QuizQuest Coding Report Summary

Group 8 - Jonathon Repta, Amal Syed, Edward Liang, Jennifer Alonso

Project Overview:

QuizQuest is a free to play browser based quiz game. *QuizQuest* combines traditional online quiz gameplay with location based features that encourage users to explore locations both near and far. Core to *QuizQuest's* design is frustration free, accessible gameplay. In order to achieve these goals, we designed the application with certain things in mind. Firstly, the application was designed from day one with accessibility in mind. In order to achieve said accessibility goals, we designed our site "mobile-first", ensured we were utilizing all of Bootstrap's accessibility features, and including homegrown usability enhancements (such as keyboard based quiz completion and a quiz browser for those who cannot use the map GUI). On top of this, we took advice given by the professor and allowed for all of the fundamental features of our application to be available to all user classes (including those who would rather play without an account).

Testing:

There were five main areas we felt needed the greatest amount of test coverage: the database helper functions, the API endpoints, the "submit quiz"/"approve quiz" frontend, and the the sign in/signup page's frontend.

For the database helper functions, we wrote and ran several suites of tests - one which submitted a single empty quiz, one to submit a quiz with empty questions, and one to submit a quiz with both questions and answers. All tests passed and no errors were found.

For the API endpoints we tested every route accessible to the user through a series of tests written in Python using PyTest. For each test case, we compared the response sent by the server with information stored in the CSV we used to insert said information into the database. All tests passed and no errors were found.

Next, we tested whether account creation by automating tests regarding use of the sign up/sign in GUI. In order to test the front end, we chose to employ Selenium as a browser automation tool. Using Selenium let us locate and interact with particular page elements in a programmatic manner. All tests passed and no errors were found.

Lastly, we concluded our front end testing by using a similar Selenium procedure to automate page navigation and find faults in our quiz submission code. All tests passed and no errors were found.

Inspection:

For code inspection, each team member inspected code for the parts of the application they wrote tests for. During inspection of each part, we found no major issues with the project. However, inspection revealed many minor issues/avenues for improvement. In regards to quiz insertion, code inspection revealed that only part of the data stored in the "pending" tables were being removed from the database when the moderator chose to approve or deny a quiz. Although this error is not major, it would lead to the database being polluted with outdated information if left unchecked. During inspection of the Quiz submission GUI, no major issues were found. However, code inspection revealed some minor UX issues regarding the "remove question" button. Inspection of the API endpoints revealed no major errors, however, inspection of the codebase revealed several legacy routes that were not being used anywhere else within the application. Every unused route was noted and added for removal. Inspection of the account creation/sign up revealed no major errors.

Conclusion/Retrospective:

Overall our project is quite polished and met most if not all the major criterias laid out by Group 2 (Fall 2016). With that said, there are still some items in the product backlog that should be addressed in future revisions of the project if development were to continue. Firstly, QuizQuest could have several features we wanted to implement if we developed a native mobile version of the application. Although this would be a moderately costly endeavor it has the potential to transform QuizQuest from a fairly standard Quiz game into something more interesting. This affects QuizQuest's space in the market as well, because in order for the game to stand out from the deluge of other similar applications, we would need to add more community based features, location based elements, and utilize them effectively to keep users attention. Essentially, we should learn from the successes of other location based games like *Ingress*/*Pokemon Go* and make changes accordingly.