

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)4

[**Evaluation**](#_2o15spng8stw)5

[**Recommendations**](#_m8aleynsvzvc)6

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/21/2022 | Veronika Skliarova | Initial draft. Executive summary, design constraints, and domain model. |
| 1.0 | 06/04/2022 | Veronika Skliarova | Added evaluation for different operating platforms |
| 1.0 | 06/19/2022 | Veronika Skliarova | Recommendations section |

**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)

Our new client, The Gaming Room, needs a new game developed. “Draw It or Lose It” is loosely similar to the 1980s television game Win, Lose or Draw, where teams compete to guess what is being drawn. The application should render images from a large library of stock drawings as clues. A game consists of four rounds of play lasting one minute each. Drawings are rendered at a steady rate and are fully complete at the 30-second mark. If the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess each to solve the puzzle with a 15-second time limit.

## [Design Constraints](#_2et92p0)

Since the game is web-based, the users will be accessing it from different kinds of platforms and browsers, which means we have to consider restrictions of each platform and make sure the application runs smoothly. Development varies for different platforms, so we will either need multiple teams working on different implementations, or we should expect the project to take longer.  
A capable server is a must-have to ensure stable performance when multiple players join.

We will use singleton pattern for the game service and unique identification system for games, teams, and players, to make sure that everything stays synchronous, and we don’t end up with duplicate games/teams/players by accident because of unexpected timing.

## [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 is the base class to hold common attributes and behaviors. Game, Team, and Player classes all inherit from entity. This is an “IS A” relationship. Classes Team and Player have a “HAS A” relationship, where Team’s instance can contain multiple instances of Player. Same goes for Game and Team classes, where Game’s instance can contain multiple instances of Team. GameService class utilizes a singleton pattern by using a private constructor and a public getInstance() method. GameService contains static attributes for the unique identifiers system for classes Game, Team, and Player. GameService has a “HAS A” type relationship with Game class. ProgramDriver is the class that contains a main() method. SingletonTester is a class that contains unit tests, which are called from ProgramDriver class.

**"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** | Uncommon choice for hosting web-based applications. Expensive and impractical, but the most secure. | Most popular option for hosting web-based applications. Reasonable price doesn’t require a lot of resources, especially when compared to MacOS or Windows. Allows to implement our own security features. Easy to maintain and scale as the popularity of Draw It or Lose It grows. | A popular, albeit least secure choice. Expensive and requires a lot of resources to run. Requires use of Microsoft proprietary software, which means the software will be only Windows-compatible. | Not suited for hosting of web-based applications. |
| **Client Side** | Google Chrome and Mozilla Firefox are both available and widely used on MacOS, Linux, and Windows. A responsive web-design is required to scale the application to the screen’s size of whatever device the users are accessing it from. Prioritizing those two browsers will allow to cut down costs and development time, since they can be used from any operating platform. | | | Since the application already exists on the Android market, we can reuse the same design for iOS. Need to allocate enough time for development and testing, since iOS platform requires different tools. |
| Default browser is Safari. Will require more testing to ensure the same performance level as Chrome/Firefox. | Most distributions come with Firefox by default. | Default browser is Microsoft Edge. It uses Chromium engine, same as Google Chrome. No expected compatibility issues. |
| **Development Tools** | For backend we will use Java for its out-of-the-box running capabilities on any platform. IDE is IntelliJ Community, which has plenty of features for speedy development at no cost. For UI we will go with open source React JS framework and VSCode IDE. For deployment we will be using Docker. All tools are available for each macOS, Linux, and Windows. | For backend we will use Java for its out-of-the-box running capabilities on any platform. IDE is IntelliJ Community, which has plenty of features for speedy development at no cost. For UI we will go with open source React JS framework and VSCode IDE. For deployment we will be using Docker. All tools are available for each macOS, Linux, and Windows. | For backend we will use Java for its out-of-the-box running capabilities on any platform. IDE is IntelliJ Community, which has plenty of features for speedy development at no cost. For UI we will go with open source React JS framework and VSCode IDE. For deployment we will be using Docker. All tools are available for each macOS, Linux, and Windows. | For backend we will use Java for it’s out-of-the-box running capabilities on any platform. IDE is IntelliJ Community, which has plenty of features for speedy development at no cost. For UI we will go with open source React Native JS framework and VSCode IDE. The benefit of React Native is that it runs on both Android and iOS devices, and is essentially the same as React framework we’re using for web applications. |

## 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 Gaming Room should host its application on Linux platform. Linux is open-source, and it supports most programming languages, whereas Windows will require the use of Microsoft’s expensive proprietary software. As the application and its customer base grows, Linux will be more than capable to handle increasing traffic.
2. **Operating Systems Architectures**:   
   Linux OS Architecture consists of a few layers: hardware, kernel, shell, and utilities. Hardware layer consists of peripheral devices, such as CPU, HDD or SSD, RAM. Kernel is the core of the system; it interacts with software and provides low level service of upper levels. Shell is the interface between kernel and user, can be graphical or command-line. Utilities are programs that provide most of functionalities to the user.

1. **Storage Management**:   
   For storage management I would recommend a cloud service. This way we will not have to maintain our own data center and will be able to easily scale the storage size as the platform grows.
2. **Memory Management**:   
   Memory management system in Linux employs partial memory execution through demand paging and virtual memory. Virtual memory is a layer of memory addresses that maps to physical addresses and is allocated to the program when an instance of said program demands more RAM than available (*What is virtual memory and demand paging?*, 2016). Demand paging is a type of swapping done in virtual memory systems, where the data is not copied from the disk to the RAM until they are needed or being demanded by some program (*What is virtual memory and demand paging?,* 2016).
3. **Distributed Systems and Networks**:   
   As the game Draw It or Lose It becomes more popular, more users will be joining. One of the biggest benefits of distributed systems is the fact that more machines can be added at any time with no issues. If one of them goes down, users shouldn’t notice the difference, since others will “pick up the slack”. They can be difficult to manage, since a lot of monitoring, logging, and load balancing functions need to be added for visibility into the operation and failures of the system (*What is a distributed system? - an introductory guide*, n.d.).
4. **Security**:   
   By default Linux OS uses best practices when it comes to security, such as limited user privileges/permissions, file system security, audit trails, and others. We can build on that to customize the security that The Gaming Room requires. On top of everything development team will be following the best practices, and before the release a round of pentesting will be conducted.

References

*What is a distributed system? - an introductory guide*. Confluent. (n.d.). Retrieved from https://www.confluent.io/learn/distributed-systems/

Jithin. (2016, November 29). *What is virtual memory and demand paging? - interserver tips*. Retrieved from https://www.interserver.net/tips/kb/virtual-memory-demand-paging/