

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 | 11/11/2022 | Calvin Cottman | Added executive summary, design constraints, and domain model. |
|  |  |  |  |

## [Executive Summary](#_sbfa50wo7nsh)

Creative Technology Solutions would like to create a team-based game loosely based on the 1980’s television game “Win, Lose or Draw”. In this version of the game, we will create an application that renders an image steadily over 30 seconds while a team tries to solve the puzzle. If the team does not guess the puzzle before the time expires, the remaining teams have a 15 second window to solve the puzzle. Each round lasts one minute and there are four rounds.

## [Design Constraints](#_2et92p0)

* One or more teams can participate in the game.
* Each team can have multiple players assigned to it.
* Game and team names must be unique, and users must have the ability to check if their name is unique.
* Using a unique identifier, each game, team, and player can exist in memory in only one instance.
* This application will be a 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)

Below is our UML class diagram. We have a “ProgramDriver” class which is responsible for driving the application with the main function. The Program Driver class uses the “SingletonTester” class. This class is used to test the singleton design pattern which ensures that there is only one instance of a created class which we use for creating one instance of a game.

Next, we have the “Entity” class which is an abstract class that defines the components of the game, including the game. The “GameService”, “Game”, “Team”, and “Player” classes are all derived from this entity class and inherit the traits and functions of this class. The Entity class consists of a private ID variable and a private name string. The functions of the entity class consist of a constructor to create. It also contains a method to obtain the private variable “ID”, and a method to obtain the private variable “name”. Lastly it has a function to convert data into a string.

The GameService class consists of a list of games, and methods to return the game objects via a name, or game ID. It can also return the next player ID or next team ID. This class has a constructor to create an instance of a game service and there is also a function to add a game to the list of games. This class facilitates the games so that there is one instance of a game happening at a time. This is how this class relates to the game class.

The game class consists of a list of teams. There are functions to add a team to the list and create an instance of a game. There can be multiple teams in a game and that’s how this class is related to the team class. The team class similarly has a list, but this is of players. There is a function to add players to a team, and a constructor to construct a team. Lastly there is the player class which defines a player ID.

**"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 makes use of multiple languages. HTML, C, Objective-C, Swift (for apps). Mac is a secure OS good for web-based use. Mac also has great integration between its OS and iOS. HTML would be a perfect fit for this application. Great host for mac, ios, and Android. | Open-source OS. This OS is more secure than Windows, and even a bit more secure than MacOS. This platform would be difficult to get to host a web-based program because it is mainly run in C Language but is a good interpreter language. Good host for Windows and Linux. | While Windows is not the most secure OS, it has a lot of pros. Windows is the most used OS of all of them. This is big when you want to attract users. C-Language is a good interpreter. Uses .net Web developers (C#).  Great host for Windows, Android and Linux. | Mobile devices have two major operating systems. Android and iOS. Both of which have integration with Windows and MacOS respectively. Objective C can be integrated with C-Language and iOS would need interpretation to host multiple clients. |
| **Client Side** | Mac uses HTML which is great because the working app wants an HTML interface to run. This will not be costly as the language does not need an interpretation and could be done in less time than other platforms. | C-Language is used to structure middle-level programming and develop low-level application which is fine in this case. To work with other applications, there would need to be an interpreter and that means more developers on the project. This would be more costly than the other three. | While Windows is a very mainstream OS, the implementation of the client side would require some interpretation since the programming language does not directly work with HTML, but rather net. Web development. A bit costly but easier than other solutions. | The mobile devices have two different languages in which at least one would need interpretation to the server. This would be costly and time consuming as there would need to be a developer specializing in both languages or multiple developers. |
| **Development Tools** | Mac makes use of a few different languages. This may be expensive to develop as you may need more than one specialist or a developer who specializes in multiple languages. | Linux uses C language in their OS. This is the only language used, but there are different variations of their OS. C-Language is a good language for beginners so most programmers should know this language. | Windows uses C-Language and C#. C# helps developers create NET-connected applications for Windows to the internet. C# is ideal for beginners so it would not take long. Developers would be easy to and inexpensive. | Mobile apps require different languages from their respective OS but integrate tightly with them. Android apps require Java and iOS require Objective-C. This will require at least two different developers which can make is more expensive than the other choices. |

## 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**: A great operating platform to run this software would be Windows. With Windows being widely used by most people it makes sense. Since it uses C-Language, it will be easier to integrate another OS into this operating platform as C-Language integrates with all of the other Operating Systems.
2. **Operating Systems Architectures**: Windows uses C-languages which include C, C++, and C#. These languages integrate well with others and is an easier language to learn. The Server side of this program will be in HTML which is supported by Windows. It may require developers who know JavaScript develop code for this Server side.
3. **Storage Management**: A good way to manage storage is through a Storage-Area Network. A SAN can be dynamically allocated to hosts. It can prohibit access to hosts. It has better performance than the other network-attached method, because the storage I/O operations consume bandwidth which increase latency. We want to avoid latency to give users playing the game a fair chance while the picture loads incrementally. A host attached storage would not be a desired method as users cannot access this remotely.
4. **Memory Management**: It is recommended that C\_SCAN scheduling method is used. This algorithm seeks to find access to memory fast. This is imperative in a game where we will be slowly rendering an image. Having fast access to memory is key when we a playing a time-based game over the web. There is not a heavy load placed on the disk, not from the library but from the amount of I/O request that can be given. The goal of this application development is to expand. This may mean thousands of players on a server at once.
5. **Distributed Systems and Networks**: The key to having great functionality in the realm of networks would be to ensure that the integration between all the different operating systems is solid. Another factor is to reduce latency as much as possible because there can be thousands of people playing this game. It is important that the server is maintained as well, this way the server operates the way it is supposed to. This can help reduce the probability of an outage.
6. **Security**: Security for this application needs to be reasonable for a game. There isn’t a high level of information being stored by the application, but it should protect from the common attackers. One way to be secure is to maintain servers. Also, passwords and usernames need to be encrypted or use of a key would help. Two factor authentication is not needed for a game like this because there is no money from the user involved. The basics of security in Windows can be used to keep this game secure and running. There is not a huge exchange of information or valuables being exchanged but it is always best to be as secure as needed, not too much more and not less.