

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 | 11/13/2021 | Byron D. Staton | Created an executive summary, listed design constraints, described domain model |
| 2.0 | 11/26/2021 | Byron D. Staton | Updated the development requirements section |
| 3.0 | 12/10/2021 | Byron D. Staton | Analyzed the characteristics of and techniques specific to various systems, made a recommendation |

**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’s popular game, *Draw It or Lose It*, is only available for Android devices at present. This title is loosely based on the popular 1980s television game show *Win*, *Lose*, *or Draw* where contestants compete to guess what is being drawn. The objective is to develop a web-based version of the game that can be accessed by users on various platforms.

## [Design Constraints](#_2et92p0)

The current version of the app runs on the Android platform. We will be using the Java programming language to develop the web-based version of the app. Since the app is web-based, it will need to be compatible with popular web browsers such as Apple Safari, Google Chrome, Mozilla Firefox, Microsoft Edge, and more. The app involves teams so performance can be affected.

## 

## [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 holds common attributes and methods. It creates a relationship with the Game class, Team class, and the Player class (child classes). The three child classes inherit the attributes and methods found in the Entity class. A “HAS-A” relationship occurs between the Game, GameService, and Team classes. Another “HAS-A” relation occurs between the Team and Player classes. The “HAS-A” types are aggregation relationships. Multiplicity is represented by the “0…\*” characters between each class, which means there may be zero to many instances of each object.

**"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** | **Advantages**  Mac can be excellent for smaller user environments. It has fewer technical constraints than Windows.  Mac offers the macOS Server formally known as OS X Server and can be installed on single Mac computer.  The cloud can be used to run a game server.  **Disadvantages**  It is difficult to integrate into larger user environments.  The potential licensing costs are high. | **Advantages**  Linux uses open-source software which drives down expenses. It’s safer than other OS.  Linux is transparent and typically provides concrete solutions to errors.  Deploying to Linux is supported for .NET Servers or the open-source Mono runtime which is free.  Cloud-based servers are available.  **Disadvantages**  The file system is susceptible to trojan horses.  There is no standard server edition for Linux.  Linux servers have poor support for games. | **Advantages**  Windows has a wide range of available software.  Many users are more comfortable with using this OS.  This OS offers the Windows Server platform.  The cloud can be used to run a game server.  **Disadvantages**  Windows is more susceptible to viruses.  Windows Server has high licensing costs.  If server hardware is needed, then costs are expensive. | **Advantages**  Frameworks for mobile help boost performance and promote scalability.  The mobile OS offers a mobile web server software for smartphones known as I-Jetty.  macOS Server works with iPhone devices.  **Disadvantages**  Generally, it requires higher costs, especially for set up processes.  I-Jetty is only available for Android devices.  The potential licensing costs are high, especially for devices with a pixel density higher than 500 ppi. |
| **Client Side** | Mac can be an expensive option.  Moderate time is required to set up projects.  Moderate expertise is required.  Developer tools built into the browser allows a developer to check responsiveness based on the type of device (iPhone, iPad, laptop, desktop). Using a cloud-based platform like BrowserStack Live allows a developer to test cross-browser compatibility with the use of real devices. | Linux comes with minimal costs to users.  Linux users may need higher proficiency to support Linux systems.  Linux requires more time to set up projects.  Using a cloud-based platform like BrowserStack Live allows a developer to test cross-browser compatibility with the use of real devices. | Windows can be an expensive option.  Minimal time is required to set up projects.  Minimal expertise is required but can have more roadblocks than Mac/Linux.  Using a cloud-based platform like BrowserStack Live allows a developer to test cross-browser compatibility with the use of real devices. | Mobile development can be expensive depending on complexity.  The average time is four to six months but could be longer.  Higher expertise is needed for implementation.  Capabilities native to a device may differ on another device if building a native app so developing a hybrid app is the better choice. |
| **Development Tools** | Programming languages include Swift, Objective-C, C/C++, Java, Python, PHP, Ruby on Rails, HTML, CSS, JavaScript  IDEs include Xcode, Eclipse, JetBrains, Atom  Other tools include Git, SQL, MongoDB  There are licensing costs associated with JetBrains IDEs.  Swift only supports iOS and macOS. | Programming languages include C/C++, Java, Python, Ruby on Rails, HTML, CSS, JavaScript  IDEs include Eclipse, JetBrains, Atom  Other tools include Git, SQL, MongoDB  There are licensing costs associated with JetBrains IDEs. | Programming languages include C/C++, Java, Python, Ruby on Rails, HTML, CSS, JavaScript  IDEs include Eclipse, JetBrains, Atom, Visual Studio  Other tools include Git, SQL, MongoDB  There are licensing costs associated with JetBrains IDEs. | Programming languages include Swift, C#, C++, Java, GO, Python, Ruby, HTML, CSS, JavaScript  IDEs include Android Studio, Visual Studio, Eclipse, Xcode, JetBrains  Other tools include Git, SQL, MongoDB  There are licensing costs associated with JetBrains IDEs. IntelliJ can be used for developing Android applications.  Android Studio only supports Android devices. |

**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**: Linux is an excellent choice for a server platform for the application. It is a free, open-source operating system so the source code can be modified to your server needs and be redistributed so long as it is done under the GNU license. Linux is the most secure OS available and because it is open-source, contributors of the software are always making sure there are no bugs or back doors. It can also be run on a variety of systems including personal computers, mobile phones, and tablets.
2. **Operating Systems Architectures:** The Linux OS contains a kernel, a hardware layer, a system library, shell, and system utility. The kernel is made up of core components that are responsible for all the major activities. The system library is made up of functions that implement the functionality of the OS and does not access the kernel directly which adds a level of security to the system. Utility programs handle individual and specialized-level tasks. The hardware layer consists of the peripheral devices in the system. The shell can either be a command line or graphical and it accepts user commands and executes the kernel’s functions.
3. **Storage Management**: Linux supports a variety of file systems because each one solves a different kind of problem, but the most used ones are the ext4, ext3, ext2, xfs, btrfs, and gfs2. The file system’s job is to control how data is stored and retrieved. The ext4 filesystem is backward compatible with legacy systems and stable which is good for the application since it will run on various systems. A better option for the web application is cloud-based storage. Instead of storing the application’s data on a user’s local storage, it can be stored at a remote location which can be accessed via an internet connection. Cloud storage is secure, cost-efficient, and scalable to say the least.
4. **Memory Management**: The application will use physical memory in a user’s system called random access memory or RAM. Any data or instructions stored in RAM will be in close proximity to the web server, and the CPU. This approach will enhance speed and performance. When the memory is no longer needed, it will be freed and reused again later.
5. **Distributed Systems and Networks**: A distributed system or application is broken up into two different programs. The first program contains the client software and the other contains the server software. Distributed applications run on multiple systems simultaneously for a single job. These apps can communicate with multiple devices and servers on the same network from anywhere in the world. This will be a RESTful application where the client, the browser, will request resources from the server over an HTTP protocol. It may be worth noting that the server is not concerned with the UI or user state and the client is not concerned with data storage.
6. **Security**: Linux is arguably the most secure OS. It is open-source and contributors to the OS are always ensuring that there are no bugs or back doors. Each process in the system has an independent virtual address which prevents different processes from affecting other processes. There is a need to explore common security vulnerabilities and establish certain requirements. The application should only accept a strong password upon a user’s registration. The information used to create their account should only be sent over an HTTPS connection. To further protect user data, multifactor authentication may be implemented.