

Draw it or Lose it WB

# **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 | 3/30/2025 | David Kim | Information were added to provide a better understanding on what the requirements are for the project “Draw it or Lose it WB”, parts added are cover page, the document revision history, the executive summary, design constraints, system architecture view, domain model and recommendation. |

**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 Company wants to design and develop a web-based version of Draw it or Lose it. The games is similar to the 1980’s television game, “Win, Lose or Draw”, however instead of images being drawn it uses a library of stock drawings as clues for the teams to guess. There will be four rounds with multiple teams participating, each lasting one minute, images will be shown to the teams at a steady rate and are fully shown at the 30 second mark. Once time expires and the team on their turn do not answer the other teams have a chance to answer within a 15 second time limit.

## Requirements

## Must allow for 1 or more teams

* Each team will have multiple players
* Game and team names must be unique for users to check if a name is available
* Only 1 instance of the game can exist at one time
* Web-based
* Four 1-minute rounds
* Pictures used as clues, gradually filling at a steady rate until the 30 second mark
* Pictures represent a title, phrase, saying, etc

## [Design Constraints](#_2et92p0)

* The Web-Based game must run on multiple platforms
* Only one instance of the game can exist at anytime
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name

## [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 diagram shows a clear and simple design for the game system. The ProgramDriver class starts the application and uses the SingletonTester to make sure the GameService works as a singleton, meaning there is only one instance of it running. The GameService manages all the game data, keeping everything organized and consistent. At the core is the Entity class, which holds common details like an ID and a name. The Game, Team, and Player classes all use these basic features from Entity, which makes the code simpler and easier to maintain. The Game can have many Teams, and each Team can have several Players. This setup matches the requirement that the game supports one or more teams and multiple players per team. The design also helps ensure that game and team names are unique, which is important for keeping the game organized. The key object-oriented ideas such as inheritance (sharing common features via Entity), encapsulation (each class taking care of its own data), and the singleton pattern (making sure only one GameService exists). These principles work together to create an efficient and easy-to-understand system that meets the client's needs.

**"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** | User-friendly environment, relatively straightforward to set up for web hosting, however Macs can be more expensive than other platforms and has less support or history compared to Linux and windows. | great for being flexible, secure, and performance under heavy loads, however the barrier to entry is more intimidating for first time users with a higher learning curve. | most accessible with compared to others, however the cost to operate the server is higher | Mobile devices can run local servers, however with their limited processing power or battery constrains make it unstable for hosting. Mobile devices are best as a client device. |
| **Client Side** | Expensive for  users  Moderate time and  expertise required.  Accurate skills and  needed to navigate  OS | Low cost, however a very niche user base | Very accessible and ease of use but higher cost | Most accessible and flexible, but lower  hardware specs can  make development  and user usage  difficult if managing  larger applications. |
| **Development Tools** | HTML,  CSS and JavaScript. IDEs include Eclipse, Intellij | HTML,  CSS and JavaScript. IDEs include Eclipse | HTML,  CSS and JavaScript.  IDEs include Eclipse, Intellij, Visual Studio | HTML, CSS,  Javascript |

## 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 provides the most user-friendly system and has a long history and running list of development tools
2. **Operating Systems Architectures**: Windows' graphical operating system lets you easily access and switch between files, software, games, videos, coding tools, internet features, and many other functions without interruption.
3. **Storage Management**: Windows 11 uses storage sense to provide the amount of storage that is being in used.
4. **Memory Management**: Windows is best in this case as it excels in memory management for saves files and photos, particularly the images from Draw it or Lose it.
5. **Distributed Systems and Networks**: Networked multiplayer systems, like online games, usually use a shared database that lets players from different locations interact with each other over the network.
6. **Security**: Windows have a built-in security protection software.