

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 | 11/14/2021 | Jordan Wentworth | Software Design Doc. |
| 2.0 | 11/27/2021 | Jordan Wentworth | Evaluation |
| 3.0 | 12/11/2021 | Jordan  Wentworth | Recommendations |

**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)

Draw it or Lose it is a web-based game that is currently only on Android. The game is loosely based off the 1980s game Win, Lose or Draw. The game includes four, one minute rounds, where the drawings last 30 seconds into the game. If the team does not guess the puzzle, the other team has one guess to solve the puzzle. They have 15 seconds to guess, or they do not steal the puzzle. The software requirements for the game are incorporating one or more teams and each team will have multiple players. The game and team name must be unique and only one instance of the game can exist at a given time.

## [Design Constraints](#_2et92p0)

As a web-based game, the game will need to operate on different operating systems. Windows, Mac, Linux and Android operating systems all use code different; therefore, every operating system has to be unique. This will require us to have multiple teams working on the project to be able to incorporate different OS. In addition, we will need multiple testing teams to make sure each OS is working properly and cross platform play is working.

## [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)

First off, the ProgramDriver has the main in the program. Then the main calls on the SingletonTester. Secondly The Game, Team and Player class all inherit from the Entity class, which will allow the three classes to inherit the attributes and code from the Entity class. Next, the GameService class extends 0 to many to the Game class. Thus, many games are played in the GameService. Then, the Game class extends to the Team class. This is where the name of the game and team will be checked for other instances. The Team classes now extends to the Player class, where you add your name to the method.

**"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** | Mac’s have easy accessibility and server configuration.  Mac’s also have flexible terminal commands to configure the server and access/make changes. | Cost friendly and, like mac’s, they have easy server configuration and access. | Out of all OS, windows have the most available software.  However, they are prone to viruses. | They are gaining more popularity, but it is better for the server to be immobile. |
| **Client Side** | They need moderate expertise and time. They are more on the expensive side. | They need maximum expertise and time. However, there are lower costs. | They need minimum expertise and time. They are more on the expensive side. | They are more difficult to implement than others, but provide flexibility. |
| **Development Tools** | When running languages we can run more option like HTML/CSS/ JavaScript while supporting libraries to support the frontend. IDE’s can be javascript, python, php. Development tools are Eclipse, visual studio and github. | When running languages we can run more option like HTML/CSS/ JavaScript while supporting libraries to support the frontend. IDE’s can be javascript, python, php. Development tools are Eclipse, visual studio and github. | When running languages we can run more option like HTML/CSS/ JavaScript while supporting libraries to support the frontend. IDE’s can be C++, python, html. Development tools are Eclipse, visual studio and github. | When running languages we can run more option like HTML/CSS/ JavaScript while supporting libraries to support the frontend. IDE’s can be html, C++, php and python. Development tools are Eclipse, visual studio and github. |

## 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 the Gaming Room to use Windows because of its flexibility. It comes with the most software available then the rest of the other operating systems. In addition, it takes little expertise to use windows. Programming languages and different IDE’s are abundant with windows.
2. **Operating Systems Architectures**: Windows architecture is great in the way you can allocate memory to where you need it. You can also control the processor in where and how much it is being used.
3. **Storage Management**: Windows, unlike other operating systems, allow you to fully customize your storage. You can change where you store your files. Managing files are easy on windows as well.
4. **Memory Management**: Like we have talked about before, the Gaming Room will need to have quick access to the data they have in their game. Windows uses RAM, which is fully customizable to add more RAM, to quickly access their data.
5. **Distributed Systems and Networks**: Networking games have a database that can link players together. This is done by a application program that can be incorporated in the game.
6. **Security**: Windows comes with basic security features like passwords. There are other added features like antivirus, firewall and malware protection. Other third-party security features can be bought and incorporated with windows.