

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 | 01/23/2022 | Evan Elrod | Initial release |

**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)

The Gaming Room is looking to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, which is currently available as an Android app only. To facilitate this, Creative Technology Solutions (CTS) will assist in setting up the environment and streamlining the development. Additionally, CTS will begin developing the game application while also addressing The Gaming Room’s software requirements.

## [Design Constraints](#_2et92p0)

As this is a web-based application, it will need to be hosted on a server that is accessible to all clients. The game will be Java-based and meet the following software requirements:

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* 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. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [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 GameService class sets up the instance and can add uniquely named games. After which, the Game class can add uniquely named teams. After which, the Team class can add uniquely named players. The Game, Team, and Player classes inherit ID and name properties and methods from the Entity class.

The objects created are concise, without excessive or unrelated functionality. ID and name properties are encapsulated to protect their values. The inheritance from the Entity class makes sense since each of the subclasses needs many of the same properties and methods. The application of these principles helps to keep the code efficient while still fulfilling the software requirements.

**"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** | Apple does offer server-based environments. This is a major advantage for Mac clients. Mac applications are fully supported.  Disadvantages include higher upfront costs and cost of maintenance. It is not ideal for environment relying on third-party applications. | Linux is open source and some distributions are free or low-cost. It is very customizable and secure. Linux supports web-hosting environments featuring many programming languages.  It can be difficult to learn and some third-party applications may not work. | Windows supports the most third-party applications and works on a wide variety of hardware.  It is less secure as it is targeted most often due to its popularity. Its large userbase does mean that it is updated frequently. | One example is Oracle Database Mobile Server. It can manage data, users, devices, and applications in large deployments.  This has the advantage of better support and compatibility for mobile clients using Android or iOS, but it otherwise impractical for web-hosting. |
| **Client Side** | Apple products feature the Safari browser but also support many popular browsers like Chrome and Firefox. Developing an app on the App Store could be done in Swift or Object-C, but would also require a developer license. | Linux supports all major browsers. There is a lot of open-source software available to support development. May have a steeper learning-curve. | Like Mac and Linux, there is support for all major browsers. It features the updated Edge browser and Internet Explorer, which is now defunct. There is a webstore, but most apps can be developed and installed without utilizing it. There is wide support for many programming languages and development environments. | Developing an app might require expensive licensing fees associated with the major mobile operating systems (Android variants, iOS, and Windows) along with expertise associated with mobile app development. Just utilizing the built-in browsers would only require accommodating for smaller screens, touch controls, and mobile versions of browsers. |
| **Development Tools** | Xcode is the primary IDE with support for many programming languages. Visual Studio and other popular IDEs like Eclipse can be installed. Native applications using the app store must be approved by Apple. The Apple Dev program costs $99/year. | There are no licensing costs. There is a huge community for open-source support. Like Mac, there is a wide range of support for IDEs and languages. | Visual Studio is native on Windows, so it allow for the most compatibility with developing Windows-based apps. There is extensive documentation and support. Licensing can be free. | There is Xcode for iOS development and Android Studio for Android. As mentioned before the Apple Dev program costs $99/year; to make money from Android apps cost $25. Developing for just mobile browsers does not require this. |

## 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**:

We recommend The Gaming Room use a Linux distribution that is hosted on a cloud server. Not having a local server reduces or eliminates costs associated with procurement and maintenance. Linux is flexible and supports most development tools. As it is open source, it is free and will help with reducing costs.

1. **Operating Systems Architectures**:

Serverless (cloud-based) architectures employ modularization at the level where logic is executed and information is passed between the user and database. This means that The Gaming Room pays based on consumption and its needed services. Linux is similarly modular where the OS kernel contains only core components and separate modules support other functions and services. The result is a lightweight, efficient solution that requires less disk space and improved processing speed for a better user experience.

1. **Storage Management**:

On the client-side, the application is web-based, so an internet connection is already required. Besides the client application itself, there is likely no need to have more than that and a few of the 200 images stored. New images could be downloaded as the game proceeds and replace already used images to keep the amount of storage used to a minimum.

On the server-side, all code and images should be stored in the cloud server. As the game grows in popularity, The Gaming Room can purchase additional space without having to buy hardware. Amazon Web Services or Google Cloud would work well for this project.

1. **Memory Management**:

Unused code should not be loaded into memory. Some operating systems allow for fast and efficient loading from memory through disk and demand paging. Space on the hard disk can be reserved for use with the RAM with disk paging. Demand paging loads only what is needed for immediate processing by separating processes into smaller tasks. Finally, as the client application will be developed in Java, memory management will be handled automatically with its garbage collector memory management system.

1. **Distributed Systems and Networks**:

The cloud server and client machines communicate over the internet. Each session has a unique identifier. The client initiates messages that modify or get data from the server database. The actual physical server that is hosting the application and data is likely not tied to it, such that if it goes down due to a crash or hardware failure, the application and data can seamlessly run on another physical server. As all major browsers are supported, we need not consider the client’s operating system.

1. **Security**:

A simple username and password scheme would work best for this application. Multifactor authentication and biometrics would be overly burdensome in this situation. Many cloud service providers have their own level of security so we need not worry about implementing security systems to protect the application data or image database.