

Draw It or Lose It

# **CS 230 Project Software Design**

Version 3.0

## 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 |
| --- | --- | --- | --- |
| 3.0 | 12/13/2024 | Brad Brennand | Operating platform recommendation |

**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 is a client that already has a game developed for Android app users. They are looking to expand the game to other users such as PC and iPhone users. The game is called “Draw It or Lose It”. By expanding the game to other users will bring more clientele and boost the company’s product sales.

## 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)

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.
* The game must be able to be run on multiple platforms.
* Ensure compatibility of all platforms for proper gaming resolutions and play.
* Ensure proper security measures to prevent hackers.
* Ensure gaming has proper amount of storage so the game doesn’t crash based on amount of players and teams on a server.

## [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)

There is an Entity Class which has an Id and Name set to private so it can’t be changed. The Entity class also has three subclasses called game, team, and player. The player class has two attributes, both are public. The Player name and Id and the to String attributes that displays the player name and id. The team class has private players List where the players names will be saved once created. The player class also has three attributes, Team which gets the name and id, addPlayer attribute that adds a player, and the tostring attribute that displays the Team name and id. The game class has a private List where the teams are saved. The game class also has Game id and name, addTeam that adds a team, and tostring that displays the game. The GameService class is associated with Game, Team, Player and Entity classes but can be run by itself. Under Gameservice there is a private List of games where the games can be saved. Private GameId, private nextPlayerId, private TeamId, and a private Service which provides a single instance of a Game. There is a getinstance of gameservice, add game attribute, get game id, get game id, get game count, get next player id, and get nextteam id. There is also the program driver which runs the game and a singleton tester to ensure there is only one instance of a game, team, and 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)

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** | Servers for Mac can be pricey for set up and maintenance. | Linux is similar to Mac and can be preferred as it is customizable. Linux isn’t as user friendly as Mac but does host web abilities easily. | Windows servers have a wide range of compatibility making it widely used by many.  Windows is easy to use, has free updates and amazing tech support. | Apple and android devices are different operating systems and have different requirements.  Android and Apple have their own servers for mobile users. The two are not compatible. |
| **Client Side** | Compared to windows Mac is less used. Mac is compatible with Apple phone users. Mac has the least common languages used compared to Linux and Windows.  Can be costly as development for Mac users have some of the highest quality software and hardware for end users. To get professional tech support might require monthly subscriptions. | Linux uses common languages of coding such as Java, Python, and C or C++. Using these languages makes it easy to adapt to.  Linux is a little cheaper than Mac but still costly. Setting up the system can be difficult. | Windows consistently maintains customer support making it highly recommended by users. Windows is designed to be a multi-user platform. Also easily adaptable with many products and languages.  Windows offers free tech support with all their products. Updates are automatic until a new version comes out, they give advanced notice when version is outdated and when support will no longer be continued. Cost affordability for all clients. | Very cost effective to make mobile applications to see the amount interest and return before expanding to bigger web based or computer based devices.  Android and Apple users are separate from each other. They are not compatible. Apple user compatibility is with Mac users. Android can operate with Linux and Windows. |
| **Development Tools** | Mac can run multiple different programming languages such as Java, Python, HTML and others. | Linux works with most languages but not supported as other OS systems. | Windows is compatible with the majority of languages. | Mobile apps are easier to manage as they take up less storage. Development tools are easily accessible through different apps. |

## 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**: Windows would be a great source to start with. Windows is widely used by many and very user friendly and customizable for optimum customer performance. Windows has many development tools to assist in software development. Windows is very good at being cross compatible.
2. **Operating Systems Architectures**: Windows operates on kernel, which is an operating system used to translate commands into instructions for the computer based on the customer demands. Windows servers can be used for account manageability, especially for client server architecture.
3. **Storage Management**: Windows uses cloud storage to free up computer storage. Easily accessible storage to manage large or small files. SSD or HDD can also be used where local storage is needed.
4. **Memory Management**: Windows memory uses a 64 bit memory for faster and easier operation of the computer operation. The game images will not be saved in the main memory but should be saved in a cache memory for quick access to the main memory.
5. **Distributed Systems and Networks**: Windows can be used with a cross platform design tool such as Unity could be used to make the transition easy. Also using azure when dealing with cloud storage to help with server setup. Using azure can maintain a high connectivity rating with minimal downtime.
6. **Security**: Windows is known to use Mcaffee security which is an overall virus and firewall blocker to prevent hackers and viruses from affecting the system. Further security could be added like BitLocker to add encryption to further help with data protection. Also setting security rights for users and administrators to ensure no user has access to administrators rights.