

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 | 07/16/21 | Daniel Giannatsis | Added more design constraints |

## [Executive Summary](#_sbfa50wo7nsh)

The client, The Gaming Room, needs their Android exclusive game application “Draw It or Lose It” to be developed into a web – based distributed game application. We need to design the environment to run the game on a web platform and rewrite the object-oriented application code to work with this environment.

## [Design Constraints](#_2et92p0)

Since the game application is web based, it must be hosted on 1 or more web servers. It must be accessible from a website to be run on all web browsers.

Since the application is distributed and will be played concurrently by multiple teams across multiple browsers, the web servers must coordinate this and the software of the game must account for this.

Based on the requirement that a game will have the ability to have one or more teams involved, the software will need to have a way of making sure each team’s data is unique since games will be run on multiple computers concurrently.

Based on the requirement that each team will have multiple players assigned to it, the software needs to have a Team class that contains multiple Player objects.

Because of the software requirement that game and team names must be unique, the Game class will need to contain a list of Team objects and a way to check if the team name a user enters already exists when adding a Team to the list. Similarly, the Team class needs a method to look up Player names when adding players to the Teams list.

Since only one instance of the game can exist in memory at any given time, the software needs to create unique identifiers for each instance of a game, team, or player. This can be implemented by incrementing and assigning Id numbers for game, team and players every time one of these objects is created.

## [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 provides generic attributes (name and Id) and simple methods such as getters and a toString() that the derived classes Game, Team and Player inherit. Starting with the Player class, we see it inherits from Entity and overrides Entities’ parameterized constructor and toString() method. Team also inherits from Entity, with its own constructor and toString method. The association line from Player to Team with 0…\* and private attribute players tells us Team contains a list of Players , from 0 to many. Team also has a method to add a Player object to the list of players. Next, the Game class also inherits from Entity with its own constructor and toString method. The association line and private list attribute : teams similarly tell us that a Game can contain 0 to many Teams. It also contains a method to add a Team to the teams list. The GameService class, from the association line and the private list attribute: games contains 0 to many Games. It also has private attributes nextGameId, nextPlayerId, and nextTeamId to assign unique identifiers to each object upon creation. It also contains a private, static instance of itself, which is part of the singleton pattern designed to limit the amount of running GameServices to 1. Next it has a private constructor, another piece of the singleton pattern. GetInstance method is how we access the single static GameService instance . We can add games to the games list using the addGame method , and retrieve game information using the 2 getGame methods with respective parameters id and name. We also have methods to get the game count, the next player id, and next team id. The SingletonTester class simply runs a test to make sure that the Singeton design pattern is working correctly. Finally, the ProgramDriver class runs the program, containing the main method. It also uses the SingletonTester class when calling GameService.

****

## [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.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Deployment method on Mac includes using built in Apache and DynDNS Hosting on a mac is free with this method. The downside is it can’t be scaled up very much.  Secure platform for hosting compared to other operating systems. Expensive hardware but the OS doesn’t cost extra money like with Windows machines.  Must pay every time the OS has a new version released. Only Mac can release updates or tweaks to OS. | Deployment method on Linux includes Apache.  Since Linux is a free OS, anyone can configure the OS to fit their needs. This is useful because one can optimize the OS to run better with the hardware and with the web server. Linux has the most flexibility in terms of hardware it can be run on, leading to cost savings. Less OS updates means the server will have more uptime. The security must be configured more compared to Windows and Mac. | Deployment method uses Microsoft IIS which requires at least Windows 10 home ($119)  Must pay every time the OS has a new version released. Only Windows can release update or tweaks to OS. Hardware has more options than Mac but less than Linux.  More updates mean less uptime. | IOS:  There is a method to deploy the web app using IOS but is inconvenient and limited as it requires installation of a Linux shell app ( iSH) and use of a CLI on a smartphone. Python 3 needs to be installed on the iPhone as well. The hosting is also limited in functionality and features compared to other options. Can only run simple websites; not scalable. There are no licensing costs as the apps needed to set up the server are free. Cheap energy costs.  Android:  Deployment is simple to set up due to free hosting apps that don’t require running a shell like with IOS.  It can only run small, basic websites, not scalable. Like IOS, there are no licensing costs, and cheap energy costs due to the hardware. |
| **Client Side** | Use of frameworks that support cross-browser functionality will speed up and simplify the development process, as they manage html, css and other web specific languages. The initial time overhead to learn the framework will pay off.  Major frameworks such as Angular aren’t platform specific. XSS security like Web application firewalls, and certificates can be configured using Angular. Certificates for the client side will protect liability if there’s a data breach, and encryption for the domain, but good ones cost money ($399/year for Digicert) Benefit of digicert is it’s recognized by all browsers.  Time will be needed to do cross-browser testing with tools like LambdaTest (free , 60 mins of parallel testing/month). | Use of frameworks that support cross-browser functionality will speed up and simplify the development process, as they manage html, css and other web specific languages. The initial time overhead to learn the framework will pay off.  Major frameworks such as Angular aren’t platform specific.  XSS security like Web application firewalls, and certificates can be configured using Angular. Certificates for the client side will protect liability if there’s a data breach, and encryption for the domain, but good ones cost money ($399/year for Digicert) Benefit of digicert is it’s recognized by all browsers.  Time will be needed to do cross-browser testing with tools like LambdaTest (free , 60 mins of parallel testing/month). | Use of frameworks that support cross-browser functionality will speed up and simplify the development process, as they manage html, css and other web specific languages. The initial time overhead to learn the framework will pay off.  Major frameworks such as Angular aren’t platform specific.  XSS security like Web application firewalls, and certificates can be configured using Angular. Certificates for the client side will protect liability if there’s a data breach, and encryption for the domain, but good ones cost money. ($399/year for Digicert) Benefit of digicert is it’s recognized by all browsers.  Time will be needed to do cross-browser testing with tools like LambdaTest (free , 60 mins of parallel testing/month). | <IOS>  Real client-side development isn’t feasible on iOS. Even though there are ides, they are designed to be supplemental to a desktop ide and lack support for relevant libraries and frameworks for client-side web apps.  An iOS app needs to be developed. It’ll cost $99/yr to publish on the IOS app store. Instead of building the whole app from scratch, using REST APIs to connect to the web server will save some time. This isn’t an easy task and might take more than one person to complete, as the frameworks that can be used with Swift have a steep learning curve and knowledge is required to use the iOS SDK. There isn’t an easy, cheap way to develop this.  <Android>  Real client-side development isn’t feasible on android. There are IDEs available, but they are focused on developing android apps.  Time will be needed to do cross-browser testing with tools like LambdaTest (must pay for mobile : $99 /month mins of parallel testing/month) to test the web app on mobile platforms. |
| **Platform-Specific lamguages/tools** | Sketch for prototypes.  Frameworks: Angular  Languages: HTML, CSS, PHP, Java  IDEs (all free) : Angular IDE, Atom, Webstorm  CrossBrowserTesting by LambaTest  ($99 /month) | Invision studio for prototypes.  Frameworks: Angular  Languages: HTML, CSS, PHP, Java  IDEs (all free) : Angular IDE, Atom , Webstorm  CrossBrowserTesting by LambaTest  ($99 /month) | Invision studio for prototypes.  Frameworks: Angular  Languages: HTML, CSS, PHP, Java  IDEs (all free) : Angular IDE, Visual Studio, Atom  CrossBrowserTesting by LambaTest  ($99 /month) | Swift  CrossBrowserTesting by LambaTest  ($99 /month) |

## 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 a machine running Windows Server as the operating platform to run the server for the game application to allow Draw it or Lose it to expand to other computing environments. Windows Server has its own benefits over running a user-centric OS like Windows 10 such as support for more memory, more effective usage of the server hardware, and support for many more network connections. This choice is also largely influenced by the fact that machines running Microsoft Server have access to Microsoft Azure, which is a very powerful, flexible, cost-effective hybrid or cloud model to manage the web application.
2. **Operating Systems Architectures**: The Windows server will be using Microsoft Azure to form a hybrid cloud model to host the web application using a REST API for lightweight and fast communication with the client-side web browsers. The REST API will be managed by Postman, which works well with Microsoft Azure. Azure allows developers the option for a service to be run and maintained directly by a team locally or for the service to be managed entirely by the cloud. The architecture is flexible, and services can be added or taken down from to the cloud as the team sees fit. For example, If it would be cheaper and more secure/practical/time efficient to move databases to the cloud, the team can simply decide to do it, and if there’s an issue they can move back to local servers. This hybrid cloud architecture allows for the best choices for each need, whether they be traditional or serverless (cloud-based).
3. **Storage Management**:

Azure Storage Explorer is a tool that’ll be used to manage storage on cloud servers. It makes it easy to add new features and data to the cloud, since Draw it or Lose it will most likely expand with new features or, at the very least, add new images to be used in the game. Since Storage Explorer is a Microsoft tool, it supports Active Directory for fine-tuned access control. The actual database service used will be Azure SQL Database. This is a great choice because it will support scalability as the game expands and has its own optimization features to improve database performance.

1. **Memory Management**:

On the server side, Draw it or Lose it will entail a lot of processes in memory mainly to request player, team and images from storage, and the application code to actually coordinate game instances. Unavoidable network problems from packet loss or corruption will have clients re-requesting data, and therefore adding even more requests and utilizing more RAM. Windows Server has some features and tools that will make sure that memory is being used in the most efficient and safe way possible. To start, the machine running Windows Server has many of the staple memory management features of modern operating systems such as virtual memory and demand paging. Windows Server also has Pagefile if the server’s RAM is completely utilized, pages can be written to the Pagefile which resides on the hard drive. Pagefile isn’t something that’ll be used often, as we want the server to almost never have a RAM shortage, and also because storage is expensive. For on premise servers, Performance Monitor can be used to find bottlenecks in memory so a team can configure memory or change the application’s code to deal with memory better, especially at peak load times. Performance Monitor displays some key metrics to help with this. The first are committed bytes, which shows how many bytes of virtual memory were allocated by different processes. Working set, total shows the virtual memory in active use. Pages Output/sec shows how many pages were written to the Pagefile, which can indicate that paging is the bottleneck. Performance Monitor reports other metrics as well , and the plan is to use an on-premises server to monitor memory management as the game expands so that the application and cloud servers can be configured to be as memory efficient and reliable as possible.

1. **Distributed Systems and Networks**:

As mentioned above, the web app will use a REST API that will allow client HTTP requests to be fulfilled by the servers with simple representations of objects using a format such as JSON. This will allow game instances on all connected client computers to quickly re-request data when network problems like packet loss and data corruption occur. This is important because the game requires a decent amount of information to be loaded into the user’s memory from the client side, and because the network infrastructure connecting each player, each player’s own internet service and network devices like routers will inevitably affect network connectivity and cause problems. Since Draw it or Lose it will potentially scale all over the world , players may be very far from the physical servers which will cause latency issues. One way to solve this is to add and migrate more cloud servers with Azure that will be spread out in different time zones , therefore improving latency for clients since they’ll be, on average , closer to a server. The cloud migration will also help connections at peak usage times with tools like Azure Load Balancer. A downside that should be considered when using an increasingly cloud based infrastructure is that there isn’t much one can do when the cloud service provider (Azure in this case) encounters outages. Therefore, it’s viable to keep at least one physical server. Azure also has a few tools under Azure Network Watcher that make it easy to monitor cloud server traffic remotely at the packet level or using flow logs to understand traffic better and help make decisions about server expansion or network security.

1. **Security**: The first piece of security to consider is user authentication and authorization. Since the operating platform is a Windows server, accounts can be managed locally by Active Directory or as a hybrid model using Azure Active Directory. Using one directory as an authoritative source will increase security by decreasing complexity. Azure AD can be used to implement and enforce important security features like multi-factor authentication, secure password reset methods, and makes it easy to enforce better passwords by using banned password lists and having complexity requirements. Roles can also be managed to work with both local and cloud directories to make it easy for an administrator to create roles and control access to what functions each role has access to. Another measure that should be taken is to protect user accounts on the local server by implementing a firewall. This protects the user data in the event that a cloud-based directory is compromised. Another important feature that must be implemented at some level is encryption. One form of encryption that’ll be used is encryption at rest which will encrypt high-demand data (such as the game images) when written to disk using a key and decrypt using the same key when delivering the data to the client side. The keys will be stored in a separate, protected directory such as Azure Key Vault. This will add another layer of security on top of the proper authentication and authorization schemes.