

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## 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 | 12/18/22 | Larry Jones | Recommendations updated |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wants to bring an existing web-based game, Draw It or Lose It, to multiple platforms. The game is currently only available on Android. The premise of the game is guessing random pictures, and is played with multiple teams guessing with a time limit.

## [Design Constraints](#_2et92p0)

• One mor more teams involved

• Unique names for each player, and check whether name currently in use

• One instance of the game can exist at a time

• Must be able to run on various platforms; universal application

These are requirements for the game itself, as a guide for software features. Software development aspects need to be formalized. The Gaming Room wants to expand to universal app capabilities for this game, to be able to run on various platforms and 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)

The UML shows objects and classes within a system, and relationships between them. In this UML, Entity creates a relationship between Team, Player and Game, and the three in turn inherit information from Entity. Each class will share common references. GameService has a reference to Games, Games in turn a reference to Team, and Team has a reference to Player.

**"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** | Flexible terminal commands for configuration. Upgradable, with various choices for web hosting, but less preferred for web hosting. | Advantages in the realm of security, and is most preferred for web hosting, but difficulty in finding applications that support web hosting features. | Most widely used OS worldwide. High requirements for resources, but less loading time. Disadvantages with virus susceptibility and tech support. | Server operation is better if immobile and tracked as stationary. Mobile devices represent a wider reach but selectivity for mobile platforms is a constraint. Also, security is a variant feature. |
| **Client Side** | Moderate level of expertise and time involved. Cost is similar to that of Windows. An open question to what is required in to ensure compatibility with all web browsers and mobile devices. | Higher level of expertise and time involved. Minimal cost. An open question to what is required in to ensure compatibility with all web browsers and mobile devices. | Minimal level of expertise and time involved. Cost is similar to that of Mac. An open question to what is required in to ensure compatibility with all web browsers and mobile devices. | Flexibility for clients and developers as far as updates involved, but more difficult in regard to implementation than other types of devices. |
| **Development Tools** | Tools including Swift and Notepad++. Languages consist of but not limited to HTML/CSS/  JavaScipt, while frontend and general-purpose languages include Java, Python, PHP and Ruby. | Tools include Visual Studio, Eclipse, Notepad++.  Languages consist of but not limited to HTML/CSS/  JavaScipt, while frontend and general-purpose languages include Java, Python, PHP and Ruby. | Tools include Visual Studio, Eclipse, Notepad++. Languages consist of but not limited to HTML/CSS/  JavaScipt, while frontend and general-purpose languages include Java, Python, PHP and Ruby. | Using Android and Swift app creation is easy.  Languages consist of but not limited to HTML/CSS/  JavaScipt, while frontend and general-purpose languages include Java, Python, PHP and Ruby. |

## 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**: Begin with Windows as a starting platform to branch to. Windows has more software available, and numerous IDE’s to work with. This platform will also involve minimal expertise and cost. It is the most widely used platform.
2. **Operating Systems Architectures**: Windows architecture allows utilization of kernel processes without affecting applications that use it. It provides ample services with GUI capabilities and access to system resources. Server accounts include graphics, multimedia, messaging and other services.
3. **Storage Management**: Windows is a failsafe ecosystem that offers storage sense as a way manage files on hard drive, as well as cloud backup and recovery. Disk Management, Storage Sense, and Disk Cleanup are built in tools that help with management and maintenance of storage.
4. **Memory Management**: A dedicated database for storage of photos and other applicable media. This provides a secure and centralized area for access of these files, and can be used in conjunction with your favorite idea when opening and referencing files. Windows does have Memory Management as a built-in utility that can be used.
5. **Distributed Systems and Networks**: Use a system that enables cross-platform game creation, that in turn can export to various platforms that allow for import and cross-play if possible. Also, make sure server capabilities are sufficient for large player volumes to ensure continuous connectivity. A client-server distribution system could be used so that individual client’s system strengths could be utilized for development.
6. **Security**: Windows has native security features such as Windows Defender that are sufficient for purposes of security on windows devices, but other sources will be required for other platforms. It’s recommended to follow platform specific protocol for security of data, which may include antivirus tools if needed. Encryption will be needed for data that is sent and received.