Team Name: ……..... Date: ……..

GitHub URL: ……

Team Members

Name: ……. StudentID:…… Name: …… StudentID:……

Name: ……. StudentID:…… Name: …….. StudentID:……

Video Details: ……

0-No feature, 1 Attempt, 2 Working, 3 Excellent

	Each sub-component should be given 0-3 e.g., 2	0-3
Demonstrate Understanding of HTML, CSS and Javascript	☐ HTML, CSS and Javascript validates fully (no syntax errors, warning, issues) ☐ Creative use of CSS styling (which lines/files) ☐ Advanced CSS3 layout (which lines/files) ☐ Creative use of Javascript (which lines/files why creative) ☐ HTML5 semantic elements used appropriately (evidenced, how, has the HTML been run through HTML validation software W3C and included in the submission) ☐ Game/website should be responsive (also responsive in a mobile/different browsers) – Code for managing client/server delays, e.g., Interpolation, prediction, AI. Good used of asynchronous Javascript (avoid stalling GUI) ☐ Hand written code (clean, structured, commented, ..) ***Your Code** (e.g., all files commented, license/author details top `all' files, indentation consistent across `all` files, should be the same for all team members (project coordination)	
Client/Server	☐ Manage communication between multiple `concurrent' client/server The games should not be running `separately' but shared interactive experience – manage concurrent/client server interaction of data (e.g., collisions/data conflicts) ☐ Handle issues (delays, network issues, corruption,) Interpolation, message pausing the game (delay is over 5 seconds), smart AI system, …	
Database/Security	☐ Data is managed for the website effectively and efficiently (Server/Database) ☐ Database (working reading/writing) [stores variety of information, such as high scores,] ☐ Data is automatically backed up (server side), ☐ Whitelist/validation testing for data submitted to server ☐ Client also tests/validates data sent from server (corruption/value ranges avoid cheating) ☐ Database security checks (all SQL commands are `security checked' avoid SQL injection) ☐ Data sent/received is encoded/decoded/validated somehow (e.g., hash/pattern ☐ IP tracking (prevent spamming/attack logging) ☐ Login system (user can register and login/store/continue game progress) ☐ Reset/restore password (maximum allowed tries, attack prevention) ☐ User passwords hash encoded with salting (or similar complex encryption)	
Code Structure	☐ Clear separation of code/styling	

	☐ Tidy code organisation, file and folder names (js folders/css folder, ..) ☐ Clear variable names, constants for fixed values, avoid using `magic' hard coded numbers scattered around code, .. ☐ Appropriate code nesting and indentation (methods with clear purposes, descriptions/comments – avoid adhoc/hack fixes), scoping and not having everything `global' ☐ Informative code comments (readme, `all' files commented, every function, style, ..)	
Usability	☐ Easy to navigate and use (Website/Game) (Accessibility testing? Different browsers, screen resolution, mobile/desktop, language - `evidenced' – detailed in the documentation) ☐ Clear awareness of accessibility principles (evidenced in the website, e.g., about/help page, also in the documentation/readme/code) ☐ Disclaimers/warnings – e.g., photosensitive epilepsy/flashing images ☐ Effective navigation at all sizes, content easy to locate Font size (scales to different screens, not just `fixed/hardcoded'), text/information not in margins, bright/animated buttons/text so easy to see/identify on the screen ☐ Limitations and optimizations Profiling data statistics, jpg vs bmp, appropriate file sizes/download considerations (avoid downloading everything for anything), sending only required data to specific clients (not all data to all clients)	
Development Progress	☐ Shows even work distribution (group work) (**evidence** on code written/bugs fixed/testing/…) ☐ Evidence of task development, tuning, feature refinement (week by week log of task development, i.e., not the last week or 2 weeks – over the full duration of the project) ☐ Live site (Github website) (GitHub page, with readme, and access to material/source files) ☐ Project management (team synergy, evidence of team working together on GitHub, helping/each other, such as, task lists, issues lists, bug fix lists available on GitHub over the duration of the project)	
Testing	☐ Testing integrated into the project from the start (as each feature was added a set of tests was defined/included in the code/documentation) ☐ Validation/verification process for ensuring reliability (e.g., code, standards, ..) ☐ Documentation on the tests (e.g., where delays added to the server to simulate poor connections, automatic errors added to the send/receive data, soak testing (left running 3+ days), larger number of instances created to simulate 100s or 1000s of simultaneous users)	

Notes/Comments