

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 | 05/22/2021 | Lauren Connors | Initial Draft |
| 1.1 | 06/06/2021 | Lauren Connors | Deployment Target Evaluations |

**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)

Software needs to permit management of multiple teams of multiple players for a single game at a time, ensuring uniqueness across player, team, and game names.

## [Design Constraints](#_2et92p0)

Needs to be noted that with multiple sources of data, consistency in data needs to be handled deliberately so all connected clients are served with the same game state. Advised solution is to have the software connect each client to the same server node on a per-game basis and ensure synchronization primitives are used to prevent any data collisions.

## [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)

Game, Team and Player classes are all derived from the base class Entity. Game has a zero to many relationship with Team which has a zero to many relationship with Player. All Entity objects have an id and name with getter methods, but through obfuscation the id and name fields are constant and cannot be modified in run time. Additionally, all Entity classes implement a custom toString method. The Game Service object is a singleton pattern which ensures any access to it refers to the same instance. This instance has a zero to many relationship with the Game object instances, and tracks the next id value for each entity.



## [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** | OSX is able to run JVM and containers, but is typically not optimized to run as a web server. | Linux is able to run JVM, and many distributions have special flavors for Server deployments; makes a great target for Server application deployment. Container support is often out of the box, and installing kubernetes is relatively simple. | Microsoft offers Windows Server operating system at a premium which includes IIS. However, this competes with proposed Kubernetes solution and complicates scaling options. | Mobile devices can often be bandwidth limited in addition to poor memory, compute and storage resources compared to typical server hardware. Mobile operating systems also do not run any container run times – this would not be advisable. |
| **Client Side** | Most major web browsers run on Mac; no unique development concerns need addressing. Ensure client side application is HTML5 / ECMA2015 or earlier compliant. | Chrome, FireFox, and MS Edge are all available; no unique development concerns need addressing. Ensure client side application is HTML5 / ECMA2015 or earlier compliant. | Most major web browsers run on Mac; no unique development concerns need addressing. Ensure client side application is HTML5 / ECMA2015 or earlier compliant. | Most major web browsers run on Android and iOS; no unique development concerns need addressing. Ensure client side application is HTML5 / ECMA2015 or earlier compliant. |
| **Development Tools** | OSX is capable of running JDK and common IDES (ie: Jet Brains and Eclipse), makes a suitable development platform. | Linux is capable of running JDK and common IDES (ie: Jet Brains and Eclipse), makes a suitable development platform. | Windows is capable of running JDK and common IDES (ie: Jet Brains and Eclipse), makes a suitable development platform. | While some IDES and languages can be used for development on a iOS or Android device, hardware constraints on top of software UX make this an ill advised development platform. |

## 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**: Advise deploying this service in three layers: a database, the web application, and a load balancer. As this is a web application, scalability is a must, and using a tool like Kubernetes with Docker or similar container system would maximize reliability without complicating the code base. See Storage Management and Memory Management for further information.
2. **Operating Systems Architectures**: The server application code is running in JVM, so x86, x64 or ARM are all acceptable architectures which leaves most operating systems as an option. With the suggested Kubernetes / container approach, the host OS could be something like Google’s “Container-Optimized OS” or similar light-weight Linux distribution.
3. **Storage Management**: Advise using a relational database for storage as it has efficient built-in mechanisms to enforce the policy of unique names via table indexes / constraints.
4. **Memory Management**: Current implementation runs a risk of running out of memory for no clear purpose as there is no mechanism to release Game objects from memory, so the operating platform will simply allocate more memory until there is none available. This isn’t a memory “leak” as the objects are still reachable from code, none the less Memory allocations will eventually fail. Advise leveraging database rather than storing all Entity sub-class instances in-memory.
5. **Distributed Systems and Networks**: Adopting the Kubernetes container deployment pattern makes available automated load balancing options and enables automated scaling for web applications. Additionally, some database providers provide auto scaling database clusters (IE: Azure SQL) in the event that application load is too much for a single database node. Leveraging a load balancer greatly simplifies the networking required.
6. **Security**: In regards to security, advise serving application over HTTP on a private network reachable only by the load balancer which will terminate HTTP with a CA-signed SSL certificate to provide the application over HTTPS. This protects clients from MITM attacks, and adds a level of protection between your application servers and the internet where the LB has an opportunity to protect against eg DDOS attacks – all of which is transparent to the application itself. Regarding the proposed database solution, keeping it in a private network with the Application containers and keeping passwords updated and unique can help deter any data leaks.