Test Skeleton Weekly Deliverable

Skeleton for Unit Test

There is not much unit testing for our website because we handwrote the code. There are not many small methods within the website except for the game and the database.

* The game
  + Answer is inputted to a question in the game
    - If Correct – add to score, and move piece
    - If Incorrect – Do not add to score, and move piece
  + Make sure actionlistener is only called when button is clicked in game
* Database
  + get methods
    - Existing User – return true if username and password are found, return false otherwise
    - Scores
      * Individual Student
        + Gets students most recent score
        + Gets students highest score
        + Get students average scores
      * All Students
        + Gets the highest score
        + Gets the average score
  + Add methods
    - New User – add new user stored with username, password, email, and user-level (student, teacher, parent)
    - Scores
      * Individual Student
        + Last Score – replace existing last score with the new score, add 1 to counter
        + High score – compare new score to old high score, if new score higher than old high score, replace high score, else, keep same high score
        + Average score – add new score to total scores, divide by the counter, should calculate the correct average.
      * Student Total
        + High score – compare new score to old high score, if new score higher than old high score, replace high score, else, keep same high score
        + Average score – add new score to total scores, divide by the counter, should calculate the correct average.

Skeleton for Integration Test

* See if created user is sent and stored in the database correctly
* Test to see if you can register a new user of all levels (student, parent, teacher) and they are inputted into the database correctly.
* See if login correctly looks for usernames and passwords in the database, and if incorrect username and/or password, fail, don’t allow user to login, and prompt the user to enter a correct username and password.
* Make sure the site allows the right levels of authority to access different info
  + Teacher – high score, average, and last score for all students along with the whole class.
  + Parent – high score, average, and last score strictly for their child.
* Social Media
  + Make each icon (Facebook, Twitter, Snapchat, GitHub), and make sure if clicked, each takes you to the desired URL.
* Test the sub-titles (Game, Ticketing, and Home) to see if they lead to the correct URL that matches with the correct sub-title, even if it is the current URL you are on.
  + For example, if you are on the home page and click home, it reloads the home page.
* Test the slideshow on the home page to make sure you are easily able to go back and forth between the photos available.
* The game should run smoothly and easily, running when it’s supposed to and stopping when it’s supposed to. Allow user to play multiple times without necessarily reloading the page.
* Test that when a game is complete, the score is sent to the database
  + Make sure it is stored for the right student
  + Make sure it replaces the old high score if it is a new high score
  + Make sure it is added to the average correctly and the counter for the average also goes up
* Test that the ticketing allows you to purchase a ticket, and, if applicable, allows you to enter a discount code for the ticket.
* Make sure the ticketing gives some sort of confirmation for a ticket being purchased.

End – to – End System Test Procedure

* Open website to the home page
* Login/Register if user already exists/does not exist
* After user login/registration should send you back to the home page
* Open game tab
* Play game, get results, and if applicable (depending on how well you did in the game), get discount code.
* Go to ticketing. Purchase a ticket and, if applicable, use discount code to get ticket.
* Go to social media to further follow the game, along with get more information

Regression Test Procedure

When updates are added, start at the top first and use prioritization (the stuff most of the users will be seeing the most often, and focus on that first). This means we start at the top of the website, making sure each page still is displayed correctly. Make sure the login and register new user work for the visitors on the website. We then will make sure the game runs correctly, and if edits are done on the game, we will make sure scores are still calculated correctly throughout the game. If the database is edited, we will first make sure it is still in communication with the website, allowing the website to call to the database for usernames, passwords, scores, averages, etc. If ticketing is edited, make sure the ticketing page is not messed up, and make sure it still does the ticketing correctly, confirming the ticket and allowing a discount code to be entered. After making sure the main user-level stuff is running correctly, go through the database and make sure values are being stored correctly from the game and users, and make sure they are calculating the right values. Then we will go to all the base level methods to make sure they are still running correctly.