

# 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) 3-4

[Evaluation](#_2o15spng8stw) 5

[Recommendations](#_m8aleynsvzvc) 6-7

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/24/2022 | Tyler Morgan | Updating Development requirements |

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 wants to develop a web-based game (Draw It or Lose It) that serves multiple platforms based on their current game, Draw It or Lose It, which is currently available in an Android app only.

## [Design Constraints](#_2et92p0)

* Needs one or more teams involved
* Each team has multiple people
* Game and Team names must be unique to allow users to check whether the name is in use or free
* Only one instance of the game can exist at any time.
* Must run on multiple platforms

The main constraint will be taking this from being solely available on Android, to make it work across all other platforms. We can change the code for each platform or integrate different language files into one IDE.

## [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 a relationship between Game, Team, and Player class. This means they all inherit or get information from Entity. The UML shows this with inheritance. Each class will share common references like “name” and “id”, making Entity a superclass. When we look at their relationship, we see Team and Player is a “has a” type. While Game has teams and GameService has games. When a class “has a” it’s an instance of one class and has a reference to an instance to another class, it’s using aggregation. When we look at this diagram, we see GameService has a reference of Game, Game has a reference of Team, and Team has a reference of 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:

| Development Requirements | Mac | Linux | Windows | Mobile Devices |
| --- | --- | --- | --- | --- |
| Server Side | Flexible terminal commands to configure the server, access, or make changes. | Same as Mac but more cost friendly. | Windows has the most software available compared to the other operating systems. | Make the mobile app secure and easy to track. |
| Client Side | Moderate experience and time required. Prices comparable to Windows OS. | High level experience and time required. Lower prices than Mac and Windows. | Lowest level of experience and time required. Price comparable to Mac IOS. | The most difficult OS to implement but provides the most flexibility to clients and developers for usage and updates on the go. |
| Development Tools | Common languages: HTML/CSS/JavaScript and supporting libraries to support the frontend and general purpose programming languages like Python, Java, php, Ruby on Rails. Tools: PyCharm, Eclipse, Visual studio, Github. These languages/tools are available for all OS.  Cmake, Xcode | Common languages: HTML/CSS/JavaScript and supporting libraries to support the frontend and general purpose programming languages like Python, Java, php, Ruby on Rails. Tools: PyCharm, Eclipse, Visual studio, Github. These languages/tools are available for all OS.  GCC | Common languages: HTML/CSS/JavaScript and supporting libraries to support the frontend and general purpose programming languages like Python, Java, php, Ruby on Rails. Tools: PyCharm, Eclipse, Visual studio, Github. These languages/tools are available for all OS. | Common languages: HTML/CSS/JavaScript and supporting libraries to support the frontend and general purpose programming languages like Python, Java, php, Ruby on Rails. Tools: PyCharm, Eclipse, Visual studio, Github. These languages/tools are available for all OS.  Mobile C |

**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: The easiest platform to start on and branch off, would be Windows. Windows has the widest variety of IDE’s while being the easiest and cheapest platform to use. Using Windows as a starting platform will allow us to create different versions for other OS easier.
2. Operating Systems Architectures: The architecture of Windows gives us the ability to use all Windows based apps. Windows based apps (along with other OS) use GUI’s, or Graphical user interfaces, to show us what tasks we are trying to perform. While using this architecture we can set up users on an individual platform or run it off a server.
3. Storage Management: Windows has an easy storage system. It allows you to create files, folders and shortcuts. You can pin or place these items in other folders or on your desktops to stay organized. You can select any existing folder to store files or create a new sav location so you can separate different types of files. Along with all the options for local storage, Windows 10/11 allows you to use cloud storage, so it doesn’t affect the storage capacity on your local hard drive. You can also use the cloud as a backup in case a file gets deleted by mistake.
4. Memory Management: As stated above in storage management, you can use your own specific locations to pull files from. With memory management options, we can place all our picture files and whatever else we may need immediate access to, in a folder with the rest of our project. This will allow direct integration and remove any possibility of errors from reading files or not being able to find them. Having adequate cache space will allow for faster CPU times and less stress on the system overall.
5. Distributed Systems and Networks: Since we are using Windows as a base platform and need to enable cross-play for other OS (iOS, Linux, mobile), we need to find an IDE that will be easy to convert code or even just run the same code across all platforms. My research has found that we will need to install all the proper SDKs for cross platform development. Also, using a game engine such as UNREAL ENGINE, UNITY, CRYENGINE, etc... would help make the game run smooth across different operating systems. While having all these platform integrations, we must ensure our servers are strong enough to run these games together simultaneously, with large user volumes. Our server will be backed up by additional servers to ensure it can reboot and run, in case of an outage. Back up servers will have all proper security measures and storage requirements in place.
6. Security: Although Windows has a built-in security system, it is recommended to install an anti-virus and anti-malware on your own device. All users should take responsibility for their own security and privacy. Creating a strong password is always the first sgtep to personal security. Enabling 2FA, or two-factor authentication is a great step to ensure added security. As a game running on a server, we will make sure our servers stay secure by taking all necessary measures to protect our user’s information that we have stored. Keeping a database of usernames and passwords can be dangerous if not properly secured. It could allow outsiders to access personal information and could hold us liable for the breach. All users and developers will have certain permissions to only allow access to what they need. Installing a firewall and keeping the environments separate will help ensure that everyone is staying within there secure zones. We will stay up to date will patches and have automated updates to ensure all bugs are fixed within a timely manner.