. I will be working on both concurrently until system testing is complete.

### 7.1.1 Scope of Grant

The continuation of the project will have three components. These components are a research pilot, added functionality, and a conference or journal paper. The research pilot will focus on formal usability testing of the interfaces and effectiveness of the system. The pilot will have two volunteers who will use the system over several weeks. The goal for formal usability testing will be to test the ease of use of the system and look for areas for improvement. Verifying that the interface is easy to use will help the students focus on learning. Testing the effectiveness of the system will include testing students’ feelings about mathematics, their performance in their math class, and overall school performance over time (tbd). Qualitative data will be collected by using semi-structured interviews. This qualitative data will allow future research and improvements to students’ experience. In addition to the students’ opinions, the effectiveness testing will compare statistically the students’ performance in his or her class with the data collected by the system. In addition, a demonstration with educators is planned to show the system in use and survey their interest in the system. The added functionality will be instructor defined modules for mathematics and statistical analysis of the relationship between the students’ mathematical performance and EEG data while using the system. This will allow instructors to focus tutoring on specific topics for exams or areas where classes are struggling. Lastly, a conference paper to report findings from the research will be submitted to a conference in an appropriate domain.

# 8.0 Future Research

In the future, the system can be expanded to other domains (e.g. History, Psychology, Chemistry, etc.). Small modifications would be needed to add these domains to the system, but they would need to be coded in java. Future improvements will focus on a web interface that allows updates to the tutoring system from the professor’s office. In addition, report generation of results will be completed. Also, tutor and administrator login will be included in future versions. Currently the performance of a student is recorded, but an analysis of his or her progress would help professors and instructors focus on problem areas. Lastly, long term longitudinal studies will help verify the validity of the system and lead to improvements.

# 3.0 References

Blue, Eric. (2011). *MindStream-NeuroSky EEG Data Streamer*. Eric Blue’s Blog. Date Retrieved October 31, 2013. URL: <http://eric-blue.com/2011/07/24/mindstream-neurosky-eeg-data-streamer/>.

Bray, Mark. (2007). *The Shadow Education System: Private Tutoring and its Implications For Planners*. UNESCO: International Institute for Educational Planning. Date Retrieved Jan. 23 2013, URL: <http://unesdoc.unesco.org/images/0011/001184/118486e.pdf>

Crowley, K., Sliney, A., Pitt, I., Murphy, D. (2010). *Evaluating a Brain-Computer Interface to Categories Human Emotional Response*. 10th IEEE International Conference on Advanced Learning Technologies. Date Retrieved Jan. 21 2013, URL: <http://www.neurosky.com/Documents/Document.pdf?DocumentID=7cd1b6a1-8bb9-42af-bbf1-5cc0761d403f>

Emotiv. Date Retrieved: Oct. 9 2013. URL: <http://www.emotiv.com/>

Evans, R., James, Abarbanel, Andrew. (1999). *Introduction to Quantitive EEG and Neurofeedback*. San Diego, CA: Academic Press.

Gittis, Alan. (2005). *The Measurement of Brain Waves*. Date Retrieved: Oct. 10. 2013. URL: <http://www.psych.westminster.edu/psybio/BN/Labs/Brainwaves.htm>

Kahneman, D. (1973). *Attention and Effort*. Prentice-Hall Inc. New Jersey. Date Retrieved: Jan. 21 2013. URL: <http://www.princeton.edu/~kahneman/docs/attention_and_effort/Attention_lo_quality.pdf>

Lutsyuk, N., V., Eismont, E. V., Pavlenko, V. B. (2006). *Correlation of the Characteristics of EEG Potentials with the Indices of Attention in 12- to 13-Year-Old Children*. (2006, Feb). Neurophysiology, Vol. 38, No. 3. Date Retrieved: Feb. 2 2013 URL: <http://www.pfl.crimea.edu/art_32.pdf>

Napoli, M., Krech, P., & Holley, L. (2005). *Mindfulness Training for Elementary School 43*

*Students: The Attention Academy*. Journal of Applied School Psychology, 21(1), Date Retrieved Feb. 19, 2013 URL: <http://www.mindfuleducation.org/attention%20academy.pdf>

NeuroSky. (2010). *NeuroSky Store: MindBlaster*. Coding4Fun. Date Retrieved: October 31, 2013. URL: <http://store.neurosky.com/products/mindblaster>.

NeuroSky. (2010). *NeuroSky Store: Puzzlebox Orbit*. Puzzlebox. Date Retrieved: October 31, 2013. URL: <http://store.neurosky.com/products/orbit>.

Miller, H., P., Weiss M., G., (1981, Dec). *Children’s Attention Allocation, Understanding of Attention, and Performance on Incidental Learning Task*. Wiley, Society for Research in Child Development. pg. 1183-1190. Date Retrieved: Jan. 21 2013. URL: <http://www.jstor.org/stable/1129505>

Milo, Moryt (2013, 2013) *NeuroSky evolves from producing toys to providing treatment*. Silicon Valley Business Journal. Date Retrieved: Oct. 9 2013 URL: <http://www.bizjournals.com/sanjose/print-edition/2013/03/29/neurosky-evolves-from-producing-toys.html>

Szafir, Dan., Mutlu, Ban. (2012) *Pay Attention! Designing Adaptive Agents that Monitor and Improve User Engagement*. CHI 12. Date Retrieved: September 12, 2013.