

Draw It or Lose It

# **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 | 11/09/2023 | Jacob C Wilson | * Added ability to have one or more teams * Added ability to assign multiple players to each team * Added unique game and team names to entities * Fixed issue where multiple gaming instances could be created in memory |

**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)

Currently, Creative Technical Solutions is attempting to develop a web-based version of their current Android app, Draw It or Lose It, which allows players to form teams to guess a thing, title, or phrase. The problem currently is remaking the game to be available online for multiple platforms to be able to play the game. Furthermore, the solution to this problem will have to address the differences of browser platforms that will deliver the web-based application in question. We will also need to address any operating system differences to make the application available for all systems. The solutions seem to be to create an environment using the appropriate technologies that are platform independent.

## Requirements

*Business Requirements:*

* *Host game with a content management system*
* *Use of SEO to help reach a broader audience*
* *Accessibility is needed for all content of the game*

*Technical Requirements:*

* Deliver a gaming environment experience that is easy for users to navigate
* *Game needs to be accessible to all available browser platforms*
* *Build game for all devices such as mobile devices and desktop devices*
* *Having a database to store things such as user accounts*

## [Design Constraints](#_2et92p0)

* Will need to be responsive for all available browsers
* Will need to be optimized thoroughly to decrease operational costs
* Will need to be streamlined for different resolution devices such as desktop, tablet, and mobile
* Needs to have server-side frameworks to hold game sessions and user interactions

## [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.

(I would like to know; this would mean stuff such as server-side communication and databases. Is this right?)

## [Domain Model](#_8h2ehzxfam4o)

***Entity:***

**Entity is a parent class for all the classes present except for GameService. This class serves as the base class for all “entities” created from these other classes. Essentially, Entity is an inherited class so that all other classes have unique identifiers and names. Entity is also used to eliminate replicated code throughout each class, since they all share common methods and attributes from the Entity class.**

***Game:***

**Game is a class that is instantiated to keep track of all the game’s assets, while keeping one instance of the class active in memory while the game exists. This class is composed of a list of teams to play in the current game’s instance. This class inherits from the Entity class to have a unique ID and name for its object.**

***Team:***

**Team is a class instantiated to track the players for the respective team. You can have multiple players in the players variable and add specific players to the team. This class inherits from the Entity class to have a unique ID and name for its object.**

***Player:***

**Player is a class instantiated to keep track of an individual player and have entity traits inherited from the Entity class.**

***GameService:***

**This class keeps a list of game sessions available to be played. This class adds game instances, gets game instances and creates ids for games. This game can also consist of multiple game objects. This class is a Singleton which means it is created to simply exist as one object only in memory for users to use to access current games available for use.**

**ProgramDriver:**

**"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 must 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** | MacOS’s characteristics align with providing a stable and secure environment for hosting web applications with solid security infrastructures that helps become compatible for multiple other devices. MacOS has a harder time being scalable for bigger deployments compared to other server-oriented distributors. | Much like MacOS, the characteristics provide a stable and secure environment for hosting web applications with more-than solid security infrastructures in place to help become compatible for multiple devices. Linux offers scalability, being able to grow vertically and horizontally. | Windows offers a user-friendly environment with robust server features. Some of the characteristics of Windows include user friendly graphical user interfaces to interact with. This, however, comes with resource intensive usage that causes harm to host a web application from this operating platform. | There are many user advantages such as portability and location services that would help aid in server connections. However, there are many security concerns along with processing power that would be needed to sustain hosting a web application using the operating platforms available for mobile devices. |
| **Client Side** | Developing the designs of the game for MacOS requires each Mac device’s design principles and guidelines to ensure the game looks its best online. Developing the best user interfacing and experiences are important to comply with the design tools available on Mac’s hardware. Mac also provides little to no expertise in understanding its systems as it is extremely user friendly. | Linux is extremely cost effective for supporting multiple types of clients, as this operating platform is completely free with no licensing fees which could help host web applications across the board. Timing may be a bit more challenging with Linux, as we would need to ensure the platform-specific optimizations are in place with the testing of specific platforms as well. The downside of Linux would be within the expertise of the operating system, as Linux is incredibly expensive, it would require understanding certain distribution systems such as Ubuntu and Debian. | Because Windows is mostly separate from other distributions of operating platforms, this makes it costly to develop multiple codebases for other platforms. By doing so, this would also take further time and create harder to reach goals for the Windows system development. Developers would need to be well educated in cross-platform development to ensure consistent functionality across all platforms. | This would be remarkable like Windows, in which it may be costly to create multiple codebases for different platforms. This would also increase the time it would take to reach goals for all developments for the web-based application. Developers would again need expertise in cross-platform development to achieve a streamlined user experience for players. |
| **Development Tools** | Developing the web-based application requires multiple tools and languages such as Swift, Objective-C, Xcode, and AppCode. Developing a web-based application, however, would require a hosting service such as Netlify to launch the application on the internet. | Developing the web-based application requires multiple tools and languages such as JavaScript (Node.js), HTML5 and CSS, Python, C++ or C#, Visual Studio Code, and Unity Editor. To deploy a web-based version on Linux, it would require a hosting service such as Netlify to launch the application on the internet. | Developing the web-based application requires multiple tools and languages such as C#, C++, Visual Basic .NET, F#, Java, Python, PowerShell, Visual Studio, and various other IDEs. To deploy a web-based version on Windows, it would require a hosting service such as Netlify to launch the application on the internet. | Developing the web-based application requires multiple tools and languages such as Swift, Objective-C, and Xcode for iOS. It would involve Java, Kotlin, and Android Studio for Android. There are many other ways to develop apps on Mobile devices. To deploy a web-based version on Mobile, it would require a hosting service such as Netlify to launch the application on the internet. However, there would need to be included in responsiveness for the application through mobile. |

## Recommendations

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

1. **Operating Platform**: I would recommend Linux for scalability and to allow for further cross-platform creation to be possible. By using Linux, we might be able to achieve greater security purposes for shared information and to assist in streamlining the user experience across multiple devices later.
2. **Operating Systems Architectures**: Linux has many different system architectures that allow for flexibility in choosing the right one to power the game and help it become scalable and compatible for multiple devices. The strength of Linux lies in its adaptability which enables it to run on an extensive range of hardware architectures that will allow it to become a commonality in diverse computing environments, from personal devices to data centers.
3. **Storage Management**: The Logical Volume Manager allows for flexible and powerful storage management. This type of storage allows for easily managed, resized, and migrated logical volumes.
4. **Memory Management**: Memory management techniques such as virtual memory, process address space, and caching and buffering to achieve the best memory management we can get for the game. Virtual memory allows the game to have a larger address space than the physical RAM, facilitating the storage and retrieval of game-related data. While Process address space is used to organize the software’s code, data structures, and dynamically allocated memory. Lastly, caching and buffering is used to load in and create saves for the game for future use if a user would like to come back to the game later.
5. **Distributed Systems and Networks**: This would involve designing a networked architecture that supports seamless interaction between clients and a central server. The goal is to enable real-time communication through tools such as WebSockets and APIs, implementing mechanisms for real-time data synchronization to ensure updates to the game state for each connected client, and a consistent user experience so that users can have secure information on the game along with consistent runtimes while they play the game.
6. **Security**: Linux offers various means of protection such as firewalls, file system security, user and group management, security patching, security auditing, kernel security features and many more. These features ensure that the Linux systems are fully operational, and that data will be protected to the best interests of the client and organization.