

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 | 04/20/2025 | Darius Wilder | Changes made to the document revision history, executive summary, requirements, design constraints, system architecture view, domain model and recommendations. All sections were modified with updated information. |

**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 Company plans to design a web-based video game that is compatible with a diversity of platforms that is based on their current game *Draw It or Lose It*. Included are four 60-second rounds with pictures serving as clues, slowly appearing until the 30-second mark is reached. If the team fails in solving the puzzle, the opposing team is granted one chance to guess within 15 seconds to win the game.

## Requirements

* Must accommodate one or more teams
* Each team can accommodate multiple players
* Names for individual players and teams must be unique and allow verification of name availability
* Only one game can occur at a time
* Must be web-based with an internet connection
* There must be four rounds with 60 seconds of gameplay each
* The pictures must serve as clues that gradually appear until the 30 second mark is reached
* The pictures must represent a title, phrase, adage
* If the team fails in solving the puzzle, the opposing team is granted one chance to guess within 15 seconds to win the game.

## [Design Constraints](#_2et92p0)

* Each team should have multiple players
* Names for each player and team must be unique with verification of name availability
* The game must be able to operate on a diverse range of platforms

## [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)

This UML class diagram describes the class structure of the Draw It or Lose It application system. The main nucleus is the Entity class, which serves as a superclass for every entity of the inside the application. It contains the more generic attributes such as “name” and “id,” which enforces the constraint of having a unique name and identifier.

The Game, Player, and Team classes are the subclasses that extend from the Entity superclass. These subclasses represent the core entities within the game application: the Game is a collective of various teams, while each Team is a collective of Players.

There are several composition relationships with the game design: the GameService class shares a composition relationship with the Game class, as the former manages the instances and object references of the latter. There are also similar relationships between Game and Team classes, along with the Team and Player classes.

The ProgramDriver class contains the game’s main function which includes a singleton instance of GameService created that initiates a single instance during the game. The responsibilities of ProgramDriver are to add games, teams and players using the GameService instantiation. There’s also dependency on the SingletonTester class.

This UML class diagram represents object-oriented programming principles, such as inheritance, encapsulation, and abstraction. Inheritance is described through the relationship between the superclass Entity and its subclasses, which allows the inheriting of attributes and behaviors. Encapsulation is when the GameService class conceals its attributes in the name of data security and abstraction. Abstraction is shown through the management of games, teams and players; the emphasis of important interactions, while hiding irrelevant details. This promotes consistency in the game structure, upholds code arrangement and limits redundancy.

**"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 servers, built  on Unix, provide a  stable platform for  web apps and offer  server-based  deployment  options. Despite  being developer-  friendly with tools  like Xcode, they  come with higher  hardware costs  (around $2500) and  an additional $500  for server licenses.  However, scalability  is somewhat limited  compared to Linux  and Windows  Provides a  stable platform for  web apps and offer  server-based  deployment  options  Though developer-  friendly with tools  like Xcode, hardware costs are more expensive  (roughly $2.500), with  an additional $500  for server licenses  Limited scalability  compared to Linux  and Windows | Open-  source OS  Customization and  flexibility with a  diverse software  ecosystem.  Renowned for  scalability,  stability, and  security,  Some GUI  limitations and  hardware  considerations  Servers  ranging from  $9.99 to $399 monthly  are  commonly used for website hosting | Offering broad  software  compatibility,  strong developer  ecosystem,  supported by  extensive  hardware  compatibility and  documentation  Higher  prevalence of  known security  vulnerabilities    Windows servers  do not  have service fees,  attractive option  for website hosting | Characterized by  their portability,  touchscreen and  gesture-based  interactions within a  limited screen size    Spectrum of  hardware capabilities  across various  devices  Variations in  performance and  features  Their cost can vary  significantly |
| **Client Side** | Easy to learn  User-friendly interface  Attracting and maintaining multiple clients can increase costs  May require expertise and various skill sets | Requires time and expertise  Operating system requires Linux data  Expensive for Linux users | Licensing costs may have detrimental effect on the project budget | Limitations with connectivity and software responsiveness would have to be considered  May require the usage of features such as camera, GPS location, notification permission |
| **Development Tools** | Commonly used tools include JavaScript and Node.js  Compatible with popular IDEs such as Xcode and Visual Study Code | Languages include HTML, CSS, JavaScript  Libraries supporting front-end and languages  Systems such as JavaScript, Ruby, Python | Programming languages such as C# and .NET are used to software building and deployment | Programming languages such as Swift and Objective-C for iOS and Java for Android are used to develop software for mobile devices |

## 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**: The best platform for the project to thrive in more computer environments is Windows, due to a variety of integrated development environments (IDEs). One advantage is that Windows 10 and 11 being compatible with newer games. Microsoft DirectX is a collection of application programming interfaces (APIs) that deals with video game programming.
2. **Operating Systems Architectures**: Windows is an operating system with optimal abilities for storing files, gaming, videos, coding and other features that are used interchangeably with an internet connection.
3. **Storage Management**: Windows has the capability of analyzing and managing files in the hard drive, monitoring storage space and calculating when the storage is nearly full. One feature is Storage Sense, which allows viewing, organizing, and controlling the files located on the hard drive. It provides a visual of which folders and files occupy the most space, along with the prearranged scheduling of deleting irrelevant files from memory. You decide which files are slated from deletion and which files are spared and saved.
4. **Memory Management**: Windows can allow storage for *Draw It or Lose It* photos and data relating to player and game information into a singular and safe location. Depending on the application’s requirements, Windows automatically sizes and relegates the memory to continuing processes. To promote performance without interruption, Windows will acquire a portion of the hard drive as virtual memory to store dormant data temporarily when the physical RAM reaches capacity. Also, the system can identify and resolve memory issues that would cause crashes and inconsistencies.
5. **Distributed Systems and Networks**: Network games with multi-user interaction systems typically possess a database for all players and data. Cloud platforms such as Google Cloud (Cloud Spanner) and Amazon (DynamoDB) can provide data consistency amongst players from various locations. Using this method, developers have minimal responsibilities with infrastructure management. Open-source libraries such as ENet and RakNet offer more versatility and control with network behavior, yet the building process is needed.
6. **Security**: Windows features built-in security protection software, yet it is highly recommended to use an alternative source to secure user data and information. It is highly recommended to research features, costs and results associated with third-party security protection software. Evaluation of your budget and specific needs are recommended. It is highly discouraged to use free security software due to questionable functionality and privacy safeguards.