

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 09/19/2021 | Alana Kaiser | Establish software requirements for the game application. Complete summary, and design constraints. |
| 2.0 | 10/03/2021 | Alana Kaiser | Evaluate the advantages and weaknesses of various platforms. |
| 3.0 | 10/17/2021 | Alana Kaiser | Asses the characteristics and architectures of different operating systems and make a recommendation. |

## [Executive Summary](#_sbfa50wo7nsh)

Draw It or Lose It, by The Gaming Room, is currently an Android only app and needs to be expanded, and redesigned, to become a web-based game that serves multiple platforms like Mac, Linux, Windows, and mobile devices. The redesign requires the game to have one or more teams, multiple players on each team, unique team names, and only one instance of a game in memory at any one time.

## [Design Constraints](#_2et92p0)

* Web service for hosting and storage
  + High levels of availability
  + Scalability
  + Performance
  + Control and flexibility of configuration
* Image file type that will display correctly
* REST architectural constraints
  + Uniform interface
  + Client-server
  + Stateless
  + Cacheable
  + Layered System

## [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)

A UML diagram is a means of illustrating object-oriented programming and how that will be implemented in Draw It or Lose It. There are currently seven classes in the program. Starting with the ProgramDriver class; it includes the main method that will be in control of the game. There is a SingletonTester class that is used by the ProgramDriver to ensure that there is only one instance of the game running. The base class is Entity. Classes Game, Team, and Player all inherit common attributes and behaviors from the Entity class. As the base class, Entity provides Id and Name to the subclasses Game, Team, and Player. The Game class creates a list of all the teams and provides the method to add new teams. The Team class will do the same but for unique players instead. The Player class provides a method to create a new player instance. GameService, Game, Team, and Player classes are all associated with each other. As the UML indicates, they can have zero to many instances as needed. Finally, the GameService class interacts with Game, Team, Player, and Entity using the singleton pattern to ensure that only one instance of a game is in memory at any one time. GameService also contains the iterator to quickly iterate through the team and player names.

**"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** | UNIX-based, built-in firewall, encryption, and authentication services. Run Mac, Windows, and Linux apps. Expensive for web hosting. MacOS X Server will only run-on Mac hardware. It is recommended to use licensing software to manage licenses. Licensing costs can vary widely. | Free open-source software. Supports cooperation with little risk to the program’s core, low hardware demand.  Complex to operate, 3rd party programs need admin, some programs will not work. It is recommended to use licensing software to manage licenses, but Linux should not require licensing fees. | Beginner-friendly, intuitive, large number of 3rd party apps. High costs, vulnerable to security errors and malware, high demand on hardware. Expensive, high licensing costs. It is recommended to use licensing software to manage licenses. Licensing costs can vary widely. | It is possible, but not practical for this specific project. Mobile Web Server (as an example) is a lightweight mobile phone app that provides a simple HTTP web server. It is not capable of running a project this large. This option is not recommended. |
| **Client Side** | Mac is the most expensive option. Moderate expertise and time is required. Uses Safari web browser as standard. Can use other browsers. Must run cross-browser compatibility checks. | Cost is moderate. Will need the most amount of time to develop because it requires the highest skill. Uses Firefox for web browsing. Must run cross-browser compatibility checks. | Lowest cost option. Setup up is easy and will require less time. Easiest to implement so less skill will be required. Comes installed with Edge. Can use other browsers like Chrome. Must run cross-browser compatibility checks. | Cost depends on the device being used. Will need the most time to develop. Requires expertise in Android and iOS. Browser depends on the device; Android uses Chrome and iOS uses Safari. Must run cross-browser compatibility checks. |
| **Development Tools** | Macs can run many languages like Objective C, JavaScript, and CSS. Also supports PHP and Python. PyCharm, Eclipse, or XCode for an IDE. Licensing costs can vary widely; it is recommended to use licensing software to manage licenses. It is recommended to have someone on the team specialized in running a Mac Server and tools. | Works with Python, C++, Rust, and Linux can run other languages like HTML/CSS/JavaScript. Licensing costs can vary widely; it is recommended to use licensing software to manage licenses, but Linux should not have any licensing fees. It is recommended to have someone on the team specialized in Linux to maintain the server and tools. | Easiest to use of all the options. Works best with Visual Studio, but Eclipse and PyCharm IDEs are also used to support Java, Python, PHP, and Ruby. Languages like HTML/CSS/JavaScript are also used. Licensing costs can vary widely; it is recommended to use licensing software to manage licenses. Windows is the most widely used server software and should not require a specialist on the team. | There are apps available to build and code websites, but Android and Apple do not speak to each other. Swift to build iOS mobile apps. Kotlin to build Android OS mobile apps. Apple developer license is $99 per year and Google Play Developer pays a one-time $25 fee. Due to conflicts, these environments are not suitable for development. Mobile platforms are not suitable for development environments. |

## Recommendations

1. **Operating Platform**: Windows Server 2022 is the recommended operating platform for delivery of The Gaming Room’s new application, Draw It or Lose It. Windows Server uses the Windows Azure cloud. Windows Server 2022 has advanced multilayer security and gives developers a flexible platform for application deployment. Cloud based resources can be used by any device using any platform from anywhere in the world. This allows the game to be played globally.
2. **Operating Systems Architectures**: It is the recommendation that The Gaming Room utilize the cloud-based architecture Azure, specifically Platform as a Service (PaaS), for the deployment of their application Draw It or Lose It. Windows Server 2022 combined with Azure allows The Gaming Room to remove their dependence of on-site servers, storage, and networking framework. Cloud architecture allows for a quick delivery of updates, real-time scaling of the application as needs change, and by reducing on-site resources, they are able to cut costs. A PaaS model gives The Gaming Room networks, servers, and the storage needed to run the application, while simultaneously giving the freedom to control the application deployment and configuration settings.
3. **Storage Management**: Windows Server 2022 uses the built-in Storage Migration Service. This service allows for the easy migration of storage from source to Windows Server or Azure by using SMB compression to compress the files before transferring them over networks. That gives users a faster transfer on congested and slow networks. Using PaaS service model, Azure will manage the hosting environment along with managing the physical hardware as well, eliminating the need for on-site servers. This means multiple cloud servers are leveraged to meet the necessary storage requirements.
4. **Memory Management**: SQL Server Memory Architecture uses and frees memory as required. A goal of database software is to reduce disk I/O because the read/write operations use the most system resources. SQL Server uses a buffer pool to hold pages already read from the database. A function of Memory Management is to keep track of whether a memory location is allocated or free. One method that creates effective Memory Management is the use of virtual memory.
5. **Distributed Systems and Networks**: The Gaming Room will use Windows Server 2022’s integration with Windows Azure to communicate with the servers in the cloud. Cloud architecture scales up quickly to support usage growth. As it scales up, the application will run across many servers to handle the increased demand. This creates a distributed system. Combined with networking, it is a useful way to communicate with the different servers. Using cloud servers, the tasks are efficiently divided.
6. **Security**: Windows Server in conjunction with Azure offers advanced multilayered security. It offers encrypted HTTPS and AES-256 encryption. By default, Microsoft’s Windows Server Antimalware is installed as well. Another security to implement is authentication, using a username and password. Once a user has been authenticated, the system then checks to see what that user is authorized to access and what is their defined role in the system. To handle authorization, passwords should be stored as a cryptographically strong hash and salt the hash with a unique value for that login. A third-party identity, like Google, Facebook, or Twitter, can authenticate an identity. Other security features are reauthorizing accounts that have been idle for too long, limiting the number of login attempts, and using password reset instead of retrieval.