

Draw It or Lose It

# **CS 230 Project Software Design**

Version 3.0

## 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 | 07/14/2023 | Frederick Case | Added an Executive Summary, the software requirements, design constraints, and a Domain Model. |
| 2.0 | 7/29/2023 | Frederick Case | Added findings about operating systems under Evaluation. |
| 3.0 | 8/11/2023 | Frederick Case | Added recommendations for operating platform and its architecture, storage and memory management, distributed system and networks, and security. |

**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)

In Draw It or Lose It, we will need to create software that will have multiple teams and players in play. Each team will take turns, with one team being the drawing team and the others as spectators who have the chance to steal. Players on the drawing team will then need to create drawings from a database of stock photos so that their team can guess what they are drawing. Each drawing team has 30 seconds to guess what the drawing player is creating. After the 30 second timer, there is a 15 second opportunity for each team to offer one guess as to what the drawing is.

In this software, we will need to create the game with its rules, players within the game, and teams that each player is in. The game will need to have its rules implemented, have a stock photo database that it can draw from, have a class where we can get user input for the drawings, and a timer for each round.

## Requirements

The client has requested that these software requirements need to be met for this game application:

* The game must have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* The game and team names must be unique, which allows users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time.

## [Design Constraints](#_2et92p0)

The design constraints for this application will be converting this application from being an exclusive app on Android to a web-based game that can be used on multiple platforms. Since it will be a web-based game, we will need to make sure that this application can be used online with multiple users being able to log in to the game using their computers from home. We also need to consider how different input will be put into the application. Mobile users will be able to draw using a touch screen and web users will be able to draw using a mouse. They will also need to type in their answers.

## [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)

In this application, we will have a parent class called “Entity” where we will be able to create private “id” and “name” attributes. “Entity” will have methods that will get the “name” and “id” attributes from user input in the game. “Entity” will have child classes “Game”, “Team”, and “Player” that inherit these attributes from the “Entity” class and will have their own unique set of attributes and methods. Each of these classes have a relationship with each other through an iteration pattern and through a singleton pattern. The singleton pattern will ensure that every team name and player name will be unique, and the iteration pattern will iterate through arrays within the program to add new instances of names if it has not been used. This program will have a driver class that uses a main method to execute the application and will use the “SingletonTester” class to make sure that all the players and teams in the game have unique names. The child class “Game” will have a private array of “Teams” that will hold all the team names. It will also consist of a constructor and methods that will add the team names to the array. The child class “Team” will have a private array of players that holds a list of all the different players on each team. The constructor and methods in this class will add each unique player’s name to the team. The last child class, “Player” consists of a constructor and method to add players names to the program. We also have a GameService class that has a relationship to “Entity” and its child classes through the iteration of these arrays. “Gameservice” has private methods that iterate through the arrays created throughout the program and is tested through the singleton pattern to be unique. “GameService” also has public methods called getters that get information for each instance of the game’s name, the games id, the games count, the team names, and each player’s name. The driver will allow us to run the program and will give us output during testing to ensure the program is running without bugs.

All instances of private attributes of private methods in this application are an example of encapsulation. Having child classes that inherit data from a parent class is an example of inheritance. Since “Game”, “Team” and “Player” inherits from “Entity”, we can say that this is an example of polymorphism as well because “Entity” refers to objects from its child classes throughout the program. Since we are creating this program to be user-friendly, we are using abstraction. This is because the users will only have to enter their name, see what team they are on, and will be able to play the game without entering any complex data to run the program.

**"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** | Mac is a great option for building web-based applications because you can build applications on multiple operating systems using parallel. This means you can run MacOS, Linux, and Windows side by side. Mac also offers better security than devices that are run by Windows. | Devices using a Linux OS can build web-based applications more securely than its counterparts. Linux is also faster and does not need updates to improve its applications. However, Linux offers far less support when compared to Windows or Mac and is less user friendly. | Windows is great for building a web-based application because of its adaptability. Windows can be customized in any way the user wants and is the best OS for gaming. The downside of Windows is that it is easily susceptible to malware and viruses. | Building a web-based on your mobile device offers portability and can integrate mobile phone features like its camera and GPS. However, mobile devices have less RAM and memory which can limit the applications performance and scalability. |
| **Client Side** | Although Mac devices are generally more expensive out of the box when compared to devices with other OS’s, the hardware is comparable in specs. Mac users also generally will spend less on repairs over time. | Linux is a free to use Operating system, so it is the cheapest option to build applications on. However, admins need to spend more time on applications because every feature needs be added manually. | Windows is the most used OS because most of the low-cost devices on the market use Windows. However, the type of hardware can vary therefore inconsistencies in performance are apparent. | Creating a web-based application on a mobile device requires development for multiple platforms, which adds time for debugging and testing. |
| **Development Tools** | MacOS is the only operating system that can build applications for IOS. You are able to also run additional OS’s with their IDEs and languages on a mac using a virtualization application. | Linux applications are mostly written in C/C++, but you can also use Perl, Python, and Java. You can use Microsoft Visual studio and Eclipse on Linux. | Windows can use virtually all IDE’s and programming languages available, minus Apache or IOS. Windows users can customize their hardware easily to do whatever they want with it. | The type of programming language depends on the mobile device. IOS uses Swift, while Android uses Java. You can find many IDE’s in a mobile devices App store like AIDE for Android and Kodex for IOS. |

## Recommendations

1. **Operating Platform**: For expanding Draw it or Lose it to other computing environments, it is my recommendation to use a cloud-based server, more specifically, Amazon Web Services (AWS).
2. **Operating Systems Architectures**: AWS supports multiple operating system architectures, including MacOS, Windows, and Linux environments. This can be done by using AWS tools for PowerShell. There are also AWS SDK’s that are available for a variety of popular programming languages and platforms like Python, C++, Java, etc. This can allow developers to use any application build for any OS and have it work on this cloud operating platform (Barney and Gillis, 2022).
3. **Storage Management**: AWS uses their Amazon Simple Storage Service (S3) for scalable storage. This is used for data backup, data collection, and analytics. This will help keep user information stored for each player. AWS also has block-level storage using Amazon Elastic Block Store that saves data when using EC2 (Elastic Compute Cloud) instances. This will ensure that there is efficient storage management for virtual machines and containers within the program (Barney and Gillis, 2022).
4. **Memory Management**: AWS’s EC2 allows for efficient memory management because it provides virtual servers that assist with computing capacity. EC2 will help with faster computing times within these servers that have memory-intensive computing jobs. AWS also utilizes its Amazon ElastiCache and DynamoDB Acceleratory for real-time caches for applications. This will allow frequently used data to be accessed quicker when users are accessing images from the app (Barney and Gillis, 2022).
5. **Distributed Systems and Networks**: AWS uses Amazon Simple Queue Service (SQS) for communication between distributed applications. This can ensure that distributed systems and networks can work as intended by sending, storing, and receiving information from linked applications on the same platform. Amazon API Gateway also lets a development team create and manage APIs that allows applications access back-end data. This allows the game to communicate with other forms of the application on different platforms (Barney and Gillis, 2022).
6. **Security**: AWS boasts a plethora of security features like AWS Identity and Access Management, Amazon Inspector, provides software and hardware-based encryption, has a management console, and so much more. AWS Identity and Access Management lets admins manage user access to resources in a program. This helps keep sensitive information out of the hands of users looking to steal information for malicious purposes. The Amazon inspector assesses potential security risks by analyzing the AWS environment for vulnerabilities in security. AWS also includes tools and services that encrypts sensitive information that protects against DDoS attacks. Finally, the AWS Management Console manages resources in cloud computing, cloud storage, and security credentials so that information in the application is only accessible to those with access (Barney and Gillis, 2022). AWS has a lot to offer when it comes to security with their clients!

References:

Barney, Nick, and Alexander S. Gillis. “What Is AWS (Amazon Web Services) and How Does It Work?” *SearchAWS*, 6 Oct. 2022, www.techtarget.com/searchaws/definition/Amazon-Web-Services.