

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)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 3/19/2021 | Joshua Hampton | Updated Executive Summary and Design Constraints |

**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 Android application Draw It or Lose It needs to be reworked to be developed as a web application. The features of the game require that their will need to be at least one team with multiple players able to be assigned to each team. Game and team names must be unique, and users should be able to check if their desired name is taken. Only one instance of a game should be able to be ran at any given time. We can achieve this through the use of singleton classes and iterators patterns.

## [Design Constraints](#_2et92p0)

The game is written in Java and will be web based so it will need to run as a Java applet. To be able to run on as many devices as possible the game will need a light footprint and use limited memory. The user interface will also need to be touch friendly for mobile devices.

## [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 below provides a visual representation of the code for Draw It or Lose It. The ProgramDriver class contains the main method which is the starting point for the program and contains the function and method calls for the program. The ProgramDriver class uses the SingletonTester class to access the testSingleton method which checks to make sure the singleton GameService object cannot be instantiated more than once. The Entity class is shown as the base class of the Game, Team, and Player classes. This means that the Game, Team, and Player classes inherit the variables in the Entity class. The GameService class is associated with the Game class, such that we can create zero to many Game objects. The Game class is associated with the Team class, such that we can create zero to many Team objects. The Team class is then associated with the Player class, such that we can create zero to many Player objects.

****

## [Evaluation](#_2o15spng8stw)

**Mac**

* No server environment
* Higher cost for developing publishing Applications
* Limited hardware variations provide a more stable environment
* Higher Operating costs

**Linux**

* Largely Open source
* Smaller client base
* Stable for server environment
* Lower Operating costs

**Windows**

* Large variety of hardware options means limited ability to test
* Frequent updates can cause unforeseen bugs
* Widely used
* Server environment available
* High operating cost for server side

**Mobile Devices**

* Large variety of hardware and operating systems making it difficult to test
* Touch screen compatibility required
* Large user base
* Portable

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac does not have a server environment, any server-side resources would be from a third party. | Linux is largely open source which provides a low operating cost. Linux is also widely used and stable for server environments. | Windows is good for server applications but has a higher operating cost. Windows server is typically used for enterprise domain environments. | Hosting a web-based server on a mobile device would not work due to limited hardware resources. |
| **Client Side** | Mac has widely standardized hardware and software throughout their systems making it very stable and easier for testing. | Linux is not widely used as a client and has a wide variety of hardware that it can be installed on making testing very difficult and costly. | Windows is very widely used in homes and offices. It also is installed on a large variety of hardware making testing more difficult. | Mobile devices would be used strictly on the client side. Since there is a wide range of hardware, operating systems, and version in use testing would be very difficult. |
| **Development Tools** | Objective-C and Swift are typically used for Mac applications. Java can also be used. Development can be done on IDEs such as Xcode, AppCode. Since the software will be web based very little, if any, code will need to be written in Swift unless this was chosen as a server. | C, C++, Java, or Python could be used for programming on Linux. Linux has many IDEs including visual studio, eclipse, and NetBeans to name the most popular. | Windows can use C, C++, C#, Python and Java. Many IDEs exist for programming on Windows such as Visual Studio, Eclipse, NetBeans, and IntelliJ. | IDEs for programming on mobile devices vary depending on the platform they are being developed for. For iOS Xcode would be the most common. For Android Eclipse, NetBeans, or Android studio would likely be amongst the top choices. |

## 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 Linux to be used as the server-side operating platform for Draw it or Lose It. Linux will provide an affordable, maintainable, and scalable operating platform for Draw it or Lose it to grow on. Since the game will be a java web app it will be easy to serve all clients simultaneously without making changes to the code base.
2. **Operating Systems Architectures**: I am recommending a Serverless Architecture as the server-side application can be run using third party services to handle the authentication and database. This architecture will also provide additional affordability as there will not be the initial overhead of purchasing, configuring, and maintaining a physical server. A serverless architecture will also provide additional flexibility to scale resources as necessary and provide superior uptime.
3. **Storage Management**: Since we will be using a serverless architecture, server-side physical storage will be handled by the cloud vendor. A storage pool will be created to store all app data, therefor RAID is not a necessary concern. Client-side storage will be managed by having a local cache of images to prevent a high overhead on devices with limited space.
4. **Memory Management**: To limit memory consumption on devices with limited memory, we will keep images in cache until they are about to be used or are currently in use. This will provide increased performance without creating a high overhead on memory utilization. Once an image has been used it will be immediately removed from memory.
5. **Distributed Systems and Networks**: With a serverless architecture redundancy will be provided in the cloud system itself. If one cloud server goes down the app is simply moved to another server. This serverless architecture provides superior dispersion and uptime to a traditional client-server configuration.
6. **Security**: With a serverless architecture, security and user authentication will be provided by third party cloud services such as AWS Cognito with AWS credentials. This will provide an easy to configure and maintain login service and also provides the ability for users to authenticate through social identity providers such as Facebook, Twitter or amazon. (Bashir, 2019)

**Citations**

Bashir, F. (2019, July 13). What is Serverless Architecture? What are its pros and cons? Retrieved April 18, 2021, from <https://www.freecodecamp.org/news/what-is-serverless-architecture-what-are-its-pros-and-cons/>