

<Name-of-Software-Application>

# **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 | <01/22/23> | Louis Constante | Updated rubric items |

**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 client, Creative Technology Solutions, wants to take their existing mobile game and create a web-based version of the same game. CTS needs to set up a web-based gaming environment using an operating system that best suits their needs.

## Requirements

The game, Draw It or Lose It, needs to have the ability to have one or more teams involved. Each team needs to be able to have multiple players assigned to it. Game and Team names must be unique. Only one instance of the game can exist in memory at once.

## [Design Constraints](#_2et92p0)

Constraint #1 – Operating System:

The game currently exists only on Android OS. The existing game data from Android OS should ideally be shared with the web-based games. The chosen operating system should be one that integrates well with Android OS.

Constraint #2 – Libraries:

The game needs to be able to access large libraries of stock images to be used in play.

Constraint #3 – Intellectual Property:

The game is based on a television show that aired in the 80’s and is similar to other games on the market. CTS should ensure that they will not infringe on any other copyrights.

## [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 Class contains the one and only main method and uses the SingletonTester class to test if the singleton implementation is working correctly. The GameService class is the Singleton Class that holds information about how to uniquely identify newly created games, teams, and players. The Game, Team, and Player classes are each associated and inherit from the base Entity Class. Each Game can have 0 to any number of teams, likewise with each team to each player. Each Class in the diagram contains encapsulated data, along with methods for retrieving that data. The GameService class uses polymorphism to allow methods like getGame() to take multiple different parameters.

**"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** | MacOS does offer server hosting ability, however development can only be done using Apple hardware. The fees for hosting a server using Mac are also very high, relative to other options. | Linux does offer the ability to host servers There is no licensing fee for running a server on Linux, making it a relatively low cost option. | The licensing fees are high for hosting a server on windows. However, the quality and stability of windows servers are also high. | A mobile devices operating system would not be suitable for hosting a server. |
| **Client Side** | MacOS is built on Unix making it ideal for developing for various other platforms. Mac is also equipped with several useful development tools that allow for this kind of programming. | Linux can be hard to use for inexperienced users, so some level of expertise would be ideal for developers. The cost of developing on this platform is relatively low. | The cost of Windows hardware is cheaper than Mac. A moderate level of expertise would be required to make development on Windows smooth and secure. | On a mobile device, the game would be best suited as an app. It would be best to have developers who have expertise in developing mobile apps as it can be quite different from other platforms. |
| **Development Tools** | Swift will be the primary coding language when working on MacOS. Xcode is the IDE that would most likely be used. | C is one of the most useful languages when developing a program on Linux. Many of the most popular modern IDE’s are available on Linux, such as Visual Studio. | Windows supports many coding languages and like Linux, provides access to many high quality IDE’s. | A tool like Xamarin could be used to allow developers to create the game for all of the major mobile platforms. There are several useable languages that would be suitable such as Java, C++, or Rust. |

## 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**:

Hosting Draw It or Lose It on a Windows server will allow the client to expand their game to the major platforms that they have identified including Windows, Mac, and Linux. While the cost of a Windows server is more than that of a Linux server, it’s not as expensive as a high-end Mac server. The quality of the tools and support offered by a Windows server combined with it’s stability and security features make it an ideal choice to host the game.

1. **Operating Systems Architectures**:

The kernel structure of Windows Server 2022 will allow Draw It or Lose It to easily meet the requirement of cross platform functionality. This will be achieved through the Hardware Abstraction Layer that will allow clients to make requests of the server from various popular platforms. The 64-bit version of Windows is recommended to allow the server to handle the amount of processes needed to run thousands of games simultaneously.

1. **Storage Management**:

Some game files like images and classes can be stored on a local SSD which will reduce the

number of requests that the server will need to make of the cloud storage or database. The bulk

of the application data such as game instances and player data will be too large to store locally

and will need to be stored using some kind of database solution. SQL is a popular database

solution that works well with the recommended Windows operating platform.

1. **Memory Management**:

The application will use a combination of Window’s virtual memory as well as the physical memory of the server. Windows can manage memory swapping, giving the application the ability to prioritize processes and interchange them from main memory to secondary memory as needed.

1. **Distributed Systems and Networks**:

A simple Client Server pattern will allow The Gaming Room to distribute the functional software between the server and the client by connecting them using the web and taking advantage of HTTP methods for API calls. The client will have the required functions needed to make requests of the server and the server performs the required processes and returns something to the user. This kind of connectivity will require both the server and the client to be connected to the internet for the game to function properly.

**Security**:

The primary recommended security measure is the use of role enforcement. Each user will have

their own user information that will contain data about their role. They might simply be a user

that has access only to the games features from an abstract front-end perspective. This kind of

user would have no access to the fundamental software that the game uses, giving them no

chance to intentionally or accidentally cause issues with the back end. This role enforcement

will need to be coupled with some kind of password protection such as two factor

authentication or an encryption key.