Spring 2019

Concept Building Web Application

Jaime Becker

Concept Building Web Application

Jaime Becker

May 8, 2019

Presented to faculty of Rollins College in fulfillment of requirements for Honors in the Computer Science Major.

ABSTRACT:

In this thesis, I have ported a python software application that is used for the collection of information for functional learning classifications into a web application. I have analyzed the problems in the previous software and created the new web application to address these problems using Flask, AppEnginge, and Bootstrap. This web application mirrors the original software in look and feel but adds several functionalities and improvements like support for multiple input modalities and a separate administrative and user view. I also present a proposed evaluation of the web application.

1. Introduction

1.1. While the need for degrees in science, technology, engineering and mathematics (STEM) grows, Rollins College, along with many other universities in the United States, face high drop-out rates in these majors. In order for professors to adjust their curriculum for students who exhibit low grades in the introductory courses, interest has arisen in the variables that would predict students' success in STEM majors. Frey et al (2017) used a concept-building software to examine the variables that could predict a student's success in Chemistry. They found that there is a correlation between a student's performance in Chemistry and the score they received in the task in the concept building software.

The concept-building software has been used to abstract several rule-learning models in functional learning. Functional learning is learning based on stimulus and response patterns. Rule-learning models are derived from the user's ability to map the stimulus to the correct response based on a particular rule they have learned over time. If students are accurately able to determine the particular rule, they are classified as an abstract learner; those who cannot determine the rule are classified as exemplar learners.

In a software system developed to classify students as either as an abstract or exemplar learner, the user is shown a bar graph labeled with the stimulus at a predetermined height. They are also given an empty bar graph, in which they must use to predict the response. The particular stimulus-response pair is referred to as a "cover story". Once they mark their predicted response on the bar graph, they are shown a third bar graph which shows the correct height of the response, which is derived from a predetermined formula based on the stimulus. The user is given four seconds to observe the third bar graph with the correct result and then the bars reset to their original state and a new trial begins with a new stimulus level.

This cycle continues for a predetermined set of trials. The information that is gathered each cycle is the user's prediction, the distance from the correct response, and the time it took for the user to make the prediction. Based on results, students can be classified as abstract or exemplar-based learners. Frey, et al. (2017) found that students who classified as an abstract learner tended to receive high grades in the introductory Chemistry classes than the students who were classified as exemplar learners.

1.2. Frey et. al.'s work (2017) is a pilot test for this software, and it leaves many possibilities for future work, such as its ability to predict success in majors other than Chemistry and the possibility of professors to adjust their curriculum and offer help to those who are exemplar-based learners in order for these students to see success.

However, there are many issues that arise when using this software:

1. Timeliness of the exam: as discussed previously, there is a predetermined set of trials for the task. For the version of the software Rollins uses, student must go through 260 trials and only certain blocks of trials are used to calculate the student's score. Because there are so many trials, this exam takes students 45-60 minutes to finish the exam if they are examining each trial with care. However, it's been noticed in several studies that some students become bored and simply hurry through the rest of the exam without trying to make correct predictions in order to complete it faster.
2. Format of the software: the software is a python script run through Pygame that must be downloaded onto each computer. Downloading the software to each computer takes time and requires constant assistance from the IT department.

These issues inhibit schools from using the software to collect further data to understand if this learning phenomenon applies to other majors as well.

1.1. In order to resolve these issues, I am first replicating the software to a web application. This will allow the exam to be administered from any computer without the need to download anything. I am using Flask, which is a framework written in Python, to recreate the back-end functionality of the software. I am using HTML, JavaScript, and Bootstrap to replicate the software's front-end design.

Once the software is completely replicated, I move into the issue of making the task faster to use. As described, the task is long and tedious, resulting in most of the data being thrown out due to lack of effort in the later trials. As the software currently exists, students use a combination of the arrow keys to make their prediction and the enter key to submit their answer.

An evaluation study will compare the efficiency and accuracy of the task using the current keyboard controls and using only a mouse. Comparing these input methods will allow us to select a modality that lowers the time it takes to complete the task, in hopes that students will be able to focus for the full duration of the task.

I found this comparison to be an important aspect of the study because as I have found in the literature review, there seems to have been no recorded discussion of any of the design decision made while making the software.

2. Literature Review

Different versions of the software have been used in several psychology studies. Section 2.1 looks at the history of the software itself, how it has been used, and how it has been changed over time. Section 2.2 reviews studies done to test the efficiency and accuracy of different input modalities. The studies are focused on the mouse and keyboard as input devices, which is particularly relevant to this work.

2.1. History of the application in other studies

My work replicates the software used in Frey et al.'s work (2017) to study the concept building skills of Chemistry students. In this study, the software is used to predict whether or not students will succeed in Chemistry based on the type of learner they are assigned after taking the concept-building task through this application.

Delosh et al. (1997) developed variations of this particular software for use in their study to identify what best approximated the overall pattern of performance of the functional learning paradigm. This was the first recorded use of this software.

Delosh, et al (1997) reported on two relevant experiments. In Experiment One, the general progression of the software remained constant: participants first read the task instructions and bars from 1 to 100 (labeled every 10 units) were presented. Participants filled in the second bar to their desired prediction using the arrow keys and entered the value with the spacebar.

Several aspects of the software in the earlier Delosh, et al.'s (1997) work differed from the subsequent implementation in Frey, et al.'s (2017) work. The following list contains specific aspects of the software used in Delosh, et al.'s (1997) work that are different from the software used in Frey et al.'s (2017) work. A rationale was not presented on why these design changes were made:

1. Horizontal layout: the bar graphs that are presented to the user are horizontal rather than vertical as in Frey's system.
2. Feedback: after the initial training period and during the transfer trials, the software did not allow for the participant to see their feedback after they entered their desired value and sent them directly into the next trial. In the software used in Frey, et al.'s (2017) work, feedback is presented by displaying the difference between the user's prediction and the correct output after each trial.
3. Timed feedback: in the training trials that the participants were allowed feedback, they were able to view feedback for as long as they chose, rather than the four second time limit like in Frey, et al.'s (2017) work.
4. Cover story: the interpretation of the relationships consisted of drug dosage and arousal, rather than the absorption of fictional element "Zebon" and release of fictional element "Beros".

All other aspects of the software remained constant.

Experiment Two differed from Experiment One because Experiment Two was designed to "examine the reliability of [their] findings and to assess the generalizability of the results over variations in the procedural details of the task" (DeLosh, 974). In Experiment Two, the participants were first simply asked to learn each stimulus-response pair, but they were not told that there was any relationship between the two. The cover story was plant hormone and plant height, rather than drug dosage and arousal. The assigned function relationship between the plant hormone and plant height was changed from the assigned function relationship in experiment one by a constant of 20. All other aspects of the software remained the same from Experient One to Experiment Two.

While both Frey, et al. (2017) and Delosh, et al. (1997) state specifically that the participants used keyboard arrow keys, the spacebar, and the enter button to control the software, neither of them say why they chose to use the keyboard as input rather than a mouse. From this, I can assume that these decisions were either random, or believed to have no effect on the test itself.

2.2. Input modalities tested

As stated above, there was no rationale about the design decisions of the software in either study. Neither Frey, et al. (2017) nor DeLosh, et al. (1997) discussed why they chose to use the keyboard arrow keys, rather than the mouse. I questioned the efficiency of using the arrow keys, rather than the mouse, and researched previous work done which compared these two input modalities.

Pickard's study (1994) questioned whether or not college students developed a personal preference in input methods (keyboard or mouse) used to select a menu item. He also questioned the accuracy of each input method given the experience of the user with the input methods by tracking the user's every keystroke and mouse movement as they selected a main menu item in the software application. He concluded that users do not develop a personal preference as they get more comfortable with the application. He also concluded that if there was a development of a preference in input devices, it must occur over a larger period of time and with factors that cannot be regulated. Also, and more importantly for my research, there was little difference in the accuracy of input devices, each having a standard error rate of about six inaccurate selections per test.While my research does not account for personal preference, it is important to note that his research did not show a significant difference in accuracy of the two devices.

Trewin's work (1996) explored the impact motor disabilities have on performance with different input devices, including the keyboard and the mouse. To test the performance of the keyboard, she examined errors such as the user pressing keys in addition to the intended key, the user failing to press two keys simultaneously, and the user missing the intended key all together.

To test the performance of the mouse, she examined errors such as the user's click length, the user's time between clicks, the user's ability to position the mouse, and the user's dragging and dropping errors. While this was pilot research, she concluded that the design of the experiment was successful in providing patterns of the difficulties that the subjects had but was unable to conclude exactly which input device proved more accurate.

Isokoski and Martin's work (2007) tested several different input modalities for a first person shooter (FPS) game. The input modalities included a regular mouse, arrow keys with a regular mouse, a trackmouse (a mouse with a moveable ball on the top), and an XBox 360 controller. They used a very small sample size of only six participants, who had all played FPS games before. The game consisted of targets which the players had to select using their given input devices. Isokoski and Martin found that the regular mouse alone was over twice as efficient as the game console controller. However, most of their data pointed towards the idea that the effectiveness of the input device depended on the extent of the participants previous training with the device. While my research does not use a trackmouse or an XBox 360 controller, this study does relate to my research as it involves the user selecting a very specific target when making the prediction. It is important to note that they found the mouse to be the most efficient input device. Also, it is interesting that they used both the mouse and keyboard together as a single input modality.

In 2009, Grunzweil and Haller published research which explored the most efficient method of software interaction for preschool children. They evaluated the efficiency of mouse and the keyboard arrow keys, which are the two input methods I will evaluate. The study consisted of several experiments, first being object selection. The first experiment resulted in a significant difference in both efficiency and accuracy. The mouse had an average of 11.39 more correct selections over the keyboard arrow keys. Experiment two evaluated horizontal movements; experiment three, object movements, all of which found the mouse to be the better method with higher efficiency and accuracy.

Grunzweil and Haller concluded that the mouse, overall, was a better method for applications for preschool students for performance and accuracy. A problem they found with the mouse was for when children needed to stop on a precise point, but they concluded that this could be a result of the developing motor skills in preschool students. While my research does not involve young children, this is important research to note because this will contribute to my hypothesis that the mouse will perform higher in efficiency and accuracy over the keyboard strokes during the chemistry exam.

Zabramski's work (2011) evaluates and compares the efficiency and accuracy of three different inputs to perform a line-tracing task. She compares a mouse, a pen, and a touch input. Although my research involves keystrokes rather than touch input, it's important to note that she concluded that touch input outperforms the mouse in terms of time to complete the task, but it performed below the mouse in shape recognition, meaning the mouse could be used to more accurately trace the shape.

From theses studies, I cannot confidently conclude which input modality will improve the efficiency and accuracy of the concept-building software. While the work done by Grunzweil and Haller (2009) and Isokoski and Martin (2007) supports the mouse as the superior input modality, Pickard (1994), Trewin (1996), and Zabramski (2011) were inconclusive. When replicating this software, it will be beneficial to include options for both the keyboard and the mouse to assess which input modality, if either, increases the efficiency and accuracy.

3. Current System

As discussed previously, no rational is given for the design decisions made for the software. Section 3 is a detailed description of the current software, as well as it's purpose and limitations.

3.1. Purpose (Concept Building Theories)

This software can categorize human concept-learning into two models: exemplar and abstract models. Users who fall under the exemplar model store characteristics of the stimuli they are presented and use those characteristics to form conclusions about new stimuli. When these users are faced with a new stimulus, they respond by relating the new stimulus with similar stimuli they have seen. For example, a student using the software is presented with an input of 45. They make their prediction and find out that the correct output of the function is 80. They store this information, and several trials later, they are presented with an input of 46. Remembering the input from before, they would make their prediction a prediction of around 80.

Users who fall under the abstract model, however, tend to focus on the underlying functions of the stimuli, rather than storing and comparing certain instances. When faced with new stimuli, these users respond by applying the learned concepts. For example, when this type of user is presented with multiple trials, they will instead attempt to figure out the function between each input and output, say F(x) = 2 * x. If they are then presented with an input of 45, they will apply the function they have determined from the previous trials and make a prediction of F(45) = 2 * 45, which is 90.

The software can also classify the user as a non learner. Frey, et al. (2017) had a high of a 49% non learner rate during their testing. The software is able to determine the markers of nonsustained effort on the task by tracking the time it took for the user to complete each trial, the variety of prediction values they submit, and a high error rate across training blocks. For example, they found that 73% of the non learners averaged a <2 second trial time, while the average for exemplar and abstract was 8 seconds (Frey 1192).

Frey, et al. (2017) determined that these models correlate to student's success on examinations in typical college chemistry courses such as Intro to Chemistry I, Intro to Chemistry II, and Organic Chemistry II. They found "consistent advantages for abstraction learners across the three courses" (Frey 1190). This conclusion can be advantageous to both students and professors because it allows professors to form their curriculum around all types of learners, and it allows students to have a better understanding of their type of learning style and help them consider whether Chemistry is the correct major to pursue.

The application is able to determine the type of learner by giving the user a series of trials in which they predict the output given a particular input. Section 3.2 examines the particular software that I am replicating and shows how a conclusion is made based on the results of the trials.

3.2. Python Application

This entire application is currently written in one long Python script. It uses Pygame, a library which is used specifically to run multimedia applications, to display graphical results.

The application begins by assigning the user an identification number, which will allow the researcher to organize and identify the participants and their respective trials.

Next, the instructions begin. The instructions are there to explain the fictitious scenario that the participant is in, the cover story, and how each trial will look. For this application, the participant is informed that they have just been hired by the National Aeronautic and Space Administration (NASA) and that a new organism has been found on Mars. This organism absorbs newly found element Zebon and releases the also newly found element Beros. The name of the elements and the Mars anecdote are all part of the applications cover story.

The participant is told that scientists know that there is a relationship between the amount of Zebon absorbed and Beros released by the organism, but they do not know the exact relationship. The participant is then given the objective of their task, which is to figure out how these elements are related.

These fictional elements are related through predetermined functions. The amount of Zebon absorbed is the input and the Beros released is the output. The functions are determined from trial to trial based on a set vertex number, which is eighty in this version of the application. The functions are determined as such:

- If the input value (the Zebon absorbed) is LESS than the vertex:
  - $F(x) = 2.2 * x + 13.5$
- If the input value is GREATER than the vertex:
  - $F(x) = 365.4 - 2.1992 * x$

The user is not shown the functions, as it is the objective to figure it out based on the trials that they are given. Their ability to figure out the underlying function contributes to their placement as an exemplar, abstract, or non-learner at the end of all of the trials.

The participant is then given instructions on how to complete each trial, which are given as such:

> *Two bar graphs will be shown on the screen. The first bar graph will display the amount of Zebon absorbed. From this, you are to predict the amount of Beros released on the second bar graph.In order to make your prediction, use the ARROW KEYS on the right hand side of the keyboard. Press the UP and DOWN*

ARROW KEYS to get yourself in the ballpark of your prediction, then use the RIGHT and LEFT ARROW KEYS to fine tune your prediction.

This is followed by an actual test trial, which allows the user to get comfortable with the controls before they begin the actual trials.

When they have completed the instructions and the sample trial, the user is begins the training blocks. The trials in these blocks are not recorded and are there so that the user can either learn patterns between the trials or develop a function based on the results of each trial. However, the user does not know that these trials are not recorded. The participants completes thirteen training blocks, which each consist of twenty trials.

Each trial is completed as such:

1. Participant is shown three bar graphs labeled Zebon Absorbed, Prediction, and Beros Released from left to right, respectively. The “Zebon Absorbed” bar graph is showing the specific level between one and two hundred. The “Prediction” and “Beros Released” bar graphs are empty.
2. Participant makes their prediction on the bar graph labeled “Prediction” using the arrow keys. As stated before, the participant uses the left/right arrow keys to move their prediction by one and the up/down arrow keys to move their prediction by a pre-set “jump value”, which happens to be ten in this application.
3. Once the user has placed the bar graph on their prediction, they press “ENTER”.
4. The “Beros Released” bar graph is filled to the correct height, and the participant is given feedback based on their prediction. The feedback given is the error between the correct height of the bar graph and what they predicted. The “Zebon Absorbed” and “Prediction” bar graphs remain unchanged in this window.
5. After four seconds, the system resets the bar graphs to the next trial, and the user begins this process again.

After each block, the user is given feedback based solely on the result of that block. The feedback given is the average error between their predictions and the correct amount over those twenty trials. Although the bar graphs go from 0 to 200, the inputs in the training trials (the height of the Zebon absorbed bar graph) range only from 60 to 99. This causes extreme repetition over the 260 trials in the training blocks, but it allows for the user to get an understanding of the relationship.

Once the user has completed all thirteen training blocks, the user is then unknowingly moved in the transfer block. The transfer block looks the same at the other trials, but these trials are recorded and used to determine the participant's placement as an exemplar, abstract, or non-learner. The transfer block is thirty six trials in a row. The inputs of these trials are much different, as they are there to test whether or not the participant picked up on the functions. These inputs include a few of the numbers the participant has seen before, but they are sprinkled in between large outliers such as 33 and 123.

The system is tracking the prediction error over these trials because in these trials, the user is shown inputs they have never seen before. When given a trial with these outliers, an abstract learner will be able to apply the functions he/she has learned. However, an exemplar learner will use the closest approximate trial they have seen to make their prediction.

This data is analyzed by an R script which determines which learner they classify as.

3.3. Workflow

This section illustrates how the system is used in a study from the side of the participant and the side of the researcher. This is important because it highlights some of the problems that I intended to fix with my application.

3.3.1. Participant Workflow

When a participant is selected to use the application, they must first be assigned a specific computer to run the application on (more about why this is important in section 3.3.2). Once they are assigned a computer with the necessary systems installed, they click through the instruction with the enter key, do the sample trial and begin the training blocks.

As mentioned above, the training blocks consist of thirteen blocks of twenty trials each, resulting in 260 training trials. The user must complete all 260 trials before even getting to the trials that determine what type of learner they are. After these blocks, they are taken to the transfer block, which is another 36 trials.

Once they have predicted the output of almost three hundred input values, the user has completed the task, and no further data is collected from this user.

3.3.2. Research Workflow

When a researcher wants to use this system, the first step is to reserve a computer lab. Each student must use a computer that a researcher supplies for them; they are not allowed to use their own personal computer because it will not have the application installed.

Once the researcher reserves a computer lab, he/she must contact IT to assist in downloading Pygame to each computer that the researcher wants to use. The software must be installed to each computer because Pygame is responsible for the graphical display of the application.

The researcher can then start the participant testing. The researcher must schedule each participant into a particular time window that they have reserved in the lab. When a participant completes all of the trials, a file is saved locally to the hard drive. This file is what the researcher is interested in; it contains all of the data collected from the transfer block trials as well as their placement as a learner. The researcher must individually retrieve each of the participants data files from the respective computer that they used immediately after the participant has finished because the file will be deleted if the computer is logged out or restarted.

Once the data is collected from the local computer, the researcher must then manually put the data file through an R script. This script analyzes the data and saves a file to the system with the participant's classification. The researcher can then use this information collected for the purpose of their study.

3.4. Problems

There are many problems in the current application and workflow that have been presented in section 3.2 and 3.3. The following are problems that stood-out from these sections:

1. Reliance on IT: As described in section 3.3.2, IT affects many parts of the researcher's study. First, the researcher needs IT to download PyGame on every computer so that the application can be run. Second, the researcher must abide by IT policies, which include deleting all files when logged off, restarted, etc. The researcher must collect the data immediately after the participant is done to prevent the data from being lost.

2. Limitation on flexibility of participant scheduling: As again described in section 3.3.2, the researcher must reserve time in a computer lab. It is important that every participant is able to attend this reserved window because if the computers are turned off, restarted, logged off, etc., PyGame and the data collected will be deleted off of the computer.
3. Timeliness of research workflow: Running a participant using the process described in section 3.3.2 is a lengthy process. The researcher must first try to reserve a computer lab for a minimum of two hours, which can be difficult to schedule in an academic environment with classes going on. They then must schedule a member of IT to come before this time slot to set up all of the computers, which can take up a lot of time depending on how many participants. The task itself takes about an hour to complete, the researcher must individually collect the data, and then must manually run every individual data file through the script. This process is unnecessarily time consuming for the researcher.
4. Potential loss of data collected: As touched upon in problem 2, there are many ways for participant's entire data to be lost. One case would be if the computer was logged off after the participant finishes the task. On Rollins computer lab computers, when a student logs out, all of the information on their account is deleted. Thus, logging out would delete the data file. This would also occur if the computer was shut down or restarted. This is a problem because the data file is only locally stored on the computer's hard drive with no backup version available. If a data file is deleted, that participants trials are gone.
5. Timeliness of trials for participants: As described in 3.3.1, participants are put through almost 300 trials over the entire task. This task can take up to an hour to complete, and each trial looks the same as the last, besides the change in input. The test is tedious and leads many students to get bored and stop trying. The script can detect when a student stops trying by the time between trials, the same prediction being used over and over, and the overall average error. The script will

classify this student as a non-learner, and the trials are basically useless to the researcher.

6. Keyboard use only: As again described in 3.3.1, the keyboard arrow keys are the only way the user can make a prediction, and once they are done with a trial, they hit the "ENTER" key. Every move the participant makes is done on the keyboard. There were no known input modality design choices made in previous work (see section 2.2), so this raises the possibility that the test may be less time consuming if the participant could use their mouse rather than their keyboard.

4. New System Design

4.1. System Modeling

The structure of the system I am replicating is relatively simple. The user is shown the level of "Zebon" on the leftmost bar graph and attempts to predict the level of "Beros" using the middle bar graph. Once the prediction is made, the correct level of "Beros" is shown to the user, as well as feedback regarding the difference between their prediction and the correct level of "Beros". This then repeats until all of the trials have been completed.

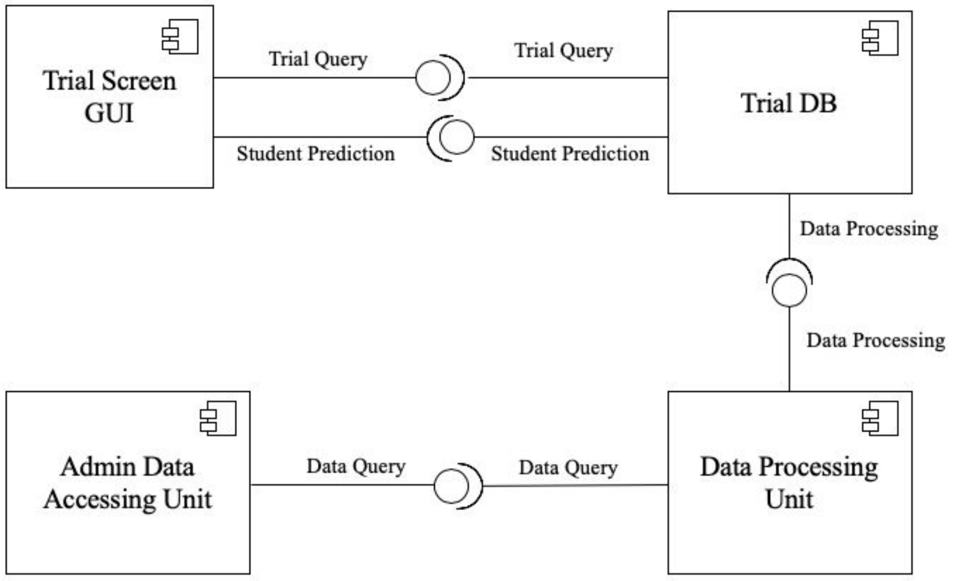


Figure A. Component Diagram of the Web Application.

To replicate this system, I used software models to capture the static and dynamic structure of the system. As seen in Fig. A, this software requires four main components: the Trial Screen Graphical User Interface (GUI), the Admin Data Accessing Unit, the Data Processing Unit, and the Trial Database (DB). The Trial DB is responsible for providing the GUI with the fictional levels of "Zebon" and "Beros" calculated from a given function. The GUI presents the user with the "Zebon" level before they make their prediction and the correct "Beros" level after they make their prediction. When the student makes their prediction, the GUI sends the prediction to the Trial DB to be stored.

The Data Processing Unit receives the trial information from the Trial DB and processes the information. This unit will calculate the error between the user's prediction and the actual function output and will also make the calculation deciding whether the student is an exemplar- or rule-based learner. After all the trials are complete, the Admin Accessing Unit allows the admin to retrieve the calculations made by the Data Processing Unit.

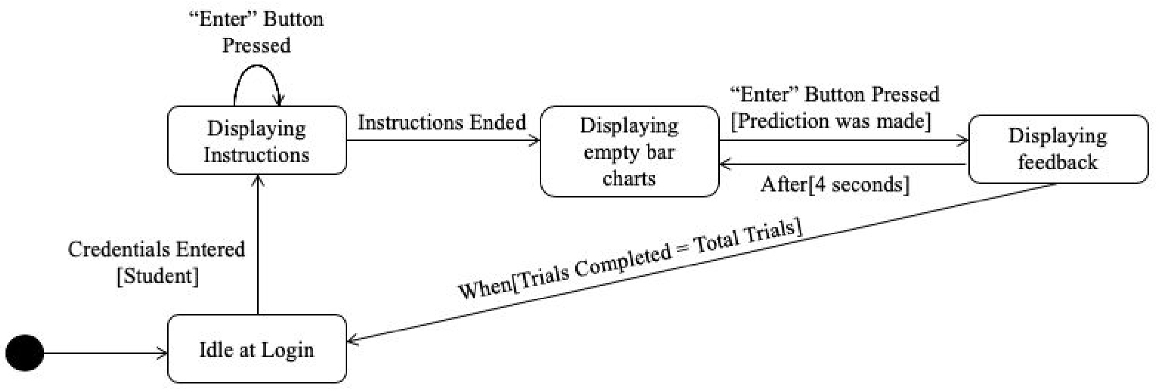

"Enter" Button Pressed
Displaying Instructions
Instructions Ended
Displaying empty bar charts
"Enter" Button Pressed [Prediction was made]
Displaying feedback
After[4 seconds]
Credentials Entered [Student]
When[Trials Completed = Total Trials]
Idle at Login

Figure B. State Diagram in User's View.

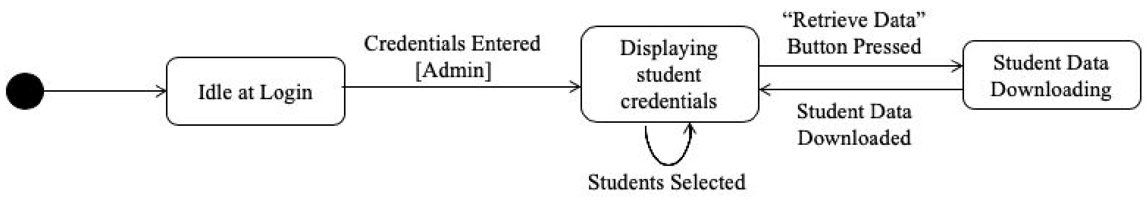

Idle at Login
Credentials Entered [Admin]
Displaying student credentials
"Retrieve Data" Button Pressed
Student Data Downloading
Student Data Downloaded
Students Selected

Figure C. State Diagram in Admin's View.

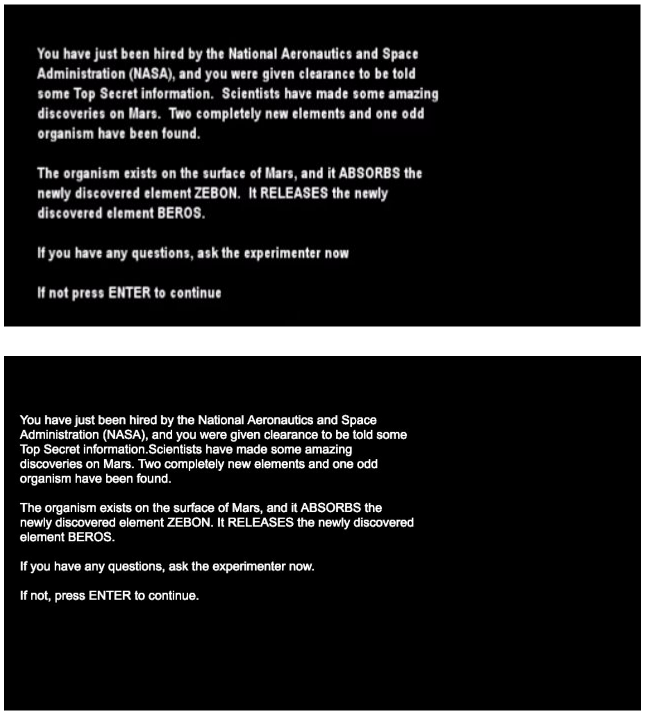
The functionality of the application and state interactions are very simple. Fig. B shows the states of the web application from a user's perspective. If student credentials are entered, the application will then display the instructions, and then display the beginning of the trial, where the "Zebon" level is displayed on a bar graph, but the prediction and "Beros" bar graph levels are blank. The user enters their prediction, which changes the state to displaying the feedback to the user. After four seconds, the feedback disappears and the next trial begins. When the user completes all of the trials, the application is reset back to the login page.

Fig. C shows the states of the web application from an admin's perspective. If admin credentials are entered, the admin is taken to a page that allows them to select the desired student's and download all of their information regarding the trials and their placement as a learner.

4.2. Replication Look & Feel

I use the Flask web framework, which is written in python, to handle the back-end functionality of the system ("Flask", 2019). I used the App Engine on the Google Cloud Platform to test the progress of the application, both in functionality and design ("Cloud Computing Services", 2019). I stayed as close as possible to the design of the original software application. I use a combination of HTML and Bootstrap to maintain similar fonts, colors, and positioning ("Bootstrap", 2019).

Fig. D shows the first page of the instructions from the software I am replicating (top) and the web application I created (bottom). I remained consistent with the spacing and capitalization of the words in the instructions. The instructions consist of five more pages, one being an example of what each trial will look like. This example allows the user to practice using their given input modalities on the bar graph.


You have just been hired by the National Aeronautics and Space Administration (NASA), and you were given clearance to be told some Top Secret information. Scientists have made some amazing discoveries on Mars. Two completely new elements and one odd organism have been found.

The organism exists on the surface of Mars, and it ABSORBS the newly discovered element ZEBON. It RELEASES the newly discovered element BEROS.

If you have any questions, ask the experimenter now

If not press ENTER to continue

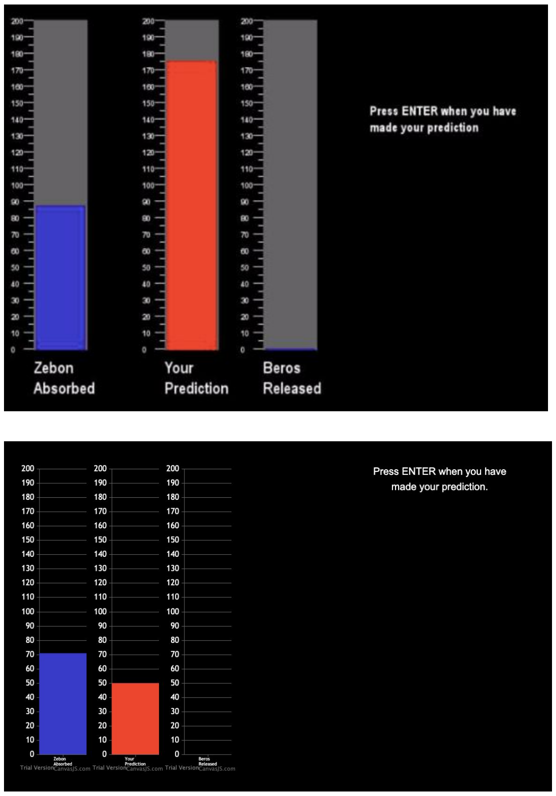
You have just been hired by the National Aeronautics and Space Administration (NASA), and you were given clearance to be told some Top Secret information.Scientists have made some amazing discoveries on Mars. Two completely new elements and one odd organism have been found.

The organism exists on the surface of Mars, and it ABSORBS the newly discovered element ZEBON. It RELEASES the newly discovered element BEROS.

If you have any questions, ask the experimenter now.

If not, press ENTER to continue.

Figure D. Instructions in the Original Software (top) and Replication Web Application (bottom).



Press ENTER when you have made your prediction

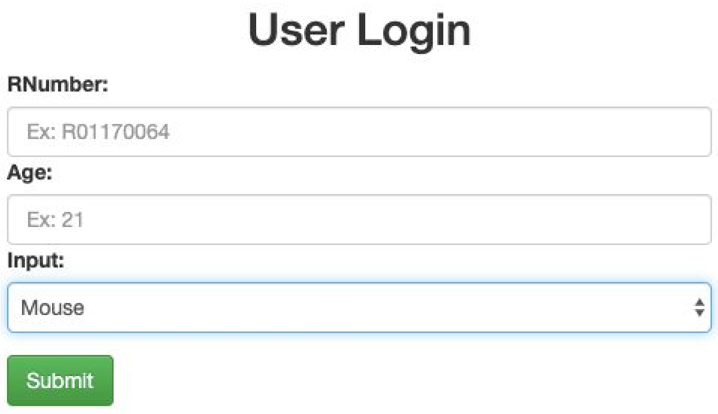
Press ENTER when you have made your prediction.

Figure E. Example of a Trial in Original Software (top) and Replicated Web Application (bottom).

As can be seen in Fig. E, the bar graphs are laid out in the same positioning: leftmost being “Zebon Absorbed”, middle being the user’s prediction, and rightmost being “Beros Released”. The coloring of the bar graphs are similar, as the “Zebon Absorbed” graph is a shade of blue and “Your Prediction” is red. When the user enters their prediction, the “Beros Released” graph displays the correct level in the same shade of blue as the “Zebon Absorbed” graph. The instructions for each trial on both applications are displayed to the right of the bar graphs.

4.3. Additions to New Application

To address the problems listed in Section 3.4, there were many additions to the new web application. First, I have added application a login page, as one was not needed in the original software. I created a dynamic login page which allows the user to submit relevant information based on if they are going to complete the task as a participant or if they are an admin looking to retrieve information. The user looking to complete the task must fill out their “RNumber”, their age, and the input modality they are using (as shown in Fig. F). The admin must submit the correct password in order to go to the admin page. The log-in page allows for problem 4 (listed in section 3.4) to be solved. Problem 4, potential loss of data, is solved because the user’s information will be connected in the database to the data that’s collected during the task, which is accessible to the researcher.


User Login

RNumber: Ex: R01170064

Age: Ex: 21

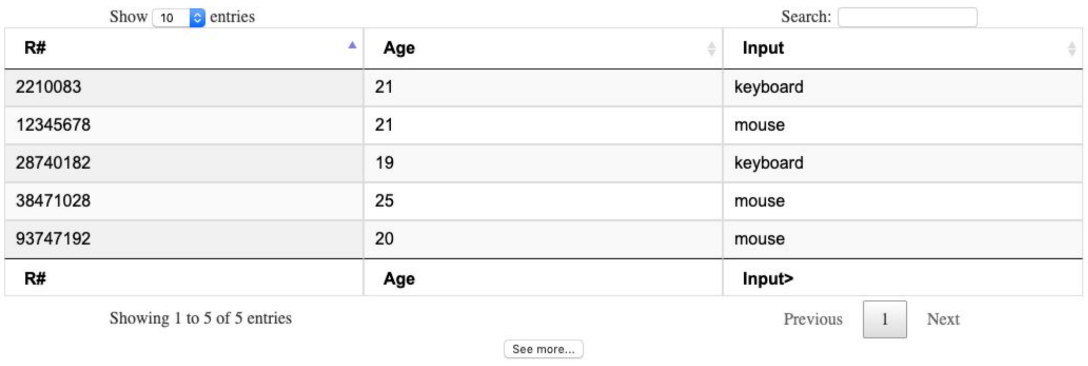
Input: Mouse

Figure F. User Login Page of Web Application.

The input modality selection box (the bottom input box in Fig. F) of the login page gives the user the choice of using the mouse or the keyboard throughout the task. For example, if they select mouse, then the participant will click through the instruction pages using the mouse, will use the mouse to make their prediction on the prediction graph during the trials, and will use the mouse to click to the next trial. The keyboard will not be needed at all with this selection. This addresses problem 6, keyboard use only, (described in section 3.4) because the participant, or the researcher, is able to select which type of input modality they want to use, and they are no longer restricted to just the keyboard.

The admin page is also a new feature from the original software. In the original software, for the researcher to acquire the user's data, the data file, which is saved locally to the computer being used, must be manually saved onto an external hard drive. This problem is described in detail in section 3.3.2, which shows the researcher's workflow.

The admin page in my web application allows the admin to search for the user(s) they want to retrieve information about. I used the Data Table API in JavaScript to create a dynamic table of every student stored in the database. The admin can search by "RNumber", age, or input modality. Fig. G shows an example of what the admin page would look like filled in with example user data.



R#	Age	Input
2210083	21	keyboard
12345678	21	mouse
28740182	19	keyboard
38471028	25	mouse
93747192	20	mouse
R#	Age	Input>

Showing 1 to 5 of 5 entries

Figure G. Admin Page of Web Application.

4.4. Updated Workflow

Given the new additions to the application, the workflows of the researcher and the participant are drastically changed to address the issues laid out in section 3.4.

4.4.1. New Participant Workflow

When a participant is selected to use the application, they will be sent a
website url which directs the user to the new web application. The user can then complete the task at their own convenience on their own laptop. The researcher will either inform the user of the input modality they want the participant to use, or they will be allowed to choose which input modality they are most comfortable with.

Once they have completed all of the trials, the web application will reset, and the participant is done with the task.

4.4.2. New Research Workflow

When a researcher wants to use the system, they no longer have to reserve
computer lab, as participants are able to use the application on any computer. The researcher could even send the website's URL to each of the participants.

As participants complete the task, the system's database will populate with
the participant's data. When the researcher wishes to access this information, they must log into the admin page using the correct password. The admin page will then allow the researcher to select which participants (based on either their "RNumber", age, or input modality) and download the data collected.

5. Proposed Evaluation

5.1. Reason for Evaluation

Frey, et al.'s (2017) work concludes that the system does accurately predict the student's success on exams in chemistry courses. Also in Frey, et al.'s (2017) study, they received a tremendous amount of non-learners (49%), which caused much of the data collected to be unusable. This is explained by the idea that participants get bored from the long task and stop making an effort to make accurate predictions. The task can be too long and tedious for some participants to remain focused.

The incredible amount of non-learners raises the issue that the test may take too long to complete. The large number of trials are necessary for the user to attempt to learn the underlying function. If we cannot reduce the number of trials, how can we make this task shorter, in an effort to keep the attention of non-learners.

As discussed in Section 2.2, there were no recorded design decisions for the application; they do not give reasons why the keyboard is the only way of operating the software. The reason I added the mouse as an alternative input method is to test whether or not the mouse is faster and more efficient than the keyboard in completing the trials. Time did not allow for me to move forward with user testing, but I have included a proposed evaluation method, which is described in the next section.

5.2. Possible Evaluation Method

To test which input method is more efficient for the application, I propose that a study is conducted with two large samples of students. The students can be of any major, as this study is not looking for their success in chemistry like previous studies. The sample sizes must be large because even though I expect the number of non-learners to decrease, there were so many classified as non-learners in previous studies that I do not expect them to disappear all together. We need to account for this by adding more students, so that we have potentially more data to analyze.

Group 1 of students would take the test using the mouse option while Group 2 would take the tasking using the keyboard option. Each group would complete the task and two aspects would be analyzed from each participant: the time it takes to complete each trial and the time it takes to complete the entire task.

From this data, we could draw a conclusion as to which input modality increases the efficiency of the task, if any, as well as whether or not the number of non-learners is decreased for the mouse condition.

6. Discussion & Future Work

Replication of this software into a web application proves to have many benefits for both the researcher and and the participant. The web application allows for more convenient scheduling for the researcher, as it allows for participants to complete the task on any computer rather than having to reserve time frames in a computer lab. The researcher also does not have to rely on IT for set-up and data collecting because the web application does not require the installation of Pygame and also does not save the data locally.

However, there are limitations of the software that are many due to limited time. Below is a list of future work to be done on this software:

1. Implement the database: One of the most important problems that I aimed to solve by creating this web application was the potential loss of data. As discussed in Section 3.4, the data file created when the user completes the task has the risk of being deleted if the computer is restarted or logged off. This required the researcher to immediately retrieve the data file from the participants computer. However, the database allows all of the participant's data to be saved through the web application, without the risk of being deleted.. In the future versions of the web application, Fig. F shows how the data will be stored. Each student entity will have a unique ID, a Rollins College ID number, an age, and the input method they selected (mouse or keyboard). Each student participates in many trials, each of which stores a unique ID, the participant ID (p_ID), a trial number. There are also attributes of the relationship, which include all of the information about a specific trial.

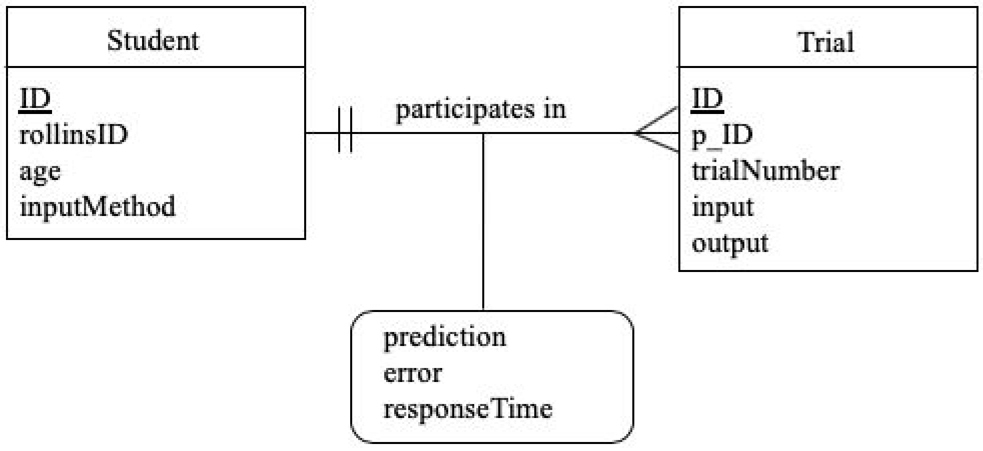


Figure H. ER Diagram of the database used to store participant information and trial results.

2. Ability to download data files and run them through the R script for analysis: With the implementation of the database, we have all of the data together in one spot, but that does not help the researcher analyze the data. Through the admin page, the researcher can currently view the data about the participants who completed the task. Moving forward, a system should be implemented in which the researcher can select participants and download either a raw data file from their task or a file that has been run through the R script which analyzes the data and classifies them as their respective type of learner.

After these systems are added to the web application, the next stage is to run the the evaluation that I proposed in Section 5.2. This evaluation would be conducted in hopes of discovering which input method, whether it be the keyboard or the mouse, is more efficient. Efficiency is important in this task because as the task can take an hour to complete, student struggle to remain persistent through the entire task. If a more efficient input method was found, this could lead to students remaining focused throughout the entire task, decreasing the number of non-learners and increasing the amount of usable data. Further work is needed to know if data collected from a keyboard and data collected from a mouse are comparable.

7. Conclusion

In this thesis, I have presented:

- An analysis of problems in the previous software used for collection information for functional learning classifications.
- A Web Application version of the software based on Flask, AppEngine, and Bootstrap which mirrored the original software in look and feel
- Several added functionalities and improvements in the Web Application including:
  - Support for multiple input modalities
  - Separate Administrative and user view
- A proposed evaluation of the Web Application

Going forward, with the implementation of a database and a built-in script analysis, an evaluation can be done to determine whether the mouse or keyboard is more efficient and whether or not the input modality decreases the percentage of non-learners. Regardless of the findings on input modalities, the Web Application can be used to collect data to further study of functional learning classifications.

References

(2019, April 17). Flask(A Python MicroFramework). Retrieved from http://flask.pocoo.org/.

(2019, April 17). Cloud Computing Services. Retrieved from https://cloud.google.com/.

(2019, April 17). Bootstrap - The most popular HTML, CSS, and JS library in the world. Retrieved from https://getbootstrap.com/.

DeLosh, E. L., Busemeyer, J. R., & McDaniel, M. A. (1997). Extrapolation: The sine qua non for abstraction in function learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 23(4)*, 968-986. http://dx.doi.org/10.1037/0278-7393.23.4.968.

Frey, R. & Cahill, M. & Mcdaniel, M. (2017). Students' Concept-Building Approaches: A Novel Predictor of Success in Chemistry Courses. *Journal of Chemical Education.* 94. 10.1021/acs.jchemed.7b00059.

Grünzweil, B. & Haller, M. (2009). Analyzing Interaction Techniques Using Mouse and Keyboard for Preschool Children. In Proceedings of the 5th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for e-Inclusion (USAB '09), Andreas Holzinger and Klaus Miesenberger (Eds.). *Springer-Verlag*, Berlin, Heidelberg, 448-456. DOI=http://dx.doi.org/10.1007/978-3-642-10308-7_32

Isokoski, P. & Martin, B. (2007). Performance of input devices in FPS target acquisition. *Proceedings of the international conference on Advances in computer entertainment technology (ACE '07)*. ACM, New York, NY, USA, 240-241. DOI=http://dx.doi.org/10.1145/1255047.1255104.

Pickard, S. (1994). College Students' Preference of Computer Input Device: Keyboard Versus Mouse. (Ph.D. Dissertation). University of North Texas, Denton, TX, USA. UMI Order No. GAX94-24396.

Trewin, S. (1996). A study of input device manipulation difficulties. *Proceedings of the second annual ACM conference on Assistive technologies (Assets '96)*. ACM, New York, NY, USA, 15-22. DOI=http://dx.doi.org/10.1145/228347.228351.

Zabramski, S. (2011). Careless touch: a comparative evaluation of mouse, pen, and touch input in shape tracing task. *Proceedings of the 23rd Australian Computer-Human Interaction Conference (OzCHI '11)*. ACM, New York, NY, USA, 329-332. DOI=http://dx.doi.org/10.1145/2071536.2071588.