CS5003 — Masters Programming Projects

Assignment: P1 – Project 1

Deadline: Wednesday 27th February (Week 5) 21:00
Credits: 22% of the overall module grade

NOTE: MMS IS THE DEFINITIVE SOURCE FOR DEADLINES AND CREDIT DETAILS

You are expected to have read and understood all the information in this specification and any accompanying documents at least a week before the deadline. You must contact the lecturer regarding any queries well in advance of the deadline.

Aims

The main aim of this project is to teach you to write clearly structured single-page web applications. You will also learn to use JavaScript libraries as components of your application and, in particular, to use APIs to access remote services.

Details

In this project, you will create a web version of a trivia quiz. Your single-page Web application will present a sequence of multiple choice questions, and the player will choose one of the provided answers. If the answer is correct, the player is awarded a number of points. After each question, the player can choose to quit the game, retaining their score, or continue with the next question. If the player answers three questions wrong, they lose the game and get a score of 0.

The questions will be obtained from the online trivia collection at:

`https://opentdb.com/api_config.php`

Your application should consist of client-side Javascript, using the appropriate API calls to provide suitable questions. Your submission should be written in modern JavaScript (a.k.a. ECMA Script 6) with appropriate HTML, CSS and any libraries you find useful. It should *not* be written in any other language or JavaScript dialect (e.g. CoffeeScript, HAML, Pug, etc.).

Requirements

Your application should provide the functionality described below. You must make sure you have completed all the requirements in each section before moving onto the next.

Basic

Your application should provide the following:

- Allow a player to start a new game.
- Repeatedly present the user with questions obtained from the API and allow them to select an answer.

- Track the numbers of questions a player has answered correctly and incorrectly.
- Finish the game appropriately when a player answers three questions incorrectly (i.e. the player loses the game and gets 0 score).
- Finish the game appropriately when a player chooses to stop (i.e. the player 'wins' and gets a score equal to the number of questions they answered correctly).
- After a game has finished, allow a player to start a new game without reloading the page.

Intermediate

Your application should provide all the requirements described in Section Basic, plus the following:

- Assign the player a score based on the difficulty of the question (e.g. easy = 1; medium = 2; hard = 3). Use this to calculate the player's score.
- Allow the player to request two of the wrong answers be removed for a question, leaving them with the correct answer and one wrong answer (a 50-50 lifeline). Only allow this once per game.
- Provide a countdown clock limiting the time available to answer a question. If the time runs out, the player is deemed to have got the answer wrong.
- Prevent the same question coming up more than once (unless the page is reloaded).

Advanced

Your application should provide all the requirements described in Sections Basic and Intermediate, plus the following:

- Track the top ten scores, and the names of the players who achieved them. After each game, display the names and scores in a leader board.
- At the start of each game, present the user with four randomly selected categories and allow them to choose one as their 'bonus' category. Throughout the game, if the player answer a question in this category correctly they receive double points for the question.
- Allow the user to choose the difficulty of their next question (easy, medium, or hard).
- Allow the player to pause the timer for a question for 1 minute so they can ask a friend what the answer is (an "ask a friend" lifeline). Only allow this once per game.
- Allow the player to ask the system what the answer is (an "ask the host" lifeline). The system should choose an answer and say how confident it is its answer is correct. There should be a chance the system will choose the wrong answer or say that it does not know the answer. The chance of these different outcomes happening should be determined by the difficulty of the question (e.g. the system is more likely to give the correct answer for an easy question and not know the answer to a hard question).

Deliverables

A single `.zip` file must be submitted electronically via MMS by the deadline. It should contain:

- the source code for your application, including your CSS, HTML and Javascript
- A short report (approximately 1000 words), in `PDF` format detailing the design of your solution, discussing any requirements and design decisions taken, and reflecting on the success of your application. Try to focus on the reasons for your decisions rather than just providing a description of what you did.

Submissions in any other formats may be rejected.

Marking Criteria

Marking will follow the guidelines given in the school student handbook:
https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptor

Your submission will *not* be evaluated based on aesthetic appeal (this is not a visual design course), but use of CSS and DOM scripting which enhances the experience and interactivity will be rewarded.

Some specific descriptors for this assignment are given below:

- A *poor implementation* **in the 0–7 grade band** will be missing nearly all required functionality. It may contain code attempting a significant part of a solution, but with little success, together with a report describing the problems and the attempts made at a solution.
- A *reasonable implementation* **in the 8–10 grade band** should provide some of the functionality described in Section Basic, and demonstrate reasonable use of HTML and Javascript. The code should be documented well enough to allow the marker to understand the logic. The report should describe what was done but might lack detail or clarity.
- A *competent implementation* **in the 11–13 grade band** should provide all the functionality described in Section Basic, demonstrate competent use of HTML and CSS (e.g. for layout and positioning), allowing a player to play a complete game. The code should be documented well enough to allow the marker to understand the logic and should be contained in JavaScript files that are separate from the HTML code. The report should describe clearly what was done, with good style.
- A *good implementation* **in the 14–16 range** should provide all the functionality described in Section Basic and some or all of the functionality from Section Intermediate. It should demonstrate good code quality, good comments, and proper error handling. Good use of CSS for layout and styling is expected for a submission in this range (not unmodified defaults). The report should describe clearly what was done with some justification for decision, with good style, showing a good level of understanding.
- An *excellent implementation* **in the 17 and higher range** should provide all the functionality described in Sections Basic and Intermediate, and some or all of the functionality from Section Advanced. It should demonstrate high-quality code and be accompanied by a clear and well-written report showing real insight into the subject matter.
  Note: For this practical you do not need to invent your own extensions. Concentrate on providing high quality, sophisticated implementations of the requirements in this specification and writing an insightful report demonstrating understanding.

Word Limit

An advisory word limit of approximately 1000 words, excluding references and appendices, applies to this assignment. No automatic penalties will be applied based on report length but your mark may still be affected if the report is short and lacking in detail or long and lacking focus or clarity of expression.

A word count must be provided at the start of the report.

Lateness Penalty

The standard penalty for late submission applies (Scheme B: 1 mark per 8 hour period, or part thereof):
https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#lateness-penalties

Good Academic Practice

The University policy on Good Academic Practice applies:

`https://www.st-andrews.ac.uk/students/rules/academicpractice/`

Finally

Don't forget to enjoy yourselves and use the opportunity to experiment and learn! If you have any questions or problems please let me know (`mailto:ruth.letham@st-andrews.ac.uk`) — don't suffer in silence!