

# 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 | 09/19/2021 | Christian York | Initial creation |

**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)

CTS team is looking to create a web-based Draw It or Lose It team-based game.

## [Design Constraints](#_2et92p0)

The CTS team does not know how to setup the environment to facilitate the development of this game, and will require our help to get it up and running.

What we need to be able to play the game is a system that accomplishes the following:

* Only one instance of each game can exist at a time, but many game instances may exist
* Each game will have a set of teams playing
* Each team will have a set of players assigned to the team
* The same team may be able to play multiple games
* The same player may be able to be on different teams across separate games

## [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)

ProgramDriver class contains the main() method, which creates the games, teams, etc., as well as using the SingletonTester class in order to ensure that only one instance of a particular game exists at a time.

Game, Team, and Player classes all inherit from Entity class and use the Entity class structure as a base to add on to with more specific information about games, teams, or players. GameService class creates the individual instances of the games, with teams. There is a zero to many relationship between GameService, Game, Team, 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** | **Pros**: High Security levels built in, state-of-the-art hardware with high processing speeds  **Cons:** High cost | **Pros:** Low cost, high security, runs on almost any hardware  **Cons:** Requires a certain level of expertise to use, with limited support | **Pros:** Microsoft proprietary software is widely used and well documented  **Cons:** Not as secure as Linux or mac | I have never heard of using a mobile device as a server. |
| **Client Side** | Any operating system with an internet connection can host the client-side code on a supported web browser or a dedicated app | Any operating system with an internet connection can host the client-side code on a supported web browser or a dedicated app | Any operating system with an internet connection can host the client-side code on a supported web browser or a dedicated app | Any operating system with an internet connection can host the client-side code on a supported web browser or a dedicated app |
| **Development Tools** | OS specifically designed for creative tasks and to be easy on the eyes  Java and other open-source languages are easily developed with many IDEs available  .NET development can also be done easily with either virtual Windows machines or using tools such as Boot Camp | Linux is a very developer-friendly OS, with the capability to customize it to your individual liking  IDEs such as Geany come packaged in with many Linux distributions and can handle a variety of languages | Excels at developing with .NET languages using the Visual Studio IDE.  Developing with Java and other open-source languages is also easily completed with many available IDEs depending on the language | There are not many development tools for mobile devices, as mobile apps are typically developed on a PC running one of the other operating systems |

## Recommendations

1. **Operating Platform**: Windows Server 2019
2. **Operating Systems Architectures**: multi-tier architecture with one tier hosting the web-based app code and files, and another hosting various services called by the game, and another for the database which will hold game/team/player historical records and account information.
3. **Storage Management**: at least 250 GB of storage on Solid State Drives for reduced hardware failure risk and faster disk read speeds.
4. **Memory Management**: at least 16 GB of high-speed RAM to accommodate the potentially high volume of user traffic accessing the game and its associated files.
5. **Distributed Systems and Networks**: The server(s) will be physically hosted at a local server farm with dedicated high-speed internet connections and technicians available to troubleshoot any hardware issues. The Gaming Room will have access to these servers via VPN and remote desktop software. The servers will have 24/7 monitoring to ensure they are online and functional at all time.
6. **Security**: The servers will only be accessible at the local server farm or via a secure VPN connection. All traffic to and from the servers will go through a firewall and only specified ports will allow connections. The servers will have anti-virus/malware software running at all times scanning for intrusion attempts. All administrative accounts on the machines will have 2-factor authentication enforced with every login attempt. Players will need to set up an account with a username and password and optional 2-factor authentication. Player’s personal/identifiable information will be encrypted in the database and decrypted once it is sent to the client side.