ICT375 Advanced Web Programming
Assignment One Marking Guide
Updated on 1 September 2021

The marking guide has marks allocated based on the requirements as stated in the assignment question. Assessment for each component will be determined by the **quality** of your work in meeting those requirements.

Summary of solution: a solution to the problem is usually centred around three *asynchronous* client-server pairs:

1. the client sends the details of a student to the server which then add the details of the student to file `data/students.csv` and save the photo (if any) to directory `photos/`, and then provide feedback to the client (whether the submission is successful or not, eg, the id already exists)
2. the client sends the search key to the server which then searches the file `data/students.csv` for matching students, and then send the result back to the client. The client then displays the list of matching students including clickable links to students' photos (if any).
3. When the user clicks the link to the photo of a student, the client sends the request to the server which then sends the requested photo back to the client for display.

Submission of student details (35 marks)

1. (client) Collect and submit student details (18 marks)
   - allow the user to enter 7 pieces of information of a student, including selecting a photo file (5 marks)
   - validate user input and provide validation feedback (3 marks)
   - submit the student details, including the photo (if any) to the server asynchronously (8 marks)
   - and display the feedback of the submission from the server (2 marks)

2. (server) Save the student details into files (17 marks).
   - on receipt of the client request with a student's details, save the photo (if any) to the sub-directory `photos/` with the correct filename (5 marks)
   - save the seven field values including the link to photo to file `data/students.csv` as a single line (make sure no duplication). (10 marks).
   - sends the feedback response to client (whether submission is success or failure) (2 marks)

Search for students (25 marks)

3. (client) Send search key and display results (13 marks)
   - allow user to enter a search key and then submit the search key asynchronously by pressing Enter or clicking a button (3 marks)
   - display the details of matching students in tabular format including the clickable links to their photos (10 marks)

4. (server) Find matching students and send the result to client (12 marks)
   - server gets the search key from the client and searches file `data/students.csv` for matching students (8 marks)

- server sends the result in JSON format to the client (4 marks)

Display of student photo (15 marks)

5. (client) send photo request (7 marks)
    - allow user to click the link to the photo of any student (if any) and send the request asynchronously to the server (3 marks)
    - display the photo from the server in the same page (4 marks)

6. (server) send photo to client (8 marks)
    - sends the respective photo to the client. The server must be able to handle photos of multiple formats (eg, .png, .jpg). (8 marks)

Overall quality (10 marks)

7. Overall quality of your solution including aesthetical aspect of the front end, user-friendliness of the application, quality of your source code, and robustness of the program (up to 10 marks)

Other aspects

8. The pages should not reload, otherwise deduct 5 marks
9. All requests must be handled asynchronously, otherwise deduct 15 marks.

Report (15 marks):

1. Introduction/overview (1 marks).
2. Description of deployed modules (*your own and non-core only*): how they work; why they were used the way they were (1 marks).
3. Detailed description of the overall design – the different components and how they worked together as a system (including diagrams) (1 marks).
4. Description and usage of data structures: what, how and why (1 marks).
5. **Thorough testing strategy and evidence of all required functionality: 10 marks. See Testing below.**
6. Conclusion and reflection (1 marks).

Testing:

1. Each piece of testing evidence is best presented with a combination of a description explaining the purpose and outcome of the test together with the resulting test output (such as terminal output or screenshots).
2. Marks will be allocated for the initial web page design and presentation of output. The design should use relevant information gathering techniques (check boxes, radio buttons, drop-down lists, etc.). If necessary, the design should also utilize JavaScript to validate input.
3. Test rigorously with different browsers (Chrome, Firefox and Edge). Test on different devices (smartphone, pad, laptop, desktop).
4. For each claimed feature of your application listed in the Conclusion, you must include test evidence to back up your claim.