

Draw It or Lose It

# **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 | 11/18/22 | Aaron Adeyemi | Added Executive summary and identified design constraints. |

**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 would like to develop a web-based, multi-platform version of their android game, *Draw It or Lose It*. The game will operate with multiple games, with multiple teams featuring multiple players. All of those elements will need to have unique identities to keep them distinct and prevent unwanted interactions.

## [Design Constraints](#_2et92p0)

The game needs to be able to function on various platforms.

Games will need to be able to support multiple players, in multiple teams in a game across platforms.

Game and team names must be unique and the interface should be able to notify players that a given name is in use.

Only one instance of a game can exist at a time. Every game, team and player must have a unique name/id.

## [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 ProgramDriver contains the main() of the code. ProgramDriver uses SingletonTester to test whether or not an instance of GameService already exists. The Entity class serves as the parent class for the Game, Team, and Player classes, and thus all three classes inherit the Entity class’s attributes. The three subclasses are hierarchical, however. A member of the Game class can have a Team but not vice versa. A member of the Team class can have a Player, but not vice versa. And although the GameService class is not a child of the Entity parent class, it also fits into the hierarchy, as the GameService class can have a Game, but not vice versa. Every member of this hierarchy can only have one unique member in its corresponding class however. So, a GameService can only have one unique instance of a Game, and a Game can only have one unique instance of a Team, and a Team can only have one unique instance of a Player.

**"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** | Apple does have support for servers, but they are very expensive. | Linux is able to support servers at no cost. | Windows is able to support servers at a reasonable cost. | Mobile devices can be used to host servers but are not recommended. |
| **Client Side** | To develop on Mac, you must have a MacBook, which are quite expensive. But besides that, developing for Mac is rather simple and the SDK is accessible, but only if you have a MacBook. | Developing for Linux shouldn’t incur any additional costs. The only considerations would be for time and the familiarity of the developers with working with Linux. | Developing for Windows shouldn’t incur any additional costs. The only considerations would be for time and the familiarity of the developers with working with Windows. | Developing for mobile devices shouldn’t incur any significant additional costs. However, the accessibility that comes with developing for mobile devices will require developers familiar with that work. |
| **Development Tools** | Development on Mac will make use of Swift. | Coding in Linux could be done in either C, C++ or Python. | While C/C++ is the most common code used in Windows applications, virtually any language can be used to work in Windows. With that, the developer’s preferences should be considered. Visual Studio is, however, very commonly used and is also free. | Mobile devices, like Windows, are largely quite flexible on which language is used, and as such, developer preferences should be considered. However, Java is the most common language used for development for mobile apps. |

## 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 recommend Windows as an operating platform.
2. **Operating Systems Architectures**: Windows offers a largely flexible system that most clients are already familiar with. Windows is also very developer friendly and doesn’t require as much active maintenance in the long term, in comparison to Linux. Windows is also highly compatible with most software and hardware. However, due to the nature of Windows, it is more liable to security breaches, and such will have to be taken into account during development.
3. **Storage Management**: I recommend Microsoft’s Azure Stack for cloud storage. Since it is cheap, easy to set up and manage in the long term.
4. **Memory Management**: Since the number of users at any given time for Draw It or Lose It is variable, memory should be allocated as is necessary during highly active times and restricted during less busy periods. Having a real-time monitor that can allocate and restrict memory would be recommended.
5. **Distributed Systems and Networks**: Since the system will be using cloud storage via Azure Stack, any outages will not impact the data being shared between systems and connectivity to the server will be remote. The servers and networks containing the games will be independent of the users.
6. **Security**: Having a unique marker to identify the nature of a user should provide user security in that no user without proper authorization will have access to information that is restricted to them. Means such as Firewalls, data-encryption and two factor authentication can also be implemented to prevent external attacks.