

Creative Technology Solutions

# **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 |
| --- | --- | --- | --- |
| 1.3 | 08/12/2023 | Terry Christmas | Updated Recommendations page 7 |

**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 requesting basis environment creation for a new game called Draw It or Lose it. The game consists of rendering images from a large library of stock drawings as clues. The game consists of four rounds of play and each round will last of one minute. Drawings are rendered at a steady rate and are fully completed at 30 seconds of each round. If the team does not guess the clues before the time expires, the remaining team will have an opportunity to guess within a 15 second time limit.

## Requirements

* *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*

## [Design Constraints](#_2et92p0)

* Servers to host game
* End user hardware and platforms
* Compatibility on different web browsers
* Security for user data
* Network and Rendering times

## [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 Entity class is the main driver class that is the source for the Game, Player, and Team Class. These 3 classes inherit attributes from the driver class. With the 4 child classes the also inherit from each other. From the UML, we are using polymorphism to limit the amount or repeatability of the getters and setters for each class. Each of the child classes are also encapsulating methods from each class such as player to team 0 to infinity. We are using this for our methods to get or set to each class for the player having to check for repeated names, teams, or games.

**"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** | Pros: Unix based secured system, compatible with other mac devices, powerful advanced hardware.  Cons:  Limited expensive hardware choices, MacOS is licensed base dedicated for Mac products. Third party software support and update potential roadblocks. | Pros: Open Source and Free source code. This is a benefit for smaller companies that keeps the cost low. Stability and robustness requiring less maintenance.  Cons: Development and knowledge of OS. Command line-based commands. Limited support and compatibility issues. | Pros: Compatibility with most applications and previous versions. Most popular amongst other OS, which is more prevalent for web-based servers.  Cons:  Security is the weakest amongst other which is more prone for malware attacks. | Pros:  The most accessible and portable OS which has more user usage. Updates are controlled at server level which doesn’t require software updates to end user.  Cons:  Internet connectivity is needed which will require either Wi-Fi or good cellular service for limited interruptions |
| **Client Side** | Expertise developing UI/UX MacOS design guidelines. Complexity and the testing the User Interface | Most challenging OS for knowledge and programmers to support. Cost and time is the highest. | Popular OS that has a bigger talent pool for programming and knowledge. This will be the cheaper route for time and cost. | With Mobile devices, the client side has many different Operating systems such as Android or IOS. This will have cost and time dedicated to support the different compatibility aspects. |
| **Development Tools** | Native MacOS applications are Objective-C and swift. XCode is MacOS primary IDE tool. | C/C++  Python  Java  Visual Studio Code  IntelliJ Idea  Eclipse | C/C++  Python  Java  Visual Studio Code  IntelliJ Idea  Eclipse | Android  Java  Kotlin  IOS  Swift  Objective-C  Cross Platform  Flutter  IDE Android Studio  Xcode(IOS)  Visual Studio Code |

## 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**: Since the game is web based, any operating platform can be used. I recommend using Linux OS for the server. This is the lowest cost solution and is open source. Since the Gaming Room is trying to expand their clients, this will keep risks low if project fails. The stability and robustness of using Linux is beneficial over the rest OS.
2. **Operating Systems Architectures**: The Operating system is consisted only of a few components. Hardware, Kernel, Shell, System, and finally the User. Hardware is consisted of physical devices such as servers, monitors, terminals. This allows the user to use secure socket shell to target certain files, apps, or databases. Kernel is essentially the foundation of Linux OS to allow the functionality between devices and Linux.
3. **Storage Management**: Storage management is used in Linux by portioning and dividing physical storage into sections. This is done over SSH command lines. There are appropriate file systems in place to manage performance, reliability, and other features. These systems are created then mounted to the file hierarchy. These file systems can be unmounted when no longer needed.
4. **Memory Management**: For the server to handle Draw It or Lose It, we must allocate sufficient memory on the server to be able to process the data. The game would also need to use caching and prefetching to improve overall performance and speed when rendering the images. Since the game will be using a lot of memory during user game sessions. We will have to set up the best memory management in the Kernel Parameters.
5. **Distributed Systems and Networks**: I would suggest using a Cloud Server inline to support multiple OS and platforms. This can handle majority of the traffic back and forth between different operating systems. This will also benefit maintenance for the server, the third-party cloud service provider will be responsible for the functionality, security, and traffic flow. Draw It or Lose It will be responsible for the data. We would use the API structure to use get, post, rest, and delete structures between the main server, cloud server, and client end.
6. **Security**: This would be used either on the server side or cloud side for the user data. We would ensure that the data is protected by encrypted password and multi-factor authentication for the client. For the server end we will handle authorization needs. We will have an admin account for the server and user accounts for the clients. The server will be set up with firewalls to monitor traffic. We can also monitor and audit the traffic to track the activity or detect potential threats. We will also have to set up routine security patches and updates to continuously kaizen.