

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)4

[**Recommendations**](#_m8aleynsvzvc)7

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 3.0 | 04/17/21 | Mark Meyer | Implementation of simultaneous multiple-team, multiple-player gameplay with unique game, team, and player names. Only one game instance allowed. |

**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 software design problem involves a game of four one-minute rounds where teams compete to guess what is being drawn by the application each round. The client requests to streamline the ability to play each game with multiple teams with multiple players on each team. Only one game should exist at any given time in memory and all game, team, and player names should be uniquely identified.

## [Design Constraints](#_2et92p0)

The design constraints for developing the game application in a web-based environment include but are not limited to designing a user-friendly interface that is closely mirrored across multiple platforms with minimal bugs, optimal network, application speed as well as exceptional functionality. Developing a cross-platform application often requires a bigger budget, longer development time, and added resources for added planning, designing, implementation, documentation, testing, deployment, and maintenance. This often results in unforeseen technical and/or business constraints.

## [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 UML class diagram below conveys the class structure and object-oriented principles of the proposed web-based application. The “Entity” class is the parent class of the “Game”, “Team”, and “Player” child classes such that when any one of these child classes creates an object, that object inherits the attibutes, methods, and constructors from “Entity”; in other words, each object of any of these child classes becomes a unique “Entity”. This means that each child class has an “is a” relation to the parent class, but each child class possesses a “has a” relation with each other such that a “Game” has a “Team” has a “Player”. Additionally, each class practices encapsulation by protecting (ie.) unique attributes such as the “Entity” id and name, list of games, list of teams, list of players , or a game service instance. Lastly, the application uses polymorphism because each child class defines its own ‘toString()’ method which grabs the unique identifier from the inherited “Entity” class to output that information to the user. Utilizing object-oriented principles protects player privacy and allows for consistently unique identifiers.

****

## [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** | Costly.  User-friendly GUI for users and developers alike. Fluid workstation with relatively easy accessibility and server configurations. | Cost effective.  Difficult to navigate for users or devs who are used to GUI platforms. Linux uses command shell similar to Mac for ease of accessibility and server configurations. | Can be costly.  User-friendly GUI and popular. Software options for Windows are in abundance and generally well-supported. Has command shell for lower level configurations. | Inexpensive.  Mobile device specifications vary for each user. Potential challenge in developing a game that is compatible with all mobile platforms, but majority are Android or iOS. |
| **Client Side** | Mac has a similarly high cost to Windows, and typically always more than Linux or Mobile. Support for Mac users requires an average time but adequate skill. | Linux has minimal cost. However, support for Linux users requires high level of expertise with Linux and a maximum allowable time for troubleshooting. | Windows has a similarly high cost to Mac users, but the support for Windows users requires minimal technical skill and minimal time to troubleshoot. | Potentially high costs for support as the difference in challenges device-to-device can get very proprietary, so the tech will generally need to be well-versed across several mobile platfforms. Maximum time should be considered. |
| **Development Tools** | Programming languages for Mac include JavaScript, Java, Objective-C, C/C++, Ruby, and Python. Other languages are used as well for things like markup and styling such as HTML5, CSS, and XML. Common IDEs/tools for Mac include Git, Github, Eclipse, IntelliJ by JetBrains, CLion, Visual Studios, Xcode, and Android Studio. | Programming languages for Linux include C/C++, Java, JavaScript, Python, Ruby, and Bash.  Other languages are used as well for things like markup and styling such as HTML5, CSS, and XML. Common IDEs/tools for Linux include Git, Github, PyCharm, Eclipse, NetBeans, QT Creator, IntelliJ, and Eric. | Programming languages for Windows include  C#, C/C++, Java, Python, JavaScript, and Objective-C. Other languages are used as well for things like markup and styling such as HTML5, CSS, and XML. Common IDEs/tools for Windows include Git, Github, Visual Studio, NetBeans, Eclipse, Code::Blocks, and IntelliJ. | Programming languages for mobile development include Python, JavaScript, Java, C/C++, Scala, Kotlin, PHP, C#, Objective-C, Ruby, Rust, and Lua. Other languages are used as well for things like markup and styling such as HTML5, CSS, and XML. Common IDEs/tools for mobile development include GitHub for Mobile, IDE For Android, Android Studio, IntelliJ, Visual Studio, and Xamarin. |

## 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**: I recommend Windows as an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments. Windows is a very popular platform that allows streamline accessibility, a user-friendly interface, a copious and versatile offering of software packages, and a wide variety of software engineering tools for developers.
2. **Operating Systems Architectures**: Windows supports x86 and x86-64 architechtures for 32-bit and 64-bit platforms respectively. X86 is heavily supported in the community as well as by many major brand hardware manufacturers for both Intel and AMD. Windows offers extensive support for a number of programming languages, software packages, and virtual machine support so developers can open numerous different types of operating systems at once. Server configurations are manipulated from PowerShell, Window’s command line. Windows is also very quickly and seamlessly deployable to multiple users/developers.
3. **Storage Management**: Windows offers free, but upgradeable, 5 GB of cloud storage with every license purchase. Windows also offers a simple configuration interface for partitioning and formatting storage as well as exceptional RAID software support for backup redundancy. I would recommend using a solid state drive (SSD) for efficient data reading, writing, and retrieval operations. An SSD with M2 and/or NVMe support is ideal for optimal speed but not necessary.
4. **Memory Management**: Window utilizes virtual and physical address space for up to 4 GB of memory for 32-bit platforms and for up to 8 TB of memory on 64-bit platforms. Windows allows its virtual address space to be accessed by all threads of a process; however, processes belonging to one thread cannot be accessed by processes of a different thread. This holds the interest of protection from process corruption which assists in Windows applications running smoothly.
5. **Distributed Systems and Networks**: Network games that require connectivity of multiple user interactions typically share a physically-distributed database that communicates and updates over the network. Network developers must create this communicative architecture from scratch and often run into routing, queuing, and congestion issues. Although these systems offer easy coordination and connectivity, the distributed systems may also experience failed hardware components as well as computational redundancy which may cause hardware and/or connection lag.
6. **Security**: Windows OS requires a license purchase but comes equipped with a well-supported, simplistic GUI design for users and developers alike. Windows provides user account control and antivirus software to assist with securing data. Windows support VPN and SSH service capabilities to optimize security when having to use remote sessions and/or virtual networks. Also, Windows support Java development which is an ideal programming language to use when security is a concern because of its inherent ability to clean up objects from memory after they’ve been used. This deters unwanted memory leaks and promotes an optimal, secure performance.