

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 | 06/18/2023 | Joshua Brown | Added Specifications for “Draw It or Lose It” |

.

## [Executive Summary](#_sbfa50wo7nsh)

Our client The Gaming Room wants to develop a cross-platform web-based version of their mobile app “Draw It or Lose It,” which is currently only available for Android. The application’s game will be four rounds long; images from a library will be rendered over a period of 30 seconds; players on a team must guess before the time limit or the remaining teams will have an opportunity to offer one guess to solve the puzzle.

## Requirements

* Web-based game application
* Designed the game after their current Android App Game.
* A game will have the ability to have one or more teams involved.
* Each team has multiple players.
* Game and team names must be unique.
* Only one instance of the game can exit in memory at any given time.

## [Design Constraints](#_2et92p0)

* This web-based game will need to run on all platforms.
* Should have the same look and feel as the Android App version of the game
* Since the code is already on Android, we may have to consider writing the code for this web application in other languages that can be used by other OS; however, Java is a pretty versatile programming language, and with some additional tools (such as a cross-platform framework or an iOS port) we may be able to allow the application to run on all platforms. Since the application is already being written in Java, I’d recommend first trying to implement those tools for cross-platform capabilities. If that doesn’t work out, then we may have to consider rewriting the code in another language.

## [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)

In the UML diagrams, Entity serves as the base class or superclass for the Game, Team, and Player classes. To reduce redundancy, the commonly shared variables and methods that were included between the Game, Team, and Player classes were all incorporated into a superclass called Entity. By having these subclasses inherit from the Entity class, the common methods can be used without needlessly repeating them, which will cut down on errors, and allow for the subclasses to focus more on code that is unique to their specific class. GameService has a zero-to-many association with Game; Game has a zero-to-many association with Team, and Team has a zero-to-many association with Player. Lastly, the ProgramDriver houses the main and uses the SingletonTester to test that the program only initiates one instance of a game at a time.

**"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** | **Pros:** -Tend to be more secure; less likely to be hacked  -Can use open-source servers, such as Apache.  **Cons:** -The most costly option.  -Not the preferred web hosting option. | **Pros:** -The cheapest option.  -Free software from open-source library.  -Easily modifiable  **Cons:** -The least secure; open source doesn’t mean secure.  -Web host will have to incorporate their own security measures. | **Pros**: -Extensive amount of applications and tools not available for other platforms.  -Cost is roughly in the middle.  **Cons**: -Due to popularity, it is the most prone to cyber attacks. | **Pros**: -Very cost efficient  -Broader reach; many people have mobile devices vs computers.  -Very popular  -Normally cloud based  **Cons**: -Not very secure. |
| **Client Side** | **Consideration:**  -Moderate to little know how.  -Frequent updates make it more secure on the client side  -Require up-to-date hardware to run the applications. | **Consideration**:  -Extensive know how.  -Requires more experience.  -Less Resources required to run. | **Consideration**:  -Very little expertise needed  -Requires less experience  -Frequent updates and maintenance to prevent crashing. | **Consideration**:  -Very little expertise  -Provides flexibility for users  -Ease of access; can often be access offline depending on the application.  -Requires frequent updates; however, updates are easier than most. |
| **Development Tools** | **Languages:** Swift is the primary and preferred language; however, other languages such as C# and Java are possible with the use tools.  **Popular IDEs**: Visual Studio Code, Eclipse, XCode | **Languages**: Can use most languages (C, Java, Python, etc.)  **Popular IDEs**: Visual Studio Code, Eclipse, IntelliJ Idea | **Language**: Can use most languages (C, Java, Python, etc.)  **Popular IDEs**:  A plethora of options: Visual Studio, Eclipse, etc. | **Language**: Can use the most common languages with tools (swift, java, C, C#, etc.)  **Popular IDEs:**  Android Studio, Qt IDE, Eclipse, Visual Studio 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**: Considering that the game already exists on mobile devices (for the Android), I would recommend that The Gaming Room host their web-based application on Windows. The benefits are that it wouldn’t cost as much to host as it would on Mac, and it will require less experience and know-how than that of Linux, preventing mistakes that would lead to vulnerabilities in the web-based application.
2. **Operating Systems Architectures**: Windows offers more when it comes to tools and IDEs to make sure your application has everything you need. Hosting on Mac might limit some tools that we need to make the application run on multiple platforms.
3. **Storage Management**: Windows storage management is a storage system that comes with Windows and has several features such as Storage Spaces (lets you combine physical hard drives into a local one), Storage Optimization, Storage Sense, and more. OneDrive is also a good cloud-based storage service that you can download regardless of which OS you are using. As a result of its cloud-based storage, files are synced across multiple devices, allowing for a team of people to all have access to the same files, making work done in teams even more efficient.
4. **Memory Management**: By storing the images for the application in a folder on your local hard drive, the application can be pulled from the file to use in the application, whether it be through actually reading the file from the application or through memory mapping to the file.
5. **Distributed Systems and Networks**: By using a framework such as Codename One (a free open-source framework that works with all IDEs) you can build an application using Java and it will be able to run on any OS. It even has a neat feature to allow you to check out the look and feel of the application as it would be on a different platform. Once the game is built, it can be accessed online from a web address. A way to prevent some outages is to plan for the amount of traffic that will be on the server. By using the already-developed Android App, as a reference, you might be able to determine just how much traffic your server will receive and can plan accordingly. The use of a Content Delivery Network or CDN, will also be a great idea to deal with unexpected high-volume traffic or DDoS attacks by sending users to other servers.
6. **Security**: Some ways to make the web-based application more secure include: making use of CDNs to prevent DDoS attacks, practice secure coding principles to limit cyber-attacks (such as input validation and SQL injection), install SSL plugins and https protocol, don’t allow user to upload anything into your website (especially if there is no need to), choose a reliable web hosting provider, and make sure any included software, security tools or plugins being used are up-to-date.