

Draw It or Lose It – Web Application

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/26/25 | Ryan Ward | Initial Draft |

**Instructions**

Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is a client that would like to take their existing Android only mobile game and produce a web-based version. Creative Technology Solutions (CTS) will provide the programming of the web application, web site, and supporting database. Maintenance of the system will be the responsibility of the client, unless otherwise specified. CTS will also provide training for the newly built system regarding setup, operation, and maintenance of the system. CTS will include the necessary hardware as part of the implementation bid, but maintenance of the hardware will fall to the client. The application will be designed to run across multiple web browsers and meet the requirements detailed below.

## Requirements

The following software requirements have been identified:

* A game will be able to have one or more teams involved
* Each team may have multiple players assigned to it
* Game and team names must be unique (should allow users to check names when choosing a team name)
* Only one instance of the game can exist in memory at a time.

## [Design Constraints](#_2et92p0)

* As this is a web-based game, various browsers must be taken into account. This may lead to extensive testing as there are many different web browsers available across the various operating systems.
* Servers must be able to run the game for multiple users concurrently. We will need an estimated user base to determine hardware requirements.
* An operating system for the web server(s) must be chosen. This operating system must be capable of meeting the specifications of the web application.
* A programming language for the application must be chosen. This should be chosen based on portability, upgradability, and maintainability.
* There should be a dedicated database in place to store prior game and user information. This system must be set up for security and scalability.

## [System Architecture View](#_ilbxbyevv6b6)

Please note: There is nothing required here for these projects, but this section serves as a reminder that describing the system and subsystem architecture present in the application, including physical components or tiers, may be required for other projects. A logical topology of the communication and storage aspects is also necessary to understand the overall architecture and should be provided.

## [Domain Model](#_8h2ehzxfam4o)

The program will consist of seven classes. An Entity class, which serves as the basis for the Game, Team, and Player class. These three classes inherit from the entity class. There is a zero-to-many relationship between the GameService class and the Game class, the Game class and the Team class, and the Team class and the Player class. The GameService class is set up with a singleton pattern, so there can be only one GameService at any time. The GameService can support multiple games, and each Game can support multiple teams, with each Team in turn supporting multiple players. The other two classes are the driver class, which serves as the entry point to the program, and the SingletonTester class, used for testing purposes. Encapsulation is used, particularly by the GameService class, which hides its fields as private, then provides only the necessary accessor and mutator functions. The GameService class also makes its constructor private, resulting in the singleton pattern mentioned above. The Game, Team, and Player classes all make use of polymorphism by overriding the toString() method to each produce their own unique output when printed. Abstraction is used by the entity class, which cannot be instantiated itself, but rather serves as the base class for the Game, Team, and Player classes.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#_2o15spng8stw)

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client’s requirements and look at the situation holistically, as it all has to work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac OS has a more expensive hardware cost, but is stable, with a user-friendly design. There would be a higher maintenance and upgrade cost due to proprietary hardware. There is also limited availability of server specific software. | Linux is a lightweight OS highly suitable for operating a server-side application. It is a lesser-known OS with less hardware support that is less user friendly than other options. However, there would be minimal cost for setup and maintenance. It is generally considered more secure than other options. | Windows is a well-known OS, with broad support for a wide range of hardware. It also has frameworks in place for server specific functions, although it is generally considered the least secure of the options. The cost will be moderate. | Mobile devices would serve as a poor choice for server-side applications, as they are intended to run client-side applications on devices with less storage, memory, and other necessary functions. There is a general lack of support for operating a server ecosystem on mobile architecture. |
| **Client Side** | Mac OS is a very popular operating system with a very user-friendly interface. It does have a higher hardware cost, resulting in increased upgrade cost. OS is limited to specific, proprietary hardware (unless running a virtual machine). Since this would be a web-based application, most of the client-side architecture would be handled by the web browser. We would need to ensure the application is supported by a range of browsers, especially the most popular ones (Edge, Firefox, Chrome, etc.), as well as Mac specific browsers like Safari. | Linux is the least used client-side system, with the least amount of application support. It is also considered less user-friendly. Many distributions of the OS are free to install and use. Since this would be a web-based application, most of the client-side architecture would be handled by the web browser. We would need to ensure the application is supported by a range of browsers, especially the most popular ones (Edge, Firefox, Chrome, etc.) | Windows is the most widely used operating system, providing a wide array of application support. It is a user-friendly system, although it can be prone to bugs and errors that can cause the system to crash. Windows supports a broad range of hardware options. Since this is a web-based application, most of the client-side architecture would be handled by the web browser. We would need to ensure the application is supported by a range of browsers, especially the most popular ones (Edge, Firefox, Chrome, etc.) Extra consideration may be needed to ensure security due to the prevalence of Windows based security flaws | Across mobile devices there are many different operating systems and versions to contend with. There would be different, OS specific applications to maintain (iOS and Android, at a minimum). Creating applications for specific devices would entail a larger cost and time investment and would require teams that specialize in each OS to be developed. |
| **Development Tools** | Java would be an ideal programming language, as Java can run on all platforms that support the JVM, so it would require just one team that is able to deploy the application via web-based environments. The team could select a single IDE such as intelliJ or Eclipse for production. Development may require other services such as Maven and Spring Boot, as well as a database management system. Choice of OS for development would be a matter of personal preference. | Java would be an ideal programming language, as Java can run on all platforms that support the JVM, so it would require just one team that is able to deploy the application via web-based environments. The team could select a single IDE such as intelliJ or Eclipse for production. Development may require other services such as Maven and Spring Boot, as well as a database management system. Choice of OS for development would be a matter of personal preference. | Java would be an ideal programming language, as Java can run on all platforms that support the JVM, so it would require just one team that is able to deploy the application via web-based environments. The team could select a single IDE such as intelliJ or Eclipse for production. Development may require other services such as Maven and Spring Boot, as well as a database management system. Choice of OS for development would be a matter of personal preference. | iOS devices do not support the JVM, so additional support and development costs would be necessary for Apple devices. This would require broader knowledge and/or more specialized teams for development. Required languages would include Java/Kotlin for Android environments (with Kotlin being the preferred) and Swift for iOS. |

## Recommendations

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

1. **Operating Platform**: A Linux-based operating platform would provide the best performance and reliability for maximum uptime. Linux servers are capable of servicing various web browsers across varying operating systems, thereby satisfying the need for expandability to other computing environments. Linux based systems are also generally less expensive than other options, with a solid track record of being secure and reliable. A web host would that provides a Linux-based server would be ideal for this application.
2. **Operating Systems Architectures**: Linux comes in numerous distributions based on the UNIX operating system that may be chosen for their strengths. For example, CentOS is widely regarded as a solid choice for web hosting. Linux uses a layered approach including the hardware layer, which consists of the physical components of the system. Above that is the kernel, which is responsible for managing the hardware of the system, such as memory allocation and file management. The application layer is on top of that, which communicates with the kernel and houses user-facing applications, such as a web browser that will be needed to play the game.
3. **Storage Management**: Storage should be provided by the chosen web hosting service. A good host will offer options for scalability. For instance, if the number of images used in the game is to increase, or if the user base has a large influx, more storage would be necessary to manage the data used by the application. For this application, I would recommend using a storage medium with contiguous allocation for the image storage, as these files are not likely to change and have a consistent size. For the storage of game history, players, and teams, I would recommend a paging system. This allows only the currently needed information to be loaded, such as the list of active players in a game.
4. **Memory Management**: Linux uses a virtual memory management scheme to help make the most of the physical memory available. What this means is that the details of the physical memory allocation and usage are abstracted to make memory management more accessible to an application. Additionally, virtual memory creates the appearance of a larger memory space than is available from physical memory. This is achieved by paging, which is the process of breaking up the data into blocks called pages. The system is then able to swap data that is not used frequently into a secondary storage medium. This frees up main memory for performing tasks that require more immediacy.
5. **Distributed Systems and Networks**: By using a distributed system and network, issues such as limited connectivity and outages may be minimized due to the nature of distributed systems. The application and its storage are not centralized to a single point of failure but rather distributed across varying physical and logical spaces to increase availability and uptime of the system and network. For this reason, potential web hosting services should be carefully evaluated for performance and reliability, as well as customer support.
6. **Security**: As security is a great concern, features of the Linux operating system must be used to protect the system and its users. Items such as user authentication are paramount to securing the system. Not only the end users (players of the game) must be authenticated, but system administrators as well. This is to prevent intentional or accidental harm to the system. By adhering to a set of clearly defined user roles, users of the system at every level will only have access to the parts of the system they need. Also, the database for housing user information must be secure to avoid leaking personal data about the players, such as their email addresses. Additionally, data being transmitted should be encrypted to ensure the security of user account information. Any decent web hosting service should provide at least modest security features, but advanced security options may come at a premium.