

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 | 03/19/2023 | Achna Hettiarachchi | Finalized Executive summary, Design constraints, domain Model. |
| 1.1 | 04/02/2023 | Achna Hettiarachchi | Finalized Evaluation for the “Draw It or Lose It” game development. |
| 1.2 | 04/16/2023 | Achna Hettiarachchi | Final recommendations for the project. |

## [Executive Summary](#_sbfa50wo7nsh)

Our client The Gaming Room wishes to build a web-based game that works on multiple platforms grounded on their current game, Draw It or Lose It. The game is currently available in Android app only. There will be multiple teams involving several players competing on four rounds of play that take one minute each. The app will extract images at a constant rate from a large collection of stock drawings as clues within 30 seconds. The team must guess the puzzle before time ends, or else, the other teams get a chance to submit one guess each to work out the puzzle with a 15-second time constraint.

## Requirements

* A game must be able to have one or more teams participating.
* Each team will consist of several players.
* Game and team names must be unique to let players to check whether a name is in use when selecting name for the team.
* Only one instance of the game can be present in memory at any given time. I.e., have unique identifiers for each instance of a game, team, or player.

## [Design Constraints](#_2et92p0)

The software design must meet the following requirements.

* Needs one or more teams participating
* Each team has several individuals
* Game and Team names must be unique to let player to check whether name is in use or available
* Only single instance of the game can be present at any time
* Must work on multiple platforms

The client also needs the app to work on all devices including Windows, Linux, and Apple machines instead of just on Android. So, we need to think about ways to modify the existing code to comply with other languages distinctive to those devices.

## [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)

Game, Team, and Player classes all extend the Entity class, which means that they inherit Entity's name and id. GameService class doesn't explicitly extend the Entity class, it uses the Game class object. The Game class is composed of Team class objects. Team class is composed of player objects. ProgramDriver and SigletonTester models require each other to function.

**"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)**

First, I will evaluate various platforms for their characteristics, advantages, and weaknesses for hosting a web-based software application. I will consider whether each of the operating platforms offers a server-based deployment method where the website will be hosted, and what are the potential licensing costs to the client, The Gaming Room, for the server operating system.

Then I will summarize the software development considerations (cost, time, expertise) that are necessary for supporting multiple types of clients. To ensure the application is compatible with all web browser platforms and mobile devices we must use web standards like HTML, CSS, and JavaScript; adopt a responsive web design, do cross-browser/mobile device testing, and utilize the latest development tools like Vue.js.

Finally, I will identify the relevant programming languages and tools (IDEs and other tools) that are used to build this type of software for deploying on each operating platform. When developing software compatible with different platforms, technical requirements impact the development teams a lot. It demands expertise in different programming languages, tools, and frameworks and requires added testing and debugging. Multiple development teams may be beneficial to tackle the technical requirements of such a complicated project. For instance, several teams can develop the software for different platforms parallelly. Further, there can be licensing costs related to the development tools that we must consider at the early stages of the project.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | **Characteristics:** user-friendly interface, strong security features, high-performance hardware, high-quality displays, and dominant graphics capabilities. **Advantages:** Robust Unix-based system well known for stability and security. Easily upgradable. **Disadvantages:** Limited software/hardware options and certain compatibility issues. **Deployment:** MacOS Server can host websites and web-based applications. It has flexible terminal commands to construct the server. still less preferred for web hosting services due to being an expensive platform. | **Characteristics**: open-source, flexible, secure, low cost, high level of customization and control. **Advantages**: Strong command-line interface allowing developers to easily manage and configure web servers and databases. **Disadvantages**: complexity, lack of technical support, and limited hardware compatibility. **Deployment**: Many Linux distributions have LAMP (Linux, Apache, MySQL, PHP) stack pre-installed. | **Characteristics**: ease of use, compatibility with a wide variety of software and hardware, and huge user base. **Advantages**: offers a diversity of tools and technologies, like Microsoft IIS web server, ASP.NET framework, and SQL Server database, which are perfect for web development. **Disadvantages**: more susceptible to viruses and malware and insignificant technical support. **Deployment**: Windows Server has the web server Internet Information Services (IIS). OS and product licensing make it more expensive. | **Characteristics**: widespread due to portability and cost-effectiveness, need a link to a computer with internet access to combine further relay steps, and is related to security issues**.**  **Advantages**: Mobility, ability to personalize, convenience. **Disadvantages**: limited resources, and reliance on internet connectivity. **Deployment**: Mobile operating systems, like iOS and Android, are not normally used as server OS. Mobile web servers, like KWS and Bit Web Server, permit developers to host websites and web-based applications straightforwardly from mobile devices. |
| **Client Side** | Moderate expertise and time required. Most Expensive. | Maximum expertise and time required. Minimum cost. | Minimum expertise and time required. Expensive. | Adaptable for updates but difficult to implement compared to other devices. |
| **Development Tools** | Macs support HTML/CSS/JavaScri  Pt and languages like Java, Pyhton, PHP, and Ruby. IDEs include visual studio, eclipse, PyCharm, IDLE. Tools include Atom, Delphi, Cloud 9, Embold, Kwatee, Zoho Creator. | Linux supports HTML/CSS/JavaScri  Pt and languages like Java, Pyhton, PHP, and Ruby. IDEs include Eclipse, visual studio, IDLE. Tools include Atom, Delphi, Cloud 9, Embold, Kwatee, Zoho Creator. | Windows support HTML/CSS/JavaScri  Pt and languages like Java, Pyhton, PHP, and Ruby. IDEs include visual studio, Eclipse, IDLE. Tools include Atom, Delphi, Cloud 9, Embold, Kwatee, Axure, Zoho Creator, Jira, Azure. | Languages like Java, Pyhton, PHP, and Ruby can be used for mobile app development. Android studio, Xcode, Eclipse, Visual Studio Code are among best IDEs for mobile devices. |

## Recommendations

Here I analyze the characteristics of and techniques specific to various systems architectures to make a recommendation for The Gaming Room.

1. **Operating Platform**: I recommend Windows platforms due to the availability of software, handiness of IDEs, minimum expertise needed, and reasonable cost associated.
2. **Operating Systems Architectures**: Windows operating systems support several architectures, that are intended to support different hardware platforms and provide upgraded performance, memory capabilities, and power efficiency for specific use instances. Specifically, x86 is the most widely used supporting both 32-bit and 64-bit versions, ARM is designed for mobile devices supporting both versions, Itanium is for high-end servers and workstations, and AMD64 is used for improved performance and larger memory support in both desktop and server operating systems. Further, Windows API is the programming interface to the Microsoft Windows OS, supporting Windows-based applications to provide a Graphical User Interface (GUI), retrieve system resources, integrate audio, etc. that will be useful in our project.
3. **Storage Management**: Windows 10 “storage sense” feature allows analysis and management of files on the hard drive. It’s a useful tool for managing storage space as we have to deal with large amounts of data, including images, sound files, and other media. With this feature, we can easily identify and delete temporary files or older versions of the game assets that are no longer required, freeing up valuable storage space on the hard drive. Mainly during the development phase of a game, where files are continually being created and altered. In addition to the built-in storage structure, cloud storage options such as OneDrive or Dropbox can be useful as it permits us to access game files from different devices or locations. Cloud storage allows us to store game files securely in the cloud and easily restore them in the event of a hardware failure or other data loss situation. This backup feature will ensure that highly important game files are not lost and hence development goes uninterrupted.
4. **Memory Management**: Designing this game requires a large database with loads of pictures and we will have to use memory allocation to securely store these pictures together with other project files instead of the default picture folder. Windows has several memory management techniques to manage the large database of pictures required for the game, counting virtual memory to use the hard drive as temporary storage, a memory manager to assign memory as per requirements, memory mapping to load only required segments of large files into memory, and tools like Task Manager and Resource Monitor to screen memory usage and recognize any issues. These techniques will certify that the game goes smoothly and efficiently.
5. **Distributed Systems and Networks**: To facilitate cross-platform communication for our game, we should adopt a cross-platform game engine like "Unity" that supports multiple platforms and utilize a service-oriented or microservices architecture with APIs or message queues to communicate between them. We should also carefully decide on the programming language and framework and ensure that our servers are powerful enough to support large player capacities with a backup plan for power outages. It is important to have a reliable and secure network with measures in place to avoid security threats. We can use techniques such as caching, load balancing, and distributed data stores to minimize network latency.
6. **Security**: On Windows, we can protect user information through security features like Windows Hello, a biometric authentication system that uses facial recognition or fingerprint scanning to verify user identity. Windows also has a built-in firewall that can block unauthorized access to the system and block attacks from malicious software. Yet we better use more reliable antivirus software to protect user data as Windows is susceptible to virus attacks and do regular backups of user data. It is recommended to have a plan to respond to security incidents (if any occur) such as protocols for notifying affected users and authorities. Encryption (e.g., HTTPS to encrypt web traffic and VPNs to create secure connections between devices over the internet) can be used to secure data user information on various platforms. We must further prompt users to create strong and unique passwords for their accounts, educate them on how to avoid phishing and other types of social engineering attacks that try to steal their personal information, and employ multi-factor authentication as an additional layer of security to user accounts.