Education Application Future Developers Guide

This document details all the section we (the original project members - Jeremy Dunnet/Patrick Crockford/Ethan Bell/Casey Rogers/Oliver Yeudall) have identified as needing to be changed/revised or considered when taken further than the current state the product is currently in.

The current state of the product: The Education Application is currently a 100% client-side served, privately hosted web application that contains mostly demo/junk content.

The future of the product: This product was design and is envisioned to be live hosted on a hospitals web server and full of medical content.

Each file in the repository that has changed listed below includes

Method(s): a list of all methods that require a certain change

Changes: the changes suggested

Reasoning: why we believe the above changes should be considered

# Files

## Browsertest.js

Method(s): Entire file

Changes: All functionality should be converted to a server-side scripting language (PHP etc.)

Reasoning: This file tests whether a browser the user is accessing the site is one we support (to ensure all HTML code works and preventing older browser version being exploited). We recommend this be made server-side to reduce the amount of times it is run (only on session connection) and so that users cannot edit the javascript using developer tools and then access site anyway

## JSON files

Method(s): All current json files (located in json folder)

Changes: Convert to database tables

Reasoning: This change enables most of the following changes to work (since they would access a server-side database of information) as this is more common practice. JSON files are parsed in their entirety - so conversion to database would reduce memory usage by only transmitting required data.

## Media files

Method(s): All media files (located in the media folder)

Changes: Storage in external database

Reasoning: This change is mainly to reduce the web application memory size - as if the media is decoupled from the source code it can still be linked when requested from a database.

## popup.js

Method(s): loadDefinitions, loadDefinitionsJSON and parseDefinitions

Changes: Convert from javascript JSON parsing to server-side database script parsing

Reasoning: This code should be made server-side to reduce attackers ability to mess with it (retrieve information they want), reduce memory (only request what popups you need) and to frontload all loading (as JSON parsing may load screen with no content since script is still asynchronously waiting- while server-side waiting would be a blank screen).

## sanitise.js

Method(s): All methods

Changes: A copy of this code should be converted to server-side script and connecting to any server-side requests from user input

Reasoning: While the santisation is performed well in this javascript - it can be messed with in developer tools and request packets can be modified before they leave the user's machine so additional server-side sanitization should be performed being accepting any user input sent.

## search.js

Method(s): All methods

Changes: This could be done as a server-side script

Reasoning: This is a consideration depending on how much information you decide the HTML should contain (since for this implementation a lot of search terms are placed in data HTML attributes to allow search to find them) and if you wish the search to be dynamic (search results provided in popu as user types search terms) or static (results are displayed on a search page after user submits their chosen search terms). The dynamic search must have almost all functionality as javascript while the static can be server-side scripting which would allow easier sanitisation and control of input/output.

## quiz.js

Method(s): loadQuizJSON and setQuestions

Changes: This functionality should be server-side script.

Reasoning: The loading of questions should be done by requesting information from the database (needs server-side script) and random number generation can also be done their (prevents user knowing what numbers equal what questions and therefore prevents them modifying the numbers before requesting the questions attached).

Method(s): showResult

Changes: Part of this method should be server-side (the correct answer checking part)

Reasoning: The answers should be checked by requesting the server to check if the answer given is the same as one in the database so that users cannot view the javascript to find the answers. The rest of the script can remain as javascript (the changing of correct/incorrect marks and button changes) as that needs to happen client-side anyway and makes no difference if modified.

## chapterSwitch.js

Method(s): loadChapterJson, loadChapter, displayChapter, loadQuiz and selectChapter

Changes: All code should be moved to a server-side script

Reasoning: All of the above methods relate to either the retrieval of content(loadChapterJSON and loadQuiz) or the parsing of the retrieved content to inject into the page (loadChapter, displayChapter and selectChapter). This can all be done server-side to retrieve only the required data (reduce memory) and to preload information (speed up site loading).

Method(s): loadBookmarks

Changes: Storage should be changed to a server-side based mechanism (sessions/cookies)

Reasoning: The bookmarks currently maintain persistent marking on the site as long as the user uses the same browser and the same machine to access the site again. This is not convenient for a user and so the bookmark storage should be placed server side so that as long as the same IP/Account (if you choose to implement) is used the same bookmarks are loaded.