

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 | 2/4/2024 | Madison Marbach | First Version |

**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)

Creative Technology Solutions took on a new client called The Gaming Room and was asked to take a game, Draw It or Lose It, which is an Android app only and develop a web-based game that serves multiple platforms.

The game needs to allow multiple teams to vary in size and it needs to render images and each team will have 30 seconds to guess what it is. A singleton pattern will be used to prevent multiple game instances and an iterator pattern will be used to prevent conflict of teams and/or members of the team.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

Because this is already an Android app, the web-based game needs to run similarly to the app.

There needs to be an option to have multiple teams, and the teams need to be able to vary in the number of players.

Only one instance of the game can exist at any time and unique identifies need to be available to use for each instance.

## [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 UML class diagram shows the different classes that com.gamingroom consists of.

The program driver is a main driver that creates games through the GameService class.

GameService is the class that initiates creating the game, teams, and players and it follows a singleton design so only one GameService class can exist at a time which prevents multiple game instances.

The Game class allows a unique ID to be created for the game and allows teams to be added to the game.

The Team class allows a unique ID to be created (team name) and allows players to be added to the team.

The player class allows a player to be added by a unique ID (name).

Entity is made up of the subclasses, Game, Team, and Player. That means that Entity is the parent class and is inherited by the Game, Team, and Players classes. Each of the subclasses inherits properties from the entity class and can add their own public and private functions.

**"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)

Using your experience to evaluate the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices, consider the requirements outlined below and articulate your findings for each. As you complete the table, keep in mind your client’s requirements and look at the situation holistically, as it all has to work together.

In each cell, remove the bracketed prompt and write your own paragraph response covering the indicated information.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Characteristics:  -More stable environment  -User-friendly UI  -More reliable for server environment  Advantages:  -Well suited for web hosting  Weaknesses:  -Software options are limited.  -Potential licensing issues | Characteristics:  -Powerful and efficient platform for servers  -cost effective  Advantages:  -Customizable  -cheaper than Mac or Windows  Weaknesses:  -Not as much support  -requires more knowledge to setup | Characteristics:  -User friendly  -cost effective  Advantages:  -User friendly  -Easy to navigate interface  Higher performance  Weaknesses:  Higher Licensing costs | Characteristics:  -Portable and requires cloud services  Advantages:  -Cloud hosting allows higher scalability  Weaknesses:  -Limited control over server hosting enviroment |
| **Client Side** | Considerations:  Cost and time are relative, the more time the higher the cost can be. May require more testing due to safari.  Development requires web technologies and responsive design | Considerations:  Similar development as Mac with web technologies. More compatibility testing is needed meaning more time and costs. | Considerations:  Standard web design practice. Both cost and time are low because of standard web design. | Considerations:  Needs to have knowledge in mobile app development which is more specific meaning cost and time can be higher. Because mobile has multiple platforms, more testing needs to be done. |
| **Development Tools** | Languages and IDEs:  Web technologies such as HTML and JavaScript. Commonly used IDEs suchs as Visual Studio. | Languages and IDEs: Web technologies such as HTML and JavaScript. Commonly used IDEs suchs as Visual Studio or JetBrains. | Languages and IDEs:  Web technologies and IDEs can be Visual studion | Mobile app development like Java for Android and Swift for iOS  IDEs can be Xcode or Android studio. |

## 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**: I think due to how common, and the affordability of Windows makes it the best operating platform. Along with the vast number of tools and open environment to test applications makes it more diverse than the other platforms.
2. **Operating Systems Architectures**: Because Windows OS is a very common OS there are many people or websites that can help troubleshoot issues that arise making it easy to set up the platform customized to personal liking. Windows is also very user friendly and allows features to be used interchangeably.
3. **Storage Management**: Cloud-based storage is recommended for our client, with windows there is Windows Azure, but any cloud-based storage is recommended.
4. **Memory Management**: Windows is good at memory management as an OS because it can sort similar data together and has extra RAM, so some data is only retrieved “as needed” which reduces memory storage.
5. **Distributed Systems and Networks**: Multi-interaction systems that are network based usually must come up with original designs for player communications and data sharing. Some type of database will be used to share the data amongst players and for individual users to recall data. Microsoft Azure can store all this information at a scalable factor as it can grow with client demands. Azure has cloud-based email alerts, monitoring, and logging solutions offered. This will take the bulk of the network load from the client to focus on development and features.
6. **Security**: Security will always be an ongoing issue for whatever type of platform we use. Using windows, we can create a personalized database, and infrastructure to implement firewalls, and encryption. The ability for the company to be able to maneuver and respond as any type of hacker is incredibly important, and windows offers this versatility, in addition to teams at Microsoft that can offer support as well.