

The Gaming Room

# **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 | <mm/dd/yy> | <Your-Name> | <Brief description of changes in this revision> |

**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)

Creative Technology Solutions (CTS) is looking for a novel solution to expand their popular game   
“Draw it or Lose it”. The suggested web-based version aims to extend the games accessibility across multiple browsers while also enriching and enhancing the gaming experience. We will develop the game following the UML design laid out in this document.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

1. *Cross Platform Compatibility* – The application will need to run on various browsers, (Firefox, Crome, Safari, ect.). We will need to ensure that there is a consistent user experience across all browsers.
2. *Scalability* – The application should be able to handle scalable numbers, especially during peak gaming times.
3. *Unique Names* – Will need to implement a validation mechanism to prevent duplicate user and/or team names.
4. *Single Instance Limitation* – Only one instance of the game can exist in memory at a time so we should look at implementing a single instance limitation to manage resources and avoid memory leaks.
5. *User Accessibility* – The application should be accessible to users with disabilities so we should adhere to web accessibility standards.

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

**"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.**

**Domain Model Explained:**

The UML diagram above shows the Domain Model for the game application. The diagram shows the overview of the classes, their relationships, entities, and behaviors. Each class has the class name, class attributes, and class methods listed. Game, Team, and Player are all inherited from Entity. The ProgramDriver uses the SingletonTester Class. The use of private attributes and methods within the classes shows encapsulation. And association is shown through the relationships, like in Game to Team as a zero to many aggregations.

## [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** | **Advantages:** Unix based which is a stable OS. Increasing in popularity.  **Weaknesses**: Limited hardware options compared to Windows. macOS is not as common in server hosting as Linux and Windows are.  **Licensing costs:** Cost will be associated but generally less than Windows. | **Advantages:** Open Source allows for more flexibility in customization. Support for various web hosting frameworks. Lower resource cost compared to windows.  **Weaknesses:** Limited commercial support compared to Windows.  Gaming software may be difficult to run on Linux.  **Licensing Cost:** Linux is open source and does not generally incur licensing costs but there may be costs associated with distribution. | **Advantages:** Extensive support for software including gaming software. User friendly interface. Great commercial support.  **Weaknesses:** Licensing costs can be high. May require more resources compared to Linux. Security vulnerabilities may be more of a problem.  **Licensing costs:** Windows servers licenses will come with a cost depending on the number of users. | **Advantages:** Large user reach as many people play games on their mobile devices. App stores provide easy distribution. Built in security features.  **Weaknesses:** Mobile platforms are not generally designed for hosting web applications directly and backend services are hosted on separate servers which may incur extra costs.  **Licensing cost**: iOS development may require an annual fee for Apples Developer Program. Android development is open source but there may be a cost with distribution or some services. |
| **Client Side** | **Cost**: High  **Time**: High  **Expertise:** macOS development  **Requirements:** Develop a native application tailored to macOS.  **Considerations:** Use frameworks like React Native or Flutter for cross-platform compatibility. | **Cost**: Low to moderate.  **Time**: Moderate.  **Expertise**: Frontend Developer.  **Requirements**: Ensure compatibility with multiple browsers like Crome, Firefox, ect.  **Considerations**: Linux users often prefer Chrome or Firefox. | **Cost:** Moderate.  **Time**: High.  **Expertise:** Windows developer.  **Requirements:** Develop a native windows desktop application.  **Considerations:** Must adhere to windows design guidelines. | **Cost:** Moderate to high.  **Time:** High.  **Requirements:** Develop a native application for both Android and Apple.  **Considerations:** Adhere to platform specific design guidelines. |
| **Development Tools** | **Languages:** Swift is apple’s programming language. Can also use Java Script for cross platform applications.  **IDE’S and tools:** Xcode is the official intergraded development environment for macOS. AppCode is an alternative IDE to xcode that focuses on Swift development created by JetBrains. Github Desktop is a graphical user interface to facilitate version control and collaboration. | **Languages:** C is best for Linus as many core components of Linux OS are written in C. Kernel Level development.  **IDE’S and Tools:** Visual Studio Code is a versatile code editor with support for C. Make is a build automation tool that can simplify the process of building projects and is often used with C projects on Linux. | **Languages:** C# is a language developed by Microsoft and is integral to Windows development using the .NET framework. C# is also a commonly used language for building gamming applications.  **IDE’S and Tools:** Visual Studio is provided by Microsoft and supports C#. Git will help with version control and collaboration. WiX Toolset is used for creating Windows installer packages and is essential for creating installers for Window applications. | **Languages:** Swift is the primary language for iOS app development. Koltin is a modern language for android development.  **IDE’S and Tools:** Xcode is the official IDE for iOS development while Android Studio is the official IDE for android development. Firebase is an alternative that will work for both android and Ios, it provides a lot of tools like authentication, databases, and cloud functions. |

## 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**: Windows is the recommended operating system due to its vast support for games, familiarity, and global access.
2. **Operating Systems Architectures**:
   1. Compatibility: Different Windows versions may have variations in their architecture so we must consider backward and forward compatibility. Recommendation is to use the most up to date Windows for security and features.
   2. Windows has built in UI elements for a more familiar user interface.
   3. Windows provides an API for handling visuals and sounds.
   4. Windows has a robust security feature to ensure the integrity of Draw it or Lose It.
   5. Since Draw It or Lose It has multiplayer features, we can use windows networking features for communication between games.
   6. Easily integrate with windows file system for saving game progress or other related data.
   7. Since the game requires server-side components, we can utilize Windows built in server architecture.
3. **Storage Management**: The New Technology File System is the most used system for modern windows applications and is recommended. Key features include support for large files, advanced security features, built-in compression, and improved recovery.
4. **Memory Management**: Windows uses a virtual memory system. This system allows each process to have its own virtual address space, which is independent of physical RAM and enables the game to address more memory than available which allows for larger data and better multitasking. Furthermore, the game’s code data, and stack are organized in this address space to prevent interference from other processes.
5. **Distributed Systems and Networks**: Draw it or Lose it can use a distributed software architecture to enable communication between platforms. This will involve breaking the application into components that run on different devices and communicate with each other via network. Having a client-server architecture where the server handles requests from multiple clients and defining API’s for communication between the server and client. This will require a stable network, so an offline mode is recommended in case of network outages. This offline mode will allow users to continue some activities without a network connection and use local catching to store data.
6. **Security**: Ensuring the security of user information for Draw It or Lose It is vital and windows as a lot of built in security measures. We should utilize windows Transport Layer Security protocols for secure communication. Furthermore, we can use the windows operating system to enforce strong passwords. We should also implement a secure authentication mechanism to ensure that only authorized users are accessing authorized games. For this we can use windows built in active directory integration for user authentication. And lastly it is recommended to have two factor authentication where possible.