

# Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[Draw It or Lose It 1](#_Toc108818358)

[**CS 230 Project Software Design Template** 1](#_Toc108818359)

[Table of Contents 2](#_Toc108818360)

[Document Revision History 2](#_Toc108818361)

[Executive Summary 3](#_Toc108818362)

[Design Constraints 3](#_Toc108818363)

[System Architecture View 3](#_Toc108818364)

[Domain Model 3](#_Toc108818365)

[Evaluation 4](#_Toc108818366)

[Recommendations 6](#_Toc108818367)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/15/2022 | Eric Slutz | Wrote the Executive Summary, Design Constraints, and Domain Model sections. |
| 1.1 | 07/26/2022 | Eric Slutz | Filled out the Evaluation table. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wants a web-based gaming application called Draw It or Lose It. It is loosely based on an 80s gameshow called *Win, Lose or Draw*. The team at The Gaming Room are not familiar with setting up the needed environment. The development team at CTS will help to simplify the development of the web-based version of the game application. Additionally, hardware requirements will not be assessed until after the software application decisions.

## [Design Constraints](#_2et92p0)

* The application is written in Java
* The application is web based
* Only one instance of the game can exist in memory at any given time
* Ability for one or more team to play
* Multiple players on each team
* Game and team names must be unique
* The game consists of four rounds of play lasting one minute each
* Drawings fully render at a continuous rate for 30 seconds
* Images sourced from a large library of stock drawings
* The first team has those 30 seconds to guess correctly
* Otherwise, remaining teams each get 15 seconds to make one guess

The above listed design constraints include the client requested software requirements. Furthermore, there are constraints dictated by the nature of the game and how it is supposed to be played. Lastly, given the web-based nature of the game application, those constraints need to be considered as well. The implications of these design constraints on the development process are that they help to set clear expectations for the application and how it is expected to work. The constraints will work to ensure that the client’s needs are being met.

## [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 shows that the Game, Team, and Player class all inherit the Entity class; thus, showing the OOP principle of inheritance. Those four classes, along with the GameService class all exhibit encapsulation through using private attributes with public methods to access or modify them. Additionally, there are zero to many relationships between many of the classes. Such relationships exist between Team and Player, Game and Team, and GameService and Game. There are multiple examples of abstraction throughout the diagram as well. A few of them include addPlayer(), addTeam(), and addGame(). The process of adding these different objects is hidden from the user; all they see is that the object has been added. Finally, the UML Class Diagram show the relationship of the ProgramDriver class using the SingletonTester class.

**"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** | * It is possible to use macOS as a server to host a web application. * You would be required to use Apple hardware which can be cost prohibitive. * There is no licensing cost for macOS (comes with the hardware).   Due to cost, would be difficult to scale up for more users. | * It is possible to use Linux as a server to host a web application. * There are many varieties of Linux available from general use to dedicated server versions. * Licensing fees vary from free to having yearly fees. * Overall, licensing can be much cheaper with this option.   Can use commonly available hardware. | * It is possible to use Windows as a server to host a web application. * A specific version of Windows, Windows Server, should be used as it’s designed for this purpose. * Of all the options, this has the highest licensing fee for any version of the Windows OS. * Licensing costs could make scaling up cost prohibitive.   Can use commonly available hardware. | * It is technically possible to use a mobile device as a server and host a web application.   Cost of scaling up and the number of devices needed, reliability, and performance are just the tip of the reasons why you shouldn’t. |

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Client Side** | * The cost of developing a web app for this platform is shared with the other desktop OSs. * This, with the other desktop OSs, are the most cost effective because you get to cover 3 operating systems with one browser-based client. * Must ensure compatibility with major browsers. * Slight additional cost, skill, and time to include Safari (only available on Mac). | * The cost of developing a web app for this platform is shared with the other desktop OSs. * This, with the other desktop OSs, are the most cost effective because you get to cover 3 operating systems with one browser-based client. * Must ensure compatibility with major browsers. | * The cost of developing a web app for this platform is shared with the other desktop OSs. * This, with the other desktop OSs, are the most cost effective because you get to cover 3 operating systems with one browser-based client. * Must ensure compatibility with major browsers. | * Additional time to developer a client for mobile devices. * Different skills needed to develop client for mobile devices. * Within mobile devices there is further division between iOS and Android * Different skills needed between iOS and Android clients. * The different skills and additional time will increase cost to develop client. |
| **Development Tools** | * For server development macOS can handle any language that is used for web app development. * For client development macOS can handle any language that is used for web app development. * Most IDEs have versions compatible with macOS. * iOS apps are typically developed with Swift. * Xcode, VS Code, or AppCode are IDEs that can be used for iOS development. * Java or Kotlin are primarily used for Android development. * Most major IDEs can be used for Android development. | * For server development Linux can handle any language that is used for web app development. * For client development Linux can handle any language that is used for web app development. * The selection of commonly used IDEs for Linux is limited. * iOS apps are typically developed with Swift. * VS Code can be used for iOS development on Linux, but it is a limited experience. * Java or Kotlin are primarily used for Android development. * Android studio can be used for Android development. | * For server development Windows can handle any language that is used for web app development. * For client development Windows can handle any language that is used for web app development. * Most IDEs have versions compatible with Windows. * iOS apps are typically developed with Swift. * VS Code can be used for iOS development on Windows, but it is a limited experience. * Java or Kotlin are primarily used for Android development. * Most major IDEs can be used for Android development. | * It is not recommended to do development on a mobile device. * Both screen size and computing power would limit the ability to do work. * While there is a browser-based version of VS Code and some other online IDEs they’re still designed for a desktop experience. * A lot of the languages are not available for mobile, so if you write code you will not be able to compile, build, or test the code. |

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>
2. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>
3. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>
4. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>
5. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>
6. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>