Year 12 SDD
Project Proposal and Design Documentation
A^3

Defining the problem

1. Identification of the Problem

Going through a past paper with the class as a teacher isn't very easy. You have to either print a copy of it for every student, show in on a whiteboard with no real or convenient way to gather answers, or spend time preparing a software by manually writing down the questions and manually giving the students options. All of these options are inconvenient and cost either time or money. A simple software of gathering answers from students and displaying them in an easy to understand and easy to individualize the answers is the goal of this software.

There are existing programs that do similar tasks, but have problems and bugs, such as:

- Disconnecting and reconnecting resets the player's points and sometimes crashes the whole server
- Can't change answer
- Can't use keyboard to answer
- Slow to allocate points and collect answers

2. Functionality requirements

- Teacher to start a server with a GUI
- Students to connect to server using a client with a GUI
- Allow teacher to change student's display name
- The teacher to choose either True/False, Multiple choice, buzzer or one-word answer (the one word can be a number)
- The teacher to choose how long students have to answer
- The students be able to answer, corresponding to the option given
- The teacher to stop receiving answers and the answers be displayed utilizing a graph
- Teacher to select correct answer and choose who to allocate points to
- Output all results out into an excel file, containing names, points and all answers ordered by the question number and clients.

3. Generation of ideas/possible alternatives

I had an idea to give other answers like a short answer question or a diagram answer, but that would make no difference as to do without the program, so it's useless. I had other ideas like giving answers such as "Higher" or "Lower", but for that the question can be just changed so that true would be higher and false would be lower.
In the GUI, I can add multiple little things, like a question counter or a timer, that will help the student.

The existing software, SRN, has 3 boxes that allow you to communicate in different ways: raise hand, yes or no and multiple choice. These 3 answering options give flexibility to answering a lot of questions, but there are more options that can be used, such as a one-word or one-number answer. Especially for mathematics questions, where the answers should all be somewhat similar, and if there are multiple solitary answers, they'll be visualized in the graph as "other". This increases the amount of flexibility of the program.

4. Communication with others involved in the proposed system

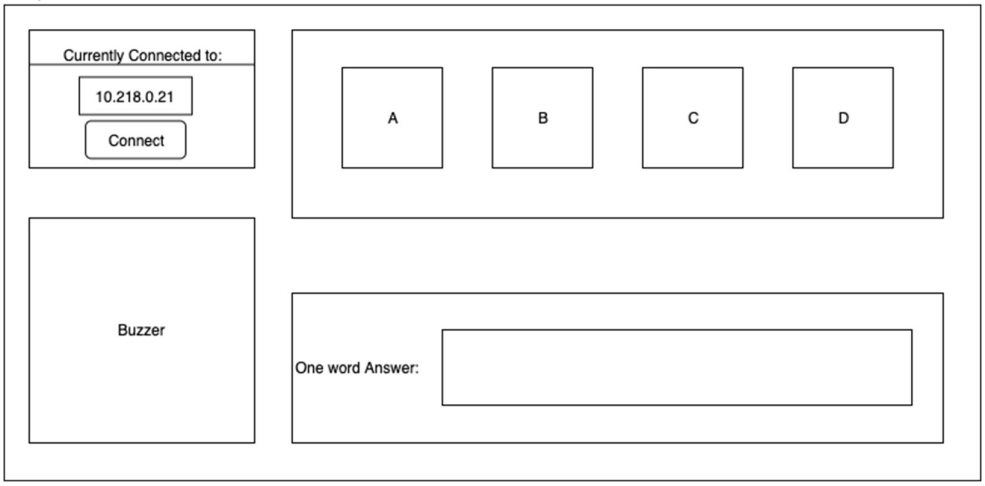
All developers will follow the given schedule, but work on different modules of the program. For example, while developing the main program, one developer will work on the networking, while the other works on the answering and displaying answers. Each developer will keep a logbook and submit a report to me at the end of every week via email.

Developers working on the base program will additionally partake in a fortnightly meeting to make sure that everything is going to plan, and will also do small compatibility tests to make sure basic parts work together.

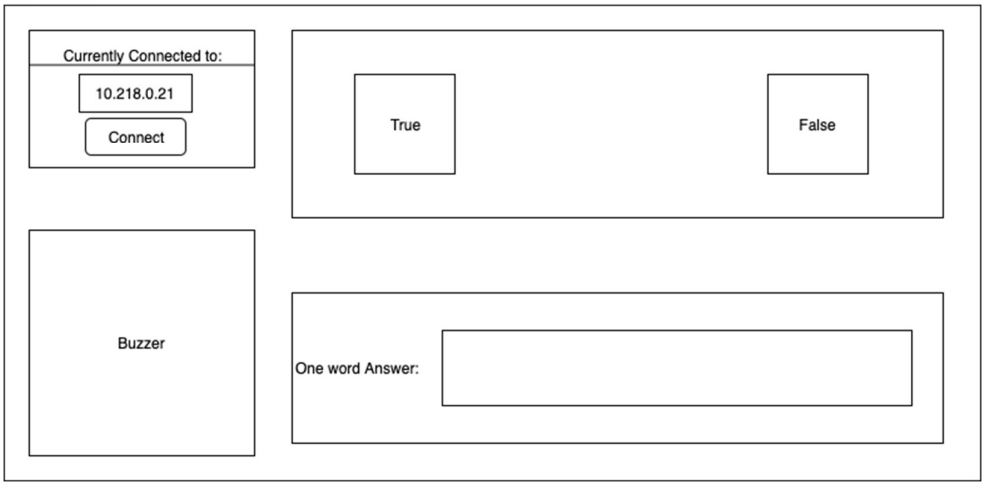
Understanding the problem

Client:

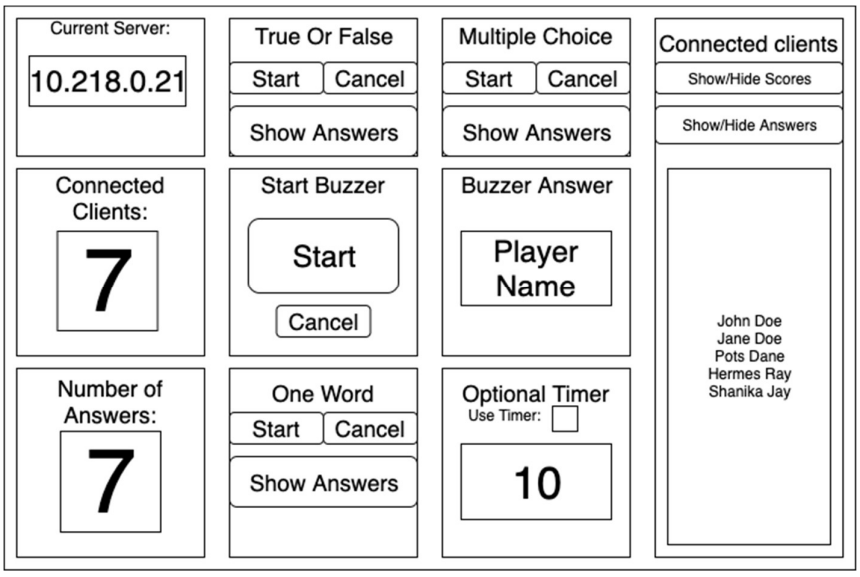
Multiple Choice:



True or False:



Server:



Explanation:

This is the interface of my proposed program. The first two diagrams show the same window, just the differences when the teacher/server changes the type of answer it wants. The one-word answer box is a textbox that allows one word/number to be typed in and submitted.

The server uses a grid type interface to separate the different modules/options. The connected clients, number of answers, optional timer, buzzer answer and connected client's list's contents will change according to the situation.

IPO diagrams:

From server

Input	Process	Output
True/False or multiple choice or buzzer start button	Sends instructions to all clients to allow answers	Allows clients to submit an answer
Text answer start button	Sends instructions to all clients to allow text answers	Allows clients to submit a text answer
All "show answers" buttons	Organizes data based on the answers and create a simple graph	A graph that shows the number of each answer
Show/hide scores button	Organize the clients based on scores and stop hiding them	The clients list becomes a leaderboard in terms of score
Show/hide answers button	Organize the clients based on the answer given and stop hiding them. Exempts text answers	The clients list becomes organized in terms of answers. Exempts text answers

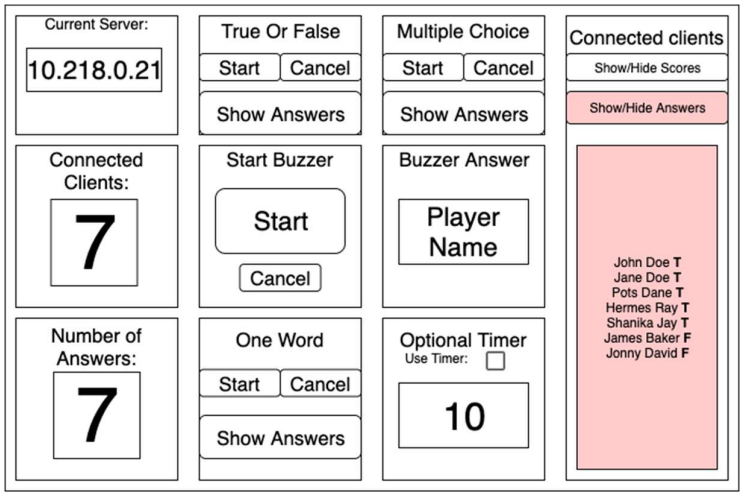
From clients

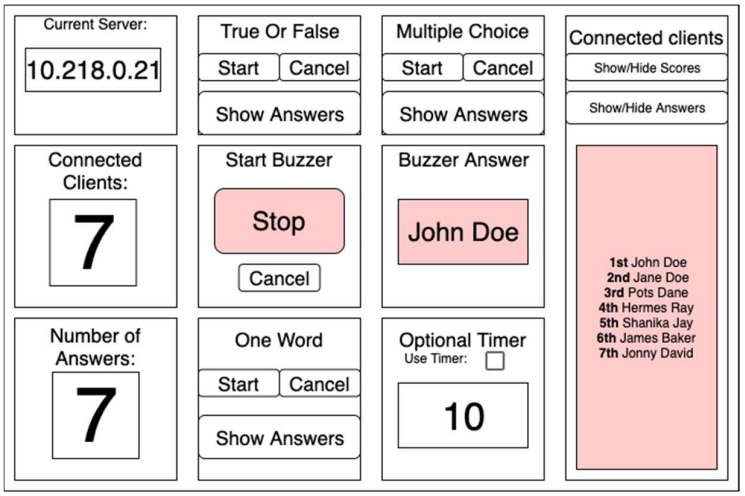
Input	Process	Output
True/False/multiple choice answer/buzzer	Send appropriate answer to the server Server stores the answer	Server receives and displays answer
Text enter	Sends the inputted text to the server Server stores the answer	Server receives and displays text answer

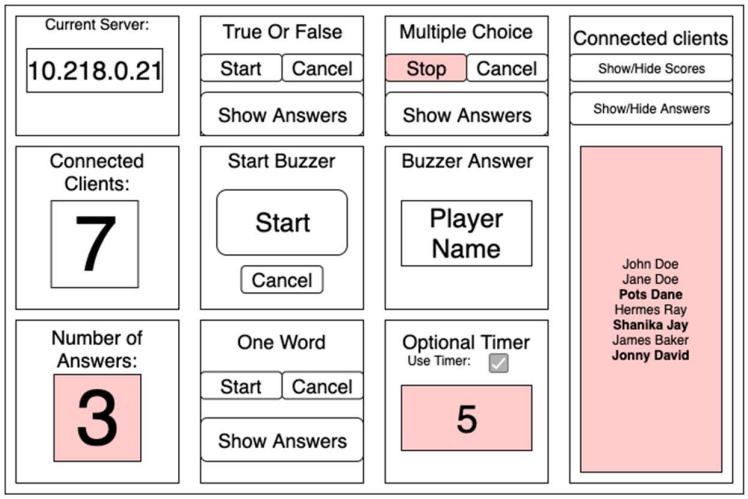
Storyboards

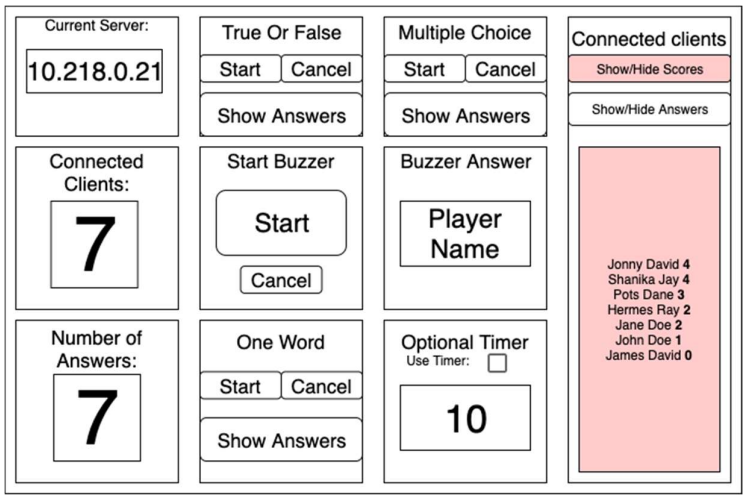
The highlighted parts are what changes based on the button pressed, which is also highlighted.

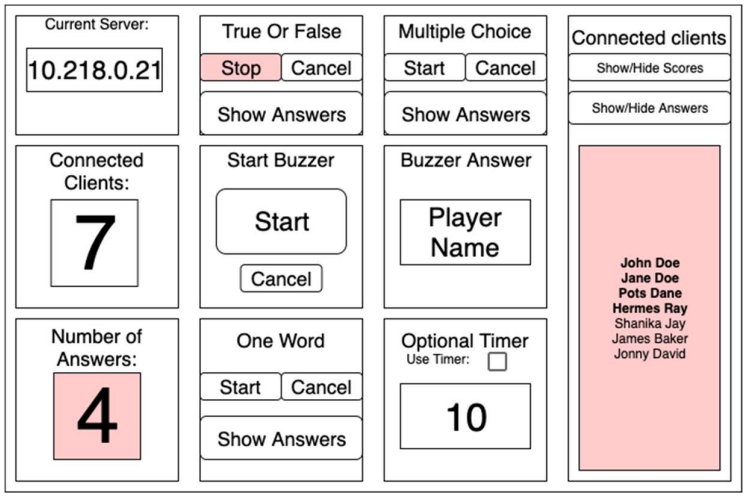
Server:



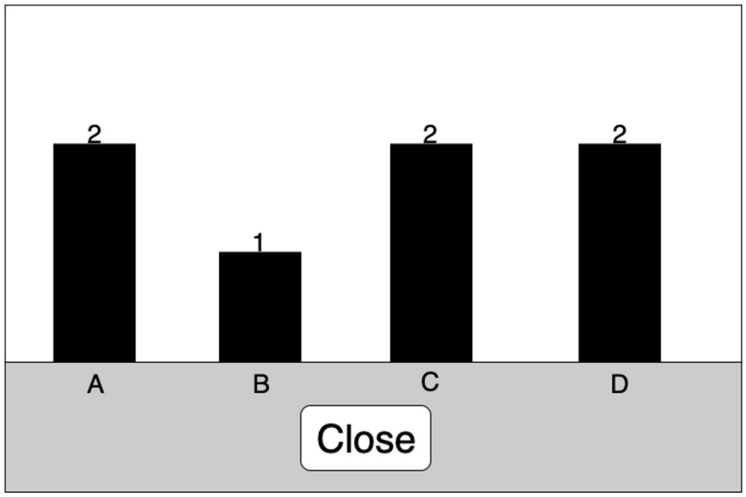




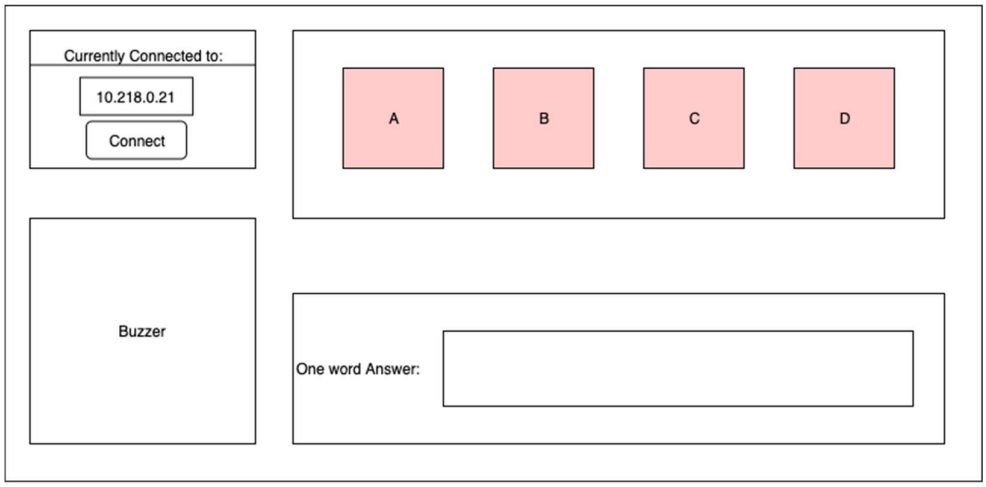


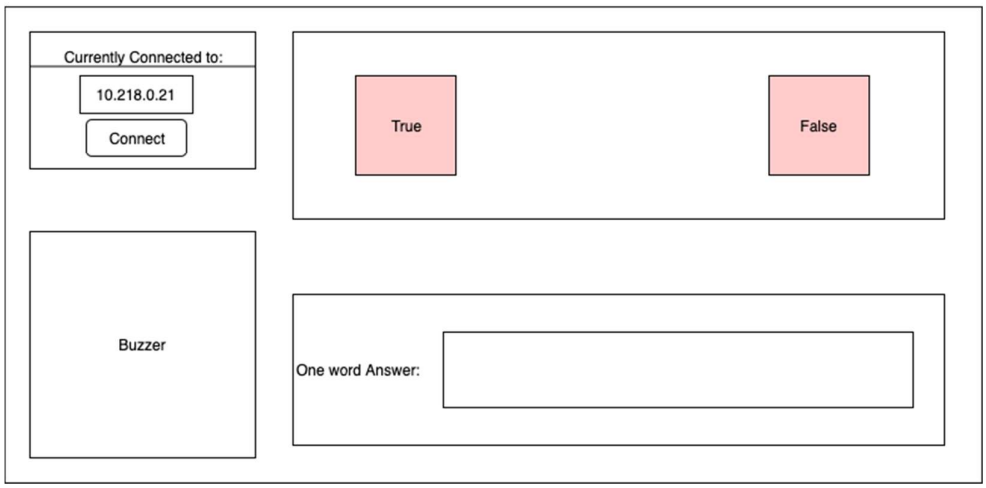


These are the graphs shown when you press “Show Answers” on the True/False, Multiple Choice and One Word.

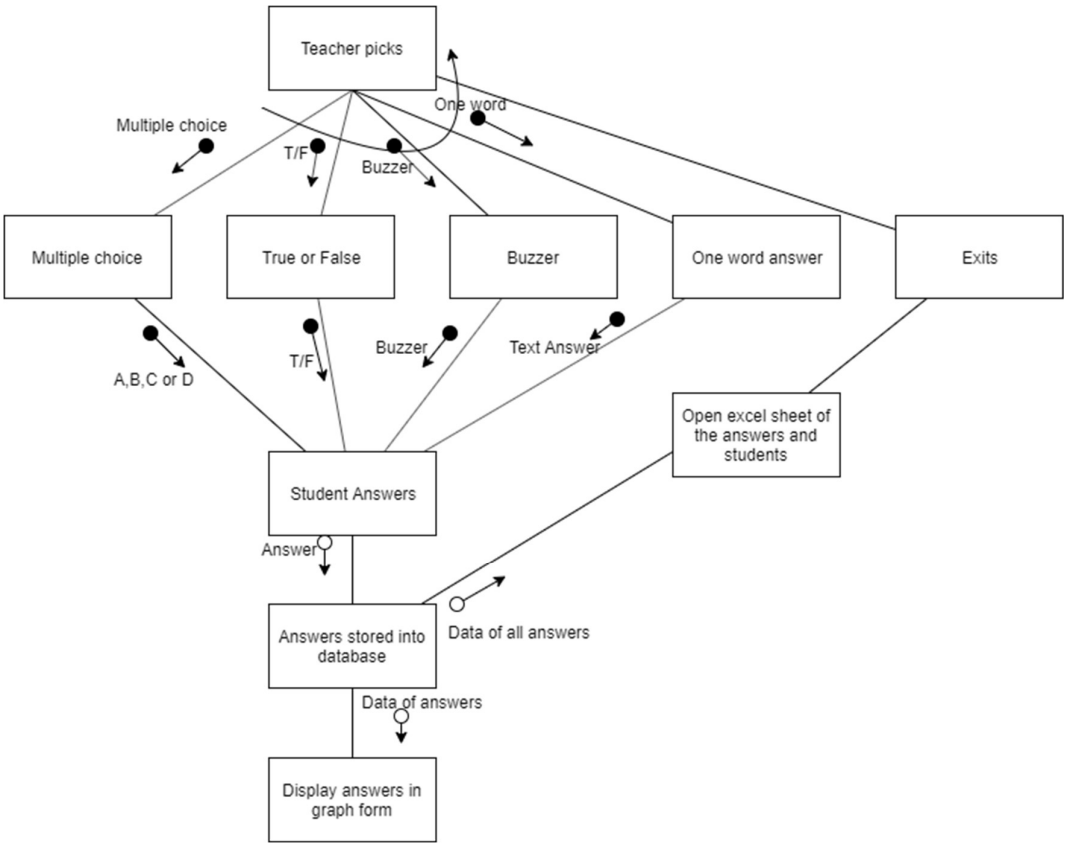


Client: (The interface changes based on the answer option given)

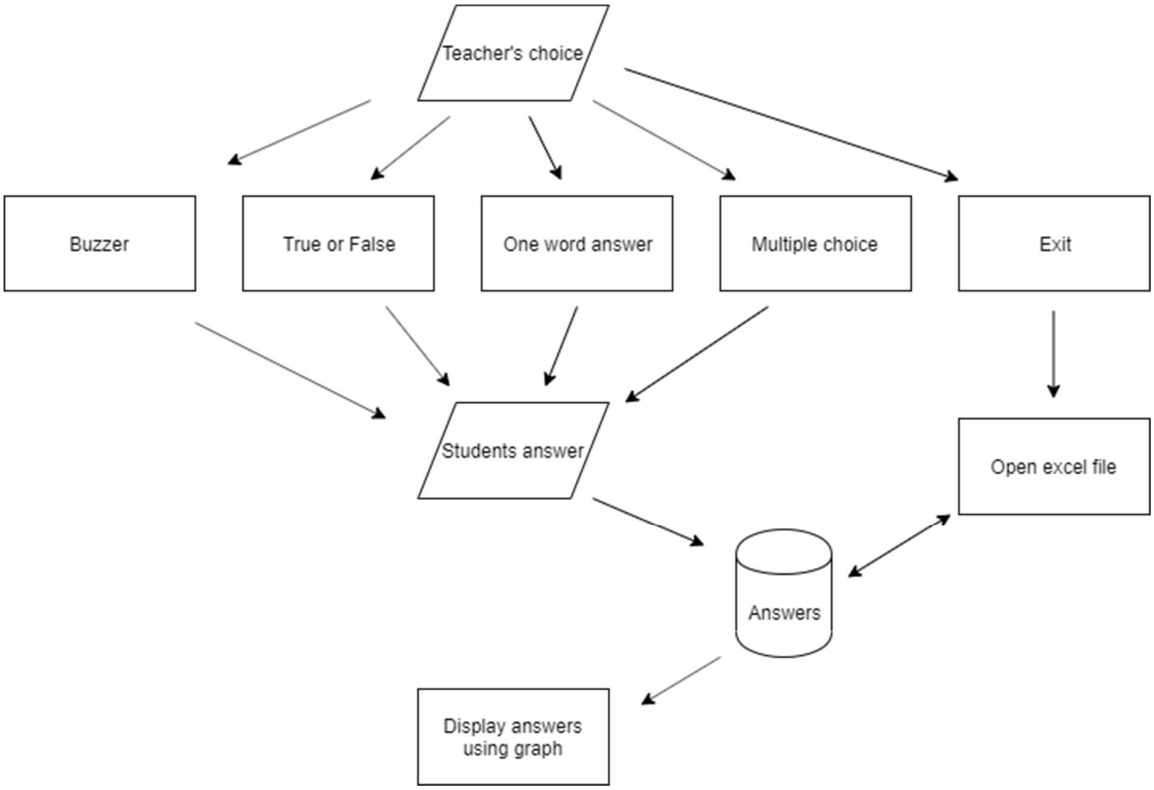




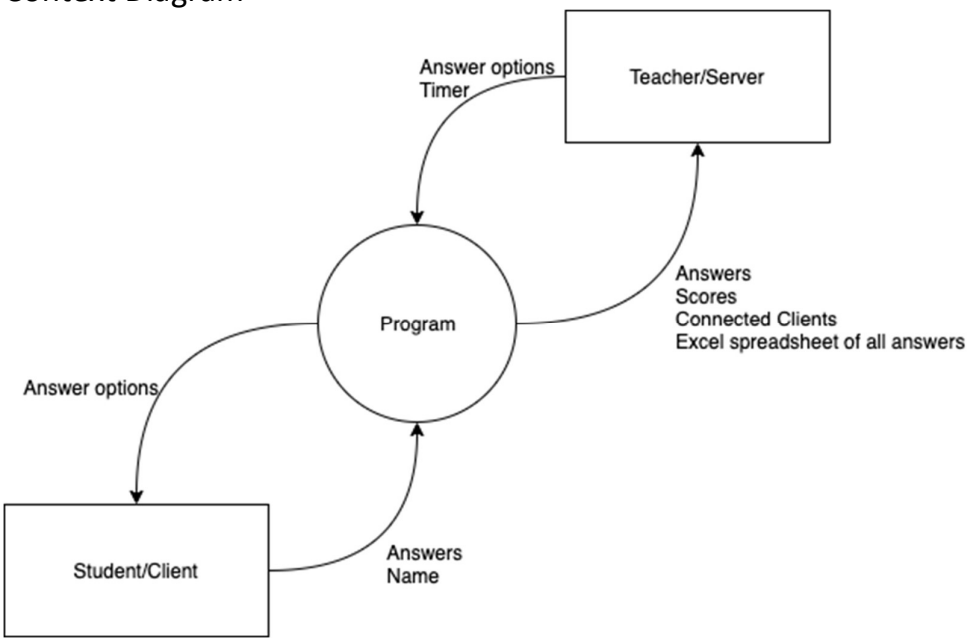
Structure Chart:



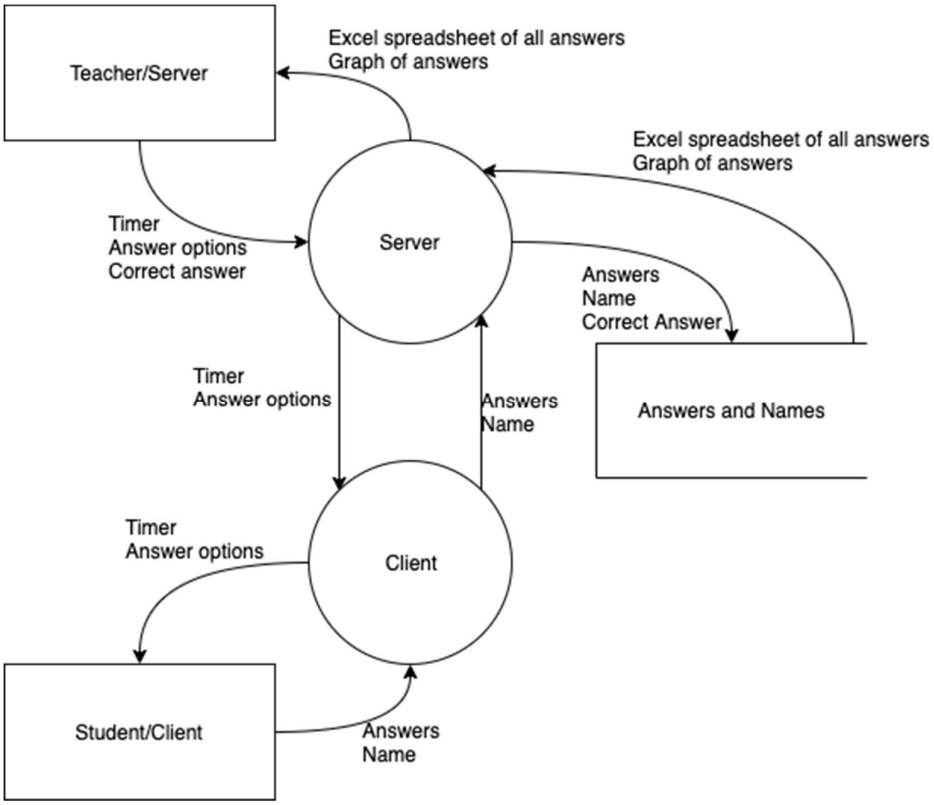
System Flowchart:



Context Diagram



DFD



Selection of software environment and identification of appropriate hardware

Most of the calculations, storing and processing will happen on the server computer, so the program can run on computers with minimal power, which is very fitting for a school environment. In addition, schools are more likely to have windows OS than mac OS, making the software environment windows make the most sense.

As python can be used to program both the program and the interface, and have it work together without compatibility problems, python will be utilized. As python version 2 is approaching its end of life stage early 2020, python version 3 will have to be used.

Minimum Requirements:

- 1 gigahertz or faster processor
- 2 gigabyte RAM
- 16 gigabyte storage
- DirectX 9 or later with WDDM 1.0 driver Graphics Card
- 800*600 Display

Recommended Requirements:

- 2.5 gigahertz Dual Core
- 4 gigabyte RAM
- 320 gigabyte storage
- Integrated Graphics
- 1920*1080 Display

Data Dictionary

Field name	Data Type	Data Format	Field Size	Description	Example
Server Address	String		15	The address of the server	10.218.0.21
Client Address	String		15	Address of the client	10.218.0.21
Client List	Array		15	List of all clients	Array[1] = “john doe”
Client name	String		30	Name of the client	john doe
Display name	String		30	Display name of the client	John Doe
Client Score	Integer	NNN	3	Score of the client	5
Client Answer	String		15	Answer of the client	True
Answer Option	String		8	Answer option given by the server	True/False
Timer number	Integer	NNN	3	The given time to answer in seconds	10
All Answers	Array		15	An array of all given answers	Array[1][1] = “Bluejay”
Connected Clients	Integer	NNN	3	The number of all connected clients	6

Design of appropriate test data

The first test that I'll have to do is the basic functionality test, so I'll test that every part of the program works properly in its proper condition. This includes the server being able to change the client's display name, answering with the keyboard and changing answers.

Next test should be testing every part of the program in a non-desirable way. For example, not answering in the given time, answering before an answer option is given, giving multiple answer options, connecting to an invalid address, disconnecting and reconnecting, answering with a different option, etc.

The next test should be a limit test, seeing if the server can handle the required number of clients, and see how many answers it can handle. This will most likely depend on the computer the server is on, but the software should be well optimized to run on a less powerful computer.

The last test should be testing the functionality of the one-word answers, as that is the most difficult and complicated answering option. The tests should confirm that the answers are received separately, and do not correlate with any other answer options. For example, when a client types in "True", it should not read the answer as a True/False answer.

Feasibility of the solution

Constraints and Scope –

Due to the nature of the program, the options of answering are extremely limited. As the purpose of the problem is to gather, store and display data of a large number of people in a simple way, individualized and specific answers cannot be processed. This means that the options given to student will be limited to tur/false, multiple choice, the buzzer and the one-word answer.

Plus, the time given constricts the amount of functions we can add to the program.

For the program to be considered functional, it should:

- Allow multiple clients to successfully connect to a server
- Server to be able to pick the answer option, timer and visibility of answers and scores
- Clients to answer to the respective options given
- Answers to be received, stored and displayed to the server, with a graph for each question and a spreadsheet of all questions at the end

Financial –

Given that the standard pay rate of a programmer is $100/hour, and the given time is 26 weeks, with 5 workdays and 8 hours of work each day, the cost comes to $104,000 per developer. With 4 developers, the cost of developing this program would be $416,000.

By signing a yearly site license with schools and companies, the cost will eventually reach its break-even point, and continue making profit as time goes on.

Operational –

The program is designed so that it fits a school/workplace environment, by designing it to be installed in an environment with a network already set up and designed to work on less powerful computers, such as generic school or workplace computers. So that students and teachers can easily understand and utilize the program, it is made simple and intuitive.

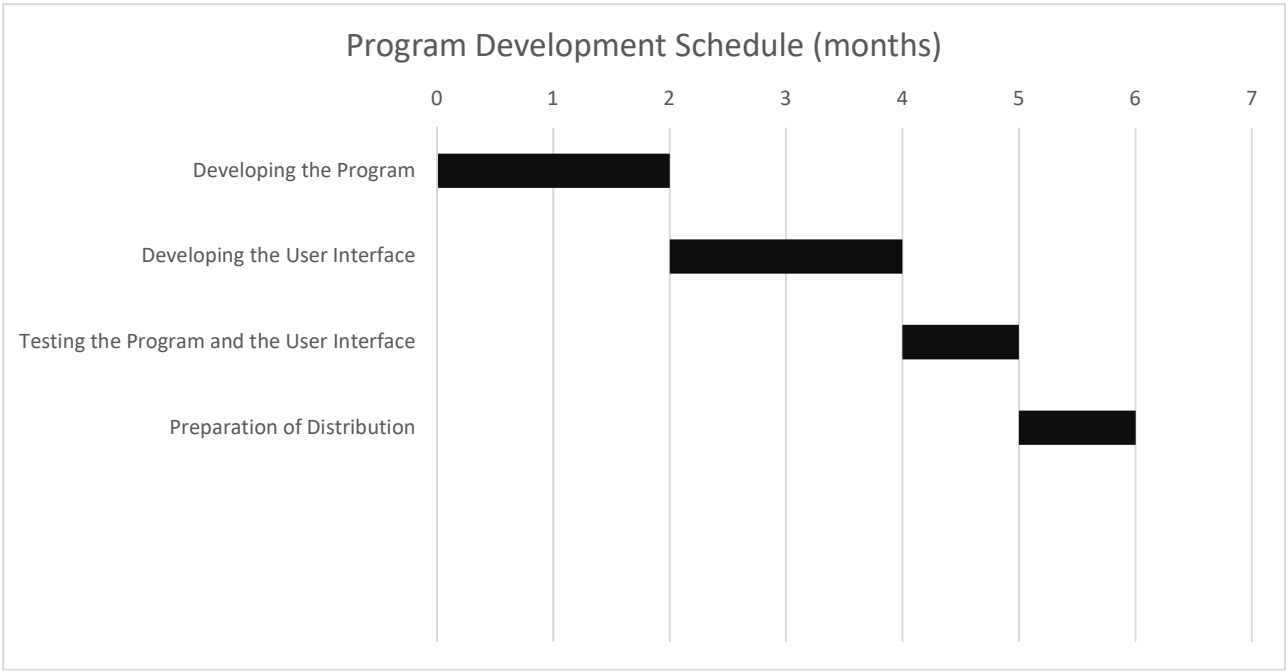
Technical –

Admittedly, the hardest part of this program would most likely be the installation on school/work grounds. Most school and work grounds have security proxy(s) installed, and that may cause a problem for the installation. However, if the program is successfully installed, as long as all of the client's computers and server is all connected via a network, the program should run smoothly.

Scheduling –

The program is due to be done by end of June 2020, which is more than sufficient time to program all parts of the program and prepare it for distribution. The scheduling will be divided up into 4 parts, which are:

1. **Developing the program and its modules**
   The developing of the base program only using a text window and ensuring all functionalities work.
2. **Developing and designing the user interface**
   Developing the user interface and integrating the base program into it, again ensuring all functionalities work.
3. **Testing the program and user interface**
   Testing the program and ensuring all functionalities work, and all unacceptable inputs are dealt with properly.
4. **Preparation of distribution**
   Preparing the program for distribution and designing a physical copy



Social Ethical Issues

As the program does collect information of members in a school or workplace, security of the data must be integrated to ensure the safety of both the school/workplace and its members. To do this, and for other reasons, only clients connected to the same network as the server can connect to the server, and even then, only if they have the port/address given by the server. This will increase the difficulty and discourage attempting to breach security.

As the data is used to determine the skill or opinions of the student/employee for the teacher/employer, the accuracy of the data is paramount to the program. If the program was to make a mistake and display the wrong answer from the student/employee, it could mean their results or their job. To ensure that this doesn't happen, when the server receives an answer from the client, it will verify the answer by pinging the answer back to the client and check.

As the data collected can be used to collect private data of students and or workplace staff, the data collected should be used for the one reason it was collected and no more. To ensure that data is not misused, only the server's computer will have access to all the data collected at the end and have the choice to show it on the screen. This will limit access to the data from the people who should not see or own it.

Inclusivity is an issue that must be addressed to ensure that all individuals can utilize the program, to widen the target audience. The program will avoid any mention of race/religion and derogatory comments. The coloring of the interface should be carefully picked so that being colorblind does not conflict with the use of the program.