Edinburgh Napier UNIVERSITY | School of Computing, Engineering & the Built Environment

SET0801 2024-5 TR2 001
Web Technologies
Coursework Report Part 2
Group 23

Manik

Khadiya

40652166

-- Words

Contents

1. URL of deployed Site

https://manikkhadiya.github.io/webtechg23/

2. Plan vs Implementation

Planned Feature	Implemented	Note
Multi-page quiz site	Landing page has 4 different categories(one per person with different quizzes inside). I have pages for the quiz, logging in and leaderboard. I tried not ot make too many pages as I did not want ot make a basic serial of pages instead I wanted to learn how to use containers and wrappers to make component html files and then inject them with js which was really cool and interesting. it really changed my whole view on web work as a whole.	As planned
User log in and persistence(leaderboard)	The current username entered in the log in page Is stored in the localstorage.	I would love to learn how to manipulate the data and maybe interface with a server of sorts to introduce server side data storage so IoT devices can see others on the leaderboard which would be super cool
Componentised navbar/footer	Main.js loads components/navbar.html and ../footer.html using the fetch command	I think I will try to procedurally generate a web based game as a future project as I would like to test both this techs max capabilities and what I'm actually capable of.
Random quiz launcher	Uses .math and rounding *100 to pick between 0 - 3 which would be a respective key to the questions that are read from data/questions.json.	I would like to see if its possible to store non sensitive data from the user on a static site like the one we built as I wasn't familiar with JSON files but now I think im comfortable in experimenting with them.

Multi question types	I wasn't able to implement different question types due to sudden personal circumstances and events going on. Though I was able to procedurally generate the quiz form without changing html files which Is a new skill for me.	I really wanted to implement sound based questions, drag and drop, and ordering question types as they would have been a good learning experience for me though I will have to do this in another project.
Timed question	10sec per question; if the timer runs out the question is marked incorrect with a sfx matching.	
Keyboard controls	Though the page is usable with TAB I would not could this as properly implemented and I envisioned a much more sophisticated system that could answer questions using the number keys on the keyboard	I don't think it would be too difficult to implement keyboard friendly inputs.
Sidebar nav	This was not implemented due to personal available time restrictions.	I may have been a little too ambitious with my features though ive learnt from this project how to better manage my time and for that im thankful.
Responsive favicons	Multiple different favicon sizes are in the head of the html file which should resize to whatever size they need to be	
Accessibility menu	Pretty happy with how this turned out, dark mode worked and wasn't difficult to use, the volume slider controls volume in real time and text is able to be enlarged at will.	I would change the text size checkbox to a slider to allow the user more control. Another issue is that I noticed that the accessibility options don't persist when changing pages. I should have used cookies or the DOM storage to store the state of the options to increase fidelity.

Summary:
We, as a group, implemented core client-side functionality: logging in/out; dynamic quizzes, timed questions, skip question buttons, random quiz buttons, persistent leaderboards, componentized footer and navbar, favicons,

3. Features to Add/Improve

1. Additional Question Types
    - Fill-in-the-blank, True/False, Slider, Matching: implement parsers, custom input widgets, and validation logic to expand beyond multiple-choice.
2. Full Accessibility Menu
    - Inject the HTML for the contrast toggle, text-size controls, and volume slider so that accessibility.js can wire up ARIA attributes and live region announcements.
3. Persistent, Shared Leaderboard
    - Migrate from localStorage to a lightweight back-end (e.g. Firebase or Netlify Functions + JSON store) so scores and user profiles are synced across devices.
4. Dark Mode & Theming
    - Build a toggle that persists user theme preference, plus smooth CSS transitions between light/dark palettes.
5. Voice & TTS Support
    - Use the Web Speech API to read questions aloud and accept spoken answers, improving both accessibility and engagement.

4. Reflection

Challenges Faced

- Module loading & strict mode: switching quiz.js to an ES module (for import { saveScore }) caused “import declarations must be top-level” errors until I changed the <script> tag to type="module".
- Promise chaining: forgot to return the fetch in loadComponent(), so .then(setupNav) was undefined.
- Element timing: wiring up accessibility.js needed a waitFor() helper; without it the toggles never bound to their buttons.
- Scope drift: planned many quiz types & sidebar nav; had to triage to core features when time ran short, which was tough but taught me to prioritize MVP.

Achievements Made

- Robust quiz engine: a fully dynamic, JSON-driven question loader, with timer, skip, and instant audio feedback.
- Login & personalization: persisting usernames
- Leaderboard: custom sorting and storage of {username, quizKey, score, date}, plus display of the quiz name alongside each score.
- Clean, componentized structure: navbar and footer imported via JavaScript fetch calls, giving a single source of truth for site layout.

Conclusion

Groupwork Reflection: coordinating quiz-data formats across teammates taught me the importance of clear JSON schemas, communication and the necesities of having a leader of role based developement. One aspect that i was really let down by was teh consistent poor communication, especially at the start of the project, where replies would be scarce. This lead to me not having a clue of what was going on at the start of the project and left me to guess what everyone one was going to do. This lead me to make a full pipeline on github with a proper sophisticated directory where everything was organised. This was not used at all by the others and instead of using github regularly and keeping each other in the loop, no one but myself and David spoke. One major approach that i would change about how this project turned out would be to instead work only locally then upload all the work at once would be to approach this like we are in the industry and build the site feature by feature so when it comes to making content we dont have to make something someone else has already made for thier quiz. This would have saved so much time and effort; making the whole process of making this site cleaner and simpler. Our approach has our site looking unprofessional and scattered. This project has taught me the importance of thorough planning and clarification early on in the developement especially with unfamiliar colaborators.

References