

**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 | 10/01/22 | Connor Ebel | Creation of document |
| 1.1 | 10/08/22 | Connor Ebel | Revision of document |
| 1.2 | 10/16/22 | Connor Ebel | Revision and finalization of document |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is inquiring about the development of a web-based game, based on their current Android-only game app Draw It or Lose It. They, The Gaming Room, want the game to serve multiple clients and platforms. They have requested the game has one, or more, teams involved in a game, each team in a game have multiple players, each game and team name must be unique (and allow users to check for name availability), and for only one instance of the game existing in memory at any given time (Okere, 2022).

## [Design Constraints](#_2et92p0)

A primary concern for design constraint regarding the development of Draw It or Lose It is the ability for multi-platform use, aside from their current platform Android. Another concern is the use of authentication methods for security of user information and platform security. Finally, storage and memory management; cloud storage, as a possible method, and speeds by which images might be rendered to maintain a good user experience between platforms.

## [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)

There is an inheritance relationship between Entity, as the parent, and Game, Team, and Player as child classes. Game, Team, and Player inherit “id” and “name” from Entity. This method of inheritance from Entity satisfies the third customer requirement of game, team, and player names, and IDs, being unique and checking for availability for each upon creation of a game (Okere, 2022).

The associative relationship between GameService, Game, Team, and Player are shown in the UML below. The associative relationships between Player and Team, Team and Game, Game and GameService are all “zero to many”, meaning no players to multiple players can join a single team, no teams to multiple teams can join a single game, and GameService can have zero to many games active. This relationship helps satisfy the customer’s first, second, and fourth requirements of a team being able to have multiple players, a game can have multiple teams, and only one instance of a game existing in memory at any given time (Okere, 2022).

**"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** | **Kernel**: UNIX kernel with own proprietary features. Low level task management, faster processing speeds.  **Licensing**: Proprietary  **Security Features**: FileVault automatically encrypts on disk, Firmware password protection, Mac Firewall, and built-in antivirus software (gizmogrind, 2022). | **Kernel:** UNIX Kernel. Low level task management, faster processing speeds.  **Licensing:** Free and open-source  **Security Features**: File Permissions (permissions to use files are assigned on class basis), Data Encryption/Decryption, no built-in antivirus software (Killi, 2022). | **Kernel:** Windows NT kernel. Low level task management but processing speeds slows down over time.  **Licensing:** Proprietary  **Security Features:** Built-in anti-virus software and customizable security features (on firewall, account protection, and device performance) (Microsoft, n.d.). | **Kernel:** Very fast processing speeds, but, in time, iOS will slow down much faster than Android.  **Licensing:** Proprietary for both  **Security Features:** Both Android and iOS have very strong security measures, no need for antivirus software. |
| **Client Side** | Macs have compatibility with majority (popular) web browsers on this platform.  **Popular Browsers:** Safari, Chrome, and Firefox | Linux has compatibility with majority (popular) web browsers on this platform.  **Popular Browsers:** Chrome, Opera, and MS Edge | Windows has compatibility with majority (popular) web browsers on this platform.  **Popular Browsers:** Edge, Chrome, and Firefox | Mobile devices have compatibility with multiple browsers.  **Popular Browsers:** Chrome and Firefox |
| **Development Tools** | **Programming Languages:**  Most popular is Swift. HTML/CSS/JavaScript for front-end work.  Java, Python, and Ruby for back-end work.    **Tools:**  Notepad++  **IDEs:**  AppCode (for Swift), Netbeans for back-end work, and Atom for front-end work. | **Programming Languages:**  A popular language for Linux use is Java. HTML/CSS/JavaScript for front-end work.  Java, Python, and Ruby for back-end work.    **Tools:**  Notepad++  **IDEs:**  Netbeans for back-end work, and Atom for front-end work. | **Programming Languages:**  Most popular is C# for App Development in Windows. HTML/CSS/JavaScript for front-end work.  Java, Python, and Ruby for back-end work.    **Tools:**  Notepad++  **IDEs:**  Visual Studio for back-end work, and Atom for front-end work. | **Programming Languages:**  Most popular are Kotlin and Swift. HTML/CSS/JavaScript for front-end work.  **IDEs:**  Android Studio for Kotlin, Atom for Swift, and Eclipse for back-end work . |

## 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**: Due to ease of use, Windows would be a preferred, initial, platform for Draw It or Lose It. There is a multitude of languages, IDEs, libraries, and tools to work with on Windows. Overall, Windows is highly compatible with the vast majority of software/programs available today.
2. **Operating Systems Architectures**: Windows has a low-level kernel which makes for fast processing speeds, but the system is known to slow down over time with subsequent updates. While licensing through Microsoft is proprietary, due to the huge presence of Microsoft, licensing is quite easy to obtain.
3. **Storage Management**: Windows utilizes a built-in storage management system. The user may elect to dictate the location files are saved, preventing loss of files. The storage management system in Windows machines allows for easy file creation, access, and deletion as well as allocation.
4. **Memory Management**: With the allowance of user dictation for file locations, on Windows, the user may save projects and other important aspects, such as images, to specific, and secure, locations on their local machine.
5. **Distributed Systems and Networks**: Strong server network is required as the game is highly dependent upon the connection between multiple users to one server. End-user applications will highly depend on the one server application of the game.
6. **Security**: The Windows platform comes with built-in security software called Windows Defender but is known for not being highly efficient. For true security, The Gaming Room may elect to utilize third-party, such as Cisco or Norton, to truly secure the transfer of information between the network itself and transfer of user information. Some practices The Gaming Room can use with active, or potential, security threats is the prevention of the use of the application until updates are completed once they’ve gone live.

# **References**

gizmogrind. (2022, 03 15). *10 macOS Security Features You Need to Know*. Retrieved from gizmogrind.com: https://www.gizmogrind.com/blog/10-macos-security-features/

Killi, A. (2022, 06 17). *7 Useful Linux Security Features and Tools*. Retrieved from tecmint.com: https://www.tecmint.com/linux-security-features-and-tools/

Microsoft. (n.d.). *Stay Protected with Windows Security*. Retrieved from support.microsoft.com: https://support.microsoft.com/en-us/windows/stay-protected-with-windows-security-2ae0363d-0ada-c064-8b56-6a39afb6a963

Okere, L. (2022, 09 28). *Project One Grading Completed*. Retrieved from snhu.edu: https://learn.snhu.edu/d2l/le/news/1158571/5088047/view?ou=1158571