

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 | 09/12/23 | Jade Watson | Complete Executive Summary, Design Constraints and Domain Model |
| 2.0 | 09/28/23 | Jade Watson | Server Side, Client Side and Development Tools |
| 3.0 | 10/10/23 | Jade Watson | Completion of Recommendation Section |

**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)

We have been contacted by the team at The Gaming Room to develop a web-based application called “Draw It or Lose It”. This application will be developed to allow for multiple teams that consist of multiple players with unique and customizable team names. These factors allow for only one instance of the game to exist at a time while creating an easily accessible game across any tool with access to the web.

## Requirements

* *Allow multiple teams per game.*
* *Allow teams to hold multiple players.*
* *Restrict for unique team names.*
* *Store team names in database.*
* *Allow for only one instance of the game at any time.*

## [Design Constraints](#_2et92p0)

* Multiple teams per game.
* Multiple players allowed in each team.
* Team names must be unique.
* Only one instance of the game can exist at one time. Web based application accessible on multiple systems.
* Application must contain a database to hold team names and ids.

## [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)

In the UML diagram below, there are several elements and entities represented within the picture. Each class is represented within its own box to distinguish the separate classes. Within the boxes representing the classes are the variables and functions held within each class. The relationships between classes are identified by the lines between the classes with each line type identifying the type of relationship. “Entity” is a superclass within this diagram as the “Game”, “Team”, and “Payer” classes are all identified as having an “is a” relationship with the “Entity” superclass. Due to this “is a” relationship, all three of the classes share the attributes “id” and “name” that are within the “Entity” superclass. There are several “has a” relationships within the diagram, also known as aggregation, as they have a “one-to-many” relationship. There is one “Team” to many “Player” classes, one “Game” to many “Team” classes, and one “GameService” to many “Game” classes.

**"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 must 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** | * Easy to utilize terminal to alter server-side systems. * Advantages:   Easy to add upgrades to utilize with IDEs.   * Disadvantages:   Not as commonly used | * Relatively inexpensive * More secure than other systems * Advantages:   High monitoring for security issues  Commonly used.   * Disadvantages   Linux applications don’t usually support web applications | * More commonly used than Mac systems * Advantages:   Relatively easier to work with because of high performance time and resource accessibility.   * Disadvantages   Not as much security monitoring or tech support | * High popularity * High portability * Advantages:   High compatibility with multiple device types   * Disadvantages:   Poor service means no connection restricting portability |
| **Client Side** | * Moderately easy to utilize. * Moderate cost | * Very difficult to work with * Low cost | * Very easy to work with * Moderate cost | * Varies in difficulty for use and cost. |
| **Development Tools** | * Better prepared for development work * Able to run all programming languages. * Support open-source libraries for development. * Often preferred for development of software | * Development tools are easy to use but very restricted * Able to run many programming languages mostly for front-end development | * Easy to use than Linux. * Tools are like Linux and restricted to being better for front-end development. | * Must be defined for Apple vs Android applications. * Software development languages are similar for all applications. |

## 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 advise utilizing a Windows operating platform for a multitude of reasons. The platform itself provides an easier system for developers when designing the application and coding the requirements of users and stakeholders. For the application users themselves, the Windows platform allows easy customizability as well as a low cost and it is simply more widely used than other operating systems. Windows is less restricted when it comes to use of Cloud storage systems which paves the way for a simple conversion to other operating systems and more universal availability.
2. **Operating Systems Architectures**: I would advise the layered architecture for application development. This architecture divides the different elements of an application into different layers within the application architecture that makes it compartmentalized and easier to organize different elements within application especially during development. The use of layers creates a tiered system that allows for easy editing and maintenance within those layers. This architecture also adapts better for scalability when developing an application for multiple platforms.
3. **Storage Management**: I would advise utilizing an SSD storage system for this application. The use of a solid-state drive in a server-client structure is designed to optimize speed for the application. This storage method decreases loading time for the application to be visible and increases the speed of application runtime because of the increase speed of each process. The use of SSD is more durable than HDD storage systems which make them more reliable as well.
4. **Memory Management**: I would advise the use of segmentation to optimize memory use for the application. This separates the application into separate sections based on their functions and size. These segments are loaded with the main memory when the application starts to reduce load time, but processes are loaded when needed. The use of segmentation restricts loading times but improves performance while aiding organization.
5. **Distributed Systems and Networks**: I would advise the use of the client-server architecture for application organization is the deciding factor when it comes to cross platform applications. The server side of this architecture is able to hold all of the basic data and user data within one database that is compatible with multiple different client-side platform types. The server would hold the game logic and SQL database which provides a single location for all important information that all users see the same. Each platform type would require it’s own game development code that would be able to access the server-side logic but perform the required functions on their own platforms.
6. **Security**: I would advise the utilization of roles within the application to define the security layers needed within the application. This sets up the application to have the general user with many restrictions of code that they can access, moderators with a little bit less restrictions and the admin with the most amount of application access. This division of access can be a simple but effective security strategy that requires little upkeep but can easily distinguish security access.