

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 | 06/04/2023 | Joshua Lewis | Continued backend development |

**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 client wants only one instance of a game to exist in memory at any given time. Each game will have only one or more teams involved and each team will have multiple players. Game and Team names must be unique. The client needs us to develop the web-based version of the game app. To solve these issues, I suggest creating an Entity class from which the players and team are derived. When adding a team or player name an iterator should be used to verify the existence of an already present Team or player name. The use of a singleton should be used for each game to avoid multiple instances existing in memory. I would also choose a web-friendly language (java) which can also be used on multiple platforms.

## Requirements

* *Each game must have the ability to have one or more teams*
* *Each time must be able to have multiple players*
* *Each game and team must be unique*
* *Only one instance of the game can exist in memory at any given time*
* *Must be web-friendly*
* *Must be able to be used on multiple OS*

## [Design Constraints](#_2et92p0)

* *Game must consist of 4 rounds lasting 1 minute*
* *Drawings are rendered complete at the 30 second mark*
* *Team must guess puzzle before timer expires*
* *If team does not quess, remaining teams have an opportunity to guess with a 15 second time limit*

*These constraints will allow us to start planning future development for the game. This will help us design the game the way the client intended and ensure that the program functions as expected.*

## [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 Diagram below shows the relationship between the Entity classes and the Game, Team, Player classes as well as the GameService Class. The entity class creates a new entity with an ID and name. A Player, Team and Game class all extend the Entity class. A player can be part of a team and a team can be part of a game but a game cannot be part of a team and a team cannot be part of a player. The game Service class extends the Game Class and contains the methods for accessing and creating a new game.

**"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** | Mac has flexible terminal commands to easily configure, access and change the server. Mac has X server which is used to configure, deploy and manage network services. | Most flexible and cost friendly of the three options Linux is great for server hosing. Like mac it uses a command shell for server configuration and access. | Widely available and commonly used, Windows has plenty of server hosting options available (Windows Server and datacenter edition) and is cheaper than Mac in terms of performance and cost. | Servers are better if they are immobile and tracked to a sole location. I would not recommend the server be hosted on a mobile device. |
| **Client Side** | Mac OS is the first option between Windows and Linux. MacOS requires more expertise than windows, takes slightly more time for development and is more expensive than Windows and Linux. | Linux is the more complex yet cheaper long-term option. Linux is not as popular as Windows or MacOS. Linux users need maximum time and skill to support Linux systems. | Cost is more expensive than Linux but is comparatively cheaper than MacOS. Windows is the most accessible and time friendly of the options. Users do not need to be as technically experienced as Linux. | Mobile OS like Andrioid and iOS are difficult to port between PC OS and mobile devices. The mobile platform does provide flexibility and convenience to users. Apps are slightly more difficult to implement on mobile than on PC. |
| **Development Tools** | Mac for web development can use Swift, HTML, JavaScript and CSS. IDE’s can be for Python, Javascript, PHP, Ruby and Java. IDE’s commonly used for MacOS are Eclipse, GitHub, and PyCharm. | Linux supports libraries for HTML, CSS and JavaScript. Common IDE’s used for Linux are GitHub, Visual Studio, Eclipse, PyCharm, and Notepad++ | Web Languages consist of HTML, CSS and JavaScript. Languages Utilized for Windows OS are (C languages like C++, C#), Python, and Java. | Android uses Java, Python, PHP, Ruby, and Kotlin with java being the cheaper of the options. Apple iOS uses Swift, and Java with Java being the cheaper of the two options. |

## 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 would recommend Windows as the Operating platform. The widespread use of Windows and its multiplatform compatibility with Android and XboxOS is perfect for games and to reach the largest audience. Windows is easy to use for both developers and and users, is time efficient and cost effective in regard to development. Server Hosting and Maintenance using a Windows OS is more costly than Linux but comparable to MacOS.
2. **Operating Systems Architectures: All Windows-based applications have services provided by Windows that show a GUI while accessing system resources including Web-Services, Graphics and Multimedia.**
3. **Storage Management**: Windows has built in Storage Management capabilities, there is also third-party compatible storage management software available. Storage Sense by windows helps manage local files but a similar third-party application example would be TreeSize. These allow you to determine both the actual allocation size for files and executables. Another alternative would be to use Cloud to store the library of images needed for the game, considering the core users will be on mobile devices.
4. **Memory Management**: I would recommend a log file for testing and a file to store the game information for later retrieval or loading/saving/continuing a game but only allow one active game in memory at a time. I would recommend the developers also limit the maximum heap size, specify the initial heap size, and set the thread stack size. The developers should keep in mind to minimize cached processes to improve optimization for mobile users.
5. **Distributed Systems and Networks**: To develop the game I would recommend using Visual Studio IDE and Unity. Unity is a game engine with a simple UI that can be easily ported between multiple platforms and OS. Using Visual Studio is ideal because Unity integrates well with Visual Studio and the code used is C# which is compatible with all Microsoft products. In addition, Windows Server OS offers simple communication between different processors between many single PCs and Android Devices.
6. **Security**: Windows is designed with security protection though I would not recommend Windows to secure user information (server side). Windows scans locally for malware and security threats in real time and the system is routinely updated to assist in keeping user information safe. Which is normal for most users. I would also suggest using a login system with username and password with an automatic sign-in feature based on IP address. If a device with a different IP address tries to sign in, then the main device or an associated email should approve the sign in.