

Draw It or Lose It

# **CS 230 Project Software Design Document**

Version 3.0

## 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)6-7

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 3.0 | 12/15/2020 | Tyler Coplan | Added Executive Summary, Design Constraints, Domain Model, Evaluation and 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)

The Gaming Room is looking to expand the reach of their popular Android app “Draw it or Lose it.” A web-based solution would allow people to play the game on multiple platforms over the internet. A game service program is required to manage each game and keep data consistent across all users. The game service program will manage multiple games, having one or more teams per game, with multiple players on each team. All games, players, and teams will have unique names and ids. The web-based app will allow for rapid and low-cost development across multiple platforms; however, it will lack some of the more polished features that a platform native app can offer.

## [Design Constraints](#_2et92p0)

Application Requirements:

* Each game will have one or more team
* Each team will consist of multiple players
* Each player, team and game will have a unique name and id#
* Must work on multiple operating platforms

In addition to the technical requirements, a web-based app will be constrained to running through a web browser. This will limit the options for input, impact display layout, and supporting multiple web browsers must be considered. However, the web-based app will take advantage of the existing functionality of the user’s operating system and web browser, providing a fine user experience with minimal development.

## [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 diagram below shows how the application will be structured. The program will be run out of the ProgramDriver class, which also uses the SingletonTester for testing purposes. The GameService class will manage the data of the program, keeping a record of all games, players and teams that have been created. To ensure that all users access the same data, the singleton pattern is used to create the GameService. The singleton pattern ensures that only one instance of GameService exists, which will then provide all users with access to the same data since all users will access the same object in memory. The data for teams, players and games exist in their own respective classes, but will inherit some of their functionality from the Entity class. The Entity class serves as a super class that implements all the data members and functions that are shared between games, teams and players. Since every game, team and player will have a unique name and id, and each entity needs a function to access this data, we can use the Entity class to consolidate the codebase of the program, making the code easier to maintain.

****

## [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 are specific to Mac devices only. Web hosting software can be used through OS X. Hardware and software are expensive. There are better choices for hosting. | Many free and open source distributions to choose from. Most common OS for servers. Large support community. Requires a more involved role in server maintenance. | Offers a dedicated enterprise level server operating system. Quick security patches. Integration with cloud environments on Azure. Higher costs than hosting with Linux, but affordable. | While there are some unique solutions for hosting low traffic websites with Android smartphones, this is not a recommended solution for large scale projects. |
| **Client Side** | Mac runs only on Apple hardware. Apple produces high quality products; however, repairs and upgrades are either expensive or impossible. High costs. Must use Apple App store for native apps. | Many free operating systems to choose from. Dedicated program developers prefer Linux. Open source software with no licensing fees. Requires a higher level of technical skills. | Most common OS for users. Apps can be downloaded from Microsoft store or the web. Requires the most security precautions. Most versatile OS for the average user. | Intuitive designs. Accessible from anywhere with an internet connection. Bluetooth allows for other input devices to connect as well. Web pages must scale to many different screen sizes for a good user experience. |
| **Development Tools** | Proprietary IDE Xcode for development on Mac. Xcode is free. Supports Swift, Apple’s native programming language. High hardware costs.  Other IDE options. | A wide array of IDE’s from lightweight to feature rich and memory expensive. Support for all non-proprietary programming languages. | Only OS that supports Visual Studio. C# is Microsoft’s proprietary programming language. Visual Studio has a free version that should suit needs. | Not recommended for software development. Lacks sophisticated IDE’s and has significant input constraints. |

## 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 recommended operating platform is Azure. Azure Kubernetes Service allows for rapid scaling, deployment to all regions across the world and can be customized to save costs. There is no need for the upfront purchase of expensive server equipment. Since the game is popular on Android, it is hard to say how many will migrate to the new web-based version of the game or how many new players the new platform will attract. The cloud-based solution will give maximum flexibility that is scalable within seconds.
2. **Operating Systems Architectures**: The biggest drawbacks of cloud platforms are that the system is not customizable and the server hosts multiple clients. A dedicated server can be optimized to suit your specific needs and you do not have to worry about potential interference from other clients on the cloud. However, I do not think that these issues would be a problem for “Draw It or Lose It.” The interference from a neighboring client is a security concern, but cloud servers are generally secure and precautions can be taken to protect sensitive data. The customizable part is not a problem either because the game is not computationally intensive so any efficiencies gained through a customizable server would not be worth the time and cost.
3. **Storage Management**: Azure Cosmos DB for users (NoSQL). It is a toss up for an SQL vs. NoSQL database without more information about advanced features of the web app. NoSQL will allow much more flexibility though and is better for real-time applications. There are advantages to SQL for things like global leaderboards and fastest times by picture, however if you do not have a solid understanding of how you want your data to relate then NoSQL is the way to go. Any changes to the data structure in an SQL database are difficult to implement. This will be stored on the cloud as well through rented SSD memory storage. The images can be kept in a similar way just as plain file storage.
4. **Memory Management**: This is the biggest advantage of Azure Kubernetes Service. Set up your nodes in advanced that will automatically scale within seconds to reduce memory use during low traffic and increase during high traffic times. It will cost money for someone to manage the cloud server use, however, these costs exist for paying someone to manage and scale a physical server too. I think the cloud option will be cheaper over time.
5. **Distributed Systems and Networks**: A cloud service is essentially a distributed system. The physical components of the game will exist in many locations across the infrastructure of the cloud. The Kubernetes Service will automatically manage all of this for you. Developers can access everything through a virtual machine that makes everything feel like it is located on the same system. This is all handled through the internet, a WAN that is accessible through the TCP/IP protocols.
6. **Security**: Security is possibly the largest concern with cloud computing. The physical security of the hardware is out of your control. There are things that can be done to protect the data though. Keeping admin logins to the service secure is vital, so ensure that only people who need access to everything have it. Data can be encrypted in house before being sent to the cloud, so this can protect admin logins and the most sensitive company data. User data, however, is vulnerable to attackers. This is not to say that cloud services are unsafe, just less safe than dedicated servers. I would suggest handling payments through a third-party vendor or at least not allowing users to store their credit card information on your accounts. Keep personal data limited, optional, and clearly define its use to the users in order handle business operations in an ethical manner. Also, having a two-factor authentication option that users can switch on or off will allow each user to decide how secure they would like their account to be, since two-factor authentication is substantially harder to attack.