

<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 | <03/17/23> | <Cole Baxendale> | <Initial software development> |

**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 currently would like to develop a web-based game that serves multiple platforms based on their current game. Currently it is available as an Android app only. This game includes multiple games, and each game has multiple teams with multiple players. The game consists of four rounds each lasting one minute. Drawings will be complete by the thirty second mark, if the team does not guess the puzzle before the time expires the remaining teams have a chance to have one guess with a fifteen second time limit. The staff of The Gaming Room do not know how to set up the environment.

## Requirements

*Only one instance of the game can exist in memory at any given time.*

*Game and Team names must be unique to allow users to check whether a name is in use when choosing a team name.*

*Each team will have multiple players assigned to it.*

*A game will have one or more teams in it.*

## [Design Constraints](#_2et92p0)

IOS, the web, and Android all use different software development kits.

The API needs to be able to work with all three of the different platforms.

Use unique IDs for each instance of a game, team, and player to limit the instances of the game to one.

A feature to let the creator of the team pick a new name if their choice is already taken.

## [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 static main method. ProgramDriver class uses directed association with the SingletonTester class.

The SingletonTester class has one public method which tests if there is already an instance of GameSerivce.

GameService class is a singleton class. It has five static private class members, games, nextGameId, nextTeamId, nextPlayerId, and service. It has a private default constructor making this class a singleton class. It then has a public static method to get the only instance of GameService, getInstance(). GameService then has six more public methods, addGame, getGame(long), getGame(String), getGameCount, getNextPlayerId, and getNextTeamId.

Entity class is the parent class to Game, Team, and Player. It has two private class members, id, and name. It then has a private default constructor. Entity class then has four public methods, one public parameterized constructor with id and name as parameters, getId, getName, and toString.

The Player object only has two public methods and that is all, a public parameterized constructor with the parameter’s id and name, along with a public toString method. Since Entity is Player’s parent class it inherits all required attributes from Entity. Player class has a many to zero relationship with Team class. This means a player cannot have a team, but a team can have a player.

Team class has one private class member, a list named players which holds Player objects. It then has a public constructor with parameters id and name. Lastly it has two more public methods, addPlayer and toString. Since Entity is Team’s parent class it inherits all required attributes from Entity. Team has a many to zero relationship with Game. This means a team cannot have a game, but a game can have a team.

Lastly Game object also inherits all of Entity’s required attributes. It then has a private class member; a list name teams which holds Team objects. It then has a public constructor with parameters id and name. Lastly it has two more public methods addTeam and toString. The Game class has a many to zero relationship with GameSerivce, this means Game cannot have a GameSerivce but GameSerivce can have Games.

**"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 has flexible terminal commands to configure access or make changes to the server. The cost is expensive though. | Linux is very similar to Mac it can do anything Mac can but is much more cost efficient. | Windows has more software available compared to other OS, but the cost is also expensive. | It is better if the server is immobile and can be tracked to a single place, they lack the power of other OS systems as well. |
| **Client Side** | It has a moderate amount of expertise and time required the cost is expensive and is around the same price as Windows. | There is also a minimal amount of expertise and time required but the cost is much cheaper than both Windows and Mac. | There is a minimal amount of expertise and time required but the cost is still expensive like Mac. | Mobile provides a flexibility to clients and developers to see updates from anywhere. It is slightly more difficult to implement than other devices though. |
| **Development Tools** | Visual Studio, PyCharm, Eclipse, Xcode, SQL and MongoDb.  HTML, CSS, JavaScript, Python, Java, C++, C#, Ruby, PHP. | Visual Studio, PyCharm, Eclipse, SQL and MongoDb.  HTML, CSS, JavaScript, Python, Java, C++, C#, Ruby, PHP. | Visual Studio, PyCharm, Eclipse, SQL and MongoDb.  HTML, CSS, JavaScript, Python, Java, C++, C#, Ruby, PHP. | Xcode, IDLE, Android Studio, Visual Studio, Eclipse.  HTML, CSS, JavaScript, Python, Java, C++, C#, Ruby, PHP. |

## 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**: Since the game is going to be operating on multiple platforms a good choice would be using HTML, CSS, and JavaScript. These languages together can run on Windows, Linux, Mac, Android, and IOS. It is also very popular so many developers will understand the languages used. I would recommend using a Linux Server to host Draw it or Lose it, this is because Linux has the capabilities of all other OS but also costs cheaper than the others.
2. **Operating Systems Architectures**: Using the Linus Kernal is secure and stable. The clusters allow for easy separation of systems, it will even have easy separation of hardware needs. This is the best choice for Draw it or Lose it.
3. **Storage Management**: You can use an SSD or HDD for storage. I recommend using an SSD considering this is the newer technology which is much faster than HDDs. Using an SSD will allow the user for faster load speeds and start up time which is a huge must for some users. This game should use SQL for a database, it is easy to use and set up. You also can start small and up the cost as you need more database space.
4. **Memory Management**: You may use a watcher for the load on the system. This will give you the ability to lower the memory usage during times the load is low and up the memory usage during high load. This will also help the cost be cheaper.
5. **Distributed Systems and Networks**: The system will be on the cloud; this means your game will not have any issues if the server or system crashes. Another server will start if this happens, this will give you user a better experience during times of trouble with the server.
6. **Security**: You can keep your users secure by using a system that gives each user a role. For example, a developer may have an admin role which allows them to see information a regular user cannot see. By doing this it will make it very difficult for any user that does not have a special role to access information they should not be able to access.