

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 | 03/21/21 | James Kraatz | Initial submission |

**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 does not have experience in developing web-based applications and what to release their current Android application, Draw It or Lose It, as web-based multi-platform game. Draw It or Lose It will be played via web on multiple new platforms like Mac, Windows, Linux, and possibly iOS. The application will have to have a single code base to accommodate these different platforms and their different hardware. Java is a great code candidate to develop a single code base providing a common application to all four new platforms along with common IDE choices that can be used on at least 3 of the 4 new platforms. Representative hardware of each platform supported will be needed for proper testing and verification of the game application.

## [Design Constraints](#_2et92p0)

In order for Draw It or Lose It to be a successful web-based application, it will have to be a common program distributed from a web-server that many different types of hardware, having different operating systems, will be able to run. With Java chosen as the development language, this will require the Java Runtime Engine to be installed on any intended client wanting to run the application. The game will require a constant network access with the internet and will need to be developed to overcome or bypass different internet connection speeds and timing to be fair to all systems playing. Since the application will be web-based, consideration needs to be given to who has access to the application, one option would be to put the server behind a protected firewall or other protection mechanism. Images, answers, user-scores and other information will need to be in storage on the web-server. During application runtime, games scores, information will need to be updated locally during gameplay.

## [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 encapsulates two data points and methods to access the data that is used by three derivative classes Game, Team, and Player. GameService is a Singleton object where only one GameService can load into memory at a time. GameService encapsulates a list of Game objects and provides methods to add and retrieve Game objects to the list and provide the new ID’s for new Team and Player objects. Game inherits from the Entity class and encapsulates a list of Team obejcts along with method to add a Team to the list. The Team object also inherits from the Entity Class and encapsulates a list of Player objects and provides a method to add players to the Player list. The Player object inherits from the Entity class as well. The GameService is associated with zero to many Game objects, each Game object is associated with zero to many Team object and each Team object is associated with zero to many Player ojbects. Lastly, ProgramDriver is used to call the SingletonTest object to test the GameService object.

****

## [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** | -Most Secure  -Highest cost  -Best uptime  -Does not run ASP, -.NET, ACCESS, MSSQL | -Secure  -Least cost  -High uptime  -Does not run ASP, .NET, ACCESS, MSSQL | -Least secure  -Mid-cost  -Frequent updates impacting server uptime.  -runs ASP, .NET, ACCESS, MSSQL | -Expensive hardware  -Limited hardware capacity  -Expensive and limited connection cost.  -Not viable for web-based game serving platform |
| Client Side | -Common Java Platform  -Common development software  -Potential closed hardware not supported | -Open source and FREE OS  -poor driver support potentially needing extra in-house development  -Common development software | Common Java Platform  -Common development software | -Common Java Platform  -Screen sizes different  -Extra development tools needed |
| **Development Tools** | -Homebrew  -Oracle Java Development Kit  -Cask  -Maven  -Netbeans  -jEnv  -git | -Oracle Java Development Kit  -Netbeans  -Maven | -Oracle Java Development Kit  -Netbeans  -Maven  -git | -Oracle Java Development Kit  -Netbeans  -JavaFX (IOS)  -Maven  -Xcode (IOS)  -git |

## 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**: The recommended platform for The Gaming Room GameService is Java running on either a OSX or Linux platform serving Java applications for their web-based games. It is believed that the OSX system is more secure, although more expensive, and both systems benefit from infrequent needs to be rebooted. Windows platforms are powerful and is the only platform to offer access to some technologies like .NET but suffer from frequent and uncontrolled updates from Microsoft increasing down-time to The Gaming Rooms product offerings.
2. **Operating Systems Architectures**: All three operating systems support secure directory services such as LDAP. OSX is a closed operating system that natively runs on Apple computer systems. The term “Closed” describes the fact that the source code to the OS is not available from its manufacturer Apple. If any problems arise between the operating system and the hardware the manufacturer will be the only ones to rely on for a solution. Linux, on the other hand, is an open software and benefits from a large community of support for operating system issues. Windows, from Microsoft, is also closed and like OSX, requires any patches to come from the manufacturer. All three platforms have Java Run Time Environments available for them that allow code written in Java to run on them. Care has to be made to avoid using any platform specific code or, if writing platform specific code, to write code to accommodate when the platform is different.
3. **Storage Management**: Storage management should be a cloud or network storage connected to the web-server for account histories, game details, and game applications. The local client systems should dynamically get game details and applications and the users details as needed from the server once logged on to the service. If games storage requirements get too large or to reduce network traffic during game it is possible to download encoded static game data to the local client and allow the client to decrypt and apply game data as needed. This would require the client to update when new releases or updates are available
4. **Memory Management**: Memory paging will be used to move data and code in and out of memory on demand. Both Linux and OSx use demand paging to manage memory allocation between active memory and secondary storage.
5. **Distributed Systems and Networks**: As a distributed application, the software will be asynchronously messaging and non-blocking. At no time can the server be stopped waiting for response from one client due to any loss of connectivity. Also, the application itself will need to be always responsive.
6. **Security**: All private data will be encrypted before being sent between client and server. All locally stored data files will also be encrypted.