

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.3

## 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 | 03/24/2024 | Carl LaLonde | Initial Draft |
| 1.1 | 04/07/2024 | Carl LaLonde | Updated Evaluation |
| 1.3 | 04/19/2024 | Carl LaLonde | Updated 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)

Creative Technology Solutions has been tasked with developing a web-based game based on our clients current game, Draw It or Loose It. The game will slowly reveal, to the player/team, images from an already stocked library of drawing, as clues, where the player/team will guess what is being drawn. Each game will consist of four rounds, each round lasting one minute. And each drawing is fully revealed after 30 seconds. If the team does not guess the what the image is before time expires, the remaining teams have an opportunity to make a one time guess to solve the puzzle with a 15 second time limit.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

* The game must be web-based.
* Has the ability to have solo or multiplayer mode.
* Game and team names must be unique to allow users to check weather a name is already in use.
* Only one instance of a game may exist at any given time.

## [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 Entity class has a relationship with Game class, Team class, and Player class. The open arrow from the three classes to Entity class indicates that these three classes inherit attributes and methods from Entity class.

GameService class is connected to Game class using a solid line and (0…\*) indicating a one-to-many relationship. This means that one instance of GameService may be related with multiple instances of Game class.

Game class is, again, connected to Team class using a solid line and (0…\*) indicating a one-to-many relationship, signifying that one instance of Game class may be related with multiple instances of Team class.

Team class is also connected to Player class using the same solid line and (0…\*) indicating the same one-to-many relationship.

The ProgramDriver class is connected to SingletonTester class using a straight solid line with a closed arrowhead at SingletonTester and also with <<uses>>. This means that the PrgramDriver class uses the SingletonTester class to run the program according to design specifications, i.e. there can only be one instance of a game at any give time.

**"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** | **Characteristics:**  -Stable and reliable platform.  -Built in support for hosting web-based apps.  -Offers features for security.  **Advantages:**  -Simplified setup and management process.  -Highly reliable  **Weaknesses:**  -Higher costs  -May lack advanced customization options. | **Characteristics:**  -Widely used  -Stable, secure & flexible  **Advantages:**  -Open-source availability  -Flexibility for different usage  -Stable uptimes and performance  -Ample security features.  **Weaknesses:**  -Less known system  -Compatibility issues | **Characteristics:**  -Support for management and security features  -Widely used  -Stable and secure  **Advantages:**  -Compatible with other Microsoft products  -Familiar user interface  -Documented support options  **Weaknesses:**  -Higher costs  -May require more maintenance  -Compatibility issues with other platforms. | **Characteristics:**  -Offers portability and accessibility  -Hardware and OS may vary  -Access capable with internet.  **Advantages:**  -Portability  -Touchscreen interaction  -Web-based applications may make use of device features such as camera or gps.  **Weaknesses:**  -Limited processing power, memory, and battery  -Varying screen sizes  -Browser compatibility issues. |
| **Client Side** | **Cost:**  -Testing may require costly Mac hardware.  -Development Tools may require addition costly resources.  **Time:**  -Learning Mac development tools and frameworks may require more time.  -Updates and maintenance for compatibility must be accounted for.  **Expertise:**  -Developers must have an understanding of MacOS and iOS designs.  -Understanding of cloud may be required | **Cost:**  -Open-source typically implies lower costs.  -Open-source development tools.  -Cost may be tied in to testing.  **Time:**  -Learning Linux platforms and frameworks may require more time.  -May require testing over many distributions.  **Expertise:**  -Requires expertise in Linux development designs.  -May require familiarity with different distributions. | **Cost:**  -May require upfront costs for licensing fees.  -Development tools may require additional costly resources.  **Time:**  -May require learning Windows platforms and frameworks.  -May require testing across multiple versions of Windows OS.  **Expertise:**  -Requires knowledge in Windows development designs.  -Requires knowledge in Windows frameworks and platforms. | **Cost:**  -Testing may require additional hardware running different OS’s.  -IDE’s typically free  -IDE’s may require additional costly resources.  **Time:**  -May be required to develop separate versions for iOS and Android.  -Learning multiple platforms and frameworks may require more time.  -Testing across different hardware with different OS’s will require more time.  **Expertise:**  -May be required to understand platform specific libraries and frameworks.  -May require familiarity with user interface designs. |
| **Development Tools** | **Programming Languages:**  -Swift  -Objective-C  **IDE’s & Tools:**  -Xcode  -Visual Studio  -Homebrew  -Git | **Programming Languages:**  -C/C++  -Python  -Java  **IDE’s & Tools:**  -VS Code  -Eclipse  -Git  -Docker | **Programming Languages:**  -C#  -Visual Basic  -JavaScripts  **IDE’s & Tools:**  -Visual Studio  -Eclipse  -.NET Framework  -Git | **Programming Languages:**  -Swift  -Objective-C  -Java  -Kotlin  **IDE’s & Tools:**  -Xcode  -Android Studio  -Flutter  -React Native |

## 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**:

Linux is the recommended operating platform for hosting Draw It or Lose It. Linux is known for its stability and reliability, making it a qualified choice for hosting web-based applications such as Draw It or Lose It. Linux is also an open source operating system, which means that it is freely available and offers flexibility for customization. Open source also implies lower costs. Linux operating systems also provide plenty of security features which is helpful to protect the game and users data.

1. **Operating Systems Architectures**:

Linux operating system is a layered operating system with a modular architecture. The core of the operating system or the “kernel” manages activities such as task scheduling, memory management and disk management. The kernel interacts with System Libraries, acting as an interface between the kernel and applications. System Utility Programs offer users the tools to perform such tasks as file management, system monitoring, network configuration, and user management. The Shell acts as an interface between the user and the kernel. There are two interfaces for the shell, command line and GUI Graphical User Interface. The architecture of Linux is highly scalable and customizable making it a good fit for hosting Draw It or Lose It.

1. **Storage Management**:

Linux supports connectivity to Storage Area Networks. Storage Area Networks or SANs are high-speed networks that connect servers to storage devices, allowing servers to access the shared storage as if it were a drive directly attached to the server. Container storage solutions such as Docker can be used to manage storage for applications on Linux servers while ensuring consistency and portability across different user environments.

1. **Memory Management**:

The Linux kernel utilizes virtual memory management, which allows the efficient allocation of memory resources to processes. Demand paging is a technique which loads only the necessary portions of a program into memory. Linux also incorporates techniques for memory protection, ensuring security and stability of the system. By utilizing these Linux memory management features, Draw It or Loose It will remain at optimal performance.

1. **Distributed Systems and Networks**:

Draw It or Lose It can be designed as a distributed system by using service-oriented architecture or microservices architecture. There could be separate services for managing game sessions, user authentication, and drawing recognition. Each service enables platforms to interact with the desired function. RESTful API’s may be used for starting game sessions, submitting guesses, or managing user accounts.

1. **Security**:

Linux offers built-in security features to protect user information across various platforms. File permissions and user authentication may be controlled at the operating system level to control access to sensitive data. Encryption may be utilized to secure data transfer over networks. By implementing these methods, Draw It or Lose It will better protect user information across different platforms.