ECE 3553 – Multifarious Systems Project

**“Snake Game” with MySQL+PHP Integration**

By: Andrew Nguyen

**Problem Statement:**

This project involves creating a JavaScript based game and displaying the game in an HTML based environment. The game must take account of which user is playing the game, as well as account for some method of scoring. Upon completion, the project must update a MySQL database using PHP queries with the score and username and take the user to a “High Scores” page. In this implementation a classic game of “Snake” is implemented, and the code preforms the aforementioned.

**Literature Review:**

The concept of playing a game and having the scores and name of a player updated to local or external database is not new. In fact, scoring systems can be dated to the 1980’s in various local arcade machines with popular games Galaga, Asteroid, or Space Invaders. The key difference between this project and these machines is that the early machines saved the scores local rather than a server based location (The design choice for the arcade machines to store data locally was out of the influence of a lack of wide spread internet).

**Detailed Explanation:**

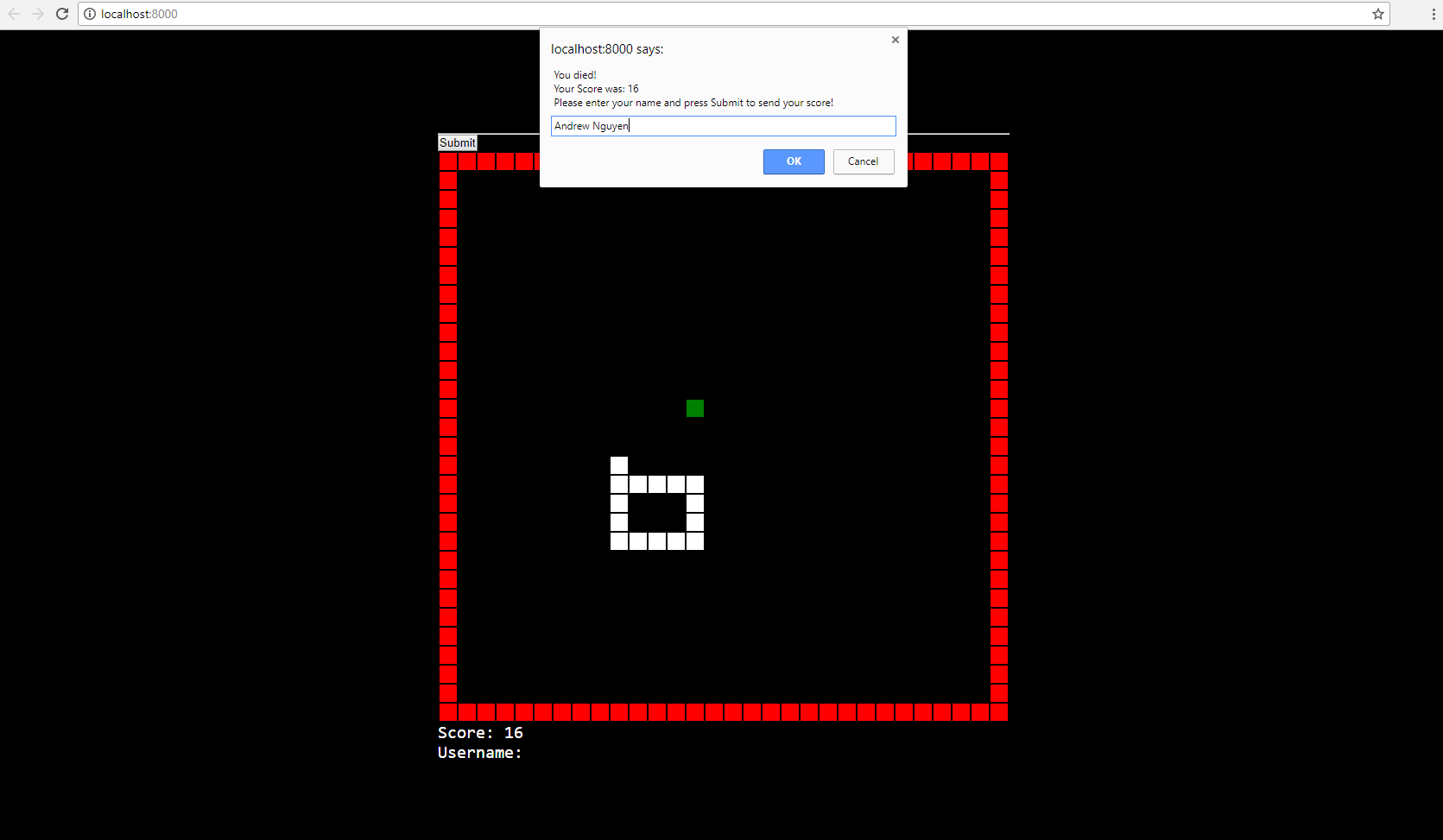
This project combines the usage of JavaScript, HTML, PHP, and CSS all together to make an HTML based experience for a user to play a game of Snake. The project will not use any languages or associated Java libraries not discussed during the class ECE 3553 (Please See Concluding Remarks).

The game begins at a home page (PHP calls it “index”) which prompts them with a JavaScript(JS) based snake game. The user will than play the game using the WASD keys to move. Upon completion, the user will be prompted with an JS alert asking them to input a username for their score to be saved as. This alert will also prompt them to hit the submit button to write the scores to a MySQL database and redirect them to a “High Scores” page. What the user does not see clearly on this page is the implementation of a form used to submit the scores. The JS script itself autofill’s two forms to contain the username and scores of the player that will be submitted to MySQL.

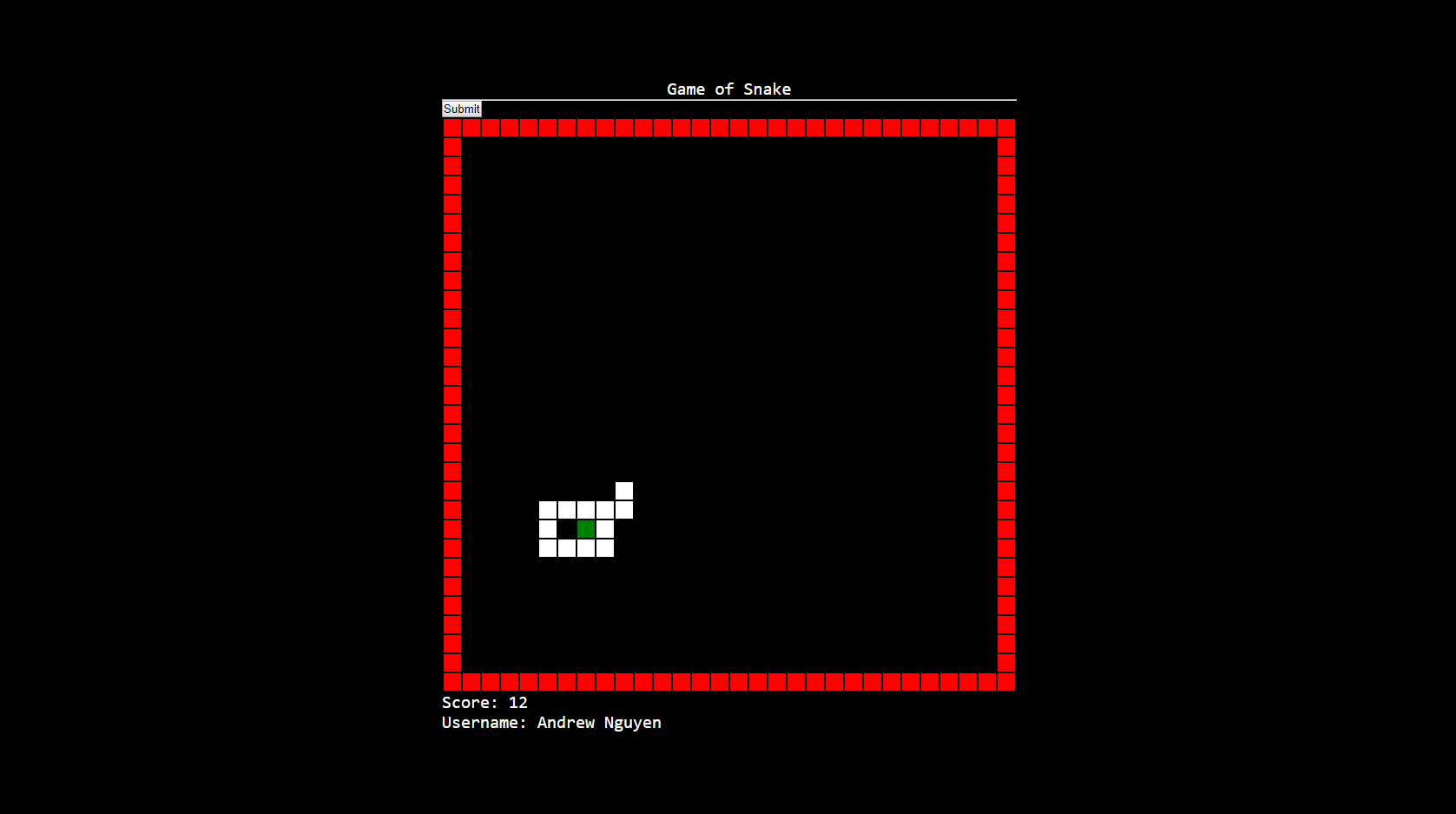
The “High Scores” page will display a table of only the top 10 highest scores in the game. If user’s scored identical scores, the first entry will always be displayed first followed by the others. This table is created through PHP queries to MySQL to generate an HTML table. Behind the scenes, this page is also responsible for writing the scores and username from the submitted form to MySQL using PHP. This task is done immediately when the submit button is pressed before the page itself is generated.

To discuss shortly about the JavaScript (This is more a web development class than a programming class) the code is a looping code that executes every 1s and draws the results to the “Snake” game being played on the home page. This is done by having the JavaScript draw the table (“Referred to as map in code”) in HTML and update that table with the results each time the code finishes a successful pass. The code logic for generating the snake keeps track of the positioning of the tail and head of the snake, as well as any collision conditions that may occur. The body of the snake is updated based on the snake and tail positioning, and the movement control is done by adding EventListener to bind keys to direction flags in the script.

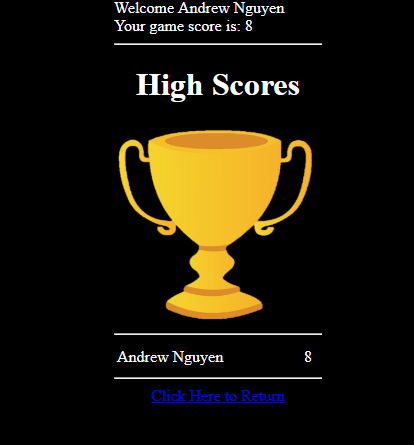
**Results:**

****

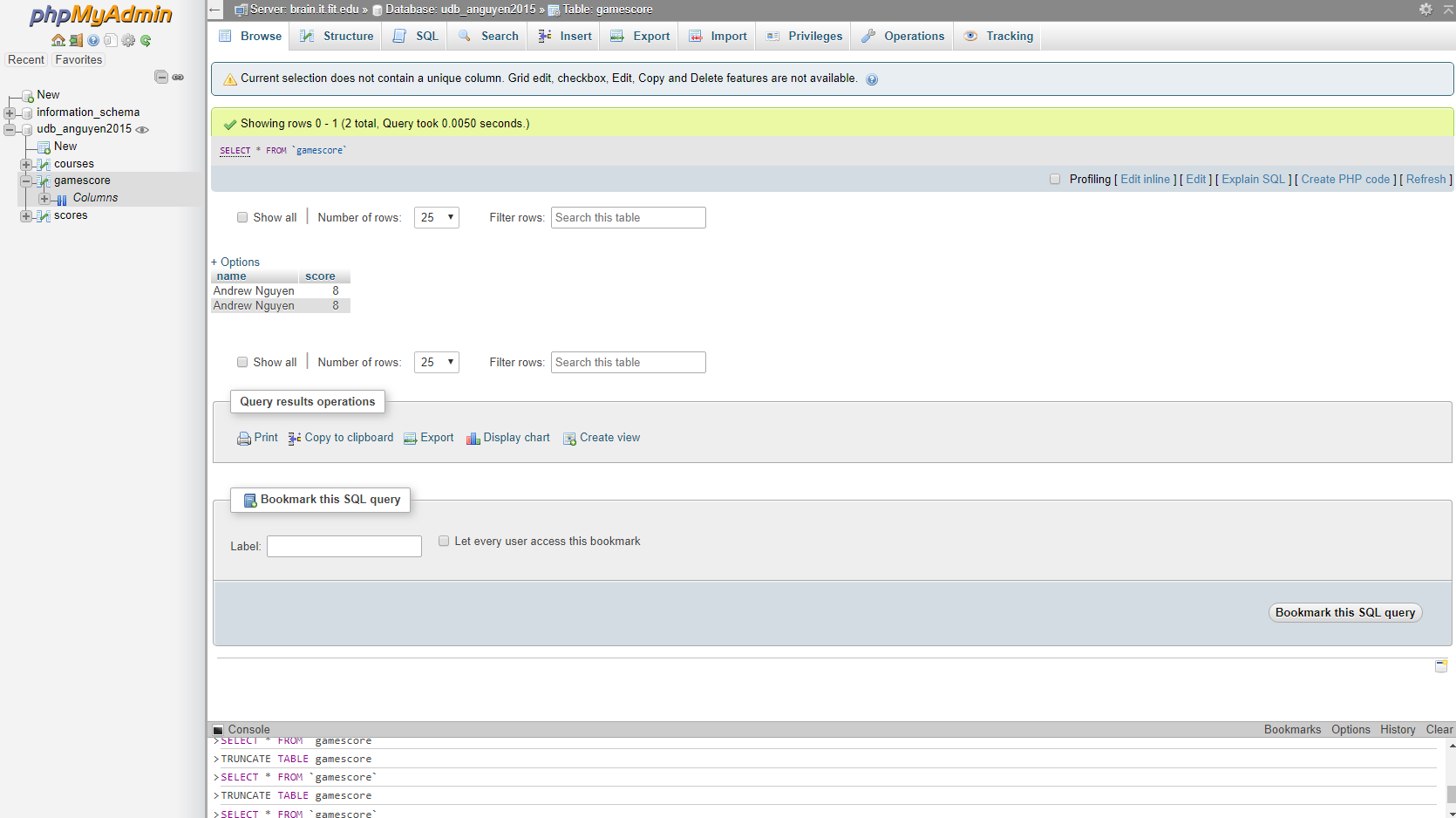
This is the result the user gets after playing the game. They will be prompted with an alert declaring a game over and asking them to enter a username to save the score as.

****

Pressing OK on the alert will update the username text under Score. The user will than press submit to send their score.

****

This is where the user is taken after submitting their score. They are taken to a page with a “High Scores” Table which will be updated as more runs are played.

****

Behind the scenes score storing in a server sided MySQL database.

**Concluding Remarks:**

There are some quality notes that must be addressed that were flagged above that must be considered if this was project was implemented into a large-scale user environment.

The first, is that the usage of a form as a method of submitting the scores to a server is a crude implementation to bypass basic JavaScript server-side limitations and to use PHP to send the scores instead. A method undiscussed in class (which I didn’t want to use) that I would implement if I had the change to redo the project would be to use the Node.JS library to write to MySQL directly in the game code rather than asking the user to press two buttons to submit the form. Using Node.Js it is possible to have the entire form submit and proceed to “High Scores” after closing the alert.

The second and final remark is that there is no data control for the scores that are submitted to the database. This means that all scores regardless of how “high or low” are submitted to the database. The query to the table will only show the highest 10 scores, but all the scores exist in the database. This means that eventually the server will run out of resources with the amount of entries contained in the table. So, the table now most be manually maintained. In the future, it would be best for the scores to only be submitted if it is higher than the lowest score out of the 10 highest, and the lowest score would then be deleted from the table.

**References:**

[**https://en.wikipedia.org/wiki/Score\_(game)#In\_the\_media**](https://en.wikipedia.org/wiki/Score_(game)#In_the_media)