

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 | 05/19/23 | Joshua Sellers | Initial Draft |
| 2.0 | 06/03/23 | Joshua Sellers | Evaluation |
| 3.0 | 06/13/23 | Joshua Sellers | Recomendations |

**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*** wishes to expand their web-based game ***Draw it or Lose it*** to run on multiple platforms. Currently the game is available on Android app only and needs assistance setting up the environment to achieve that goal. The hardware, software, and specified OS needs to be compatible for the desired platforms to mimic their already existing game to multiple platforms.

## Requirements

* Needs to model after the already existing game.
* Has to function for multiple OS.
* Software requirements needs to be addressed.

## [Design Constraints](#_2et92p0)

* Needs to model after the already existing game.
* Has to function for multiple OS.
* Only one instance of a game exists at a time.
* A game will have the ability to have more than one team at a time.
* A team can have more than one player on the team at a time.
* Team names and player names must be unique and are checked by the system to ensure no one else has this name.

## [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)

There are 7 classes in the **com.gamingroom** application. Some work together while others test coding methods that will assist the programs development. **GameServices** will assist the program and the classes **Game, Team,** and **Player** will contain objects for the organization for players to be assigned teams and teams will be assigned to a game. The parent class **Entity** will contain attributes and methods that all these classes can use for their process. The classes **Game**, **Team**, and **Player** inherit the attributes and methods from the class **Entity**. This prevents repeated code and these sub-classes inherit the super class Entity’s attributes and methods for their own use. **GameService, Game,** **Team**, and **Player** are all linked by association with each other. As seen through the connections, a 0 to all is stated between the relations. Meaning that either the classes use nothing between each other or a lot of things between each other depending on the circumstance the application calls for. **ProgramDriver** uses **SingletonTester** with a direct association for testing the singleton design method.

Polymorphism – Polymorphism is generally defined as “Appearing in multiple forms.” This means that different types of objects can be accessed through a common interface. For example, inside the **Game** class, the object **teams** is created inside the **Game** class and can be accessed through the Super class **Entity**.

Encapsulation – Encapsulation protects an Objects data components from external entities. Good practice in programing uses public and private states of values, methods, classes, and objects **GameServices** has several private values (**nextGameId, nextPlayerId,** and **nextTeamId**) and they can only be accessed and altered through the proper coding standards of their setters and getters.

Inheritance – Inheritance refers to the extention of other classes. As seen in the diagram, The classes are extentions of the previous classes. The children **Game, Team,** and **Player** inherit the attributes and methods from the parent class **Entity**.

Portability – Portability is the modularity of the program. Easy naming conventions and appropriate assignemnts to individual classes can assist in the design of a program. For example, if **Player** had a coding issue and an error populates, we would know that it was in the **Player** class and not in another class. If we programed the entire program on one class, narrowing down the issue would be difficult. This is the benefit of portability.

**"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** | Characteristics -  Flexible terminal commands for configuration of the server.  Advantages - It is upgradeable, it has great security, a lot of support is available, it has various options for different web hosting requirements.  Disadvantages - It is not preferred for web hosting services. | Characteristic- Best security among others. Most preferred.  Advantages - it is the most preferred choice for web hosting services. Security risks are caught before they become an issue.  Disadvantage- Difficult to find applications to support web hosting. | Characteristic-It is more popular than other platforms.  Advantages-, More widely known in usability. less loading time.  Disadvantage-High resource requirements and more virus susceptibility. | Characteristics -Most everyone has a mobile device.  Advantages- Most people have a mobile device (wider audience). cost-effective.  Disadvantages- It is highly selective to various smart mobile devices Poor security. |
| **Client Side** | Moderate knowledge of the OS and time to learn. Cost is same amount as windows. What is required of the application development process to ensure the application is compatible with all web browser platforms and mobile devices. | Maximum knowledge of the OS and time to learn. Minimum cost. What is required of the application development process to ensure the application is compatible with all web browser platforms and mobile devices. | Minimum knowledge of the OS and time to learn. Cost is same amount as Mac. What is required of the application development process to ensure the application is compatible with all web browser platforms and mobile devices. | Flexible for clients and developers to see updates required. However, it is slightly more difficult to implement than the other listed devices. |
| **Development Tools** | Swift is the more popular option on Mac. Macs are capable of running all language integrating tools like notepad++. The languages we can use vary from HTML/CSS/JavaScript while supporting libraries to support the frontend and general-purpose languages like Python, Java, PHP, and Ruby. | Linux can work with integration tools like visual studio, eclipse, along with notepad++. Linux can run all languages. The languages we can use vary from HTML/CSS/JavaScript while supporting libraries to support the frontend and general-purpose languages like Python, Java, PHP, and Ruby. | It is easier to use, but can run the same as Linux. So visual studio, eclipse to name a few of the many IDE’s. The languages we can use vary from HTML/CSS/JavaScript while supporting libraries to support the frontend and general-purpose languages like Python, Java, PHP, and Ruby. | You can create a lot of apps using android and swift. Both those languages and software can be run on all three OS listed before. The languages we can use vary from HTML/CSS/JavaScript while supporting libraries to support the frontend and general-purpose languages like Python, Java, PHP, and Ruby. |

## 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**: The Gaming Room should use Windows Devices for projects. Windows is the most familiar OS among users and developers. It would be more cost efficient for experts to hire for Windows OS and theirs more software in getting projects going for Windows. Also, more IDE can be used in Windows platforms.
2. **Operating Systems Architectures**: Windows has a service that uses all Windows applications that can use a Graphical User Interface (GUI) to assess systems resources. These applications refer to web services, messaging, and Graphics with multimedia. They can be accessed by the user account or the server itself.
3. **Storage Management**: Cloud can be utilized for storage if necessary. As covered before, Cloud storage offers a lot of benefits for only reserving the necessary amount of storage needed for the application. You can also use a Windows 10 feature called Storage Sense, which allows management of files and storage to consolidate data in easy-to-find locations as long as it is managed properly.
4. **Memory Management**: The creation of the game will need to focus on having a database or library to access the pictures. With proper memory management, people on the web-based game can retrieve these pictures at faster rates. The library needed to access the “clues” given to players needs a database. Recommended that these pictures remain in a secure folder so that way it cannot be altered accidentally.
5. **Distributed Systems and Networks**: The game will be web-based only running through multiple platforms. Based on research, I found Unity to be a popular choice. It is commonly used for cross-platform gaming and offers web platform gaming. Since Unity is also one of the most popular development engines, it offers plenty of useful materials, guides, and community support.
6. **Security**: Windows is commonly a victim of security threats among all OS. I would recommend another source for security that is properly equipped. Also certain languages like C++ that are commonly used for game development are also more prone to attacks. I’d install SSL certificates, firewall protection, effective passwords, set up access limitations, upgrade software regularly, and configure for file backups just in case.