

Draw It or Lose It Web Version

# **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)

[**Design Constraints 3**](#_Toc115077322)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 9/16/2023 | Aidan Gorospe | Added Executive Summary, Requirements, Design Constraints, System Architecture View, Domain Model, Evaluation, and Recommendations. |
| 1.5 | 9/29/2023 | Aidan Gorospe | Added more information to Evaluation |
| 1.8 | 10/13/2023 | Aidan Gorospe | Added more information to Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

Game Development company The Gaming Room is looking for assistance in putting their current game, Draw It or Lose It, on web-based platforms to branch out their business. They hired CTS to help them start the process of making a software design document and start the initial development process.

## [Design Constraints](#_2et92p0)

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.
* The web-based version of the game must be able to work on iOS, Android, and web-based browsers.

## [Domain Model](#_8h2ehzxfam4o)

The Game, Team, and Player Class are all child classes of the Entity class. The GameService class makes sure there’s only one instance of a Game running. The game class counts and identifies teams, and teams count and identify the players. The ProgramDriver Class has the main method, and is directly associated with the SingletonTester class.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac could be good as a server; however, the requirements of Mac computers and licensing make it a costly decision. | Linux is free and is well-versed when it comes to hosting web-based applications. | Windows server running capability is good, but the cost of licensing is pricey. | Most mobile OS are not strong enough to handle the strain of hosting a server. |
| **Client Side** | Coding would have to be done using swift. The previous issue of cost is involved with the fact you need mac computers to develop for Mac. | Python would be the ideal language to code for this OS, but the cost of using an IDE like IntelliJ could bring up the cost. | Windows is a good choice due to the .net framework being effective both security-wise and capability-wise | Developers that are accustomed to making apps are required, as things such as UI and Display are specific to the mobile OS using it. |
| **Development Tools** | Mac Computers come equipped with iCode and Swift, with access to other IDEs like Appium and ClickUp. | Python is automatically installed with most versions of Linux, and IntelliJ can be used as an IDE. | Visual Studio is the standard when it comes to coding Windows Application and uses many different languages. C++ or C# is the standard, however. | There is not much to do when it comes to developing an app with a phone, as you mainly have Android Studio. |

## Recommendations

1. **Operating Platform**: Using Linux to host the server would be ideal.
2. **Operating Systems Architectures**: The stability of the Linux kernel is hard to ignore, and I think it would be a good pick.
3. **Storage Management**: The options are either HDD or SSD for storage. For the sake of user experience, I would recommend using an SSD as it is faster accessing assets and loading things such as pictures to the browser.
4. **Memory Management**: As the server is running on Linux, I would also suggest the memory management be run on the same OS. Having both the memory management and the server contain the same OS would make compatibility issues less of a factor.
5. **Distributed Systems and Networks**: Being on the cloud, there is not much to do in terms of specifics or networks, if the current server is acting up, you can move to another server, the only thing you don’t have hands-on control is with the users themselves.
6. **Security**: By using a role-based security system there can be separation between the different classes and the user must work significantly harder to snoop around and breach security.