Testing Plan & Cases

Test Plan: *Quise*

Analysis:

- Quiz website for people to play and gain score.
- Keep track of their score with profiles
    - Profiles are user specific
    - Login and logout pages, with home/quiz page
        - Will prompt login/registration before they can start the quiz
        - If username exists but password is wrong then try again
        - If the username does not exist, bring it to the register page.
- Leaderboard to see how you rank amongst other players
    - Score based upon % correct on respective quizzes and time elapsed since quiz beginning on highest scoring test.
- Home page consists of quiz category selections that start quizzes

Testing/Test Cases:

- UI
    - Profile, log in/out, quiz taker etc… UIs all require UATs.
    - **Criteria for success:**
        - UI is clean and easily readable and accessible
    - **Testing methodology:** Manual
- Profile page
    - Query returns a results for a user
        - Including a valid name, quiz record stats (failure rate etc…) & picture (low priority feature)
    - All data returned matches the data in the database corresponding to the current user
    - UI: All data is formatted correctly on the page
    - **Criteria for success:**
        - Everything works and is functional UAT
        - Users are able to easily find items (can be done through non CS students testing the website)
    - **Testing methodology:** Unit test on database query, manual testing on front end query
    - Data:
        - Login ({username: calebK, pass : 123456 })
            - Result: ‘Success’
        - Register ({username: tysonT, pass: 987654 })

        - Result: ‘Success’
- Logout
    - Successfully logs out user
    - **Criteria for success:**
        - User can logout after clicking on the logout button
    - **Testing methodology:** Unit testing to make sure the session is destroyed, manual testing to make sure redirection works properly
    - Data: Logout button
        - Result: ‘Success’
- Quiz pages/s
    - Query returns question and answer for user, with one right and 3 wrong answer
    - Right answer will let user move on to next question
    - Wrong answer will make user stay on the page until right answer chosen
    - Timer counts down for total quiz time
    - When timer reaches 0 all questions should be submitted and the user should be redirected to a page showing their score
    - When user answers all 5 questions they should be redirected to a page showing their score
    - None of the 5 questions returned should be the same question
    - **Criteria for success:**
        - Can successfully move through quiz with no issues ie getting stuck on a page
        - Score is kept correctly
        - User is updated on leaderboard correctly
    - **Testing methodology:** Unit tests on database queries, manual testing on frontend queries
    - Data: button click
        - Result: ‘Success’
- Login Page
    - User password (hash) is stored **securely** on the backend and comparing stored data to new data works as expected. Unit testing.
    - Redirect to /login when not logged in.
    - **Criteria for success:**
        - Comparing stored data to new data works as expected. (Unit testing on hashing)
        - Redirect to /login when not logged in.
    - **Testing methodology:** Unit tests on database queries and frontend js, manual testing on frontend queries
    - Data: Payload{username': 'Nikita’, 'password’: ‘0xef75146c’ (hash of 1234)}
        - Result: ‘Success’
    - }
- Register Page
    - Submit button should add a user to the database with the specified username and password

- User creation should fail if username is not unique
- Redirects to login
- **Criteria for success:**
  - Data in database is same as data entered by the user
  - Redirects to /login when user creation is successful
- **Testing methodology:** Unit tests on database queries, manual testing on frontend queries
- Data: Payload{username': 'Nikita', 'password': '0xef75146c' (hash of 1234)}
  - Result: Username is not unique, failure

Testing Overall

Note profile, log in/out, quiz taker etc… all UIs require UATs.
All unit testing will use **Jest** https://jestjs.io/.

User Acceptance Testing:

All user tests must be conducted on campus. In order to do this we can set up a few of our computers with the application running, sit at a table and provide candy for random users that want to try our application. This will provide good critical feedback to our project to make sure all user error tests cases have been covered.

Testing Environment:

For each unit test there will be categories. Each category will represent a different page for the application (log in, register, quizzes, etc.). This will keep tests organized so when we want to create a test for a certain application on a page then we can put the test under that pages category. Unit testing will be primarily done to the back end naturally, some may occur with the quiz page JavaScript.

Example would be:

Login Tests

Test user login : joe
Test user password : mama
Expect(user login)