

<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 | <3/18/2025> | <ConnorCamire> | <First draft of the Draw it or lose it game> |

**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)

Originally Draw it or lose it, was an app that was developed exclusively for android. It is now planned to create a web-based version of this application. So that any user with access to the website would be able to play with any other user including those playing on the android application. User can create teams consisting of multiple members. With games with more than one team and participants at the same time. Keeping only one instance of the game in memory at the time. Having each user access the same instance of the game. To keep progression synced.

## Requirements

1. The game must handle multiple teams/ team names. Ensure that two shared team IDs cannot exist at the same time, But still prevent two team with a shared name from being created as well.
2. Each team can have a number of player assigned to them. Again going of over player’s ID so that two plays of the same name can exist on a team. But not the same player twice(Note that player can not create the same name as another but ID still is necessary in case, it also allows player to change there names without issues).
3. Game and Team names need to be unique not only in ID but also name name.
4. Only one games instance can exist at a time. Each game will be given a unique ID to ensure no duplicates are made.

## [Design Constraints](#_2et92p0)

1. Currently since the application currently has a android version, we need the communication between the two to be seamless, That is the games can not de-sync from the web-based player’s games. And user information must be synced between mobile and web as well.
2. Game should handle multiple players and teams in a single match.
3. Game instances, team names, and player names. Should all have unique ID to prevent conflict.
4. Ensure that the web-appreciation works on most web browser.

## [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 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.

**Entity is a parent class for all of Game,Team,Player. Showing Inheritance, through the variable of id and name being shared in all three children.**

**Game Service works with 0 to many amount of Entity’s children. Which show polymorphism in how each game can have a different number of team or players, while still functioning as normal.**

**Program Driver Uses Singleton Tester. Shows Encapsulation by ensure that only one instance of each game is every being created.**

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | <Limited usage for hosting servers, but stable.> | <Famous for web servers, stable, recommended use.> | <not very stable, low recommendation.> | <Best not used for server hosting.> |
| **Client Side** | <Works on supported web browsers.> | <Works on supported web browsers> | <Works on supported web browsers> | <Android version already available, IOS could be considered, Must work with web version > |
| **Development Tools** | <IntelliJ IDEA ,VSC,Eclipse,java SDK,Spring Boot > | <IntelliJ IDEA ,VSC,Eclipse,java SDK,Spring Boot > | <IntelliJ IDEA ,VSC,Eclipse,java SDK,Spring Boot > | <Android Studio, web frameworks > |

## 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. We should use a Linux based server, thanks to stability and cost effectiveness. A common issue that arise with Linux, is that of application support. But luckily for Linux for the sake of a server does not suffer this issue. As tools such as MYSQL, just about every programming language, server tools such as Apache and NIGNX. Are all well supported on Linux operating platforms.

1. This application will use Linux for the server/back end and its front end will be on web browsers. Linux has a monolithic and modular kernel. This means each function, such as memory access drivers, the file system and any other core functions. will be handled monolithicly resulting in a server and back end that is faster than other kernel types. Since each core function will be handled in it's own kernel space.
2. As mentioned prior. Linux is well supported with various server related tools. Such as the database MySQL. This is were we will store all information retain to a user and an instance of the game. A possible example of the variable names being game ID, player names/IDs, and Team names/IDs. Information is gathered from the user and sent to the server where the information is stored. Depending on the type of data it will determine if the data will stay on the servers. Or should be temporarily store for lets say one month. An example of this, would be saving drawings made by the player’s for one month so they can access them up to a month later, should they want to save them on their own device.
3. Since we are using a singleton pattern. We can prevent multiple instances of the same game. Linux utilizes virtual memory, which is important for keeping the server’s ram from overflowing. This is if our servers ram is ever full. This would result in at best the server slowing down and most likely the server crashing. We will want to use an OS that can virtualize memory, so that in the case of the ram filling up information can temporarily be stored on the server’s SSD/HD instead.
4. Utilizing APIs to sync data between mobile and web browser. Mainly RESTful API. While communicating between our server's back end which will handle the main logic of the game. Receiving commands from the front end, the user's game instance. Which will be sent to and handled by the server’s game logic. Then the information will be return to the user's machine.

1. We can use HTTPS end to end encryption to secure all of the user's credentials while information is being sent to and from the server. Protecting the information in the case of interception. Making sure that you never store critical information server’s side or local without using an encryption method such as Hashing. Only storing critical information on user side when absolutely necessary.