

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 03/23/2025 | Ruben Rodriguez | Created changes to the cover page, document revision history, the executive summary, design constraints, system architecture view, domain model, evaluation and recommendations. |
| 1.1 | 4/6/2025 | Ruben Rodriguez | Revised and edited information within evaluation section. |
| 1.2 | 4/16/2025 | Ruben Rodriguez | Revised and expanded upon the recommendations section of the document. |

**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 wants to develop a web-based game that serves multiple platforms based on their game Draw It or Lose It, that is currently only available on Android. The purpose of the game is to have multiple teams which consist of several people going four rounds for a minute each. When a picture is displayed from a library of images, one team guesses what it is until the time runs out. If the correct answer is not given, each opposing team member gets 15 seconds to try to give the correct answer.

## Requirements

## [Design Constraints](#_2et92p0)

- Must run across multiple platforms

- Each team needs to have multiple players

- Only one instance of the game can exist at any time.

- Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.

[System Architecture View](#_ilbxbyevv6b6)

## [Domain Model](#_8h2ehzxfam4o)

The Entity class creates a relationship between Game, Team, and Player classes. They all inherit or get information from Entity. The UML diagram below demonstrates the inheritance. We can see that Team and player have a “has a” type relationship, Game “has a” team and GameService “has a” Game. GameService manages collections of the Game objects and indirectly the Team and Player objects. The ProgramDriver class contains the main() method which serves to run the application. The ProgramDriver class uses the SingletonTester class to check that GameService is running as a singleton.

**"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 easy accessibility and server configuration. It has a easy to use graphical user interface and flexible terminal commands. Supports Apache and provides polished GUI and native development tools with high costs. | Cost-friendly. Difficult to navigate. Command shell for simple server-side configuration and accessibility.  No fees. Offers a minimal footprint with strong security practices and seamless cloud integration. | Server side is expensive. User friend GUI. Command prompt allows for easy inputs of server-side commands. Windows containers are available but less mature than Linux. Fees can potentially be significant, but Windows is already supported widely by many companies. | It can be configured to have a specific use unlike other devices. Specifications may vary from users.  Not designed for hosting production web services. While local development servers do exist, they lack performance, security and manageability required for production. |
| **Client Side** | Not as cost efficient for users. Difficult to navigate OS.  Users can access the game via Safari, Chrome, or Firefox on macOS. Cross-browsers compatibility would require testing. MacOS offers Safari Web Inspector and Chrome DevTools as developer tooling. | Takes a lot of experience and time to learn/navigate through. Linux data is required to use the operating system. Testing environments can be containerized or virtualized for isolation. Provides cost effective workstations. | Costs more than Linux but less than Mac. Easy to learn and comprehend a Windows setup with little experience needed. Corporate policies and antivirus software can cause a hindrance in the installation of developer utilities. | Provides flexibility to clients and developers to see updates at any moment. Slightly more difficult to navigate and set up OS for a user. Responsive frameworks and a touch-friendly UI are critical. |
| **Development Tools** | It consists of languages such as HTML, CSS, and JavaScript. Contains libraries to support frontend development. Relevant IDEs include PyCharm, GitHub, and Visual Studios. Also uses Xcode for iOS wrapper builds and packaging. | It consists of HTML, CSS, and JavaScript programming languages. Linux systems can have IDEs such as JavaScript, Ruby, PHP, PyCharm and others. All major frameworks can be supported without licensing costs and cannot support iOS builds. | It consists of HTML, CSS, and JavaScript programming languages. Development tools include Eclipse, PyCharm, Visual Studio, the command prompt, and others. Visual Studio and PowerShell are also used. May include licensing costs. | It consists of HTML, CSS, and JavaScript programming languages. IDEs include Android Studio, Xcode, and Visual Studio. Android Studio and Xcode are available for free (Xcode can be paid). Emulators and simulators are resource intensive. Slightly more complex. |

## 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**: The best operating platform that will allow The Gaming Room to expand their game to other computing environments would be Windows. It supports a wide variety of development tools, has a strong compatibility with third-party applications and frameworks. It also integrates well with cloud-based solutions to offer scalability for future game updates and user expansion.
2. **Operating Systems Architectures**: Windows has ways to store files, run software, play games and videos, and connect to the internet. Windows allows device drivers and services to run in user mode to reduce crashes while still offering low-level access for performance. This flexibility is great for Draw It or Lose It, which requires both responsiveness and system stability across client and server environments.
3. **Storage Management**: Using Windows can allow for features such as file compression and permission-based security for files. Windows also has built-in support for Microsoft OneDrive which could make it easier to store or back up game data in the cloud. This could help in storing game assets.
4. **Memory Management**: Windows can dynamically allocate or deallocate memory as needed. Windows also allows each process to get its own virtual address space, which would improve stability. If physical memory is low, inactive pages would then get written to disk to allow for more critical processing on the RAM. This ensures that the game engine can maintain performance even with fluctuating memory demands.
5. **Distributed Systems and Networks**: Since The Gaming Room wants to communicate between various platforms, they could form a client-server architecture where they host the core game logic and data on a Windows server that has each client communicate through the server through APIs.
6. **Security**: Using Windows would allow for data encryption usings its built in BitLocker. Windows also has a built in Firewall which can protect against unauthorized connections. We can also use RBAC to ensure only authorized users or processes can access sensitive data. Together these security measures can protect user privacy and game integrity across distributed platforms.