CITS3403/5505 Final Test.

Taking your project to the next level.

*This Exam is worth **50%** of your final grade and must be done individually. For equity reasons, the teaching staff will not be able to answer individual questions between the exams release and the due date. Questions may be posted to the **Final Test** teams channel and may be answered by teaching staff. Any communication, commentary, or discussion of the test during this period will be considered academic misconduct. Submit a zip containing your HTML document and any required media files to https://secure.csse.uwa.edu.au/run/cssubmit by **5pm Wednesday, June 1, 2022***

Consider the challenging of launching your daily game, produced in the group project, as a live game used by thousands of users around the world everyday. You will have to make sure the game is as engaging as possible, the code must be reliable, the servers will have to handle the traffic, and you must respect the users data.

For the final unit test, you will need to produce a HTML document describing how you might address these challenges. The questions must be answered with respect to the current state of your group project. The document should satisfy the following constrains:

1. The document should consist of a single HTML file, with embedded CSS and JavaScript code. You may include some additional image or icon files in the zip, but you must ensure that these will render in the document when the zip is expanded.
2. You may not use any libraries or external code, like Bootstrap or JQuery, and the page must run without using live server, so AJAX should not be included in the document.
3. The page should render in a modern web browser (Chrome, Edge, Firefox, Safari etc) from the file explorer, and should be easy to read and use.

Specification: The web page should have the unit title, your name and student number as a heading at the top and the web page should have four sections, with clear and easy navigation between them. The sections should be titled *Overview*, *Client-Side, Server-Side* and *Software Processes,* and the sections should have the following specifications:

1. **Overview:** Present a description of your application from the users point of view. You should explain the way the application works, the rules of the game, and how the user can compete the puzzle and share their results. You should include a screenshot of the application.
2. **Client-Side:** Give a brief (100 word) overview of the client side architecture of your application, and provide:
    1. An interactive To Do List of features to add and issues to address in the client-side application. The list should allow a user to add new items to the list, and remove existing items. Populate the list with 3 items that would need to be done to scale the application, with a description of what each item involves and why it is important (2 sentences each).
    2. An alternative presentation theme for the application. Specify colours, fonts and styles for an alternative theme, an explain your choices (1 paragraph)
    3. The application could use server-side rendering, (utilising Jinja and Flask for example), or could use client-side rendering (using JavaScript and a REST API). Present and explain the advantages and disadvantages for each option side by side in the webpage, and colour the background of the prefered option green (about 100 words for each option).

3. **Server-Side:** Give a brief (100 words) overview of the server side architecture of your application, and provide:
    1. An interactive To Do List of features to add and issues to address in the server-side application. The list should allow a user to add new items to the list, and remove existing items. Populate the list with 3 items that would need to be done to scale the application, with a description of what each item involves and why it is important (2 sentences each).
    2. Describe what would need to be done to deploy your application to allow it to scale with thousands of daily users, and highlight the relevant characteristics of the Representational State Transfer Architecture that would enable such scaling (about 200 words).
    3. The privacy and confidence of users will be important to the applications success, but you will also want to track users engagement. Your application could require users to register accounts (active user tracking) or could use cookies or local storage to passively track users. Give the advantages and disadvantages of each approach side by side, and color the background of your prefered option green (about 100 words for each option).
4. **Software Processes:** Making a robust and reliable application will require a team of developers to build and maintain the application. Describe what you think would be the ideal team to work with to deliver and maintain the application (100 words), and provide:
    1. An interactive To Do List of tasks required to deploy and maintain the application. The list should allow a user to add new items to the list, and remove existing items. Populate the list with 3 items that would need to be done to scale up your application, with a clear description of what each item involves and why it is important (about 2 sentences each).
    2. Ensuring the reliability of the application as new features are added will be crucial to building a user base. Describe a testing strategy for your application identifying what kind of tests and validation processes should be used, and how they should be executed (about 200 words).
    3. Describe the process your team used in developping the application to its current state and give an honest and professional appraisal of each team member you worked with. What changes would you make within the project team if you were to continue to develop the application (about 200 words).

Marking Criteria:

Name:____________________________Student#:_______________________________

Criteria	Excellent	Good	Satisfactory	Inadequate	Comments	Weight
Basic Data.	Name, unit code and student number are clearly rendered at top of page.	Name, unit code and student number are rendered.	???	Refusal to accept marks that are clearly being given away.		/3
HTML	Valid code, well formatted, with excellent navigation	Valid code, and good navigation	Mostly valid code, adequate navigation	Invalid/incomplete code. Poor navigation		/4
CSS	Valid code with intuitive classes, and aesthetic appeal	Valid, maintainable code with intuitive classes,	Mostly valid code, using some novel style elements.	Invalid code, minimal style applied.		/4
JavaScript	Valid, well formatted code, with perfect function	Valid, well formatted code, mostly functional	Valid code meeting most of the requirement	Faulty or incomplete code.		/4
Overview of Application	Clear well formatted user guide and overview.	Clear user guide and basic overview.	Basic overview of game.	Does not convey application's purpose or appeal.		/5
Client Side: To Do List	Functional list, with clear elements and useful explanations	Functional list with sensible elements.	Mostly functional list, with some elements	Non functional or incomplete		/3
Client Side: New Theme	Well chosen style, with clear justification, and excellent CSS.	Clear style, with some justification, and good CSS.	Some style, well formatted CSS	Poor style, invalid CSS.		/3
Client Side: Rendering	Clear, well explained and reasoned pros and cons	Clear pros and cons, with some reasoning.	Sensible pros and cons.	Off topic or incoherent arguments.		/4
Server Side: To Do List	Functional list, with clear elements and useful explanations	Functional list with sensible elements.	Mostly functional list, with some elements	Non functional or incomplete		/3
Server Side: Deployment	Clear, well explained process, demonstrating an understanding of REST principles.	Clear, well explained process, covering the REST principles.	Reasonable deployment process explained	Poorly explained or invalid deployment process.		/3
Server Side: Privacy	Clear, well explained and reasoned pros and cons	Clear pros and cons, with some reasoning.	Sensible pros and cons.	Off topic or incoherent arguments.		/4
SW Process: To Do List	Functional list, with clear elements and useful explanations	Functional list with sensible elements.	Mostly functional list, with some elements	Non functional or incomplete		/3
SW Process: Test Strategy	Well considered and justified test strategy for unit and user tests.	Well considered test strategy for unit and user tests.	Reasonable unit and user test strategy.	Incomplete or ineffective test strategy.		/3
SW Process: Reflections	Insightful, professional reflections of project work, with clear plan for improvement.	Professional reflections of project work, with plan for improvement.	Some reflection of project work, demonstrating lessons learnt.	Lack of insight, or understanding of project process.		/4
Overall						/50