

Draw It or Lose It

# **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 | 07/13/2022 | Anthony Spedaliere | Initial project set-up. |

## [Executive Summary](#_sbfa50wo7nsh)

Draw It or Lose It is an Android application developed by The Gaming Room. The Gaming Room seeks to expand their application which is currently exclusively on Android. The expanded version of the application will be web based and will serve multiple platforms. It will require multiplayer capabilities and provide the user with the ability to create teams. The program will utilize the singleton design pattern to ensure that only one instance of the game can run at any given time. By expanding to the web, The Gaming Room will increase their exposure to a greater number of users which will in turn make their application more successful.

## [Design Constraints](#_2et92p0)

* Web based game
* Available on multiple platforms
* Requires access to large stock image library
* Multiplayer capabilities and multi-team capabilities
* Unique team names
* Unique game names
* Only one instance of the game in memory at a time

## [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 Entity class serves as the parent class for the three child classes: Game, Team, and Player. This is an example of inheritance. There are three associations. The first is between the Player and Team classes. There can be 0 to any number of players associated with a team. The second is between the Team and Game classes. There can be 0 to any number of teams associated with a game. Finally, the third is between the Game and GameService classes. There can be 0 to any number of games associated with the GameService. However, there can only be one GameService instance at any given time per the design constraints.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Large SaaS providers such as AWS and Azure both feature options for the Mac. These companies feature large customer service departments. By utilizing a service provider, we shift the costs of on-premises management over to the service provider. | Since Linux is open source there are 100’s of distributions. Developing an application that is a one size fits all can be difficult. Despite this, Linux is considered one of the most stable operating systems for back-end development. | Windows has the strongest support for access to large commercial SaaS providers. Node.js is commonly used in conjunction with Express and MongoDB as a means of developing backends for online applications. | Smart phones and tablets are designed with a connection to the internet in mind. Utilizing a SaaS provider will simplify the process of creating a back end by shifting the on premise management to the service provider. |
| **Client Side** | Mac products are a well contained entity. As a developer you know exactly what the user has and can track any OS updates. Safari is the proprietary browser on a Mac, but Mac supports all major internet browsers. | Linux is an open-source operating system, and there are hundreds of distributions. Developing an application with features that cross over to all Linux distributions is essential. If the major browsers are supported this should not pose a huge problem as the application will be accessed by the client through the browser. | The number of versions of Windows OS still in use varies across the globe. This variance in the Windows operating system should be considered during development. Microsoft Edge and Internet Explorer historically have had more incompatibility issues with certain HTML and CSS features than other browsers. | Smart phones and tablets vary in model and version. Globally, a lot of smart phones are still in use that are out of date. Developing for a variety of mobile devices is essential for keeping the market as wide open as possible while also excluding certain models and versions to ensure development does not get bogged down trying to support too many options. |
| **Development Tools** | Mac development tools for web applications are primarily centered around the Swift programming language. XCode is a commonly used IDE for Mac app development. | Since Linux is open-source it provides a robust community-built array of development tools such as GNOME and Ubuntu Touch. | Due to its success and long history as being the global operating system, Windows has no shortage of advanced development tools that are used in creating web applications. | Tools such as XCode and Android Studio are specifically designed for designing mobile applications. XCode uses the Swift programming language for IOs, and Android Studio uses Java for Android devices. |

## Recommendations

1. **Operating Platform**: Since the application itself will be accessed through a web browser as opposed to being downloaded onto the computer system itself, I would recommend focusing on browser compatibility. If the application is functional on all major modern browsers, then it should be accessible on all modern operating platforms. This will maximize the application exposure to the greatest number of users. It is also recommended that the applications be developed using the Windows operating system. Java should be the primary language utilized and Android Studio for a mobile application.
2. **Operating Systems Architectures**: A three-tier architectural pattern would work best for this application. It consists of a client, application server, and a data server. This division of labor allows for greater scaling.
3. **Storage Management**: Providing adequate storage capable of holding all the high-resolution images and user data can be hard to predict. The recommendation would be to use a cloud-based service. If more storage is needed, then it can easily be added. If less storage is required, it can be reduced. The use of a cloud-based service also reduces the need to store all the hardware on a physical site.
4. **Memory Management**: By using a three-tier architectural pattern memory management is optimized by spreading the processes out over three distinct stages. The client requests information from the server, and the server requests information from the data server. Each entity has its own memory and is optimized to process the information that is requested.
5. **Distributed Systems and Networks**: All modern platforms have web browsers which implement HTTP protocols. By utilizing these protocols, it will allow the client and servers to communicate. The servers can listen to client requests and update the client as needed.
6. **Security**: User security is important. All password and sensitive information will be stored encrypted on servers. Two-way authentication will be required to log in to ensure the users account is not accessed by a malicious entity. Inactive accounts will be deleted after 1 year of inactivity.