

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.1

## 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 6**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 3/19/23 | Chayne Pieri | Added Executive Summary, Requirements, Design Constraints, and Domain Model. |
| 1.1 | 4/1/23 | Chayne Pieri | Added Evaluation Response. |
| 1.2 | 4/18/23 | Chayne Pieri | Added Recommendations Response. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is looking to develop a multi-platform, web-based game based on “Draw It or Lose It”, their current Android-specific app. The game is team based with 30 seconds for the current team to guess what is being drawn or the time expires. If they are unable to guess, all other teams have one chance to guess within 15 seconds. Creating the game to be web-based will allow customers to access the game through Mac OS, Linux, and Windows operating systems, as well as mobile devices with web access.

## Requirements

* Runs as a web-based game to run on many platforms.
* Exclusively one game instance can be run at a time.
* One or more teams can be involved in the game.
* Game and Team names must be unique so users can check if a name is in use while selecting a team name.
* Each team can have more than one player assigned to it.

## [Design Constraints](#_2et92p0)

The current Android-based game will need to be implemented into a web-based application. In order to make this transition, and provide customer access from many different device types, we must provide application solutions that implement a server-client relationship. While all database information will be stored on the server, clients will retrieve this information and provide the server with required information such as user input and object creation. All validation will be completed on the server, which will be returned to the client.

## [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 Game, Team, and Player Classes inherit from the Entity Class, so they have access to all the Entity public methods. Since the relationship from Team to Player Class is none-to-many, the Team Class has Player Classes. Since the Game to Team relationship is none-to-many, The Game Class has Team Classes. The GameService Class also has Game Classes for the same reason. This workflow shows that GameService utilizes and can have more than one Game, Game utilizes and can have more than one Team, and Team utilizes and can have more than one Player in their Classes. The GameService utilizes a Singleton Pattern, so only one instance can exist of GameService at a time.

**"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 provides limited hardware options to create applications, so developing on this platform is straightforward. You must work within the limited hardware requirements. | Linux provides a very stable platform to run servers with. There is very little vendor support, and there is a steep learning curve to developing with this. | Windows as a server platform provides easy development utilizing HTML. Files are easily stored and accessed on numerous devices using Windows servers. | Mobile Servers are one of the more difficult options to work with. Since mobile provides very limited features compared to other platforms, working around the technical hurdles requires complex solutions. |
| **Client Side** | Developing client-side features for Mac OS provides the same hardware limitations as the server, but retrieving the required data through Safari or another web service is straightforward. | Most Linux services are free to obtain and developing a client within this operating system is not very complex. Since web service is readily available, developing a web-based app will not be difficult. Time, cost, and effort will be moderate. | Development in Windows OS provides a feature-rich environment. Many different manufacturers support Windows. This also causes more thought to be required in development as the client must run on as many device configurations as possible. | With limited input options available for Mobile devices, much more care must be taken to ensure access is available to all customers. Touch and typing inputs are the most common, but web access is available to all device types. Interfaces must be created in a must simpler fashion than desktop devices. |
| **Development Tools** | Mac OS development is mostly written using Objective-C. WebStorm provides developers access to JavaScript, which will make web development easy. | Linux provides developers access to many different programming environments. Java, JavaScript, and C# will provide great development tools to make a web game. | Many developers use C++ for Windows applications, but many environments are available. JavaScript is a workhorse for web-based applications. | Developing applications using mobile devices is much more limited than other platforms. Swiftic is of the few options available. |

## 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**: While it may take some time to learn the operating system setup procedures, it is recommended to use a Linux Ubuntu server as the setup is the most cost-efficient. Linux will provide the Gaming Room with a very stable operating platform to provide access to their product, Draw It or Lose It, while customers are able to use Mac OS, Linus, Windows OS, or mobile devices to access their game via the web. Because Linux servers work on an open-source platform, a web-development community provides constant improvement to the operating system.
2. **Operating Systems Architectures**: To provide users with access to stored data that will interface with other users, it is recommended to utilize a client-server structure. The server, which is previously recommended as a Linux server, will isolate data storage and provide the users with access to authorized content, as well as the ability to respond or guess what is being provided to them. Because a team-based guessing game has low interactivity, unlike first-person shooter-style games, a server-client architecture will be suitable.
3. **Storage Management**: A game centered around rendering prepared drawings to multi-platform users requires large storage requirements. As scaling grows, the storage requirements to maintain the newer prepared drawings will need to be increased as well. Utilizing Solid State drives within the server will allow rapid access to the images. Utilizing a SQL database will allow the Gaming Room to provide users with access to the required information in a timely manner.
4. **Memory Management**: Rendering images for users of Draw It or Lose It will require proper memory usage. Linux provides access to the proc filesystem to initiate data queries. While Linux memory management requires learning the architecture’s tracking and memory mapping, it becomes cost-effective in the long run.
5. **Distributed Systems and Networks**: Linux provides easy access to the use of RESTful APIs. Secure data transfer between devices on one of many platforms is supported by a RESTful API. Since RESTful APIs are used for web access, this will provide the Gaming Room a way to easily distribute their game information to any device that has access to the web, provided that the device’s operating system has been prepared for on the server.
6. **Security**: As RESTful API allows for access to the use of Transport Layer Security, it is recommended to utilize TLS to ensure the user’s private data is provided between the server and the client with little chance of data leaks. It is also imperative that authentication and role-based authorization is used so that users are only allowed to access specific information that has been granted to them. Firewalls and Denial-of-Service prevention such as OWASPs DDoS should be used to keep the application up and running even when targeted attacks are made.