

The Gaming Room

# **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 | 05/26/24 | Javier Vazquez | Project 1 submission |
| 2.0 | 06/09/24 | Javier Vazquez | Project 2 submission |
| 3.0 | 06/23/24 | Javier Vazquez | Project 3 submission |

**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 expand “Draw It or Lose It”, a popular Android application, to a wider audience by making a web-based game application. The Gaming Room must be aware that the application will be rewritten in a new programming language in order to also accommodate a different operating system out of the Android operating system. Also, design changes must be made to include features optimal for a full-screen web-based application.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

The Android game application must also be converted to a web-based game application. With that in mind, we need to build a compatible application for the Windows OS. Also, web-based applications must be written in JavaScript. Currently, the most popular JavaScript framework is React and would be the most appropriate for this application. Given that the identifiers must be unique for each instance of game, team, and player, we must implement the singleton design pattern. The iterator pattern will also be used to verify instance uniqueness. Collaborative teams must also be created between The Gaming Room and CTS to explain the functionality of the Android game design. The Gaming Room must also provide any licenses needed and access to private software components. Although collaboration will be instituted, CTS will be responsible for the creation of the web-based application.

## [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)

An Entity is a parent class to the Game, Team, and Player classes. This is an example of inheritance, as the child classes inherit the attributes and methods of the parent class. Also, since Game, Team, and Player instances are also instances of Entity, this shows an example of polymorphism. Encapsulation is represented by the + and – symbols before the attribute types and method names. These symbols represent whether an outside class can access that attribute or method. A – symbol represents a private field or method and only the class itself can obtain access to it. There are also several zero-to-many relationships between classes. This means that in several instances a particular class can have zero related instances of another class. This does not, however, work in the opposite direction.

**"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** | Great when using Mac clients and allows for easy creation of features. However, hard to scale on an enterprise level. A game like Draw It or Lose It does not need enterprise scaling and would be a great choice when targeting Mac users. Most people, on the other hand, use Windows and would be best to target them first. | Offers free licensing and major pro if cost saving is the goal of the Gaming Room. Has a lower risk of security errors. However, much more of a complex system and less support. | Great support and most users would be able to run the game with no issues. Also, very beginner friendly and optimal for this type of game. However, much more costly as each user would need a license. | Not optimal for web servers. Performance and speed cannot compete with other servers. |
| **Client Side** | Not as user friendly when it comes to games when compared to Windows. Hard to cross platform with other systems. | Linux clients can be installed and functionally run on other systems. Requires software knowledge and difficult for average user of the game when cross platforming. | Compatibility is not an issue when cross platforming. Will work on other systems with low requirements. Very user friendly and optimal for this game. | Can be converted, but not optimal. Full screen views are not as user friendly. |
| **Development Tools** | For Mac development, Swift is currently the most popular. Visual Studio Code with Swift extension and Xcode are popular IDE’s for Swift development. Need for specialized Swift developers might increase cost. | The C and C++ programming languages would be ideal for Linux development. More availability of these programmers are more readily available as they are taught in most computer science curriculums. Popular IDE’s would be Eclipse, CodeLite, and CLion. | C# would be ideal for creating applications for Windows as the ASP.net framework would allow for full stack web applications as in Draw It or Lose It. Eclipse, CodeLite and CLion are popular IDE’s for C# development. | Swift with the Visual Studios code or Xcode IDE’s would be ideal for iOS systems. Kotlin with IntelliJ IDEA and Eclipse IDE’s would be ideal for Android development. These languages tend to require more experienced and specialized programmers at a higher cost. |

## 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**:

For biggest reach, Windows is the recommended Operating System to build the game. It is beginner friendly and compatible with most users.

1. **Operating Systems Architectures**:

The two main components of the Windows OS architecture are kernel mode and user mode. Operations in user mode are prohibited at the lower level on the kernel.

1. **Storage Management**:

For storage management, a file system is recommended that uses contiguous allocation. Since the number of picture files and the approximate file size is known for each, contiguous allocation fits best.

1. **Memory Management**:

For memory management, we can use the Windows memory system. For the game, only the copy of the current game’s image file is necessary. Copies of all the other images are inefficient and unnecessary.

1. **Distributed Systems and Networks**:

A distributed system on the cloud would work well for a multi-player game such as the project we are working on. Given that the Windows OS is recommended, Azure would work best and be the most compatible cloud network.

1. **Security**:

To protect user information, a multi-factor authentication is recommended. A strong password combined with another form of authentication will aid in protecting user data. Affiliating with a VPN service will add greater protection and increase revenue.