

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 5**

**Domain Model 5**

**Evaluation 6**

**Recommendations 8**

**References 9**

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/10/23 | Marissa Sihapanya | First draft with summary, requirements, design constraints, system architecture view |
| 2.0 | 07/29/23 | Marissa Sihapanya | Second draft update with evaluation of operating platforms |
| 3.0 | 08/08/23 | Marissa Sihapanya | Final version updated with recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is developing a web-based version of their gaming app Draw It or Lose It. The web-based version must embody the core characteristics of the gaming app, which are: the ability to have one or more teams with multiple players, unique game and team names, and one concurrent instance of the game within the application’s memory.

## Requirements

1. The Draw It or Lose It game application must be transformed into a web-based application.
2. The web application must be able to render images from a large library of stock drawings.
3. A game within Draw It or Lose It can have one or more teams playing at a time.
4. Each team will contain multiple players.
5. Game and team names must be unique, and validation must be implemented to check whether names are already in use.
6. The memory holds only one instance of the game at a time, created through the implementation of unique identifiers for each game, team, or player.
7. The web game has menus across its gameplay to ensure smooth navigation of the application.

## [Design Constraints](#_2et92p0)

1. Constraint of developing a web-based application from an Android game application.
   1. Must consider what new software and programming languages will be necessary to develop the web-based application.
   2. JavaScript is intertwined between Android development and web development, but web development must consider using HTML5 as it is the main language used throughout web applications for rendering. Additionally, C++ and C# are other languages mainly used in Android development, but do not translate into web development. Development of the web game will require knowledge of HTML5 and CSS.
   3. VS Code can be used for both Android and web development, but it is not made specifically for Android. Thus, Draw It or Lose It was likely created in an Android-focused IDE. The Gaming Room will need to consider a new IDE for the web application.
   4. The logic behind the code may not directly transfer from the Android application to the web application. Code may need to be refactored to accommodate web development languages, or design may need to be restructured to properly fit new languages.
2. Constraints from the ability for the web game to render images from a large library of stock drawings.
   1. Must consider where the stock images will come from, and how credit will be cited during the development process. The stock images will likely come at a cost so must also consider the prices for each library.
   2. File sizes and compression factors of the stock images must also be considered. The images will take up space within the web game, and larger file sizes will likely be more costly.
3. Constraints for staying within budget for solid development of web game from Android application.
   1. Many factors to consider for staying within budget such as cost of library stock images, new IDE installation, and talent hiring or team upskilling for new programming languages
   2. Cost of hosting the web game on The Game Room website.
      1. Many users will be accessing the application, taking up memory on the website’s servers. The team needs to consider server capacity and their anticipated audience load.
4. Constraints regarding UI/UX design principles of web applications VS applications on Android systems.
   1. Androids are limited to a portrait, mobile screen resolution. Web applications are within web browsers which are most often accessed through a PC, where the resolution is landscape. The dimensions of the application must be reconsidered, and any math pertaining to the alignment of elements among the UI must be re-calculated.
   2. Consider design possibilities of transitioning from a smaller portrait resolution to an upscaled landscape resolution. Much more can be achieved at one time with the ability to display more concurrent elements.
   3. Determine the new controls as the application is transitioning from touchscreen to mouse and keyboard. What is pleasing and convenient on a mobile device is much different than what is pleasing and convenient on a PC.
5. Constraints on the ability to have one or more teams playing.
   1. The web game needs functionality for creating teams and assigning members to each team. Can consider user-typed entry of teams and members, voice entry, or simply selection from a collection of pre-defined team names and member names.
   2. If considering user-typed entry, validate for appropriate names.
   3. For game fairness, teams must be ensured to be of the same size. Implement code validation to ensure that team sizes are the same before game proceeds.
6. Constraints for each team to have multiple players assigned to it.
   1. Must have pre-defined minimum and maximum size requirements for a team.
   2. Ensure the code validates that one player cannot be on multiple teams at once to uphold fairness across the game.
7. Constraints on uniqueness of game and team names.
   1. Must design clear visual indicators that distinguish between each team and between players.
   2. Decisions on the best symbols and colors to use for this distinction, and the color scheme of the entire web game, must be planned.
   3. Ensure these visual design decisions are made with accessibility in mind; the distinctions must be clear enough for anyone to understand.
8. Constraint on only one instance of a game existing at a time.
   1. Consider implementing the singleton pattern to achieve only one instance at a time. The singleton pattern will ensure there are not multiple instances of the game existing concurrently.
   2. Implement terminating game functionality if players would like to begin a new game.
9. Constraint for designing start and pause menus for the web game.
   1. Menu must be designed to embody core functionalities of the game: start game, create team, add members.
   2. Pause menu must also be developed: resume game, adjust team members, exit game/start new game.
   3. These menus should be user-friendly.

## [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 ProgramDriver is simply a Driver class that uses just a main() function to drive the program. It uses the SingletonTester class to test if the Singleton pattern used in the GameService class is functioning properly with the testSingleton method.

The Entity class captures our common attributes name and id. The GameService, Game, Team, and Player classes all depend on the Entity class. They all share similar behavior in modifying name and id attributes across the game by using name and id as parameters in their various methods.

0…\* represents a zero-to-many relationship between classes. This means the class on the left can have from 0 to many associations with instances of the class on the right. The class on the left can function properly even without any associations with the class on the right. In the case of the Team and Player classes, Team can have multiple players or no players. Thus Team can manipulate objects within Player. In this 0…\* relationship, a team can have multiple players, but a player can only belong to one team.

The 0…\* relationship is a core functionality for our web game. Only one game can exist at a time, and that’s ensured by the Singleton pattern. A player can only belong to one team at a time, and the 0…\* relationship ensures that by defining that Team can have 0 to many associations with Player. Similarly, the single game can have many associations with Team, but one team cannot belong to multiple games since only one game can exist at a time. The relationship is defined in the other direction by existence rather than association. A player cannot exist without a team, a team cannot exist without a game, and a game cannot exist without the game service. All these components come together to form our game Draw It or Lose It.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | + User-friendly  + Strong security  + Simple server application setup  + Highly regarded in design industry  - High cost and high investment for the hardware  - Macs are limited hardware-wise  - Not as scalable  - Limited software  - Not a dedicated server platform  \* Mac computers start at $1299 | + Stable, reliable  + Considered strongest for hosting web servers  + Highly compatible with Python  + Low cost because open source  - Not user friendly (largely command line based)  - Restrictive  \* Linux workstations start at $299 | + Fully supports all Microsoft technologies  - Due to licensing, higher cost  + Security updates are automated  + Less effort required to set up than Linux  + Internet Information Services deploys multiple servers and offers configuration and maintenance  - OS processes eat resources simultaneously  - Upgrades induce downtime  \* Windows PCs start at $400 | + Capability to use mobile devices as web servers  - Does not have the same ability to host dedicated or cloud web servers that Mac, Linux, and Windows can  - Web application will be specific to mobile devices and not expanded to clients on other OS  \* Modern mobile devices start at $600 |
| **Client Side** | Development on the Mac must make sure the application is compatible with all possible operating systems and devices from the clients. The given IDEs must also be able to construct differing UIs adapted to each device and OS. Server setup is streamlined due to its relation to Linux, thus optimizing for the multitude of clients will be easy. Mac hardware is expensive, so must consider the costs, especially when allocating resources based on the hardware’s capabilities. | Linux covers more hardware configurations than Mac does, so cost will run high when delivering a centralized user experience. Linux also has a steep learning curve but can be tolerated with extensive documentation. Linux is also open source, so it will save costs on licensing products during development when considering a large client base. | Given a web-based application, Windows, along with mobile cloud, Linux, and Mac, require expertise in web development technologies. This means a basis in HTML, CSS, and JavaScript as the basics, then knowledge of server-side scripting. Windows must also consider the costs of supporting different hardware configurations. Windows, being under Microsoft, does come with steep licensing costs. Licensing costs will depend on scope of the application, such as if The Gaming Room would buy Microsoft’s business packages, or purchase subscription-based software to develop the application. | Any client can access a web server hosted by a mobile device, but it is not guaranteed that a web application hosted by a mobile device will fully function for clients. In other words, a mobile device is much better as an application accessor than an application server. The range of operating systems used by clients is thus severely limited. |
| **Development Tools** | - Python  - JavaScript  - HTML/CSS  - Ruby  - Xcode - Visual Studio Code  - Sublime  \* Requires one team minimum  \* Software available for free  \* All languages available for free | - Python  - JavaScript  - HTML/CSS  - Ruby  - Visual Studio Code  - Atom  - Eclipse  - JetBrains  \* Largely open source so no concerns over licensing costs  \* Requires one team minimum  \* All languages available for free | - Python  - Java  - JavaScript  - C++  - C#  - Visual Studio  - Eclipse  - AWS Cloud9  \* Requires one team minimum  \* AWS Cloud9 is not open source and does cost money for access and development  \* All languages available for free | - Chrome dev tools  - Safari mobile web developer tools  \* Additional teams not necessary  \* Signing up as a developer on Google Play Store and Apple App Store come with fees |

## Recommendations

1. **Operating Platform**: The best operating platform for The Gaming Room to expand to other computing environments is Windows. The larger portion of the population uses Windows devices to access web content. Developing on Windows will ensure that the web-based application is tailored to these clients. Not only that, but Windows as an operating platform can still ensure that Draw It or Lose It is delivered sufficiently to other environments like MacOS, Linux, and mobile devices. Windows offers insights on application performance, which will allow The Gaming Room to finetune their application to each computing environment. Visual Studio Performance Profiler and Windows Performance Toolkit are examples of Windows software that allows visibility of this granularity. Windows is the most versatile platform and offers the most options for exporting to all computing environments.
2. **Operating Systems Architectures**: The architecture of Windows consists of the operating system kernel, system services, and apps (microsoft.com). The kernel acts as the center of the Windows operating platform, acting as communication between system services and computer hardware (vskill.com). System services start upon system boot and communicate with software across Windows. Apps, while engaging with system services, interact with the user.
3. **Storage Management**: A CDN (Content Delivery Network) is a great choice for storage management in Draw It or Lose It. Players will be dispersed across many regions, and these images and other files need to be distributed to them effectively. By using an origin server, a CDN allows images and other content to be distributed to players. CDN services offer high throughput and secure storage that will allow Draw It or Lose It to handle its storage well (keycdn.com). By selecting a CDN for storage management on Windows, The Gaming Room will optimize scalability and performance for Draw It or Lose It.
4. **Memory Management**: Draw It or Lose It uses 200 8-megabyte image files. This equates to 1.6 total gigabytes for all the images. The Windows operating platform uses virtual memory and RAM (Random Access Memory) to manage files on the hard drive (windowscentral.com). Draw It or Lose It will not need to load all 200 images at once; it will only need to display the necessary images per game. Thus, Windows will initiate virtual memory by moving unused files from RAM onto the hard drive (windowscentral.com). In doing so, Windows frees up space and total memory for the application to display images. Memory management also allows Draw It or Lose It to perform optimally by calling necessary images in present instances, instead of loading all 200 images at once, and then freeing images when no longer needed (microsoft.com). Windows memory processes ensure files are accessed efficiently and are then released upon operating platform shutdown.
5. **Distributed Systems and Networks**: Draw It or Lose It can implement distributed computing to achieve their application across multiple end users. By using distributed computing and upholding fault tolerance, The Gaming Room can maintain 100% uptime regardless of if one singular component fails (imperva.com). This fault tolerance is allowed because of the communication that occurs in the network within the system to handle all processes (tutorialspoint.com).
6. **Security**: Windows can use a WAF (web application firewall) to protect Draw It or Lose It. The WAF will protect against cyber-attacks such as cross-site scripting and SQL injection, ensuring that no bad actors can infiltrate the application and compromise user data (f5.com). Since the application needs to send and accept requests from clients who may have unsafe networks, many layers of security are required for optimal application health (imperva.com). User authorization for Draw It or Lose It should be handled with the principle of least-privilege, where users are authorized to only do what they need to complete a given task. Authentication will protect user information through the requirement of usernames and unique passwords. This will ensure no one user is accessing a user profile that does not belong to them.

**References**

Wwlpublish. (n.d.). *Examine the Windows client architecture - Training*. Microsoft Learn. https://learn.microsoft.com/en-us/training/modules/explore-windows-architecture/3-examine-windows-client-architecture

Vskills, T. (2020). Windows Architecture. *Tutorial*. https://www.vskills.in/certification/tutorial/windows-architecture/#:~:text=The%20kernel%20is%20the%20most,more%20CPU%20modes%20than%20that

Huculak, M. (2022). How to manage virtual memory on windows 11. *Windows Central*. https://www.windowscentral.com/software-apps/windows-11/how-to-manage-virtual-memory-on-windows-11

Microsoft. (n.d.). Disk and Memory Performance Guidelines. Retrieved from <https://learn.microsoft.com/en-us/windows/apps/performance/disk-memory>

*Games CDN | What does Content Delivery Network for Gaming do?* (n.d.). BelugaCDN. https://www.belugacdn.com/games-cdn/

Imperva Inc. (2019, December 30). *What is Fault Tolerance? | Creating a Fault Tolerant System | Imperva*. Learning Center. <https://www.imperva.com/learn/availability/fault-tolerance/>

*Distributed systems*. (n.d.). <https://www.tutorialspoint.com/Distributed-Systems>

Stapel, G. (2023, August 7). *What is Application Security | Types, Tools & Best Practices | Imperva*. Learning Center. https://www.imperva.com/learn/application-security/application-security/