

<Name-of-Software-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 | <6/22/2025> | <Dalton Gollihue> | <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)

The Gaming Room is looking for a web type game called “Draw It or Lose It.” It is inspired by the 80s game show that is called “Win, Lose or Draw.” The crew that has been tasked with The Gaming Room is very unfamiliar with such a project, which is why CTS is getting involved.

## 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)

Code is in Java. The game can only be loaded in one instance in memory at any time. The Gaming Room is web based. The game will have one or more teams. Each team will have multiple players. The team and game names must be unique. Demands 4 rounds that are one minute each. Drawings render continuously for 30 seconds. Stock images from a fixed library. The team that guesses first gets 30 seconds, and the remaining teams all get 15 seconds to make their guess.

## [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)

We can see that the following classes are derived from the entity class: Game, Player and Team. These four classes (and Game Service) show encapsulation with private attributes and public methods that allow them to be accessed or modified. We can see that the classes don’t borrow a lot from each other. The diagram also shows us that there is a relationship between ProgramDriver and SingletonTester 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** | We need to utilize a mac server host.  Increased costs associated with apple hardware. | Linux based server host.  Linux has a bunch of variations. A lot of options.  There are fees associated with licensing.  Not burdened by hardware restrictions. | Windows based server host.  Windows is subject to frequent updates.  Subject to licensing fees.  Not burdened by hardware restrictions. | Mobile phones can be used to host the application.  Could be popular and require fast up scaling. |
| **Client Side** | Similar cost for development.  Need to make sure popular browsers are supported.  Can use this to run other desktop versions.  Not the most commonly used OS. Harder for most devs to use. | Similar cost for development.  Need to make sure popular browsers are supported.  Can use this to run other desktop versions.  Under the radar OS for a lot of devs. Harder to use. | Similar cost for development.  Need to make sure popular browsers are supported.  Can use this to run other desktop versions.  Very popular OS. Most devs can use well.  A larger variety of browsers. | Similar cost for development.  Need to make sure popular operating systems are supported.  Can use this to run other desktop versions.  Very popular versions available. Devs will be plentiful. |
| **Development Tools** | Compatible with web app languages.  Can use things such as: Xcode, Swift, and VS Code. | Compatible with web app languages.  Can use Swift and other tools, but isn’t the most compatible across the other operating systems. | Compatible with web app languages.  Probably the OS that allows for the most overlap for the other operating systems. | Development is probably possible, but too inefficient.  Imagine coding on a phone. (Yikes)  Not many tools. It would take a lot of gymnastics to be able to use this to create things for the other devices. |

## 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**: It has to be a desktop environment (for the sanity of the team). I believe Windows is the best tool for the job as it is so popular and has minimal restrictions. It also allows for the most broad coverage for the other versions of the game for Linux, Mac, and mobile devices. Additionally, we can outsource some cloud service to maximize efficiency.
2. **Operating Systems Architectures**: Windows is going to be our go to because of how it’s the king of accessibility.
3. **Storage Management**: In the early stages of the game we could store most or all things locally, but as the game increases in popularity, we could consider looking into cloud options if its required after launch.
4. **Memory Management**: Using normal technology like virtual memory and memory mapping.
5. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>
6. **Security**: All data sent between the players should be encrypted, and we will also be outsourcing the cloud service to a trusted encrypted service.