

The Gaming Room

# **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  1.1  1.2 | 09/17/2023  10/01/2023  10/15/2023 | Craig Rowell  Craig Rowell  Craig Rowell | Section Edits: Cover Page; Document Revision History; Executive Summary; Design Constraints; Domain Model  Section Edits: DRH; Development Requirements  Section Edits: DRH; Recommendations |

**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 developing a multi-platform web-based gaming app based on their existing game, Draw It or Lose It, which is loosely based on the game show ‘Win, Lose, or Draw,’ and currently only available on Android. The game consists of four one minute rounds where drawings are rendered from a library of stock drawings at a steady rate, to be fully completed after 30 seconds. If the active team fails to guess before the timer expires, the remaining teams each have 15 seconds to guess the answer.

The Gaming Room would like assistance setting up the game environment and streamlining development of the web-based app according to their software requirements, with attention to software design decisions that will determine the necessary hardware requirements at a later stage.

## 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.>*

* *<REQ-100> - A game will have the ability to have one or more teams involved.*
* *<REQ-101> - Each team will have multiple players assigned to it.*
* *<REQ-102> - Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.*
* *<REQ-103> - 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.*

## [Design Constraints](#_2et92p0)

* Able to run on multiple platforms.
* Supports multiple players per team and multiple teams per game.
* Game and team names shall be unique, and users should be notified if their team name is in use when choosing a team.
* Only one instance of a game should exist at any time.

## [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 consists of classes such as, Entity, Game, Team, and Player. The Player class provides identifiers and metadata for individual users of the application, Team aggregates groups of player identity information to a single identifier with its own metadata, and game aggregates teams to a single game session, again, with metadata. This encapsulates each instance of each class. Through inheritance, the Entity superclass provides each child class, Game, Player, and Team, with the aforementioned metadata.

**"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 OS X Server has server accessibility and configurability. Many users prefer Apple products due to familiarity with its GUI. OS X Server development is proprietary, and licensing is expensive. | Open-source means free licensing, so low cost. Linux is a popular option for web-based hosting, but it is considered difficult to configure and navigate. | With its user-friendly GUI and CLI, Windows Server is highly secure and easy to configure. It is easy to learn and support. Minimal expertise is required, but it is expensive to license. | Mobile devices are not designed to be used as servers. Although it is possible, they are not equipped with the capabilities that other options offer. |
| **Client Side** | Mac SDKs are secure and easy to use, but again, development is proprietary, which increases cost, and requires domain knowledge to develop. Hardware is also more expensive, so users without access to Apple hardware will also incur increased costs. | Open-source means high configurability, but high configurability means more time and expertise required to develop. With the smallest population of users, Linux has the least accessibility. | Expertise is the highest requirement for Windows. It is more expensive than Linux. Use the .NET framework for security and capability. | Development for mobile devices requires the highest amount of expertise. Developing a mobile app is a specialized skillset that requires a strong understanding of object interaction and integration. |
| **Development Tools** | A Macbook with Apple ID Developer, Xcode, and Swift programming knowledge. Objective-C programming knowledge is also an option, though Swift is now considered industry standard.  CLIs – xCode terminal, Bash shell | Python is the most commonly used language with Linux systems and comes pre-installed on most distributions. PyCharm is a commonly used IDE for python. HTML/CSS/JavaScript, and Ruby may also be utilized with IntelliJ IDE. CLI tools – Bash, Sudo, SSH, Ping, Git | HTML/CSS/ JavaScript. Libraries to support frontend and various languages such as C, C++, C#, Java, and Python.  IDEs – Eclipse/ Eclipse CDT, PyCharm, Visual Studio/ VS Code  CLIs-Command Prompt and PowerShell | Development tools will depend on platform. Develop for android using kotlin, apple using swift, and platform agnostic using Java. Compiled languages such as C-family can be platform agnostic, but will require special configuration on each platform. Interpreted languages such as Python will also most likely require customized configuration per platform. |

## 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**: A Linux Ubuntu Server with a Kubernetes cloud setup would be an appropriate environment to host Draw It or Lose it.
2. **Operating Systems Architectures**: Linux kernel is a stable and secure environment. Kubernetes clusters will allow separation of system and hardware resources.
3. **Storage Management**: HDD storage will suffice, but SSD would be ideal if the budget allows. SSD is more expensive, but it provides faster access to stored resources. Given that this app depends on loading stored image files quickly, the user experience would be vastly improved by providing the fastest possible data retrieval. File storage can be optimized using Kubernetes nodes, and game data and user management with NoSQL nodes with the NoSQL linking the URL to the location of the pictures.
4. **Memory Management**: One efficient memory management method is to set up a watcher to monitor the system load. When usage is low, required memory can be reduced, so the maximum amount of memory is only utilized during peak hours.
5. **Distributed Systems and Networks**: Providing access to the system on the cloud prevents the need to restrict user access during server maintenance. When maintenance is necessary, simply transfer the node to another server. Set up another server to start up automatically if the system crashes. This method will host everything except for the client. Simply develop a client to access the server for each desired operating system. Kubernetes provides easy management and organization of your system using separation of functions.
6. **Security**: Separate access to information based on a role-based tier system with admin, game, team, player, and user roles. Admins will have access to all information necessary to keep the game functioning securely. Game tier has access to all information relevant to its game session. Team tier will only access information relevant to their own team within the game session. Player tier only accesses informant relevant to that player within that game session. The user role will have access to that user’s information, and only that user’s information, regardless of current game session. This will prevent users from accessing information that they should not have.