

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 05/22/2021 | Michael Jordan | First version. Filled out the executive summary, design constraints and domain model. |
| 2.0 | 06/04/2021 | Michael Jordan | Adjusted executive summary and design constraints.  Filled out the evaluation. |
| 3.0 | 06/20/2021 | Michael Jordan | Filled out recommendations. |

**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 client, The Gaming Room, is looking to adapt their current Android-only game to a web-based game that is playable on multiple platforms. This is a team-based game where the teams guess what an image is based on provided stock drawings that are slowly rendered. We have been asked to prepare a design document and begin development on the application.

These are the requirements as I understand them:

* 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.

All of these can be fulfilled using a client-server approach.

## [Design Constraints](#_2et92p0)

The server must track and restrict creation of games and teams.

There must be sign-up and login mechanics to keep track of player identity.

Testing will be needed for the game running on web browsers on different platforms.

The gameplay of the web-based application should be modeled after the gameplay of the existing Android application.

## [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 are extended from the Entity class, allowing them to inherit properties that each of them will need and saving development time. The Entity class also uses encapsulation to ensure that the unique identifier and name variables cannot be accessed directly. The Team class can contain 0 to several instances of the Player class, fulfilling the requirement that each team has multiple players assigned to it. The Game class can contain 0 to several instances of the Team class, fulfilling the requirement that a game can have one or more teams involved. The GameService class can contain 0 to several instances of the Game class. Only one instance of the GameService class may exist, and it generates unique identifiers for all Games, Teams and Players created by utilizing encapsulation to prevent direct access to the identifier variables. This fulfills the requirement that only one instance of the game can exist in memory at any given time. The diagram also presents examples of polymorphism with the overloaded getGame methods in GameService, as well as the overloaded toString method in each sub-class of Entity. The ProgramDriver is the entry point of the program that starts the GameService, and it uses the SingletonTester class to ensure the GameService class is properly implemented.

****

## [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** | Con: Systems are expensive  The web server software Apache is packaged with the OS. It is also compatible with other options including NGINX. | Pro: Minimal GUI allows efficient use of resources  Pro: No licensing costs  No web server software is packaged with the OS. It is compatible with Apache and NGINX. | Con: Significant licensing costs  Pro: Large pool of support staff  The web server software Internet Information Services is packaged with the OS. It is also compatible with Apache and NGINX. | Con: Wireless connectivity only  Con: Hardware performance is limited; will not scale well  Pro: Systems are cheap  Several web server software options including kWS, Apache and NGINX are available for Android devices. The options for iOS devices are limited. |
| **Client Side** | The testing costs for this platform may be higher since it comes as a package with the hardware.  Cross-platform languages such as JavaScript remove many of the platform-specific development concerns. | Cross-platform languages such as JavaScript remove many of the platform-specific development concerns. | Cross-platform languages such as JavaScript remove many of the platform-specific development concerns. | A specialized UI is necessary for mobile device clients. Client code may also have to be specialized to account for mobile browser features. |
| **Development Tools** | Java IDEs: Eclipse, NetBeans, IntelliJ IDEA, etc.  JavaScript IDEs: VSCode, Atom, WebStorm, etc.  HTML IDEs: VSCode, Atom, NetBeans, etc.  Tools: JDK, Node.js, Git/SVN, Maven/Gradle/Ant  IntelliJ Idea and WebStorm have associated licensing costs, the rest do not. | Java IDEs: Eclipse, NetBeans, IntelliJ IDEA, etc.  JavaScript IDEs: VSCode, Atom, WebStorm, etc.  HTML IDEs: VSCode, Atom, NetBeans, etc.  Tools: JDK, Node.js, Git/SVN, Maven/Gradle/Ant  IntelliJ Idea and WebStorm have associated licensing costs, the rest do not. | Java IDEs: Eclipse, NetBeans, IntelliJ IDEA, etc.  JavaScript IDEs: VSCode, Atom, WebStorm, etc.  HTML IDEs: VSCode, Atom, NetBeans, etc.  Tools: JDK, Node.js, Git/SVN, Maven/Gradle/Ant  IntelliJ Idea and WebStorm have associated licensing costs, the rest do not. | Java IDEs: AIDE (Android), no effective options for iOS.  JavaScript IDEs: DroidScript, AIDE Web (Android), play.js (iOS)  HTML IDEs: DroidScript, AIDE Web (Android), play.js (iOS)  Tools: JDK, Node.js, Git/SVN  AIDE and AIDE Web have associated licensing costs, the rest do not. |

## 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**: Mac, Linux and Windows would all be effective choices, but I would recommend using Windows as the server platform, as the pool of support staff will likely make it more cost effective over time.
2. **Operating Systems Architectures**: The Windows Server OS is part of the Windows NT family, which uses a layered design for its architecture. The first layer is the “user mode”, where applications (including the Draw It or Lose It server software) run. This layer has limited access to system resources and is unable to directly access the hardware of the system, relying on calls to the “kernel mode” to affect the hardware. The second layer is “kernel mode”, which has full access to the hardware and system resources. This layer handles scheduling, thread prioritization and memory management, as well as requests from the “user mode” layer.
3. **Storage Management**: A local storage system would be effective for this program, considering its storage needs. For user data that grows over time, a system that transfers it to a secondary location may be necessary.
4. **Memory Management**: Windows utilizes a virtual memory system for applications and system processes, with the hardware automatically translating virtual addresses to physical addresses. It also uses a page file system to assist with memory management, where pages of virtual memory addresses can be temporarily stored on disk if memory usage exceeds available physical memory. With current hardware, this is unlikely to be necessary for the Draw It or Lose It software, as holding a unique copy of each image in memory will only require 1.6GB. Assuming other portions of the software require memory adding up to another 1.6GB, that leaves at least 4GB of memory for the OS and other programs on the server hardware.
5. **Distributed Systems and Networks**: For the least complexity, the client application for each platform only needs to communicate with the server application, while the server application will need to communicate with every client application. With this approach, the Draw It or Lose It software is completely dependent on the server being available. However, a client losing connectivity to the server will not cripple any games the client may be in. Communication between the server and client software will be accomplished using standard internet protocols, as the intent of the software makes other network options unfeasible.
6. **Security**: The server platform will need to have firewall and anti-virus software installed to mitigate outside attacks. Communication between the server and clients should be encrypted when possible. Accounts should be set up server-side with limited capabilities to mitigate risks from physical access. If user accounts are used for authentication, then account information should also be encrypted when stored.