Quise

CSCI 3308 - Fall 2022
Group Project

Ostrich Algorithm // Quise
Group 016-03

Team Members:

Name	**Github**	**Email**
Tyson Trofino	Tyson-Trofino	tyson.trofino@colorado.edu
Clayton Dwyer	Narrosh	Clayton.Dwyer@colorado.edu
Jesse Black	jebl8843	jebl8843@colorado.edu
Caleb Kumar	cekcreator	caku9316@colorado.edu
Joseph Kneusel	jkneusel	joseph.kneusel@colorado.edu joseph@kneusel.org
Simon Walker	simondoesstuff	simon@simonwalker.tech

Application Description

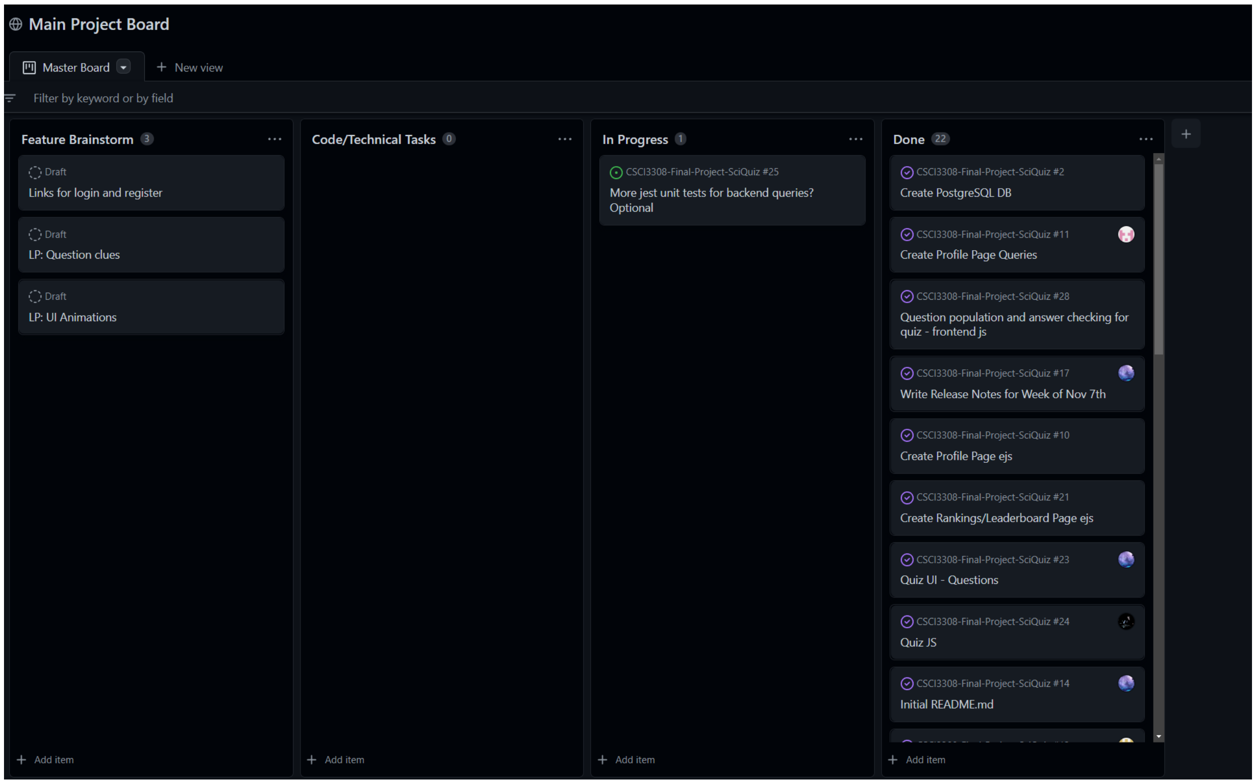
Players are given a very fun sets of timed quizzes of varying topics. Each player is given a score based on correctness of their answers, but keep in mind, it is important they answer quickly or they will be immediately failed. Quizzes are built off topics the user picks. The topics could be physics, chemistry, general engineering, computer science, etc. There are many topics and many difficulties for a very large total question pool. The questions are randomly selected, but the user can choose the category they are most interested in.

The application will also include the ability for players to create their own profile to keep track of their quizzes, their scores and times. These profiles will then be ranked according to their scores in a global leaderboard so users can view their ranking in relation to other people.

Our project aims to be a fun experience for anyone willing to learn or test their existing knowledge. Players will go onto our website and choose a variety of topics and include instant feedback on their success. Even better, over time players can accumulate data in their profiles and contribute to the global leaderboard which creates an air of competition. Overall, our project is not only a quiz, it is a fun competitive experience.

Project Tracker

[Main Project Board (github.com)]



Main Project Board

Master Board | + New view

Filter by keyword or by field

Feature Brainstorm 3

- Draft
  Links for login and register
- Draft
  LP: Question clues
- Draft
  LP: UI Animations

+ Add item

Code/Technical Tasks 0

+ Add item

In Progress 1

- CSCI3308-Final-Project-SciQuiz #25
  More jest unit tests for backend queries? Optional

+ Add item

Done 22

- CSCI3308-Final-Project-SciQuiz #2
  Create PostgreSQL DB
- CSCI3308-Final-Project-SciQuiz #11
  Create Profile Page Queries
- CSCI3308-Final-Project-SciQuiz #28
  Question population and answer checking for quiz - frontend js
- CSCI3308-Final-Project-SciQuiz #17
  Write Release Notes for Week of Nov 7th
- CSCI3308-Final-Project-SciQuiz #10
  Create Profile Page ejs
- CSCI3308-Final-Project-SciQuiz #21
  Create Rankings/Leaderboard Page ejs
- CSCI3308-Final-Project-SciQuiz #23
  Quiz UI - Questions
- CSCI3308-Final-Project-SciQuiz #24
  Quiz JS
- CSCI3308-Final-Project-SciQuiz #14
  Initial README.md

+ Add item

Video

[CSCI3308-Final-Project-SciQuiz/demoVideo.gif at main · jebl8843/CSCI3308-Final-Project-SciQuiz (github.com)]

VCS

Repo Link: [GitHub Repo]

Contributions

Simon Walker

Regarding the technical, I implemented a typescript wrapper for the OpenTDB api to make accessing data from it on the front end easier. I also helped implement /quiz which filled data from my wrapper to display questions and implemented /quizOver. Additionally, with my extra experience, I made high-level decisions around the architecture, especially around the pages/UI, and helped generally manage things and help with various bugs. Worked with all technologies.

Caleb Kumar

On the the technical side I worked on both ends, with quiz.ejs, home.ejs, the API JS calls for login and register. I helped out with figuring out how to display and work the OpenTDB API on quiz/home and correctly structuring the quizzes. I helped record meeting logs as well.

Joseph Kneusel

From a technical standpoint I worked mostly on front-end development and was responsible for implementing the front-end HTML/CSS for the login/register and home pages, respectively, as well as assisting with the other front-end pages such as the profile page. My main contribution however, is the overall "look" and artistic design of the application as I created the gradients, marquees and formatting used for the majority of the ejs files in the project.

From a non-technical standpoint, I helped plan meetings, troubleshoot ideas, created the use-case-diagram for the project and tried to keep moral up.

Jesse Black

Regarding the technical, I worked mostly on the backend of the project. Specifically on the profile queries (to get user information and update win statistics), other database manipulation (for example \gentest which creates a few test users for manual testing), and the user ranking system. I also redid some of the login code to better facilitate user sessions.

From a non-technical standpoint, I managed the Github repository and a good amount of the Project Board as well.

Clayton Dwyer

Regarding the technical side of the project, I worked mostly on front-end development and interface and spent time fixing certain bugs, especially those regarding deployment and Docker. I primarily managed deployment to the VM.

From a non-technical standpoint, I also managed a lot of the Github Project Board and the closing/opening of issues.

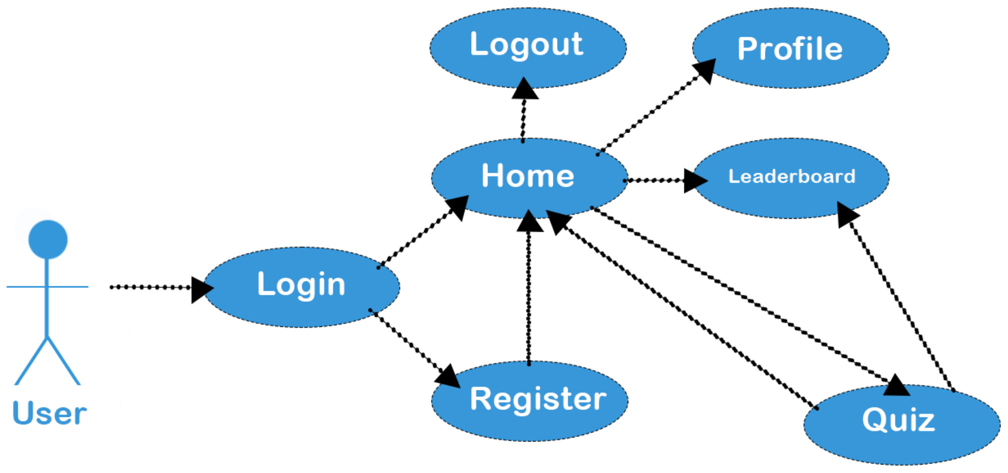
Tyson Trofino

Regarding the technical, I worked mostly on front end stuff and a little bit of the backend. I developed the headers for the login/register page and the header for once you are passed the login page. The reason this was done was to avoid users going to a profile that didn't exist yet. I also wrote the API code for the leaderboard page. I also wrote the front end code for the leaderboard page. I then spent time working on the appearance of the login and register page. It originally started out as a big thing you had to scroll through but after iterating and playing with it, I was able to make it more presentable. I also was a part of developing the database for this project.

From a non technical standpoint I helped run and manage meetings, helped organize and delegate tasks and created and organized the google drive for this project. On top of that I also reminded folks each week of the meetings we had with Nikita.

Use Case Diagram:

SciQuiz Case Diagram



Testing

- 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
    - **Results: Through User Testing, Users were able to understand the UI, navigate to options clearly, and understand each page presented to them.**

- Profile page
    - Easy to Read, Understand. Can See
    - **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:** User tests
    - Data:
        - Login ({username: calebK, pass : 123456 })
            - Result: ‘Success’
        - Register ({username: tysonT, pass: 987654 })
            - Success = Displays correct profile and profile stats
    - **Results: Through User testing, Users were accurately able to navigate to the Profile page and see their correct profile details. Along the way we did discover a bug and were able to correct it. Everything is now working as expected.**
- Logout
    - Successfully logs out user
    - **Criteria for success:**
        - User can logout after clicking on the logout button
    - **Testing methodology:** Manual testing
    - Data: Logout button
        -
    - **Results: Through User Testing, users were successfully able to logout from any page after login.**
- 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:** Manual tests on quiz and home pages
    - Data: button click
        - User

    - Success = displays the correct question, answer options and which answer is correct
    - **Results: for each category, each question and answer/s were relevant to category and able to clicked on.**
- 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:** Manual Testing, testing different accounts with different pass words
    - Data: Payload{username': 'Nikita', 'password': '0xef75146c' (hash of 1234)}
        - Success = loging in to correct profile with stat loaded in profile page
    - **Results: Through User testing, users were successfully to login after registering and when coming back to the website. We did discover a bug along the way that had to do with the hashing of the password. But we found the bug and tested again and everything is now working as expected.**
- 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:** User tests, have friends and family create accounts from their computer to see out put
    - Data: Payload{username': 'Nikita', 'password': '0xef75146c' (hash of 1234)}
        - Success = successful creation of account
    - **Results: Users were able to successfully register and keep using their account, with their stats tracked accurately**
- Open TDB Api
    - Needs to pull trivia questions from the remote DB based on a category of choice.
    - **Criteria for success:**
        - All data coming from the API is of the correct data and, according to a UAT, seems appropriate for the category.
    - **Testing methodology:** Mainly unit testing (Jest), partially UAT
    - Data: GetQuestion(Category)
        - Result: Data returned includes, "Question", "Answers", and "Correct" and is of correct types.
    - **Results: All test cases passed perfectly and the content seems appropriate.**

Deployment:

We deployed our project on the CU Boulder Department of Computer Science servers allocated for the CSCI 3308 Course. These servers are SSH accessible from the CU Boulder campus intranet and are running Red Hat Enterprise Linux 8.6. The deployed version of the application accessible via the link: http://csci3308.int.colorado.edu:3000/ . The application was deployed via a Docker container we configured using Rootless-Docker and Docker-Compose.