

Draw It Or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/23/2025 | Summer Stinnett | Initial version of document. Provides project summary, requirements/guidelines, design constraints, system architecture overview, UML diagram for documentation, evaluation of platform considerations, and recommended approach to completion of project |
| 1.1 | 04/06/2025 | Summer Stinnett | Updated draft to refine verbiage and include further elaboration |
| 1.2 | 04/20/2025 | Summer Stinnett | Updated document to expand upon the recommendations section |

**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)

This document will detail the proposed solution for the implementation of a web-based port of The Gaming Room’s game, Draw It Or Lose It. Currently the game is an exclusive Android application. The game consists of a number of teams that play in a series of four rounds. During each round a team will be given 30 seconds to guess a drawing that is rendered on their screens. If they fail to do so, another team will be able to steal it but only get 15 seconds to guess instead. The client wishes to produce a product in a manner that accurately reflects their vision with this game to enable an enjoyable interactive experience for their end users. Detailed below include things such as a reiteration of requirements, an explanation of design constraints for client consideration, and an architecture overview to facilitate conversation on how the product is built. We will also provide UML diagrams and descriptions that demonstrate a glance into the software development design, compare platform-relevant considerations, and lastly, provide a list of recommended implementation specifications that we feel would allow for an ideal product.

## Requirements

* Business Requirements
  + Create a feature-complete adaptation of the existing game Draw It Or Lose It
  + Ensure design/aesthetic is familiar to players of original game
  + Ensure development is cost-effective and efficient
* Technical Requirements
  + Ability to run games with one or multiple teams in a game that enforce unique names
  + Ensure only having a single instance of the game running per client/user at a time
  + Ensure web-based compatibility across platforms

## [Design Constraints](#_2et92p0)

* Applications need to adhere to all listed application security / design guidelines recommended by each platform
  + Proper auditing of protocols may need to be performed to ensure user data is secure
* Design of the game should be familiar to users coming from the existing Android application
  + Users coming over from the existing application may find it jarring if the design is vastly different from what they are used to
* UI should be intuitive and responsive; players should know what buttons on the screen do
  + Users may be turned away from the game if they are performing actions they do not intend to do or have actions that are not being performed due to a bad implementation. Application should use modern, practical frameworks in its development process
* Images used in the game should not infringe on copyrights and be easily retrievable
  + A copyright-free cache of images should always be available to the application and not rely on external sources that may be taken down without notice

## [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 for The Gaming Room allows for a visual representation for the organization of the code that ensures efficient management of game instances, teams, and players. The GameService class follows the singleton pattern, ensuring only one instance exists per client. The Game, Team, and Player classes inherit from the Entity superclass, which has attributes like id and name. This allows for easy code reusability and maintainability and is a good exemplification of inheritance as an OOP. The associative relationships between Game, Team, and Player (denoted as 0..\*) establishes a clear relationship where a game can contain multiple teams, and each team can consist of multiple players. Overridden toString() methods ensure appropriate string representations per relevant class. This also exhibits the OOP polymorphism. Encapsulation is maintained through controlled access to properties. and unique game and team instances are enforced through ID management within GameService. The ProgramDriver and SingletonTester classes exhibit a good case of abstraction by separating the functionality of the code into distinct parts. This architecture effectively meets the requirements for multi-team support, unique entity identification, and controlled game instance management in a web-based distributed environment.

**"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** | Not commonly used for hosting web applications because it’s more focused on personal use and development rather than running servers. While it’s Unix-based and stable, it doesn’t have as many tools or support for large-scale web hosting. Tends to have higher costs despite hardware limitations compared to Linux and/or Windows. | Free, secure, and highly customizable. Servers using Linux have many tools at their disposal. It doesn’t require expensive licenses. However, using Linux servers often requires command-line knowledge in order to fully utilize it. | Very user-friendly and accessible, especially with its graphical interface but can be more expensive compared to other choices. Can be resource intensive. Good for services that rely on tools like .NET and/or SQL. | Not well suited to host web-applications on anything more than a small scale. Lightweight and doesn’t have many tools compared to others. The Apple ecosystem allows for multi-system support in comparison to Android based devices. Both devices also have official development tools but development is not typically preformed on the device itself. |
| **Client Side** | Developing software for macOS means using tools like Xcode and programming languages like Swift and Objective-C. It can be expensive since Apple requires specific hardware and developer fees. Since macOS isn’t as widely used as Windows, testing and compatibility can be a challenge. Mac users could be operating on browsers like Chrome, Firefox, or Safari. | Linux isn’t as common for everyday users, but it’s popular for developers. Apps for Linux need to be tested on different versions, which can take extra time. However, since Linux has many free tools and open-source software, development costs are lower. Linux users could be operating on browsers such as Chrome or Firefox. | Probably the most widely used operating system, so developing software for it is essential. Software needs to be tested on different versions but there is sufficient support for that task. Windows based users may be operating on browsers such as Microsoft Edge, Chrome, or Firefox. | Requires support for both iOS and Android, increasing development complexity. Mobile apps also require optimization for different screen sizes and user interactions. Regular updates and compatibility testing with OS updates are necessary. Android and iOS users can be operating on Chrome or Firefox, but iOS users have the additional option of Safari. |
| **Development Tools** | Development for macOS apps primarily uses Swift and Objective-C. The main tool is Xcode, though other editors like VS Code and JetBrains. Often may include the use of a variety of other tools to accomplish things which can overwhelm teams if they aren’t familiar. Mac hardware costs can become expensive. | Typically involves Python, Java, C++, and JavaScript. Common tools include VS Code, Eclipse, and JetBrains IntelliJ. The command line is widely used, along with tools like Docker, Kubernetes, and Jenkins. Fortunately, a lot of these tools/licenses are free. | Development for Windows applications often uses C#, .NET, JavaScript, and Java. The primary development tool is Visual Studio, but JetBrains Rider and VS Code are also widely used. Windows supports Azure which is useful for enterprise software development. Will require Windows Pro license usually. | Mobile app development uses Swift (iOS), Kotlin/Java (Android), and JavaScript for hybrid apps. Tools include Xcode for iOS, Android Studio for Android, and cross-platform frameworks like React Native and Flutter. IOS dev program has a yearly cost. These tools are only available on their respective supported platforms. |

## 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**: A Windows-based solution recommended. Windows Server provides a stable and widely supported environment for hosting applications, making it a good choice for scalability and integration. The game’s client-side can be built using web technologies for compatibility across different operating systems. Interface across platform is friendly and intuitive. Can leverage Azure to easily spin up ready to go templates to mitigate development costs.
2. **Operating Systems Architectures**: The recommended architecture consists of a Windows-based server backend, utilizing .NET Core for flexibility and cross-platform compatibility. Windows utilizes Internet Information Services (IIS) to allow the application to run the web server and provides a host of tools to manage and monitor the application. SQL server, which is native to Windows, can be utilized for database needs and can be configured to use Windows user credentials to access. Tools such as Windows Event Viewer can be used to see details of the instances when errors occur either at the OS or application level. Windows allows configured admin accounts to easily manage these resources to facilitate development and the structure of the project.
3. **Storage Management**: For storage, a combination of Microsoft SQL Server, Azure Storage, and a local file system for local server and application files is recommended. SQL Server will manage structured data such as user accounts, game progress, and statistics, while Azure Storage will efficiently store game assets and multimedia. SQL can be managed/utilized with SQL Server Management Studio which allows the creation of necessary queries and stored procedures which run various tasks that involve moving data back and forth between the user and system. Azure resources (the images used in the game, in this case) can be configured to be easily accessible via API calls. Frameworks can be put in place to easily add more images as necessary to seamlessly integrate into game. Local storage will serve to host application and server files such as configurations, logs, or cached data for debugging purposes.
4. **Memory Management**: The Windows server will handle memory management to optimize performance. Windows will adjust dedicated memory based on need and performance and these tasks can be configured through task manager to be given priority if necessary. For the application itself, the use of IIS (Internet Information Services) ensures efficient handling of web applications. IIS also helps ensure services are kept isolated from each other. Other Windows backend processes will be managed using Windows Containers and Hyper-V virtualization to allocate resources. Additionally, if more memory is required, Azure can be configured to allow for scaling of that based on need as well as provide other metrics and alerts.
5. **Distributed Systems and Networks**: To enable communication across different platforms, a distributed system utilizing Windows-based cloud infrastructure is recommended. Microsoft Azure provides robust networking services to help in ensuring low-latency performance. Users of the application communicate to the server via secure API calls through a RESTful framework which is cross-compatible. Redundant systems can be put in place for the service to ensure functionality. An example of this can include a backup source of images that can be called if Azure Storage is not functional. During development, emphasis of using platform-independent libraries should be stressed. It may also prove beneficial to have servers hosted in different regions across the world. Additionally, traffic to the site should be managed in a way such that servers are not bogged down too much by heavy traffic in one region. The implementation of Azure can help manage this as well as having useful timeout systems in place.
6. **Security**: Windows systems use their own comprehensive security features. These include firewalls, active directories, and anti-virus software. Accounts should be secured with two factor authentication whether via by email or a third-party device. Sensitive user information displayed on the website should follow typical safeguard practices at the very least (hiding password input, automatically signing out, etc). On the web app itself tokens will be implemented for user authentication. Actions can only be taken with the user if they are valid. Data transmission will be secured using SSL/TLS to ensure data is kept private. Sensitive data on the database should always be encrypted. Administrator account access for maintenance work should always be utilized in a secure way meaning only those who should have admin access can use admin features. All code should be routinely reviewed for potential security vulnerabilities as well (SQL injection, bugs, etc). Tools such as static code analyzers can be leveraged as well as thorough QA testing before releases.