

<Name-of-Software-Application>

# **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 | <mm/dd/yy> | <Your-Name> | <Brief description of changes in this revision> |

**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, The Gaming Room, already has an app available for Android platforms called “Draw It or Lose It”. They would like to expand their current scope and develop a web-based game that operates on multiple platforms. The app will provide images from a library that will allow the players to guess the image. Each round lasts one minute, and each game has four rounds. The drawings will render over a period of 30 seconds and if the team cannot guess the image prior to the one-minute mark, the other teams will have 15 seconds to provide a guess.

The client has listed off the following software requirements:

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

Our proposed solution to these requirements would be to implement both the Singleton and Iterator design patterns. The Singleton design pattern will only allow one instance of the game at any given time, while the iterator pattern will allow multiple players on multiple teams, with unique names for each player, team, and game.

## [Design Constraints](#_2et92p0)

We will be working with a diverse arrangement of operating systems and platforms, so it goes without saying that we will have both software and hardware as technical constraints. On the business side, this is likely to be a large project with many moving pieces. Budgets, deadlines and personnel will be large factors to consider as well. We will need a team that is proficient with Swift, Android, Windows, Mac, Android, and web-based applications. Briefly listed below are the previously mentioned constraints:

**Technical:**

* Software
* Hardware

**Business:**

* Deadlines
* Budgets

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

**"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.**

The Game, Team, and Player classes all inherit from Entity. GameService has a “zero to many” multiplicity with Game, and the same goes for Game/Team, and Team/Player. We can see that the Entity and GameService classes have private attributes and a private constructer method. This is will be used to limit the number of class instances that can exist at any given time via the “Singleton” design pattern. Game and Team both have private attributes and public methods, while Player only offers two public methods. The main() method is in the ProgramDriver class, which uses the SingletonTester class. We do not implement the Singleton pattern on the Game, Team and Player classes because we want to allow multiple players, teams, and games.

## [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** | Popular for web hosting, but not the preferred choice. | Most preferred, and more cost-efficient. Most secure, but not a very popular OS. Lack of applications that support it. | Popular, high Availability, ease-of-use, but has a history of security issues. | Poor security, high mobility, also, very specific. |
| **Client Side** | Well-versed knowledge of Mac will likely be needed. | Linux is not a very popular operating system compared to the likes of Mac and Windows. It is likely that very specialized help will be needed. | Windows is very popular, and with its ease-of-use it shouldn’t be much of a chore making it compatible. | Mobile devices are usually very simple to use. Development is not very complicated. |
| **Development Tools** | Swift is the preferred option for Mac, but it can run most languages as well as the libraries. | Linux is less user friendly but can still be used in mostly the same fashion as windows. | Probably the easiest to use for the majority. There are a wide variety of resources for development. | There are plenty of resources for mobile app development. We may require specialized knowledge to ensure that the app is compatible with both. |

## 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**: I would definitely recommend Windows, since it is very popular and there are a wide array of resources for development on windows.
2. **Operating Systems Architectures**: Windows has a user-friendly interface with multiple applications running at once. You can do almost anything simultaneously.
3. **Storage Management**: Windows allows use of local storage as well as cloud storage.
4. **Memory Management**: This Game will require a database of pictures, so it would be beneficial to use copies and have accessible back-ups. This can be easily done in Windows.
5. **Distributed Systems and Networks**: In order to reduce the amount of complication in this cross-platform application, I would recommend a server-based game that operates via browser, Similar to how Runescape and other MMORPGs work.
6. **Security**: Windows monitors itself for threats on a regular basis, and it even comes with additional software to do so as well. There are many resources for security on Windows