WADD Summative Assessment Report

Introduction

The web application is called 'QuizMyBrain.' This application is as trivia quiz game, challenging the user's knowledge in various categories with multiple choice questions. The site uses the Open Trivia Database API to source the trivia questions. The user can choose a category and difficulty, and receive questions based on their choice. Their score is then saved, along with the scores from previous attempts which can then be reviewed. This report will provide a justification on how the application is designed for a successful user experience, an in-depth review of the technical implementation and a critical reflection on the achievements and failures of the application.

User Experience

Usability and Accessibility

The website provides a high level of usability and utilises conventional usability heuristics (Nielsen, 1994) to achieve an experience which is easy to understand and navigate for the user. A usable application is important to allow the user to learn and understand the features of the website and provide an experience which is flexible and efficient, something that the web application created enables.

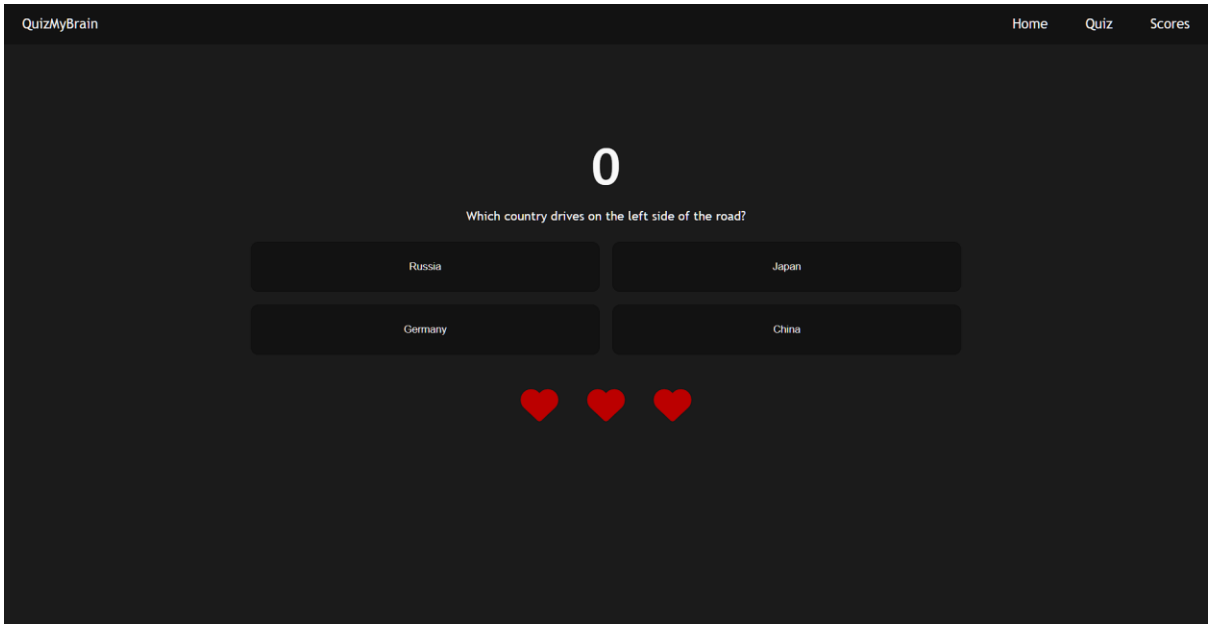

Figure 1 Quiz page

To facilitate an application which is easy to navigate, a navigation bar is present on every page. This consists of the name of the website, and links to the home, quiz, and scores pages. This provides the user with the control and freedom required to freely explore the website, ensuring they can correct any errors that they make and guarantee that any misinputs can be corrected by going back to any page. Conventional website icons and features are present throughout to

ensure an experience which is easy to learn. For example, inputs like buttons and drop-down lists, or icons like the 'hamburger' icon are used to provide the user with recognisable features that will provide them with a positive user experience which is simple and not complex. Design features are similar across each page, ensuring a sense of familiarity and uniformity to create a consistent experience. The status of the website is clear to the user when processes are made in the background. For example, the quiz page will display a spinner icon for when the data is being requested from the API and will subsequently display a message if the loading will take longer than usual. This information about the system status again allows the user to be in control and be informed about the state of the website.

Accessibility was also an area of importance. Each page achieves a score of 100 on accessibility based on Google Lighthouse. All text on the page contrasts well with the background colour, meaning they can be easily read by someone with a visual impairment. In addition to this, all inputs across the pages are focusable by a keyboard, and the website can therefore be fully used without a mouse.

Visual Design

Achieving a successful user experience was a priority during the web application's development. Early during the planning phase, the decision was made that a simple and elegant design was ideal for the website. This meant that only the essential features would display on the page at any given time, and that the page was not excessive with information. The wireframes constructed in the planning phase reflected this.

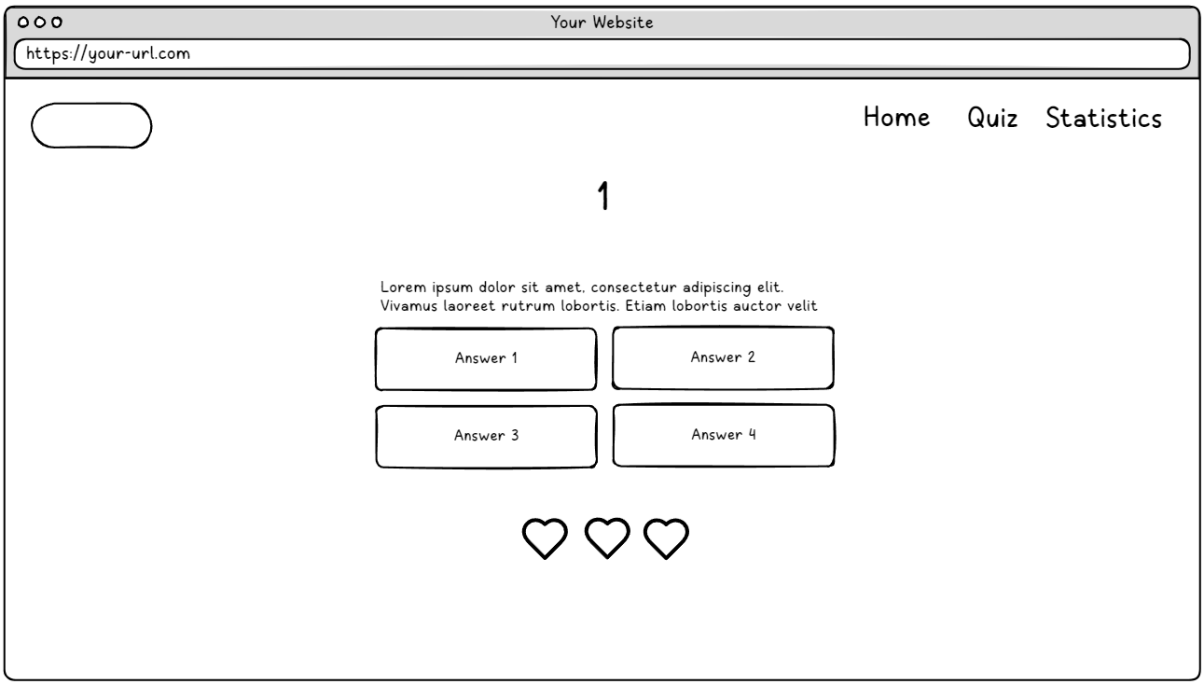

Figure 2 Quiz page wireframe

The website employs this approach across all pages, and only the information that is required on each page is shown to the user. This creates an experience which is simple and straightforward; one that does not overwhelm the user with useless information. One font type, 'Trebuchet MS,' is used across the website to guarantee uniformity across the website. Colours are kept simple, with a core black and grey background, along with a main red accent colour, often used for the hovering of links or icons. The quiz page uses a conventional green and red

colour scheme for correct and incorrect answers. The simple appearance of the interface contributes to the user experience and minimizes any possible distractions to provide a visually appealing design.

Information Architecture

The web application employs a flat hierarchy across three pages, with the quiz page being a subpage of the select page. This structure can be seen in the site map below:

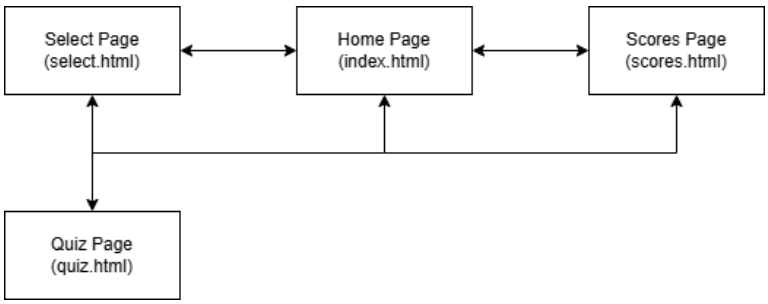

Figure 3 Site map

This hierarchy works well for the website and ensures a navigable experience, allowing the user to easily access content in a specific page when they need it. The quiz page needs to be kept separate from the rest of the pages as it requires the user to input data before accessing, but this is still integrated well into the site and does not detract from user experience. The website conforms to the LATCH framework, such as with the scores page, which sorts its data from the oldest to newest result. The web application also utilises some of the principles of Information Architecture (Brown, 2010), such as with the 'Principle of front doors,' where the home page does not contain everything and allows the user to arrive through other pages, or with the 'Principle of growth,' where the score page accommodates for a growing number of results to be displayed.

Technical Implementation

General HTML and CSS

The general HTML across all pages follow the common structuring conventions, with the use of semantic elements to define the contents of the HTML and non-semantic elements to create containers with classes which are styled with CSS. 'styles.css' is the main stylesheet for each HTML page and styles the HTML elements by using inherited styles. The CSS facilitates responsiveness depending by using relative units throughout. There are also several animations that enhance the visuality of the application, such as with the loading spinner icon.

Navigation Bar and Media Queries

The navigation bar is present in each HTML file within the '<nav>' semantic element. The container is a flexbox which allows for list items to displayed alongside each other. The list items contain links to the home, select and scores pages. When the window size is less than 800 pixels, the navigation bar changes to a mobile layout to prevent links appearing incorrectly. This is done using a 'max-width' media query. The previous links are replaced with a menu bar, which when pressed reveals the page links. This media query also adjusts other elements, such

as the button grid on the quiz page, adjusting from two to one column, to allow for more suitable sizing on smaller displays.

Form Input and Session Storage

The 'select.html' web page contains a form which allows the user to input data relating to the quiz they want to complete. The user must enter their name, select a category and select a difficulty. An event listener is attached to the button, so that when it is pressed, the data is saved to session storage. Session storage is used to ensure the data is saved only for the current session, as the data inputted is not needed for longer use. Firstly, it checks if the name input field is empty. If it is, an error message is displayed stating that the user must enter their name. Then, the inputted data is retrieved from the fields and three separate items are saved into the session storage, titled 'quizName,' 'quizCategory,' and 'quizDifficulty.' This information will be used on the next page to generate a URL for the API.

```
function saveFormData() {
    if (checkName()) {
        let formName = document.getElementById("form-name").value;
        let formCategory = document.getElementById("form-category").value;
        let formDifficulty;

        for (let i of document.getElementsByName("difficulty")) {
            if (i.checked) {
                formDifficulty = i.value;
                break;
            }
        }

        sessionStorage.setItem("quizName", formName);
        sessionStorage.setItem("quizCategory", formCategory);
        sessionStorage.setItem("quizDifficulty", formDifficulty);

        location.href = "quiz.html";
    }
}
```

Figure 4 Form saving code

API and Quiz Functionality

The Open Trivia Database API is used to source trivia questions based on a selected difficulty and category. Rather than having to come up with questions, pulling from a database of questions made for this purpose is more ideal. The 'quiz.html' page includes score text, question text, four answer buttons, and three heart icons. The score signifies the number of questions the user has gotten correct, and the hearts indicates the number of lives left before the game ends. To begin the quiz, the application generates an API URL that concatenates the inputted data from the form on 'select.html.' Using the fetch() function, the API is called. Provided there are no errors, the data will be saved as a variable. If the result returns a '429' status code, it means the data has been requested too many times. To handle this, the API is recalled after five seconds to ensure the website does not break. If another error occurs, such as with the API suffering an outage, backup data is used instead.

```
function callApi() {
    document.getElementById("loading").style.display = "flex";
    document.getElementById("content").style.display = "none";

    fetch(generateApiUrl(category, difficulty))
        .then(function (result) {
            if (result.status === 200) {
                return result.json();
            }
            else if (result.status === 429) {
                setTimeout(() => {
                    callApi();
                }, 5000);
            }
            else {
                getBackupData();
            }
        })
        .then(function (data) {
            jsonQuestions = data;
            refreshQuestion();

            document.getElementById("loading").style.display = "none";
            document.getElementById("content").style.display = "inline-block";
        })
        .catch(function (error) {
            loadingMsg.innerHTML = "This may take longer than expected...";
            console.log(error);
        });
}
```

Figure 5 Call API code

Based on the question number, the question is saved to a string variable, and the answers are pushed to an array. The elements in this array are then shuffled to ensure that the correct answer is not in the same button each time. The score, question, answers and hearts are then displayed on the page. For each button, there is an event listener which will call a function to check the user's answer when clicked. If the button pressed is equal to the correct answer, the button will turn green and no lives with be deducted. If the button pressed is equal to the incorrect answer, the button will turn red and the correct answer will turn green, while an animation plays for the respective heart icon and will subsequently deduct a life. This process repeats until the player has run out of lives.

Scores and Local Storage

When the quiz ends, an item called 'quizScores' is set in local storage. Local storage is used because this data should be accessed across sessions. An object literal is used to save the user's name, category, difficulty and total score. This information can then be viewed on the 'scores.html' page. This page contains a table which will be used to display this data. For each score saved in local storage, a new row is created, with the respective data being displayed.

Critical Reflection

The web application succeeds overall in delivering an engaging user experience, however there are elements which could have been improved. In terms of functionality, the web application works exactly as expected. The user can input data into the form on the select page as is typical with the correct inputs. The quiz page functions properly, with buttons all behaving how they are supposed to. The scores page shows all the previous attempts and displays them in a table format. All the functions seem to work without any error, however some of the current functionality could have been improved. The scores page could have been improved to allow more control over what is displayed, filtering results by name, category, or ordering scores from

highest to lowest. This would give greater incentive to pursue a higher score and create an improved user experience. I also believe that the form could have been improved to collect data in a more efficient manner. The drop-down menu could have been changed to a more visual selection of categories to place greater emphasis on choosing a category and provide a more engaging experience.

Visually, there could have been more improvements. There was a focus on creating a simple and elegant design for the interface which would focus only on delivering essential pieces of information. While this was an effective approach, the home page especially looks slightly uninteresting, with there only being text and a button to go to the select page. To improve this, the home page could have done a better job consolidating the different pages together to create a centralised hub of information with links to select and score pages. The use of colour could have also been more prevalent. While the black background looks professional, the use of more colours throughout would have been more visually interesting and would reflect the application's function. A greater use of animation, such as with navigation bars and transition could have created a more dynamic and fluid experience and would have contributed to a greater user engagement. Despite these setbacks, these issues are relatively minor and do not detract from the user experience all too much.

Lastly, the web application achieves a good level of usability. The user can navigate through the application with ease, with conventional icons used throughout to provide a familiar and easily understandable experience. However, there could have been some more work to accomplish a greater level of usability. The use of back buttons or breadcrumb navigation could have been used to provide the user with more navigational aid. More accessibility features could have also been provided to facilitate all users. Features like light and dark mode could have been added to accommodate user preferences, or the use of sound to identify whether a question is correct or incorrect.

Bibliography

Brown, D. (2010). Eight principles of information architecture. *Bulletin of the American Society for Information Science and Technology*, 36(6), 30-34, Available at: doi:0.1002/bult.2010.1720360609.

Nielsen, J. (1994). *10 Usability Heuristics for User Interface Design*. [Online]. Nielsen Norman Group. Available at: https://www.nngroup.com/articles/ten-usability-heuristics/ [Accessed 28 May 2025].