**CITS3403 Project Due on** **12noon, May 23, 2022**

**Web Project Specification: Online Daily Puzzle**

Project Description

For this project you are required to build an online daily puzzle, operating in a similar manner to [Wordle](https://www.nytimes.com/games/wordle/index.html). The app should have a short daily exercise that users do; the users receive a score, or a rank, or some kind of feedback; they can share their performance with others through social media or similar mechanisms; and they can track their progress and statistics. It should also be convenient and easy to play via mobile devices and engaging enough for the users to want to return regularly. The application should be written using HTML, CSS, Flask, AJAX, JQuery, and Bootstrap. (Any additional technologies not mentioned in lectures will require special approval from the unit coordinator).

**Examples of Daily Puzzle Games:**

[**https://www.dailypuzzle.com/**](https://www.dailypuzzle.com/)

[**https://www.mindgames.com/Daily+Puzzles**](https://www.mindgames.com/Daily+Puzzles)

In addition to the web application, you should create a private GitHub project that includes a readme describing:

1. the purpose of the web application, explaining the design of the game
2. the architecture of the web application.
3. describe how to launch the web application.
4. describe some unit tests for the web application, and how to run them.
5. Include commit logs, showing contributions and review from both contributing students

**Criteria: Front-end (50%)**

1. The web application must be functional so that the user can easily access the puzzle.
2. The webpage must be implemented using HTML5, CSS and Javascript (or a subset thereof).
3. All resources used (inlcuding pictures, javascript libraries, css) must be full referenced.
4. The website must use HTML5, and CSS. The HTML and CSS must pass this [validator](https://validator.w3.org/).
5. The website must work on Chrome, Firefox and Microsoft Edge, and render well on mobile devices.
6. The website should have at least three pages/views:
   * one promoting the theme and purpose to users and explaining the rules.
   * one presenting the puzzle/game.
   * one page showing aggregate results and usage statistics
   * optionally it may also have registration and login pages. If registration and login aren't used, some other mechanism should be used to track return users;
   * there should also be a mechanism for administrators to upload new puzzles, or vet generated puzzles. These maybe through command line operations, etc, but they should be robust so that administrators will not cause the app to crash.
7. There must be a consistent style (via css file) for all pages yet each page should be easily identifiable.

**Marking Scheme**

* HTML5 - style, maintainability, validation **10%**
* CSS -style, maintainability **10%**
* Javascript-code quality, validation of user generated data, execution **15%**
* Style - look and feel, usability **10%**
* Content - coherence, effectiveness **5%**

**Criteria: Backend functionality (50%)**

The second part of the project criteria is the back end functionality of web application. The web application should be implemented using Flask (any additional libraries/modules require unit coordinator approval), and provide at least the following functionality:

1. A user account and tracking feature.
2. A method to store puzzles and results.
3. A method to update and vet puzzles.
4. A mechanism for users to share their achievements.

Students should submit a complete Flask application providing the functionality of the project. This should be submitted as a zip including

* a full readme.md, describing the design and development of the application, and giving instructions on how to launch from local host.
* the git log, showing commits from both partners
* all source code, with comments and attributions for any external libraries.
* a requirements.txt file, listing all packages used. To build the requirements.txt file for your virtual environment, use the command: pip freeze > requirements.txt while your virtual environment is active.

**Do not** submit the virtual environment directory, or the .git directory. These are overly large, not required and will result in a penalty if they are submitted.

Submit your zipped file to [cssubmit](https://secure.csse.uwa.edu.au/run/cssubmit), making sure that you submit it to the correct unit code, as CITS3403 and CITS5505 have the same name.

**Marking Scheme**

* Codecode quality, complexity of task, execution **10%**
* Persistence and User authentication Database schema and models**10%**
* Testing Unit tests and Selenium Tests**10%**
* Design Purpose and level of complexity**10%**
* Collaboration Git logs and agile processes **10%**

The [Marking Criterion](https://teaching.csse.uwa.edu.au/units/CITS3403/projects/Project2Criterion.pdf) will be available by the midsemester break.

#### **Demonstration Schedule**

In week 12 all members of each team must present and demonstrate their project. A schedule will be set up for demonstrations by week 8.