

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.01

## 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 | 09/18/2023 | Corey Embrey | First release of the “Draw It or Lose It” for demonstration purposes |
| 1.01 | 10/2/2023 | Corey Embrey | Updates to recommendations regarding client needs |
| 1.02 | 10/17/2023 | Corey Embrey | Further updates to recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wishes to develop an environment for their web-based gaming app “Draw It or Lose It.” The game will require the ability to have multiple teams with multiple players playing in multiple games, and only one instance of a game can appear in memory at a time. As a result, the game must use a singleton pattern to ensure only one game is in memory, and a UML diagram has been developed to streamline the process of writing code for the classes which will be developed.

## Requirements

* Only one instance of a game can exist in memory at a time
* The application will be web-based
* The application must support multiple teams, with multiple players in multiple games
* The application must perform checks to ensure the names of the players and teams are unique, and display whether a name is already in use
* The application must render images from a library of stock drawings, which are to be complete by the 30-second mark
* The game will have parameters involving multiple rounds, with special parameters for a tie-breaker scenario

## [Design Constraints](#_2et92p0)

Only one instance of a game can exist in memory at any given time, so a singleton pattern will be necessary to ensure this is the case. The application is web-based, so functionality on the most common web browsers should be taken into consideration during development (Chrome-based browsers, Safari). The application must utilize the singleton pattern to create games, players, and teams. Players and teams will need additional methods using an iterator pattern to ensure that unique names are used for both players and teams, and displayed to the player when a name is currently in use using a list. Each game will render stock images from a library over a period of 30 seconds, and the players will be able to input answers to guess the drawings as they are being rendered. Each game will have 4 rounds lasting one minute each, with special parameters for a tiebreaker if the scenario permits. If the developers intend to make the app accessible on mobile devices, developing a separate mobile app should be strongly considered, as web-apps tend to be less user friendly when using mobile browsers.

## [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 diagram helps tidy up the code from our previous project, especially regarding inheritance. Rather than multiple, identical chunks of code in multiple classes, this UML moves many of the functions into the entity super class, which are inherited by the Game, Team, and Player class, which operate in similar ways. The getters for the Team and Game classes are also moved into the GameService class with other similar functions. The code is encapsulated into multiple classes, with the Game class dealing specifically with functions related to adding teams, and the Team class with functions specifically related to adding players, utilizing private lists. Abstraction is used in multiple classes using private methods and hides information from other classes and is primarily used in the GameService class which is utilized by the Team and Game classes. Polymorphism is used in multiple classes with overloaded constructors.

"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)

**Note:** The Gaming Room will be able to host a server for 1000 users in house, but a cloud configuration should be considered and prepared for a larger user base should this occur more quickly than expected.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | MacOS server has been discontinued as of 2022. Investing in Mac hardware is an expensive endeavor compared to the other options. A server-side client could be hosted on Mac using Maven, but it would not be a wise investment considering the application is currently designed for Android and this would create unnecessary financial overhead. An HTML protocol will be used to create the server. | Linux is the most common choice for a server environment because of its reliability, speed, and security. Linux server editions are typically free of charge, although it has licenses for businesses which provide technical support through a subscription-based service. Creating a webserver on Linux is viable and a great option in terms of reliability and should be the number one choice for hosting a webserver on in-house hardware using our projects IDE, Eclipse, development which contains our development tools and Maven which we will initially be using to deploy our server. | Windows has tools to create a server environment and is a solid choice to host a server with IIS, which can be customized with a GUI and includes security features backed by Microsoft. Using a cloud solution with Microsoft requires a premium, but I believe the server should be run on Linux if using in house hardware, especially if it is being hosted on a dev machine because the overhead from the OS could get in the way of server performance when the user base increases. | Mobile devices can host servers but are more complicated to set up, and the hardware is limited. The server should not be hosted on a mobile device. |
| **Client Side** | To support a Mac client with a web-based application the app needs to function within the safari browser. Unless an app is specifically designed for Mac, the Safari web browser will support web-based programs that run on other popular web browsers. | Linux distributions have a low market share in terms of desktop Linux and most commonly include the Firefox browser. Some attention may be needed to ensure the webapp functions with Firefox but a webapp developed with a more popular browser should function correctly in Firefox without modifications. | Windows uses the Edge browser by default. While Chrome is the most popular browser and the most used browser on Windows, Edge is based on chrome and should not have any problems running webapps designed with Chrome in mind. Due to Chromes market share, targeting the Chrome browser is a necessity when developing the webapp. | Mobile markets will require full applications to run with an interface that is acceptable to mobile users. The Gaming Room is currently developing an android application; however, a Mac will be required to develop an application for IOS by rewriting code to function in objective C. |
| **Development Tools** | Mac uses the Safari browser and because the application is a webapp, the game simply needs to be compatible with the Safari web browser. Mac recently gained access to Visual Studio, a Microsoft exclusive IDE, but has access to the most popular IDEs in addition to tools to develop applications for Mac exclusively which are not available on other platforms. At least one Mac workstation will be required to develop a mobile application for IOS using Xcode, which will require hiring a developer with Mac experience to write the code in objective C to develop and maintain the project. Fortunately, Xcode is freeware and will not add any additional overhead to the budget. Mac also has access to our current development IDE, Eclipse. | Linux has access to most popular IDEs including Eclipse and the JetBrains suite of products, as well as Visual Studio Code, a popular code editor with extensions that can make it function like a full IDE. Unfortunately, it does not have access to Visual Studio, the premier IDE for dealing with Microsoft specific tools, which are focused on C#, C++, Azure, and .NET Frameworks. The current project is being developed in Eclipse, which has full functionality on Linux and is available for free. Once the development environment is configured, working in Linux is like working on Windows, so this will be an extremely easy transition for developers who are currently familiar with Windows. | Windows has access to nearly all popular languages, and its versatility is only restricted when dealing with Mac applications. Thanks to the inclusion of WSL2, Windows is now able to run a virtualized Linux kernel within the operating system with minimal overhead. Windows also has native support for Visual Studio, and functions with the least number of bugs on a Windows environment. Windows natively supports Eclipse, which is our primary development tool and is available for free. Windows is the most popular OS for both users and developers, and as such most developers will have at least some experience with Windows. | Mobile devices are extremely limited in terms of development applications. Typically, mobile devices are targeted for client-side applications and software is developed on a computer with the aid of virtualized mobile software to test compatibility. |

## 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**: Linux should be chosen as both the operating platform and the server for this application. Not only is Linux desktop free to use, but it has all the necessary tools to work on the project, which is intended to be compatible with Android systems. Linux is known for its security because it separates individual users, who have access to programs on the operating platform and “super users,” which have access to the root and require a password to be input to access elevated functions and has low overhead, allowing it to run smoothly on older hardware which would save on costs. The company should choose the paid version of this server to deal with its security needs: support is much cheaper than Windows servers and will alleviate the company's team regarding server security, allowing it to allocate more resources to development.
2. **Operating Systems Architectures**: I recommend using an in-house server and storage management, The images for the application are high quality but fairly small at roughly 8mb per image, while rendering with a CPU can be an intensive task, the images with be rendered over a period of thirty seconds, making the impact on the end-users system negligible with the added benefit of the application being less taxing on the server itself due to having to render the image for multiple games. This allows the server to simply deliver the image to clients, and due to the rest of the application being a CLI interface will allow for better scalability.
3. **Storage Management**: The Gaming Rooms application is a low physical memory application, and so an in-house SSD can be used at low cost and will be more responsive than a cloud server. However, a cloud server should be selected by the company in advance so the company can properly introduce a scaling solution should the applications popularity increase to the point that more storage is needed than what is acquired in the initial purchase of the physical hardware.
4. **Memory Management**: The webapp itself ensures the no more than one game instance can be loaded into memory at a single time. The functionality of the application is text-based utilizing images to play the game, and because of this, 3 images can be stored in RAM at the start of each game and then cleared at the end of a match. The application has a naturally low memory footprint by design.
5. **Distributed Systems and Networks**: The Game Company is developing a web-based application, so the functionality for the game will be through the web browser. A reliable cloud server should be chosen to ensure maximum uptime of the server, and a cloud network should be used in favor of the original in-house network as soon as the game increases in popularity to the point that the physical hardware starts to become overly taxed.
6. **Security**: User privileges must be separated from administrative privileges. The company's server-side client must be the only client with access to administrative privileges. Users should have accounts that only allow them to access the required functions to properly play the game (creating a user account with restricted privileges, creating, and joining teams, inputting answers). The support from Ubuntu Server when addressing security issues; while in-house security could be developed by a team using a RESTful API, as the application grows it will be imperative to have more robust security measures in place as the application grows and becomes an actual target for attacks.