

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 | 11/12/2023 | Patrick Marshall | <Brief description of changes in this revision> |
| 1.1 | 11/26/2023 | Patrick Marshall | Updated evaluation and requirements |
|  |  |  |  |

**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 client is looking to make a game like the TV Show game *Win, Lose or Draw*. The game involves teams of multiple players, timers, images, and rounds. The client is seeking to make this a web-based game.

## Requirements

* A game will have the ability to have one or more teams involved.
* Game must be distributed cross-system (PC, Xbox, PS, etc.)
* Each team will have multiple players assigned.
* Game and team names must be unique.
* Only one instance of a game can exist in memory.

## [Design Constraints](#_2et92p0)

1. The game will be web-based:

The company wishes for this to be a web-based game, which will limit what libraries to be used, and helps narrow down what languages to use as some are better suited for this task than others.

1. The game will be written in Java:

[I’d normally actually recommend this be done in JavaScript, but we’ll assume this company is more used to using Java, and because that’s what we’re using in class]

This is the language most at our company are experienced in, and its flexibility towards what hardware it can be developed for, Java gives us the most wiggle room for options.

## [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 generic Entity class contains the id and name variables that all its children will make extensive use of, and the mutators for each. Each method has a hierarchy of what contains each from Game -> Team -> Player.

The ProgramDriver class has access to this hierarchy through the GameService singleton instance, which can itself make games, then can add more entities further down the hierarchy through those game variables.

**"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** | License purchased with hardware. Can host web server. | Highly customizable, price depends on distribution and might be considerable for The Gaming Room. Can host web server. | License is paid separately from hardware. Can host web server. | Not fit to host server-side logic. |
| **Client Side** | Limited (but not small) and baked-in customer base. Can likely transfer current mobile UI with little change if desired. | Little to no main-stream web-based use cases. Can likely transfer mobile UI with little change. | Largest general pool of users. Can transfer mobile UI with little change. | Large pool of users, though browser-based use is limited and unintuitive. Will require specific consideration when developing for each phone company’s app store. |
| **Development Tools** | Generally easier to get set up for development environments, using IDEs like Visual Studio or VSCode. Integrations with Linux for things like Docker are more smoothly done with MacOS than windows. | High bar for entry but many dev environment tools are housed here like Docker. | Most environments will work here (Eclipse, VScode, etc), but bugs and conflicts might arise when using multiple tools in concert or interacting with Linux virtual machines. | Requires specific tools that will not transfer easily to other OS’. |

## 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**: I would recommend Windows as the platform for the server side for its general purpose ease of use and wide range of development tools, with JSON messages bridging any games between different platforms the clients will be able to play the game on.
2. **Operating Systems Architectures**: Windows, like most OS’, has layers separating the user from the hardware. The user interacts with applications, which interact with the OS, which communicates with the computer, in this case through a CLI.
3. **Storage Management**: The executable and images for the game will be saved onto the user’s device hard-drive/internal storage to be later called on for use. While the game is in play, match data such as what images to call, who is in the match (team names/members), etc. can all be saved to an external server instead.
4. **Memory Management**: The game will use the RAM of the user’s device to call the stored images and game data to be rendered and manipulated; tracking time, teams, and other data that changes often.
5. **Distributed Systems and Networks**: To allow for communication between our servers and devices of varying OS’, this communication can be handled through JSON requests to the web for required requests. These will be used for communication between the client and server for information such as teams, players, and making sure all players are displaying the same information through comparation.
6. **Security**:

* If players are to have accounts, the standard username, password, email verification and two-factor authentication process will be perfect.
* For creating matches or private lobbies for games, simple passwords will be enough, and could be generated by the game locally for the host ala JackBox.
* Communication between client and server can be handled by encrypted get/post requests through JSON, and data stored on The Gaming Room’s own servers.