

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/13/2022 | Kyle Mosely | Complete Domain Model, Design Constraints and Executive Summary |

**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 staff requests our team to develop a web-based implementation of the game ‘Draw It or Lose It.’ We will design a game that will:

* Have multiple characters on a team
* Allow exclusively unique team names; A system for players to check if a name is in use
* Allow a single instance of the game existing in system memory at any given state

## [Design Constraints](#_2et92p0)

* Javascript or python must be used, to enable full functionality in any web-based environment.
* Game requirements state it must be written for multiple teams grouped by multiple players in each team, via player and team objects
* The game must check team name uniqueness, implementation via singleton pattern within game and team creation UI.
* Game must permit only a single instance of the game to exist in memory, implementation via unique id for game instance, player instance, and/or team instance through iterator patterns.

## [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)

Player, Team, and Game classes have each an inherited Entity superclass, providing operations and attributes that are constrained in Entity that is to be written only once, then used by all classes inheriting it. The Player, Team, Game, GameService classes all associate together. They utilize a zero-to-many association (using many instances of the other as needed). The ProgramDriver is used to drive the package, inheriting the singleTester class during the process.

**"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** | High Cost.  Superior UI.  Pliable workstation; accessible and simple server configuration. | Cost effective.  Challenging Navigation.  Command shell for configuration and accessibility | High Cost.  User friendly.  Large options of available software.  Command Prompt. | Low Cost.  Specifications vary depending on OS (Android vs. IOS vs. WP).  Trouble with cross compatibility. |
| **Client Side** | Average time for supporting mac-users.  Some backhand knowledge needed.  High Cost. | Higher time allocation for linux-users.  Low level knowledge needed.  Low Cost. | Little time needed for windows-users.  Some backhand knowledge needed.  Low Cost | Higher time to support multiple OS platforms.  Higher backhand knowledge needed.  Average cost. |
| **Development Tools** | Many languages and IDEs are deployed on Mac.  Several different tools are supported, with differing complexities. | More ordinary languages are deployed on Linux. Many different IDE’s and tools can be used, many need more backhand knowledge | Most languages and IDE’s deployable on Windows.  Several different tools are supported, with differing complexity. | Many languages are deployable on Mobile. Simple development tools are available. |

## 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 recommended for the ‘Draw It or Lose It’ Expansion. Using the Windows OS provides ease of use and accessibility to several software packages for users and developers. Windows offers support for engineering solutions that might be more challenging to implement on other OS’s.
2. **Operating Systems Architectures**: Although Windows does require overhead cost with the purchase of a license, it provides an easy UI design. With several software packages accessible, Windows will offer high flexibility for developers. Windows OS will provide superior development experience with several different programming languages available and access to command prompt shell.
3. **Storage Management**: Windows OS has a simpler config for storage, while having an internal memory manager. Windows OS gives access to cloud server storage, providing developers any required storage space.
4. **Memory Management**: With a memory manager offering, including physical and virtual space, allowing for up to 4 gb of memory and running applications smoothly.
5. **Distributed Systems and Networks**: Distributed systems and networks have some common challenges. Queuing repercussions, routing and congestion and some network and distributed systems issues that will be faced during development. Despite this, the systems in place offer easy communication and coordination structures. Independent failed components, an absent global clock, multi-threading computations may also lead to lagging performance and connection issues for players. These issues will need to be addressed during the Windows OS development cycle.

1. **Security**: Windows OS allows clients with user account settings, helping secure data going and out. This security ensures authorized OS platform modifications cannot be made without administrative acceptance. Windows also offers anti-spyware systems in the box, keeping vulnerabilities such as RDP stream cipher that calls encrypted protocols to get information like passwords, credit cards, etc. Windows also has DirectAccess for remote sites. This provides authentication and encrypted ESP when connecting users to business networks.