

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 | 11/11/2023 | Nicholas Altemoos | We work on executive summary, design constraints, and domain model. |
| 1.1 | 11/25/2023 | Nicholas Altemoos | We work on the evaluation of the client side, server side, and development tools. |
| 1.2 | 12/07/2023 | Nicholas Altemoos | We work on the recommendations for each component of the architecture. |

**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 project The Gaming Room is a web-based game that will serve multiple platforms based on the game Draw It or Lose It android game. The game is a 3-team max based game with four one-minute rounds. A round consists of a picture being pulled from a collection, drawn, and each team guessing the answer. The picture is drawn over 30 seconds and the team have until the end to guess before the other teams have 15 second turns to guess.

## 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)

* Unique game and team IDs
* A single game instance limit
* Multiple platform formats

## [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)

We have a superclass “Entity” that parents “Game”, “Team”, and “Player”. With the 3 child classes they each have an aggregation or a “has a” relationship with one another where we can have none or multiple “Player” in “Team” and “Team” into “Game” and then "Game” into the "GameService” class.

**"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** | Many native tools for command line utility for server configuration, better creation licenses than windows but hardware is limited and costly | Easy shell commands for accessibility and server configurability with reliability. Most cost effective of windows and mac | Familiar GUI and command prompts. Server costs can be mitigated but traditional costs are moderate. Unique and unportable software but can be simulated on other systems. | Device specifications vary wildly and has the most cost with having to build everything from the ground up to work with the cross-platform development |
| **Client Side** | Ease of use is like windows; startup costs are normally higher due to brand name prices. Common browsers are safari, google chrome, and Firefox. Regular updates with standardized design across browsers and extensive cross-browser testing. | Very cost effective and deployable on many devices and specs, many tools that work with the OS. Ease of use can be significantly less than other OS depending on flavor. Linux supports most common browsers. Regular updates with standardized design across browsers and extensive cross-browser testing. | Similar ease of use to mac, unique software that can only be simulated or sandboxed on other systems also can simulated other OS. Widespread so a variety of tools are developed. Common browsers are chrome, Firefox, edge, explorer. Regular updates with standardized design across browsers and extensive cross-browser testing. | Flexibility for client on location, needs adaptable code to adjust to screen size differences and touchscreen interface needs to be smooth and intuitive. Common browsers are safari and chrome. Regular updates with standardized design across browsers and extensive cross-browser testing. |
| **Development Tools** | HTML, CSS, and JavaScript are all available.  We can use IDEs like PyCharm, GitHub, VS Code, Xcode, Linux tools, etc. | HTML, CSS, and JavaScript are all available.  Systems include libraries for Python, Ruby, PHP, and others.  We have VS Code, command line, atom, AWS, and a great variety of tools. | HTML, CSS, and JavaScript are all available.  We can use IDEs like Eclipse, command prompt, PyCharm, VS Code, etc. | Firefox, safari, chrome, opera, brand browsers. All mobile browsers need to be compatible and tested. The app needs JavaScript to be compatible with the app store access. |

## 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 recommend windows for the server; it has a wide user base and would be statistically easier to find a developer. We can also utilize Microsoft’s azure web-based server hosting to easily scale up when demands increase. Licensing will be higher but over all windows can host ported software to help with cross-platform compatibility.
2. **Operating Systems Architectures**: Windows follows a ground up approach with resource management and prioritization. The GUI is well received and intuitive to use. Has a wide range of IDEs to help develop applications in whichever language is needed.
3. **Storage Management**: Windows primary system is NTFS, and it supports compression, encryption, and access control lists. We can also use dynamic disks to help with clustering storage and proper indexing.
4. **Memory Management**: Memory is managed by segmenting the available resources to optimize the amount allocated and used between each layer of function. With the cloud-based windows server we can have more on demand and dynamic memory and utilizing caching can allow us to have faster access during each instance of the game.
5. **Distributed Systems and Networks**: While we will have common issues such as lagging, queuing issues and overloaded servers, these will be worked out during the initial phase of roll out or testing as they arise in the system. We will probably need to have various servers to run the games on each platform, which will lead us to having the need for proper networking protocols such as TCP/IP to communicate between each server. Windows 2016 has built in network load balancing.
6. **Security**: Security will start on the design level, we need to ensure we use SSL/TLS encryption for client server communications, and end to end encryption for any information that’s sensitive, as well as file system level encryption such as BitLocker for windows. We also need to utilize MFA practices when working with the server and whenever any user logs in to protect account access. We need to also utilize hashing and salting practices when storing user passwords securely to prevent access to the data even if leaked. We can utilize BitLocker and Windows Defender for base coverage and give user and developer education on security features. We also have the option to use Azure Network Watcher for real time monitoring as well as open-source tools such as Suricata.

**References**

Andrei-Popov. (n.d.). *TLS - SSL (schannel SSP) overview*. TLS - SSL (Schannel SSP) Overview | Microsoft Learn.

Baranov, A. (2022, September 6). *Inside the windows cache manager*. LinkedIn.

*Dynamic disks with windows server failover clustering*. TECHCOMMUNITY.MICROSOFT.COM. (2023, February 9).

JasonGerend. (n.d.-a). *Network load balancing*. Microsoft Learn.

JasonGerend. (n.d.-b). *NTFS overview*. Microsoft Learn.

Lorihollasch. (n.d.). *Windows Storage Management Provider - windows drivers*. Windows drivers | Microsoft Learn.

Nohe, P. (2022a, December 14). *The difference between encryption, hashing and salting*. Hashed Out by The SSL StoreTM.

Paolomatarazzo. (n.d.). *Bitlocker Overview - Windows Security*. Windows Security | Microsoft Learn.

Halkazwini. (n.d.). *Perform network intrusion detection using Open source tools - azure network watcher*. Perform network intrusion detection using open source tools - Azure Network Watcher | Microsoft Learn.