

The Gaming Room

# **CS 230 Project Software Design Template**

Version 3.0

## 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** Error! Bookmark not defined.](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View** Error! Bookmark not defined.](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/15/2024 | Ty Votour | Changes made to cover page, document history, executive summary, design constraints, system architecture view, domain model and recommendation. |
| 2.0 | 12/6/2024 | Ty Votour | Changes made to the Evaluation Table (it is now split across two pages due to its length). Added Key Observations Section. |
| 3.0 | 12/13/2024 | Ty Votour | Changes made to the Recommendations section. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room project aims to create a web-based game that can be played across various platforms, inspired by the existing game "Draw It or Lose It," which is currently only available on Android. The objective of the game is to have multiple teams, each with several players, compete in four rounds, each lasting one minute. During each round, a picture is selected from a library of images, and one team tries to guess it before the time expires. If the team doesn't guess correctly in time, each player from the opposing teams gets a chance to answer, with a 15-second limit for each.

## [Design Constraints](#_2et92p0)

* Must be able to run on multiple different platforms
* Requires multiple players on each team
* Team names must be unique
* Game name must be unique
* There must be only one instance of the game

## [Domain Model](#_8h2ehzxfam4o)

The Entity serves as a link between the Game, Team, and Player classes, meaning that they all derive or gather information from Entity. In UML, this relationship is depicted through inheritance, where Entity acts as the superclass. In terms of their relationships, Team and Player are considered to have a "has a" relationship with one another, while Game contains a Team, and GameService contains Games. In UML terminology, this is represented as aggregation (HAS-A). When we say a user "has a," it implies that one class instance holds a reference to an instance of another class. In this case, the diagram shows that GameService holds a reference to Games, Games holds a reference to Teams, and Team holds a reference to Players.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | - Mac provides easy accessibility and server configuration options. It has an intuitive GUI and flexible terminal commands for advanced users. - Limited licensing cost for server hosting. | - Budget-friendly but can be challenging for less experienced users. - Offers command shell for server management. - Open-source, no licensing cost. | - Windows can be costly for server-side hosting. - Offers a user-friendly GUI and command prompt for advanced configurations. - Licensing fees may apply, especially for Windows Server editions. | - Server-side costs depend on mobile back-end infrastructure. Typically, mobile devices access the server, but hosting costs are not device-specific. |
| **Client Side** | - High cost for users. - Development for Mac involves moderate time and expertise. - Effective for a stable and secure environment. | - Significant expertise and time are needed. - Linux systems often require advanced knowledge. - Higher cost for Linux users in terms of setup and configuration. | - Costlier than Linux systems but user-friendly. - Minimal expertise required for Windows clients, with broad developer support. | - Flexible for users and developers to access updates from anywhere. - Development is more challenging on mobile compared to desktop due to device variability. |
| **Software Development Considerations** | - To support both Mac and mobile platforms (iOS/Android), the application will need to be optimized for cross-platform compatibility. - Expect a longer development time and expertise in responsive design and web standards (HTML5, CSS3, JavaScript). | - Linux offers flexibility but requires specialized skills for development. - Cross-platform support will require building and testing on different environments. - Time-consuming and expertise-heavy, but budget-friendly. | - Windows allows for efficient testing and development. - Cross-platform development tools (like React or Angular) can ease compatibility. - Moderate cost and time involved for app testing and optimization. | - Requires a responsive HTML interface for browsers and potentially native apps for iOS and Android. - Mobile platforms are diverse, and application must be tested across various screen sizes and OS versions. - The development process for mobile support requires additional frameworks (e.g., React Native, Flutter) and expertise in mobile-specific UI design. |
| **Development Tools** | - Languages: HTML, CSS, JavaScript. - Libraries: React, Angular, Vue.js. - Development tools: PyCharm, GitHub, Xcode, Visual Studio Code. - Must support responsive web design for cross-platform support. | - Languages: HTML, CSS, JavaScript, Ruby, PHP, Python. - Libraries: React, Angular, Vue.js. - Development tools: Vim, Emacs, Git, Eclipse. - Cross-platform tools and testing tools are essential. | - Languages: HTML, CSS, JavaScript. - Libraries: React, Angular, Vue.js. - Developer tools: Eclipse, Visual Studio, Command Prompt, GitHub. - Strong support for web development tools and resources for cross-platform testing. | - Languages: HTML, CSS, JavaScript, C++. - Libraries: React, Angular, Vue.js. - IDEs: Xcode (iOS), Android Studio (Android), and Visual Studio Code. - Tools for mobile app optimization (e.g., Cordova, Flutter, React Native). |

## 

## Recommendations

1. **Operating Platform**: The ideal operating platform for expanding *Draw It or Lose It* to other computing environments is Microsoft Windows. This is because Windows offers a wide variety of IDEs, ensuring you won't face limitations in development tools. Windows offers robust support for scalability, security, and cross-platform compatibility, which is essential for expanding the game to other devices and systems. Additionally, it supports containers and virtual machines, enabling the game to be deployed in diverse environments.
2. **Operating Systems Architectures**: Windows is a graphical operating system created and released by Microsoft. It allows users to run applications, play games, watch videos, access the internet, etc. It also uses a client-server architecture that enables it to handle multiple simultaneous connections from clients. Windows manages resources such as processing power, storage, and memory, while client devices (PCs, mobile devices) communicate with the server to access *Draw It or Lose It*.
3. **Storage Management**: Windows includes a useful feature called Storage Sense, which lets you analyze and manage the files on your hard drive, as well as monitor the space they occupy. This ensures that the game data and user progress are protected against hardware failures. It also uses caching mechanisms and tiered storage for fast access to critical game data.
4. **Memory Management**: Windows employs a sophisticated memory management system to ensure the efficient use of RAM and improve performance. It uses virtual memory to extend the apparent amount of available memory, even if the physical RAM is limited. This is useful for memory-intensive applications like multiplayer games, ensuring that there is minimal latency during gameplay. Windows also automatically offloads less-used data from RAM to disk, freeing up memory for active game processes. Additionally, Windows’ Storage Sense feature would enable organized memory management. This memory management system ensures the game runs smoothly, even with multiple users playing and interacting simultaneously.
5. **Distributed Systems and Networks**: Using Windowsfor the backend infrastructure can provide a distributed architecture with load balancing, ensuring high availability and reducing latency. In a distributed system, certain components depend on others. For instance, game state data may depend on a central database, while real-time gameplay data might rely on different servers for instant updates. Network-based multi-user interaction systems, like network games, usually involve a shared database accessible by players who are physically dispersed and communicate over the network. Currently, developers of network games must build shared database and player communication systems from the ground up.
6. **Security**: Security is critical, particularly since *Draw It or Lose It* involves user accounts and online interactions. Windows offers numerous security features to protect user data and ensure secure communication between devices. Windows supports encrypting systems to ensure that sensitive user data (e.g., passwords, game content) is encrypted. The built-in Windows Defender Firewall and Network Security Groups allow network traffic to be controlled and restricted based on rules. This ensures that only authorized devices and users can access the gaming servers.

**Key Observations**

**Server Side:**

* Mac: Offers an easy-to-use configuration with a relatively low cost for server-side hosting. It’s a good choice for smaller-scale projects but may not offer the same scalability for larger applications.
* Linux: Ideal for budget-conscious projects. It is highly flexible but requires expertise for setup and management. It's perfect for those who want low-cost, open-source solutions.
* Windows: More costly but highly user-friendly. It offers GUI tools that simplify the process for less experienced users, though the licensing fees can be high, especially for Windows Server editions.
* Mobile Devices: Hosting costs aren’t specific to mobile platforms but depend on the back-end infrastructure supporting mobile clients. Typically, a web-based back-end will scale to mobile devices, requiring minimal changes to the server-side architecture.

**Client Side:**

* Mac: While offering strong security and user experience, development for Mac is costlier and requires expertise to ensure the web application works smoothly across all platforms (including mobile).
* Linux: Requires significant expertise but is cost-effective for developers. However, ensuring cross-platform compatibility (especially with Mac and Windows clients) adds complexity.
* Windows: The most straightforward in terms of development, with broad support for web technologies. While the cost is higher than Linux, the development process is simpler and quicker.
* Mobile Devices: A flexible platform for users, but development is more complex than desktop environments. A responsive web interface will be necessary to ensure compatibility across various screen sizes and device capabilities (iOS and Android). Testing across devices is crucial, and additional mobile-specific frameworks might be needed for performance.

**Development Tools:**

* Mac and Windows offer a variety of development tools, such as IDEs (e.g., Xcode for Mac, Visual Studio for Windows), and libraries for building modern, responsive web applications. Both platforms provide powerful environments for managing and developing across different browsers and devices.
* Linux supports the same languages but offers more freedom for customization with its open-source nature, although it requires more specialized tools and environments to work effectively across all platforms.
* Mobile Devices will need additional tools for specific platforms (e.g., Xcode for iOS, Android Studio for Android) to ensure apps perform well on mobile platforms. Cross-platform frameworks like Flutter or React Native will help in reducing the time and cost for mobile development.