

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 | 03/20/2024 | Kimberly Castro | Updated the Executive summary and Design constraints. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room has an Android only application that is based off the 1980’s tv show “Win, Lose or Draw”. The Gaming Room would like our help in setting up the environment for the web-based version of the gaming app. The Gaming Room Staff needs us to develop the game application to meet their specific requirements. We will streamline development, provide a smooth transition ensuring operating systems hardware requirements are met and meeting the client's expectations. We will solve their problem by developing a robust and scalable web-based app that integrates modern web technologies.

## Requirements

* A game with the ability to have one or more teams.
* Each team can have multiple players.
* 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.
* Each game, team, or player will have unique identifiers.

## [Design Constraints](#_2et92p0)

Design constraints for this application include the following:

* Ensuring unique identification for the game, team, and players.
* Using modern web technologies to match the original game design from the android version.
* Ensuring there is data consistency between each platform to avoid security implications and bugs.

These constraints will help guide the process and aid in the development of a secure, scalable, web site for the game.

## [System Architecture View](#_ilbxbyevv6b6)

* Web browser
* Server-side application
* Data base management
* Security implementation
* Data consistency

## [Domain Model](#_8h2ehzxfam4o)

The UML diagram below uses the following classes:

The ProgramDriver is used to run the main program and uses the SingletonTester class to call the testSingleton method which checks if the singleton instance works. The Entity class is the parent class to the Game, Team and player class and is used as a base class to hold common attributes and behaviors. The child classes inherit the attributes id and name and are used in their respective constructor class. The child classes also inherit methods from Entity. The player class has a constructor method called Player, and a toString class that calls the getName and getId methods to get a text representation of the object. The Team class creates a list of currently active players. It also has a constructor method titled Team, an addPlayer method that calls the getName method from its parent class, and a toString method that calls the getName and getId method as well to have a text representation of the object. The Game class creates a list of currently active teams and has a constructor method, an addTeam method that uses the getName method from its parent class, and a toString method that calls the getName and getId method to create a text representation of the object. The GameService class creates a list of all the currently active games, and has private attributes called nextGameId, nextPlayerId, nextTeamId and service id. The class also has a private constructor class and a public accessor and uses the singleton method to create a single instance. It also has an add game method that checks for the currently active games and creates new games, and uses accessors titled GetGame to get the game based on either the game id or using the games name, a getGameCount method that returns the number of currently active games, and a getNextPlayerId and getNextTeamId to get a new player and team id. This program uses inheritance, abstraction, classes and objects, association, and encapsulation exhibiting principles of OOP. This program ensures efficiency because it facilitates modular maintainable software and fulfills the requirements by promoting code reuse, encapsulating data, and establishing clear relationships between classes.

**"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 OS offers server-based environments that are greatly beneficial to mac users. These servers can fully support mac clients and offer amazing graphic interfaces. The downside to these servers is that it is expensive to upkeep and are not good with handling third party programs giving limited ability to using and customizing third parties. | Linux is free and opensource. It offers a variety of customizations, and programming languages. The downfall to Linux is that there is a steep learning curve, and it also does not have the support for some common software that is used. | Windows is commonly used and has a remarkably simple interface with an easy learning curve. Windows does require a licensing so it can be expensive, however windows have a wide range of third-party support. Windows also has consistent updates and patches for their software. | Oracle offers web server support for mobile devices. The advantage of using Oracle includes support for iOS and android devices. They also have a large database that can manage larger software. |
| **Client Side** | A simple user interface with good graphics. Not much expertise is needed. It can be expensive for users, however. Moderate time is needed. | Linux has many different distributions and a plentitude of options; therefore, it will take an extensive amount of time is needed to learn the interfaces. Relatively cheaper compared to the other OS. | The user interface for windows is easy to learn and there is plenty of documentation to support setup. Minimum expertise is needed. Since a licensing is required, it can be expensive. | Has a very friendly interface and allows notifications to be received at any time. It is harder to implement than other devices and the tools required can be costly. |
| **Development Tools** | Development tools can include HTML, CSS, JavaScript, PHP and python. Supported IDE’s include visual studio code, PyCharm, GitHub, eclipse, NetBeans, rider, WebStorm, etc. | Development tools can include HTML, CSS, JavaScript, PHP and python. Supported IDE’s include Vim, Eclipse, Geaney, VSCode, and Gedit. | Development tools can include HTML, CSS, JavaScript, PHP, bootstrap, angular, vue.js, flutter, etc. Supported IDE’s include visual studio code, PyCharm, Rider, BlueJ, Code Blocks, etc. | Development tools include Swift, Kotlin, Flutter, React, Java, HTML, CSS, and python. Supported IDES include Android Studios, Xcode, MIT App Inventor, WebStorm, IntelliJ, JDeveloper, and Eclipse. |

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>
2. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>
3. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>
4. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>
5. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>
6. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>