

<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 | 03/24/2024 | Mary Watts | Defining the constraints, create the executive summary, and making the domain model description. |
| 1.2 | 04/21/2024 | Mary Watts | COmpleing the recommendations |

**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)

<Write a summary to introduce the software design problem and present a solution. Be sure to provide the client with any critical information they must know in order to proceed with the process you are proposing.>

The problem that is presented is taking our Clients current Android only app “Draw it or Lose it”, loosely based on the hit show *Win,Lose or Draw*, and making it cross- platform and more widely accessible. The game needs to be able to have one or more teams involved with multiple player. Each game and team name must be unique so that the users can check to see if the name is available. The last requirement is that only one instance of the game can exist at a time ( which is achievable by creating unique identifiers for each instance of a the games.

## Requirements

1. Game must have the ability to have one or more teams.
2. Each team must have multiple people.
3. Game/team names must be unique to allow for a check to see if that current name is in use.
4. There can only be one instance of the game in the memory at a time.

## [Design Constraints](#_2et92p0)

This is a currently only available via the app and they would like to be able to be access via a web-based platform as well, this also throws into question the creation of the environment around the site that is created for it. There would need to be works for network communications, security, and most importantly compatibility cross-platform.

Unique names are also one of the constraints as the system would need to enforce the creativity and uniqueness of the game/team/player names, as this prevents the naming conflicts and creates an optimal end user experience during the creating/joining of games.

There is also the matter of maintaining the proper functionality and accommodation of only having one instance of the game service in the memory.

## [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)

This UML Diagram shows the class structure of the game application system itself. At the center of it all is the Entity class that serves as the Parent class and contains all the common attributes such as the game id and the game name, making sure that each name has a unique identifier. Three classes extend the parent class: Game, Team, and Player. They are the key entities of the game application. The ProgramDriver class is the entry point and is where the main function is. It also contains the singleton instance of GameService to make sure that there is only one instance of all key components. The ProgramDriver is sole responsible for adding all key entities using the GameService instance. It also has a dependent relationship with e the SingletonTester so that it can ensure that there is only one iteration of the game/names at a time. Lastly, the GameService class has a composition relationship with the game class meaning that it manages the Game instances and holds all references. Team also has a composite relationship with the Player class and the Team has one with the Game.

The UML class diagram shows instances of inheritance through the relationship between the Entity class and the Game, Team, and Player classes. You can physically see the encapsulation of attributes as well as the abstraction

**"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** | <Evaluate Mac for its characteristics, advantages, and weaknesses for hosting a web-based software application.> | <Evaluate Linux for its characteristics, advantages, and weaknesses for hosting a web-based software application.> | <Evaluate Windows for its characteristics, advantages, and weaknesses for hosting a web-based software application.> | <Evaluate Mobile Devices for their characteristics, advantages, and weaknesses for hosting a web-based software application.> |
| **Client Side** | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mac.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Linux.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Windows.> | <Determine the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients as they pertain to Mobile Devices.> |
| **Development Tools** | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Mac.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Linux.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Windows.> | <Identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on Mobile Devices.> |

## 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 recommend using Windows OS as it is the most widely used OS. It is flexible and allows for cross-platform play. It supports the latest in networking and security, although security needs to be boosted to offer better coverage. With the amazing scalability and performance, it is amazing for a growing game and player base. It is also integration with Microsoft it is great for game development.
2. **Operating Systems Architectures**: Most OS have the same architectures, containing hardware, kernel, and the shell. The Kernel is the connector between the application and the data being processed, the hardware is the physical attributes of the operating system such as memory and CPU. Lastly, the shell is the software that provides the interface for the operating system. With Windows there is a separation between the user and kernel, this helps prevent crashes and security breaches. This also allows for high performance, security, and compatibility with different hardware and software.
3. **Storage Management**: With the game being a web-based application, it needs a decent Cloud storage system that utilizes the storage space only to keep the game rolling. With it being a Cloud-based operation, upping the memory will cost hardly anything as it is a software issue and not a hardware issue. The cheapest and easiest option for this would be Google Cloud.
4. **Memory Management**: Windows compartmentalizes the memory of the system into sections and assigns each application its individual address per se. There are tools on the Windows OS that are integrated so well that it is incredibly easy to control the usage of memory.
5. **Distributed Systems and Networks**: I suggest we use a client-server system as it allows for customization for each client. The game requires the ability of multiple clients to connect to one server for the game to work optimally. We need to have something that factors for network delays, outages, and security.
6. **Security**: Windows Defender is already integrated into the Windows System as a security system. There needs to be other measures to help with security such as audits and security testing. Not only that we need to have a strict response plan to help have quick response to the breaches.