

Draw It or Lost 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 | 03/19/23 | Jordan Dayedes | Added Entity class. Game class inherits from Entity class |

**Instructions**

Fill in all bracketed information on page one (the cover page), in the Document Revision History table, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is developing a web-based game called “Draw It or Lose It” and is currently only available on android. They are looking to release on multiple platforms. The game works in four rounds lasting a minute each. Drawings are rendered at a steady rate, and finish at 30 seconds. If a team does not guess the puzzle before the time expires, the remaining teams can offer one guess each to solve the puzzle with 15 second time limits.

## Requirements

- The game needs to be played with one or more team.

- Each team consists of multiple players.

- Only one instance of the game can exist in memory at any time.

- Each game and each team will need a unique ID in order for users to check if a name is in use.

- Must be multiplatform.

## [Design Constraints](#_2et92p0)

The game must be ported over to other platforms by creating a web-based version of the game. We’ll write the back-end systems in Java, which will allow the client flexibility in creating the front end game systems later. We will only be providing the back-end environment for the game.

## [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)

Entity creates relationships between Game, Team, and the Player Class. This means that Entity is the superclass, while the three subsequent classes inherit from the Entity class. GameService contains the singleton object and thus is the initial instance of the game all other classes are based on.

**"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** | Server access is made easy thanks to powerful terminal commands and tools. | Similar benefits to the Mac, but with more cost savings. More compatibility and flexibility than other platforms. | Software tools are widely available thanks to robust network of users and developers. | Major disadvantages to security and tools powerful enough to enable server stability and performance. |
| **Client Side** | Moderate knowledge and time required. Cost will be close to Windows. | The most knowledge and time required. However, costs will be low. | Moderate to low knowledge and time required. Cost will be similar to Mac, and moderate. | Maximum flexibility to both clients and developers. Can be difficult and time consuming on initial set up. Moderate to low costs. |
| **Development Tools** | Common Languages on Mac include HTML, CSS, and JavaScript. As well as major Languages such as Python, Java, and Ruby on Rails. Tools such as PyCharm, Eclipse, Visual Studio, GitHub. Deeper tools such as Xcode, clang and gcc. | Same basic common languages and tools as Mac. Out of the box support for installing packages on a wide swathe of languages and tools. | Same basic common languages as other platforms. The .net framework is a great tool for specific development needs. | Flexibility due to portability. It would be best to develop mobile apps on other platforms that have native support. Use Mac for iPhone devices. Use Linux for android support. |

## Recommendations

1. **Operating Platform**:

Draw It or Lose It will need to be deployed on a platform that is already easily accessible from most operating platforms. That’s why I recommend using a web-based platform, by utilizing tools such Node.js, or react.js, or Ruby on Rails. These frameworks allow for the team to quickly build modular systems for a web-based application, which will afford us freedom to easily expand the game when needed.

1. **Operating Systems Architectures**:

The application will need to be stable on multiple operating systems. By using Java, the team can write secure, and modular code that will be cross compatible across multiple operating system architecture thanks to the Java Virtual Machine. The application will need a lightweight version created for mobile devices. This version of the game will need a UI specifically designed for touch screens, as well as attention paid to memory management on these devices.

1. **Storage Management**:

Game assets need to be stored on a cloud-based storage service. The back end of the application can then use a hash table, that way the front end of the application can call assets upon loading a new instance of the game efficiently. When running on full desktop platforms, full original image files can be used. However, on the mobile version, I recommend using a separate hash table that will lower resolution images.

1. **Memory Management**:

Since the game is a web-based application, any assets used every time the game is loaded will need to be stored in system cache. This will help speed up subsequent load times, as well as

have a positive impact on performance.

1. **Distributed Systems and Networks**:

Considering the application will have multiple instances running concurrently on many different

platforms. A system such as load balancing technique algorithms will help create a stable

environment from which game clients can make requests for back-end application assets.

1. **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.>

Preventing unauthorized access to critical application assets will be key. By ensuring all assets

stored in the cloud service is set to read only key game assets will remain secure. As well as

requiring authentication for the client side of the application. Lastly, the application needs to

periodically perform security checks, and updates to ensure all security standards are current.