1. Hook up and install

A first time user would only need to copy the folder containing all the files onto a server and it should be ready to use.

2. Compile and link

The program is a browser, so there is no compiling and linking necessary.

3. Initialize

The program runs best on Google Chrome, so if it is not already downloaded onto the kiosk computer, this might be necessary.

4. Use

Please see the next page for a detailed description of how to use the program.

5. Detailed exceptions

There is no way of handling more than one user at a time, and there is no way of preventing more than one user accessing at a time. The program cannot prevent a user that is off-campus from playing; essentially, the kiosk could look like it is playing by itself. The kiosk will crash after a few minutes because the browser will give up waiting. Lastly, the internet provided by the phone’s provider does not have the processing power to run the program. The phone must connect to WiFi in order to work.

6. Hints on operation

I have only attempted running the program in Google Chrome and Firefox - never in Internet Explorer or Safari. Google Chrome seems to run better than Firefox, although I only really noticed the difference when the program crashed (so hopefully it will no longer be an issue) because I was running both the kiosk and the user side on the same computer, so when one crashed, Firefox would take the other down with it.

There is a detailed graph of how files interact after the How – To pages.

Starting the Game

To start the kiosk browser, go to:

<http://compsci02.snc.edu/~spiemc/Senior%20Capstone/kiosk_index/kiosk_title.php>

Similarly, the user browser is:

<http://compsci02.snc.edu/~spiemc/Senior%20Capstone/kiosk_index/user_title.php>

Playing the Game

The user-side browser will have a play button. Press it to start the game. As the game continues, the browser side will ask a question, and have four possible answers. The user side will have the same answer buttons, and both will have three 50/50 buttons. To play, answer the question by pressing the correct answer button on the user side browser.

50/50 Buttons

A 50/50 button can be used once per question. It will take away two of the wrong answers, leaving one wrong answer and one correct one. Each user has three 50/50s to use per game.

Lives

The user also has three lives. For each wrong answer they get, one life will be deducted. If they lose all three lives, they lose the game, and the kiosk and the user browsers will both be redirected back to their respective title screens.

Questions

Currently, there are only five questions per game, but this can be easily changed. The files that would need to be changed are:

* kiosk\_check.php
* kiosk\_index.php
* game\_questions.txt

The two .php files both have a PHP variable called maxQuestions (found near the top of the PHP portion). In order to add more questions, this variable would need to be increased, with both files containing the same number). As for the .txt file, for every 1 question, there are 3 possible questions to be randomly chosen from. So, for every additional question added to the game, 3 more need to be added to the .txt file. For example, because there are currently 5 questions in the game, there are 15 questions in the .txt file.

Text Files

Some of the text files are expected to be in a specific format very important to the game.

* game\_explanation.txt

The explanation for each of the answers to the questions must be in the same order as the questions in ‘game\_questions.txt’. Therefore, there should be the exact same number of explanations as there are questions in the game.

* game\_lifelines.txt

There are three numbers in this text file, as follows:

First number: the number of lifelines left (0-3)

Second number: the number of lives left (0-3)

Third number: the question the user is on (1- maxQuestions)

* game\_questions.txt

As stated before, for everyone one expected question in the game, there should be three optional questions to randomly choose from. Each of these questions should be on a separate line. In addition, after each question, the following line must contain the four possible answers, separated by **only** a comma, or ‘,’. Furthermore, the correct answer needs to have a slash, or ‘/’ in front of it. See below for the three possible questions, and their answers for the first game question.

Three of the following are musical notes as well as the name of a programming…

B,C#,/C++,F#

Which of the following is not a Computer Science concentration at St. Norbert?

CS,/RGB,GDI,BIS

How many first-year dorms are there on the St. Norbert campus?

/4,5,6,7

* user\_answers.txt

This text file is only used when the game is in play. The kiosk writes the four potential answers into it, and the user then reads from it. This file may or may not be empty if you look at it, but do not edit anything.

* user\_check.txt

Similarly, this text file is also only used during gameplay. The user browser needs to check user\_answers.txt to get the possible answers, but it can only do so after the kiosk has written the answers into it. Therefore, the user browser checks this text file repeatedly, looking for a ‘1’ – when the kiosk is done writing to user\_answers.txt, it will write a ‘1’ to this file, and when the user browser sees it, it reads from user\_answers.txt. There may be a ‘0’ or a ‘1’ in this file, but do not edit anything.

* user\_input.txt

This text file is also for game use only. The user browser writes the answer the user pressed into it, and the kiosk reads from it, and uses the answer to continue gameplay. Do no edit anything in this file.