

Draw It or Lose It Online

# **CS 230 Project Software Design Document**

Version 1.2

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.3znysh7)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model 3**](#_heading=h.2s8eyo1)

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 2023-07-16 | Dennis James Stelmach | Initial creation of document. Completion of all sections. |
| 1.1 | 2023-07-30 | Dennis James Stelmach | Revise Evaluation to reflect costs associated with each platform. |
| 1.2 | 2023-08-13 | Dennis James Stelmach | Complete recommendations section. |

## [Executive Summary](#_heading=h.35nkun2)

Creative Technology Solutions (CTS) is working with The Gaming Room to develop a web-based version of their game, Draw It or Lose It. The goal is to create a cross-platform game application that supports multiple teams and players. To ensure uniqueness, game and team names will be checked against existing names. This document outlines the software design and requirements to fulfill the client's needs. The proposed solution includes a system architecture design, a domain model depicting the relationships between classes, and considerations for design constraints.

## Requirements

The software design should meet the following requirements, as outlined by The Gaming Room:

* Support for multiple teams: The game application should allow the creation of one or more teams to participate in a game session.
* Multiple players per team: Each team should be able to have multiple players assigned to it.
* Unique game and team names: The system should ensure that game and team names are unique to prevent naming conflicts. Users should be able to check name availability during team creation.
* Unique instances: Only one instance of the game should exist in memory at any given time. Unique identifiers should be used to distinguish each instance of a game, team, and player.

## [Design Constraints](#_heading=h.1ksv4uv)

Developing the game application in a web-based distributed environment introduces certain design constraints that need to be considered:

* Cross-platform compatibility: The game application needs to be compatible with multiple platforms, including web browsers on desktop and mobile devices. This requires using web technologies that are widely supported across platforms, such as HTML5, CSS, and JavaScript.
* Client-server architecture: The game application will follow a client-server architecture, where the game logic resides on the server, and clients interact with it through web browsers. This implies that the server will handle the game state, manage user sessions, and coordinate multiplayer interactions.
* Network communication: As a distributed application, the game must handle network communication between the client and server effectively. This includes transmitting game data, managing real-time interactions, and ensuring data consistency and security over the network.
* Scalability and performance: The design should consider scalability to accommodate a potentially large number of concurrent players. It should also optimize performance to minimize latency and provide a smooth gameplay experience for all participants.
* Security: The game application should incorporate appropriate security measures to protect user data, prevent unauthorized access, and mitigate potential security vulnerabilities, such as cross-site scripting (XSS) or SQL injection attacks.

## [System Architecture View](#_heading=h.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](#_heading=h.2jxsxqh)

* The GameService class maintains a list of games and provides methods to add, retrieve, and manage games. It has a one-to-many relationship with the Game class.
* The Game class represents an individual game session. It contains a list of teams participating in the game. It has a one-to-many relationship with the Team class.
* The Team class represents a team within a game. It contains a list of players associated with the team. It has a one-to-many relationship with the Player class.
* The Player class represents an individual player within a team. Players are associated with teams, and each player has a unique ID and name.
* The Entity class serves as a base class for other classes and provides common attributes such as ID and name. It follows the principles of inheritance and encapsulation.

"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

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Because MacOS is not available to be installed on non-Apple hardware, Mac is not a suitable operating system for server-side operation. It is possible, but the limitation due to hardware availability and cost makes it difficult. | Linux can be installed on any system, has a low or no cost, has light resource requirements, and can be configured in a “headless” configuration, where a display is not required, which makes it an excellent choice for server-side operation. | Windows is a valid choice for server-side due its high software availability and hardware compatibility because of its large market share. Windows is also available for server operation via Windows Server. | Mobile devices do not make a good platform for server-side operation due to its limited hardware, software, and network capabilities. |
| **Client Side** | Mac, Linux, and Windows are excellent choices for client-side operation, as all that is required is a web browser. Technologies such as WebGL and WebGPU allow additional hardware resources to be utilized by the web browser to assist in performance-related tasks for the game as well. | | | Mobile devices are able to be used for client-side operation, but in order to improve user-experience, the web-based game should have progressive web app (PWA) capabilities or should be wrapped into a native app as web browsers on mobile devices are not very intuitive for web-based games. |
| **Development Tools** | As the game will be using the web browser as its window, there are no specific tools for developing on each platform. Development is also possible on all platforms, but developing on a mobile device can be difficult due to lack of support for this function.  When deploying for mobile devices, if it is decided to wrap Draw It or Lose It Online into a native app, then additional tools will need to be utilized, such as Apple’s Xcode or Google’s Android Studio. The Apple Developer Program is required for deploying applications to the IOS App Store, and it costs $99 per membership per year. There are no costs with deploying an application to the Google Play Store. | | | |

## 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**:   
    Considering the availability of all required tooling across the major desktop operating platforms—Windows, Mac, and Linux—the choice of the operating platform may not be a pivotal decision. However, when assessing an optimal operating platform for the development and expansion of Draw It or Lose It Online, Linux emerges as a front-runner due to its comprehensive feature set and cost-effectiveness.

Linux boasts several advantages that harmonize seamlessly with the outlined project requirements. Its cost-effectiveness, coupled with its minimal hardware demands, renders it a favorable choice. Furthermore, its ability to be configured effortlessly in a headless mode for server-side operations enhances its suitability for the project's needs. The open-source nature of Linux accentuates its versatility and adaptability, making it a logical selection for a cross-platform game application. Additionally, Linux's prominence within server environments underscores its capacity to deliver stability and reliability, both crucial factors for the game's server-side operations.

1. **Operating Systems Architectures**:   
   The chosen operating platform, Linux, supports various architectures such as x86, ARM, and more. This flexibility allows for compatibility with a wide range of devices, including desktops, servers, and mobile devices. The application can be compiled and optimized for different architectures, ensuring that it can run efficiently on various computing environments.
2. **Storage Management**:   
   An appropriate storage management system for the recommended Linux platform would be a combination of a relational database management system (RDBMS) and a distributed file system. The RDBMS will provide structured data storage, ensuring efficient querying and management of game, team, and player information. The distributed file system will handle large-scale data storage and management, ensuring scalability and fault tolerance for storing game assets and other non-relational data.
3. **Memory Management**:   
   Linux employs a robust memory management system that includes techniques like virtual memory, memory segmentation, and paging. For Draw It or Lose It Online, Linux's memory management will ensure efficient allocation of memory resources for the game application and its components. This will contribute to stable performance and prevent memory-related issues, enhancing the overall gameplay experience.
4. **Distributed Systems and Networks**:   
   To achieve communication between various platforms and devices, a distributed system architecture utilizing a client-server model is recommended. The server-side logic will reside on cloud servers, managing game state, user sessions, and multiplayer interactions. Clients, which can be web browsers on desktop and mobile devices, will interact with the servers over the network using HTTP/HTTPS protocols. This architecture will enable real-time interactions, game synchronization, and cross-platform compatibility. Dependencies between components should be managed by implementing appropriate error handling, data synchronization mechanisms, and backup strategies to ensure connectivity and minimize outages. With cloud infrastructure, also known as *serverless*, the requirement of constant up-time on a single service is not needed, and disruptions have much less of an impact.
5. **Security**:   
   Security is a critical consideration, and Linux offers robust security features to protect user information across various platforms. Access controls, firewalls, and encryption mechanisms can be implemented at both the server and client sides to safeguard data during transmission and storage. Regular security updates and patches for the Linux operating system will help mitigate potential security vulnerabilities. Additionally, implementing secure coding practices and conducting regular security audits will contribute to the overall security of the Draw It or Lose It Online application.