

<Draw it or Lose it>

# **CS 230 Project Software Design Template**

Version 1.3

## 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 | <07/21/25> | Cranmer, Thomas | Initial implementation |
| 1.1 | <08/04/25> | Cranmer, Thomas | Executive Summary, design constraints and domain model updates. Evaluation Table completed. |
| 1.2 | <08/17/25> | Cranmer, Thomas | Evaluation Table updates for clarification and Recommendations list filled out in lieu of Windows OS recommendations. |
| 1.3 | <08/24/25> | Cranmer, Thomas | Revised Citations, revised in-line citations for Recommendations list. |

**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 successful game show Draw it or Lose it now has an Android game where teams compete to guess a puzzle from a rendered image drawn from a large library of stock drawings. This is to mimic the old 1980s game show but ease players who may not have capabilities to draw like how the game show utilized an easel. The teams have 1 minute to guess the drawing and there are 4 rounds to determine a winner with chances for other teams to steal the round if a team cannot figure out the image within their 30 second time limit. The Gaming room wishes to have a web based version of the game developed for the game and has requested assistance streamlining the development. Client has limited knowledge on developing web-based environments, requests CTS assistance in designing, developing and releasing web-based application.

## Requirements

1. *Web based version of the Android game Draw it or Lose it mimicking the 1980s game show*
2. *Images to be rendered in matter of 30 seconds gathered from a library of stock drawings*
3. *Allowance of any team to guess drawing after 1 minute with 15 seconds to do so*
4. *Game has ability to have 1 or more teams involved*
5. *Each team has multiple players assigned*
6. *Game/Team names are unique and checkable when choosing a name*
7. *1 instance of the unique game allowed in memory at a time.*
8. *Unique identifier for player names/IDs*

## [Design Constraints](#_2et92p0)

1. Unique identifiers required: Each team, player and game requires a unique identifier upon creation to ensure no duplicates in memory. This ensures players trying to search for any game or team do not come across duplicates and memory does not override in the instance there are 2 of the same instance for the game, team or player identifier
2. Web Base application: The game to be designed is purely web based and is to mimic the android version of the game that already exists. This requires certain programming languages and styles to be utilized for web playing capabilities. Possible programming languages include: Java, JavaScript, HTML.
3. Single Game Service Instance: A single game instance is required to be performing at 1 time to ensure any players can access and play the game. This requires a constant check to be made in the files to ensure that a game service instance exists at all times.
4. Quality Images: Web-based applications are utilized on higher resolution devices than the typical smart phone. This requires higher storage per photo for increased quality, requiring increased server capabilities for storage and memory to accurately display images
5. Security Structure Differences: Web based HTML security procedures differ greatly from smart phone derived security initiatives requiring different methods for authentication of users. Necessary precautions for user accounts for client-server interactives are necessary

## [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 consists of 6 class member functions and 1 main function. The entity class is the base class function for the Player, Team, and Game classes. The Entity holds 2 private member variables that’s are utilized by the 3 classes that inherit the base: id and name. As a base class for the inheritance of the 3 other classes, this helps to ensure unique identifiers and names across the inherited classes. This is also ensured by an iterator pattern present that checks through any existing names and id identifiers for the placement of teams within a game and what players are in any specific team to prevent duplicates. The associated getter functions for the variables are also utilized by the inherited classes. Encapsulation is seen here in the hiding of the base class member variables and the private constructors utilized to help the singleton pattern be effective. Polymorphism is utilized between the base class and it’s 3 inherited classes on the toString function where they all override the function to display their own personalized output lines. There is 4 instances of 0 to many associations between GameService and Game, Game to Team, Team to Player, meaning while there can be 0 instances of the left class where many instances of the right class can exist in memory, example being there can be multiple player objects existing but not associated to a team object. A singleton pattern, which has it’s own testing class, is utilized to ensure only 1 game instance exists at any point in memory by having the GameService object exist as a null object until created.

**"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** | Heavily integrated with Apple ecosystem, but supports strong security features and ease of management. Licensing costs also hamper MacOS with subscriptions services being noted at $99/month. UNIX Based kernel makes ease of user creation for servers. Mac also boasts 1 of the highest tier security systems through the Apple ecosystem | Linux has a user-friendly integration, with open-sourced models allowing easier development capabilities. This is coupled with a range of applications, that may also seem daunting for the developer to start on. Licensing fee keep budgetary levels low with Linux being open-sourced. | Most widely utilized operating system, allows extensive community interaction. Windows also sports high scalability for larger programs. Extensive Microsoft policy and the need to understand Active Directory’s unique model is a necessity for developers. | While iOS sport the same security protocols as Mac and Android is open source allowing a variety of features and community elements. Mobile Phones are not suited for Server development and being utilized for servers nor have server components. |
| **Client Side** | Mac heavily secludes to the Apple ecosystem, which causes some integration holes. Along with specific rules necessary for Apple allowance of application. Expertise in Swift and C# also necessary. | Linux’s more open-sourced background allows easier integration for clients, and supports a no subscription approach for development. Developers are only held back by their expertise with Linux. With having flexibility with browsers, developing for any browser helps the Linux user, but popularity demands lean toward Chrome for Linux. | .NET Framework and ActiveX are supported by a wide range of IDEs, making integration easier for window’s client development. This is hampered by constant security update issues which may cost developers time, and costs. Having a developed Browser attached of Edge and pairing with Chrome’s popularity making the minimum requirement 1 of these 2 browsers. | For iOS, we have similar issues with Mac for expertise with Swift and C#, but performance and security sport the headlines for clients. Android has more open-source capabilities and a more user friendly UI to lower the expertise bar. Both have grand offline development capabilities. Predominant option in house holds, and with web browsers becoming more popular, developing for Chrome helps and not hinders |
| **Development Tools** | Languages like Shift, JavaScript and HTML will be the prominent programing languages. For IDEs, xCode and Visual Studio are most popular | With Linux’s open source identity, languages like C++, C# and Java will be the go to languages. These will utilize IDEs like JetBrains IDEA, CLion, or from my experience, Geany. | Programming languages C#, Java, HTML will be valuable languages for Microsoft’s staple OS. With valuable IDES such as VS Code, Eclipse and Pycharm (for Python coders) | This is divided into 2 for Swift and Java/JavaScript and C/C# for iOS and Java, Kotlin and C based languages for Android. With iOS utilizing Xcode for an IDE and Android utilizing it’s own Android Studio. It is said though, these must be utilized on a PC. Mobile Phones are not suited for development. |

## 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**: With the popularity in it’s web browsers, the abundance of IDE developing tools and while expensive to license, sports the biggest development community in case the workers run into a roadblock. Windows is the best option for developing the web browsing app, with a focus on Chrome and Edge development. Chrome for it’s universal usage across all operating platforms, and Edge for it’s ties with the Windows OS.
2. **Operating Systems Architectures**: Windows utilizes a kernel hierarchical architecture, where at the very base of the OS is the Windows kernel and Device drivers which make up the I/O software, virtual memory management. Those are the base operating system kernel, then you have the system services: your executive services, .NET Framework and your Windows APIs. Both are utilized to run both traditional applications and executables and Microsoft store specific applications that have been approved by Microsoft.
3. **Storage Management**: Windows utilizes a specialized, Windows based storage management system called NTFS. Utilizing a transaction-based log file (Microsoft, 2025) that maintains failure protection and quick file system restore capabilities. While also utilizing permission based permissions to protect storage for specific users by utilizing Access Control Lists. This is extremely useful to protect server side data and resources from any outside access capabilities.
4. **Memory Management**: Windows utilizes various memory management styles to ensure data is held and if overflowed, has a place to overflow without causing crashes. Through Virtual memory all processes have independent virtual address spaces to ensure no overlapping of processes and seamless operation of concurrent processes.(Rivera, J. 2021) In the point of if too many processes are conducted, through the act of paging, windows utilize disk space to temporarily maintain processed to prevent data loss, or process failure. And finally, with .NET, there are processes where unused objects are cleaned up and freed up for future process handling. For Draw it or lose it, the VM capabilities help ensure any instance of the game has their own memory address, while the paging method helps maintain instances in the event there are too many instances for the memory space.
5. **Distributed Systems and Networks**: Through the utilization of RESTful APIs, they make it possible for seamless usage of the same space for any browser or operating platform that is able to have a user created and a role assigned. RESTful APIs are non-discriminatory in their act of user management, along with being widely used by all operating platforms due to their simplicity and good security practices.(Gupta, 2023) With Windows, cross platform communication is even more easy with Windows specific APIs being able to have similar communication processes with mobile phone and web clients (including iOS). On top of that, cloud-based services make real-time gameplay more seamless by lowering latency requirements for users connecting to the server, and maintain smoother connections due to less access points to connect through.
6. **Security**: Windows sports multiple security points to protect user data and client-customer communication. NTFS permissions protect the user and provider separately on the local side through permission based access lists, while Microsoft developed firewalls, and anti-virus programs like Windows Defender, maintain constant monitoring of file systems and network connections. This last part is especially important for Draw it or Lose it as they can have constant monitoring of who is connecting to their servers to access the game. On top of locally, upon transit completion to connect to the server, Windows sports a Role-based access control style on top of a RESTful style API to ensure only those with proper permissions are allowed through to the game instances and server side data.

Work Cited:

* *Mysik, A. (2024, December 11). How to build an app for iOS and Android | Uptech. Uptech.*[*https://www.uptech.team/blog/app-development-for-ios-and-android*](https://www.uptech.team/blog/app-development-for-ios-and-android)
* Rivera, J. (2021, June 26). Windows vs macOS vs Linux for Web Design & Development. *PixemWeb*. <https://www.pixemweb.com/blog/windows-vs-macos-vs-linux-for-web-design-development/>
* Microsoft. (n.d.). *Examine the Windows client architecture—Training*. Microsoft Learn. Retrieved August 23, 2025, from <https://learn.microsoft.com/en-us/training/modules/explore-windows-architecture/3-examine-windows-client-architecture>
* Gupta, L. (2023, April 3). *Dropwizard Authentication and authorization*. HowToDoInJava. https://howtodoinjava.com/dropwizard/dropwizard-basic-auth-security-example/