

<Name-of-Software-Application>

# **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 |
| --- | --- | --- | --- |
| 1.0 | 08/18/24 | Omar Raymond | Project 1, 2, 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](#_sbfa50wo7nsh)

The Gaming Room aims to expand their popular Android game, "Draw It or Lose It," into a web-based application accessible across multiple platforms. This expansion seeks to reach a wider audience and improve accessibility, enabling teams to compete in solving puzzles based on a library of stock drawings.

To address this demand, Creative Technology Solutions (CTS) plans to develop a scalable web-based version of the game using modern web technologies for compatibility across multiple devices and platforms. The solution will emphasize user engagement with an intuitive design, ensure the uniqueness of game and team names, and manage resources effectively by maintaining a singleton instance of the game in memory.

## Requirements

Expanding "Draw It or Lose It" into a web-based application involves several key requirements. The game must be cross-platform compatible and responsive, ensuring it works seamlessly on various browsers and devices. Real-time multiplayer functionality and a robust backend for handling user accounts and game sessions are essential. Maintaining an intuitive UI and enhancing the user experience while adhering to accessibility standards is crucial. Security measures to protect user data and privacy, along with scalable server performance, are necessary. Additionally, a comprehensive marketing plan and customer support system are required to attract and retain a broader audience. Regular updates and user feedback integration will ensure the game's continuous improvement and engagement.

## [Design Constraints](#_2et92p0)

The primary challenge in developing a web-based version of "Draw It or Lose It" is ensuring real-time interaction among multiple players across different platforms. This demands efficient use of server resources to handle numerous simultaneous game sessions, low-latency data transmission for a smooth real-time gameplay experience, and cross-platform compatibility to reach a broader audience. Consequently, this requires a meticulously designed backend to efficiently manage game state, a frontend that seamlessly adapts to various screen sizes and inputs, and the implementation of web sockets or similar technologies for real-time communication.

## [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)

### Game Structure:

1. **Base Class (Entity):** This serves as the fundamental building block of the game. It includes essential attributes like a unique ID and name that are shared by all game elements, such as players, teams, and the game itself.
2. **Game Class:** This represents the entire game setup, encompassing all the components necessary for gameplay, including the teams involved.
3. **Team Class:** This class manages information about each team, including the team’s name and its roster of players.
4. **Player Class:** This class pertains to individual players in the game, detailing their names and the teams they belong to.

### Connections:

* The game comprises teams, and each team consists of players, resembling a nested structure where each component is built from the core building block (Entity).
* Both teams and players are distinct, ensuring that no two teams or players share the same name or number, like unique sports jerseys.

### Design Principles:

* **Inheritance:** This concept allows different game components (game, teams, players) to inherit common attributes (such as names and IDs) from the base class (Entity), promoting consistency and reuse.
* **Organization:** The game is structured to efficiently manage and track teams and their players, ensuring smooth gameplay and organization.

In summary, we’re developing a web version of "Draw It or Lose It" that’s engaging, visually appealing on any device, and protects team names from duplication. We’re using well-organized building blocks to create a coherent and manageable game structure.

**"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** | Macs can be utilized as servers, though the licensing is costly, and MacBooks are required for development. | Linux is well-suited for web-based hosting and is the most popular option, with the added benefit of being free from licensing costs. | Windows servers are highly secure and easy to set up and use, but the licensing costs are high. | <Evaluate Mobile Devices for their characteristics, advantages, and weaknesses for hosting a web-based software application.> |
| **Client Side** | Mac offers excellent and user-friendly SDKs, but developing for Mac requires a MacBook, which increases costs and necessitates a developer experienced in iOS. | The main expense would be in development time, and it would also require a developer proficient in Python. | Expertise is likely the most significant requirement for Windows development. It is highly recommended to use the .NET framework for its security and capabilities. | For mobile devices, it is essential to find developers with experience in app development. User interaction and display considerations must be handled differently than on the web. |
| **Development Tools** | A MacBook with, VSCode or Xcode installed is required, as all coding will be done using Swift. | Python is pre-installed on most Linux distributions. For coding, you can use PyCharm. | Visual Studio Code is the preferred and most effective tool for developing Windows applications. While you can use a variety of languages, C++ and C# are the most used for Windows programs. | When it comes to mobile app development, for Android apps, you'll need a developer who specializes in Android Studio. For iPhone apps, you'll need a developer with a Mac who can use Swift in Xcode. |

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**: Based on the client's requirements, I recommend using Linux as the server's operating platform. Linux is a popular choice for server applications because of its stability, security, and performance. It offers excellent support for web applications and is highly customizable, allowing developers to tailor the system to their specific needs. Additionally, being open-source, Linux benefits from a vast community of developers who continuously improve and maintain it. Since Draw It or Lose It is a web-based application, Linux would be an ideal platform for hosting the game.

**Operating Systems Architectures**: For the proposed distributed system, I recommend using a multi-tier architecture. This architecture separates the presentation, application, and data tiers, allowing for more efficient scaling and improved fault tolerance. The presentation tier is responsible for rendering the user interface, the application tier handles the business logic, and the data tier manages data storage.

1. **Storage Management**: Given that the game involves rendering images from a large library of stock drawings, a high-performance storage solution is essential for smooth gameplay. I recommend using a solid-state drive (SSD) for the server's storage. SSDs offer faster read and write speeds compared to traditional hard drives, which will enhance the speed of loading images.
2. **Memory Management**: The recommended operating platform, Linux, offers excellent memory management support. Linux employs a paging system that efficiently uses physical memory by swapping unused data to disk. This capability would benefit Draw It or Lose It by optimizing memory usage and enhancing overall performance.
3. **Distributed Systems and Networks**: To facilitate communication between different platforms, a RESTful API is recommended. RESTful APIs are commonly used for web applications and enable communication over the internet using HTTP. The recommended operating platform, Linux, offers strong support for RESTful APIs, making their implementation straightforward. Games should be designed to handle distributed systems and networks to ensure secure and fast communication between servers and clients. This can be achieved by implementing a load balancing system and utilizing a content delivery network (CDN) so that users are served by the nearest server.
4. **Security**: Security is a crucial aspect of any web-based application, and Draw It or Lose It is no exception. Linux, the recommended operating platform, offers excellent support for security features such as firewalls and access control. Additionally, Linux is renowned for its security and is generally considered less vulnerable to security breaches than other operating systems. To safeguard user information across various platforms, I recommend using SSL/TLS encryption for all communications. This will ensure that all data transmitted between the server and clients is secure and cannot be intercepted by unauthorized parties.