

Draw It or Lose It- Multiple Platform

# **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 | 03/19/2022 | Jennifer Wells | Executive Summary, Design Constraint Guidance, Model Explanation and 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 develop a web-based game that serves multiple platforms that is based on their current game, Draw It or Lose It, which is currently only available as an Android app.

Client requests that:

-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.

## [Design Constraints](#_2et92p0)

-Transmission delays/failures require special accomodations.

-Scalability of the application system should remain efficient even with a significant increase in the number of users and resources connected.

-Objects that represent a shared resource must be ensured to operate correctly in a concurrent environment. For example, stock drawings as clues.

Using distributed systems seem like a way to improve performance and availability by splitting the load across multiple processors. However, communication of these processors through a medium which may cause delay and exceed the time between state changes. Since we must assume that each step will fail, we must ensure that code will behave correctly in light of those failures.

To address scalability of the application, we will need to regularly perform performance tuning. We will need to create a list of very specific functional requirements as well as increasing cache on the outside of the web app(keeping most requested “writes” information upfront, increasing performance), which is less invasive and requires less time and budget to be conducted successfully.

We want our shared resource to operate concurrently in a separate state. To achieve this, we need to choose a concurrency model that works best with the requirments set by the client.

## [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)

In this UML Diagram, we can see that the Game, Team, and Player classes inherit from the Entity class.

We see that that for GameService, Game, Team and Player that every object in each of these classes has an association with at least 0 or more objects within these classes.

The program driver contains the main() method and uses the SingletonTester so that we only have one instance of GameService in memory.

We see encapsulation in that the Entity superclass, GameService, Game and Team have private attributes.

**"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** | -Top of the line hardware  -Costly upfront costs for hardware + licensing (from $450-$999)  -most reliable  -negligible risk for security threats  -Hosts run Apache servers similar to Linux which allows basic web code  -limited compatibility | -Open source  -Stable  -High processing power  -allows for compatibility  -runs on Apache servers and can run basic web code  -user friendliness depends on distribution  -Most affordable  -uses less resources  -may require software add-ins  -limited availability of Linux hardware | -best for running applications like ASP, .NET, Microsoft Access, MSSQL databases.  -Easy to use  -prone to blue screens and crashes  -huge risk of security threats  -Can coexist on local networks with Windows, BSD, Macs, Unix-like systems  -includes support for Windows-friendly/Microsoft-based technologies  -moderate licensing cost ($501-6,155, depending on size.) | -minimal programs and algorithm simulator applications for android devices  -time-consuming and inefficient  -difficult coding and debugging process  -low cost  -allows for compatibility  -negligible risk of security threats.  -potential for unreliability.  -not as powerful and scalable as other devices. |
| **Client Side** | -Unix based OS designed with programmers in mind  -built-in terminal capable of handling many command line tasks  -programmers receive complete toolkit which allows them to take full control of the operating system  -very little extra setup needed  -allows for much accessible cross-platform development  -much more expensive, but much more user friendly for non-developers as well as time saving. | -free and open licensed  -very customizable and developer friendly  -flexible  -easy to install  -not very user friendly or having a “straight out of the box” experience.  -takes lots of time to learn  -there can be issues with older hardware | -requires extra setup for functionality similar to macOS.  -would need to install a virtual machine to run macOS  -support for a wide range of hardware  -cheaper development costs  -popular and familar interface which is user-friendly. | -development would take much longer  -debugging, coding will be time consuming  -difficult to use  -cross-platform frameworks may lack features that users enjoy  -HTML5 is compatible with many platforms and may require minimal changes for complete functionality on each OS.  -prices can range from $5,000-$15,000, plus additional costs, but is only worth it for a worthwhile app.  -failure to keep up with development updates can cause cybersecurity issues and can also be delisted for incompatibility issues. |
| **Development Tools** | -React Native  -C, Swift, Java, .NET  -C#, F#, Python, Razor Engine  -Flutter SDK  -Dart Virtual Machine  -Ionic SDK (HTML5 SDK) | -Eclipse  -Java, C, C++, Perl  -Visual Studio Code  -Elixir | -Python, Java, JavaScript, Ruby, Perl, .NET core, C#, HTML,  -Visual Basic  -Netbeans | -Java, Kotlin, C++, C#, Python, HTML, CSS, JavaScript  -React Native  -Flutter SDK  -Ionic SDK  -Android studio(one time $25 fee)  -Swift  -Netbeans |

## 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**: Apple Mac OS.
2. **Operating Systems Architectures**: The structure of the Mac OS includes multiple layers. The base layer is Darwin which is the Unix core of the system. The next layer is the graphics system which contains Quartz, Open GL, and QuickTime. Then there is the application layer: Classic, Carbon, Cocoa and Java. Then we have the top layer. The user interface, Aqua.
3. **Storage Management**: Amazon Elastic Compute Cloud (EC2)
4. **Memory Management**: The operating platform will consistently need to access objects in a shared resource used by multiple computing devices on a network. As well as increasing a cache on the outside of the web app for this application. This operating platform divides available RAM into two broad sections: It reserves a zone or partition of memory known as System partition. The system partition always begins at the lowest addressable byte of memory and extends upward. The system partition contains a system heap and a set of global variables. All memory outside of the System partition is available for allocation to applications or other software components. When an application is launched, the OS assigns it to a section of memory known as its Application partition. An application uses only memory contained in its own application partition.
5. **Distributed Systems and Networks**: Distributed systems are computing environments where various components are spread across multiple computing devices on a network. The processing load is split up among these devices, coordinating their efforts to complete a job rather than a centralized system. Using a distributed system can increase processing power, speed and increase availability for users. These systems reduce the risks involved with having a single point of failure, increasing fault-tolerance and reliability. However, each computing device is considered a potential increased opportunity for failure. Also, challenges can arise with synchronization processes and network failure, requiring careful programming to avoid transmission delays that can result in errors and data corruption in a complex system such as a multiplayer video game.
6. **Security**: Security vulnerabilities depend on the platform and how well the code is written. The security risk can be higher with cross-platform applications because of the added vulnerabilities that affect native code and web browsers. To secure the user information, we would need to employ the use of tools to control the services that are exposed to the web using load balancers, firewalls, VPNs as well as encryption. We could also employ a verification method for users, as well as ensuring regular updates and maintenance. Mac OS has built-in security features to counteract potential attacks such as: Encrypted disk images, FileVault, Encrypted Virtual Memory, Secure Empty Trash, Secure Erase, Application signing and firewall, Application/service sandboxing /tagging/quarantine along with the added benefit of anti-virus and anti-malware services.