

The Gaming Room

# **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 | 01/06/25 | Daniel Okonkwo | Changes were made to the cover page, the document revision history, the executive summary, design constraints, system architecture view, domain model and recommendation. |
| 2.0 |  | Daniel Okonkwo | Changes were made on the Evaluation section |
| 3.0 | 06/28/2025 | Daniel Okonkwo | Changes were made to the Recommendation section to reflect some Architectural recommendations |

**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 project is to develop a web-based game that serves multiple platforms based on the current game draw it or lose it, which is only available on android. The purpose of the game is to have multiple teams consisting of several people going four rounds at a minute each. When a picture is pulled from a library of images one team guesses till time runs out. If not answered each opposing team member gets to answer till 15 seconds runs out.

## Requirements

The Gaming Room requires a web-based version of their existing Android game, Draw It or Lose It, that can operate seamlessly across multiple platforms. From a business perspective, the application must support multiplayer functionality, allowing each game to include one or more teams, with each team consisting of multiple players.

## [Design Constraints](#_2et92p0)

* Must run on multiple platforms
* Each team should have multiple players
* Only one instance of the game can exist at any time.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.

## [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)

The UML class diagram represents a hierarchical model using object-oriented principles such as inheritance and composition. The base class 'Entity' includes common attributes 'id' and 'name', which are inherited by 'Game', 'Team', and 'Player' classes. The 'Game' class aggregates multiple 'Team' instances, and each 'Team' aggregates multiple 'Player' instances, demonstrating composition relationships. 'Game Service' acts as a Singleton, managing centralized control of all game instances. This structure efficiently supports the software requirements, particularly the uniqueness of names and the single instance constraint, while demonstrating encapsulation, abstraction, and polymorphism.

**"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** | Mac has easy accessibility and server configurability. Easy to use graphical user interface.  Flexible terminal commands. | Cost friendly. Difficult to navigate the platform.  Command shell for simple server configuration and accessibility. | Server side is expensive. User friendly GUI.  Has a command prompt. | Specifications are better in other devices. Mobile device specifications vary from user to user. |
| **Client Side** | Expensive for users.  Moderate time and expertise required. Accurate skills and needed to navigate OS. | There is a lot of expertise and time required. Linux data is required to use the operating system.  Maximum cost for Linux users. | More expensive than Linux systems.  Easy to learn and understand how to support a Windows setup. Minimum  expertise needed. | Provide flexibility to clients or even developers to see updates at any place. Slightly more  difficult to  implement than other devices. |
| **Development Tools** | Languages that consist of HTML, CSS and JavaScript. Libraries to support frontend development. Other development tools include PyCharm, GitHub, Visual Studios etc. | Languages that consist of HTML, CSS and JavaScript.  Libraries to support frontend and languages. Other. Linux systems include JavaScript, Ruby, PHP and Python. | Languages that consist of HTML, CSS and JavaScript.  Libraries to support frontend and languages. Developer tools  include Eclipse,  command prompt, PyCharm,  Eclipse, etc. | Languages that consist of HTML, CSS and JavaScript. Libraries to support frontend and  languages. IDE’s for programming languages consist of HTML, php, C++ and Python |

## 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 recommend using **Amazon Web Services (AWS)** as the operating (server) platform to allow The Gaming Room to expand *Draw It or Lose It* to other computing environments. AWS supports scalability, flexibility, and global distribution, which will help the game reach a wider audience while maintaining performance and reliability.
2. **Operating Systems Architectures:** AWS supports various operating system architectures, including **Linux and Windows-based servers**. Using Linux architecture for the server is cost-effective, stable, and offers strong community support. It uses monolithic **kernel architecture**, which allows direct communication between system resources, providing efficient process and memory management while reducing overhead.
3. **Storage Management**: AWS offers **Amazon S3 (Simple Storage Service)** for scalable object storage management and **Amazon EBS (Elastic Block Store)** for persistent block-level storage. S3 can store user-generated content like images and drawings, while EBS can handle game state data and logs efficiently. Both services provide automatic replication, backup capabilities, and easy integration with the game server.
4. **Memory Management**: The recommended AWS Linux server platform uses **virtual memory management**, which allows processes to use more memory than what is physically available through paging and swapping. This ensures that *Draw It or Lose It* can handle multiple players without running into memory issues, and memory can be allocated dynamically to the application as needed.
5. **Distributed Systems and Networks**: To allow Draw It or Lose It to communicate across various platforms, AWS Lambda, API Gateway, and DynamoDB can be used to build a serverless architecture, enabling the game to scale easily while reducing server management needs. AWS uses a high-speed, redundant network infrastructure that supports communication between services globally. Dependencies between components are managed through reliable APIs, and AWS provides monitoring and automatic failover to address connectivity issues and outages quickly.
6. **Security**: AWS provides **end-to-end encryption** using SSL/TLS to protect user data during transmission between platforms. User authentication and access control can be managed using **AWS Cognito**, ensuring secure user sign-ins and data handling. Data at rest in S3 and EBS can be encrypted using AWS Key Management Service (KMS).