

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_gjdgxs)

[**Table of Contents 2**](#_30j0zll)

[**Document Revision History 2**](#_3znysh7)

[**Executive Summary 3**](#_2et92p0)

[**Requirements 3**](#_tyjcwt)

[**Design Constraints 3**](#_1t3h5sf)

[**System Architecture View 3**](#_4d34og8)

[**Domain Model 3**](#_2s8eyo1)

[**Evaluation 4**](#_17dp8vu)

[**Recommendations 5**](#_26in1rg)

## [Document Revision History](#_lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <04/16/2024> | Jason Tutterow | Project 3 |

**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](#_35nkun2)

<Write a summary to introduce the software design problem and present a solution. Be sure to provide the client with any critical information they must know in order to proceed with the process you are proposing.>

## 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](#_1ksv4uv)

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

## [System Architecture View](#_44sinio)

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](#_2jxsxqh)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

"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](#_z337ya)

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** | Strengths: Renowned for its stability and security, macOS is user-friendly for developers, offering an array of development tools natively.  Limitations: Higher initial investment for hardware. Its usage as a server platform is less common compared to Linux and Windows, potentially limiting support and resources.  Server Deployment: Supports standard web hosting services, albeit with specific configurations.  Cost Implications: The primary expense is in the hardware procurement; the OS itself does not incur additional licensing fees. | Strengths: Open-source and highly customizable, Linux dominates web server environments with extensive community support. It’s known for scalability and cost-effectiveness, with many distributions available for free.  Limitations: Demands a steeper learning curve for effective administration and customization.  Server Deployment: Excellent support across a spectrum of web server technologies; the go-to choice for scalable web applications.  Cost Implications: Generally free, though some enterprise distributions charge for support services. | Strengths: Offers seamless integration with Microsoft technologies and services, backed by intuitive management tools. Ideal for .NET framework applications.  Limitations: Higher licensing fees; potential for greater security vulnerabilities.  Server Deployment: Utilizes Internet Information Services (IIS) for web hosting, supporting a wide range of web applications.  Cost Implications: Licensing costs can be significant, varying with the edition and scale of deployment.  Mobile Devices | Android devices are not designed for server-side hosting but rather function as clients that interact with applications hosted on servers. Therefore, evaluating Android for server-side capabilities does not apply. |
| **Client Side** | To ensure a seamless experience on Mac, the application must be responsive and perform well in Safari, with design elements that complement macOS's aesthetic standards. | Development for Linux requires testing on multiple distributions and browsers, emphasizing performance and adaptability to cater to its diverse user base. | For Windows, compatibility with Edge and a focus on accessibility are key, ensuring the application is intuitive across various versions of the OS. | For Android, the focus is on ensuring performance and compatibility across a wide range of devices and screen sizes, particularly with Chrome, to accommodate the platform's fragmentation. |
| **Development Tools** | Toolkit: Includes Xcode for native iOS applications and a variety of web development tools (e.g., Visual Studio Code, Atom). | Features a wide array of open-source development tools (e.g., Eclipse, Git). | Encompasses Visual Studio, .NET Framework, and SQL Server for comprehensive development capabilities. | Utilizes platform-specific IDEs like Android Studio and Xcode, along with cross-platform frameworks (React Native, Flutter). |

## 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**: Adopting a cloud-based platform such as AWS, Azure, or Google Cloud is highly recommended for the expansion of "Draw It or Lose It." These platforms excel in providing scalable, flexible environments that support the game’s operation across diverse computing devices, ensuring a seamless experience for players whether they are on PCs, smartphones, or tablets.
2. **Operating Systems Architectures**: Cloud platforms afford the flexibility to deploy the game on a variety of operating systems, including Linux and Windows. This adaptability is akin to being multilingual in the digital realm, offering the game accessibility to a broader audience without being constrained by specific system requirements.
3. **Storage Management**: Cloud-based storage solutions are suggested for managing the game’s data securely and efficiently. These services act as robust, scalable digital repositories that can be accessed from anywhere, providing a reliable backbone for storing game assets, player information, and more.
4. **Memory Management**: Cloud platforms are equipped with dynamic resource allocation and scaling capabilities, ensuring that "Draw It or Lose It" remains responsive and efficient under varying loads. This intelligent memory management facilitates optimal resource use, maintaining game performance across different user engagement levels.
5. **Distributed Systems and Networks**: Leveraging cloud infrastructure enables the game to operate on a distributed network, enhancing global accessibility and reducing latency for a worldwide player base. Cloud services come with advanced networking tools to manage connectivity, scale resources during peak times, and maintain consistent game performance, ensuring the game’s availability around the clock.
6. **Security**: Cloud platforms prioritize security with comprehensive measures designed to protect user data. Features such as data encryption, identity, and access management, and network security protocols are integral to safeguarding player information against unauthorized access and cyber threats, instilling confidence in the game’s safety among its user base.