Web Development Exercise
SPECIFICATION
ECM1417 – Web Development
Module Leader: Dr. Wang Miao
Academic Year: 2024/25

Title: **Web Development Exercise Assessment**

Submission deadline: **18th March 2025**

This assessment contributes **70%** of the total module mark and assesses the following **intended learning outcomes**:

Module Specific Skills and Knowledge

- Demonstrate knowledge of web architecture and design patterns for web programs.
- Develop programs that run on Web clients and Web servers, and work together.
- Discuss the difficulties in achieving usability, accessibility, efficiency and robustness when developing Web programs.

Discipline Specific Skills and Knowledge

- Choose appropriate techniques and tools and implement a program to meet the given requirement specification.
- Critically evaluate how well a program meets a given requirement specification.

Personal and Key Transferable / Employment Skills and Knowledge

- Demonstrate an ability to develop usable, accessible, efficient, robust and secure Web sites.
- Demonstrate awareness of the importance of usability, accessibility, efficiency and robustness in computer programs.

This is an individual assessment, and you are reminded of the University's regulations on collaboration and plagiarism. You must avoid plagiarism, collusion, and any academic misconduct behaviours. Further details about academic honesty and plagiarism can be found at https://ele.exeter.ac.uk/course/view.php?id=1957.

AI Supported: Please see:

Using Generative Artificial Intelligence (GenAI) tools in academic work - Referencing - LibGuides at University of Exeter

1. Requirements

In this assignment, you will deploy a LAMP stack web server to your Azure VM and design a website that enables a mini-game to be played in a browser. The requirements detailed below specify exactly how this website should be built.

1.1. Web server architecture

The web server will be hosted on the Microsoft Azure platform that was introduced in week 2 and the code for your submission will be in the ELE. Each student has been allocated an individual VM on which you will be expected to configure and deploy your LAMP stack. This means on your Linux VM you will need to install and run Apache http and php.

1.2. Website design

1.2.1 Webpages

Four webpages should be created using php to run the website. The landing (index) page, registration page, leaderboard page and the game page. These webpages must use the following names:

- index.php
- registration.php
- pairs.php
- leaderboard.php

1.2.2 Page content and layout

All pages will have a shared navbar menu at the top of the pages. The navbar menu must include the following content items with their corresponding name attribute:

1) Home - name“home”
2) Play Pairs – name = “memory”
3) If the user has registered a profile display a link to leaderboard
   Leaderboard – name = “leaderboard”
4) Else, if the user hasn’t registered a profile display a link to register (without leaderboard)
5) Register – name = “register”

The navbar items must use the Verdana font in bold with the font size of 12px. The Home item should be aligned on the left and the Play pairs and Leaderboard items on the right. The navbar background must be ’blue’. If an emoji has been selected as the user image in the registration process, this should also be shown on the navbar.

Each webpage must then contain the rest of the content for the webpage in a div with id attribute value of “main”. The main div will have a background image, this must be the Arcade image (arcade-unsplash.jpg) linked below, set at the full width of the viewport and in an absolute position in the centre, vertically and horizontally.

Image source: Photo by Carl Raw on Unsplash

Advanced layout features: To make the webpage more advanced you can use bootstrap for mobile and responsive layout. You can also add animations to your game to make it more engaging for the player. For example, when the player clicks on a card, you can add a flipping animation to simulate the card turning over. You can also add an audio track that starts when the game is playing.

All layout and content features will be assessed across all pages.

1.2.3. Landing page

The landing page (index.php) must display the welcome message “Welcome to Pairs” if the user is using a registered session, followed by a button with the content “Click here to play” that contains a hyperlink to the pairs.php page. Alternatively, if the user isn’t using a registered session the landing page should provide a paragraph with the content “You’re not using a registered session? Register now” where the Register now text has a hyperlink to registration.php.

1.2.4. Registration

The registration page, registration.php, will set session variables and cookies to generate a profile for the user. Note that all user profile information should be stored in cookies and there is no need for a database.

The user profile registration form will include:

- Username/nickname
  – If the input contains any of the invalid characters, the program should return an error message and show this message under the <input> tag to the user to enter a new username. The invalid set includes: ” ! @ # % &^* ( ) + = { } [ ] — ; : “ ’ < > ? /
- Avatar selector/generator
  – Simple - default image is set as the avatar for all users
  – Medium - select an image from a set of images
  – Complex - select and configure features to assemble an emoji avatar/image

1.2.5. Pairs webpage and emoji matching game

’Pairs’ is a memory game that is played by selecting pairs of cards that are face down on the table. This page, pairs.php, will enable users to play this game virtually in the browser. This webpage will have a div with a grey background and a 5px box shadow where the game can be played. The game should implement the following features:

1) The game should start when the user clicks the button ’Start the game’. The button should disappear when the game starts.
2) Simple: The game should display 6 ’cards’ (3 pairs) with preset images and allow two pairs to be turned at a time. If the cards match the cards should remain face up. The game should make a record of points scored based on how many attempts the user takes to match all the cards.
3) Medium: The game will display 10 ’cards’ (5 pairs) and the emoji images on the cards will be generated by combining features (colour, smile and eyes) randomly for each run of the game. The game should record the number of points and time taken. Points

are calculated based on how many attempts and how much time the user takes to match all cards.

4) Complex: The user will progress through levels of the game with an increasing number of cards at each level. As the levels progress, users will also have to match 3 and 4 cards rather than pairs. Again, random emojis configured from features should be used as the card images. The game should record the total number of points as well as the number of points per level for each user based on how many attempts and how much time it takes to match all cards. The background colour of the content div should change to gold (#FFD700) during gameplay if the user exceeds the previous best score for the current level.

On completion of the game, if the user is in a registered session, they should be shown their score and asked if they want to submit their score or 'Play Again'. If the user clicks 'submit' this should send a post request to the leaderboard page. If the user clicks 'Play again' you should restart the game and reset the score to zero.

Here is an alternative plain English description of the game.

The game page: Once you have the basic layout of the game board, you need to implement the game logic using JavaScript. The game should begin by shuffling the cards and placing them randomly on the game board. Then, when the player clicks on a card, the game should reveal the image on the card. If the player clicks on a second card, the game should check if the two cards match. If they do, they should remain face up and the player should be allowed to click on another pair of cards. If they don't match, they should be flipped back over, and the player should try again.

Add game logic: Once you have the basic interactivity working, you can start adding the game logic. When a user clicks on a card, you should keep track of which cards have been clicked and compare their contents. If the contents match, keep the cards flipped over. If the contents don't match, flip them back over and allow the user to try again.

Add game state management: You'll need to keep track of the state of the game, such as the number of moves the user has made, the time elapsed, and whether the game has been won or lost. You can use variables to store this information and update it as the game progresses.

Add win and lose conditions: Finally, add win and lose conditions to the game. The game should end when all the cards have been matched or when the user has made too many incorrect guesses.

Timer and score: You can add additional features to the game such as a timer and score to make it more challenging. For example, you can set a time limit for the player to match all the cards and keep track of the number of attempts it takes the player to complete the game.

1.2.6. Leaderboard

The leaderboard page, leaderboard.php, will have a div with a grey background and a 5px box shadow where a formatted table that contains usernames and best scores will be displayed. Complex solutions with levels will display the best scores per level and the total

best scores. The table should have border spacing at 2px, and the background for table header cells should be 'blue'.

To enable multiple users in the leaderboard the php code will need to populate an appropriate data structure to store submissions from multiple users on different browsers. Note, no additional marks will be awarded for using a database for this feature and multiple users can be augmented by creating private browser sessions, so session variables don't conflict, and cookies aren't stored.

2. Submission Requirements

Your submission consists of a code part and a report part. You need to ZIP both parts into one file and submit it *electronically* in the ELE.

The Code Part. Your code part includes all your HTML, CSS, JS code and assets. At the root of your code ZIP please also include a README file that links to your Azure lab's VM (named **"ECM1417-2020"**) and provides a bullet point list of the features you've completed for each webpage. This README file will guide the marking process as we will not be able to discover all features without them being mentioned here. The README should not exceed 250 words and must only contain concise bullet points.

The Azure VM will be used as the primary resource for code marking. The teaching team can 'start' your VM so the VM can be turned off for submission but will be launched by us for marking. Your submission will enable us to open the README file, launch the VM and navigate to website in a browser, open each webpage in sequence and check all features listed in README.

Note, we will validate that the relevant VM has been linked from the README and any attempt to link to another student's VM will be considered as plagiarism.

The Report Part. The report (max 5 pages), with minimum 2 cm margins and 11-point text, should contain two items listed below.

- A (max 4 page) document *detailing* your design and reasons with respect to both your production code, and any known design issues. This part of the document should be no more than four sides of A4.
- A (max 1 page) development **log**, which includes date, time and duration of web development. This part of the document should be no more than one side of A4.

References are not considered for page limit.

3. Mark Scheme

Marking Scheme	Description	Mark
The Report		
Structure and contents of the report	The report is well structured and presented. The design is well explained and matches the specifications provided and the implemented code. The development log contains the specified information.	40%
The Code		
Landing page	Evaluate the features and content of the landing page, including the welcome message, click button, registration message and hyperlinks. Proper implementation of the layout, styling and structure of the landing page.	5%
Registration page	Evaluate a user profile registration form, which can return the error message for invalid username inputs and generate avatar for different user profiles. Proper implementation of the layout, styling and structure of the registration page.	10%
Pairs game page	Evaluate the functionality requirements of the pairs-game, including game page, game logic, state management, timer, and score display. A simple implementation can score up to 50% of the marks for this section, medium up to 75% for this section and complex 100% for this section. Proper implementation of the layout, styling and structure of the pairs-game page.	30%
Leaderboard page	Evaluate the features and content of the leaderboard page to display the best scores for users. Proper implementation of the layout, styling and leaderboard table structure of the leaderboard page.	5%
README file and code quality	Evaluate the details of the structure of the code. The description and webpage link in the README file. Proper use of language syntax with detailed comments.	10%
Penalties		
Penalty	Submission is greater than the stated number of pages (5), excluding the ELE sheets.	No penalty applied, but only the first 5 pages will be marked.
	VM can't be successfully launched, and the markers can't navigate to website in a browser.	-20%

Please contact the Harrison Hub if you have any submission issues.