

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 | 09/19/21 | Francisco Ortega | Entity class added to program. Changes made to Player, Team, and Game classes |

**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 wants to develop a web-based game application that is accessible to users via multiple platforms. Recommendation to create the application using the Vue JS framework on a containerized environment to take advantage of using Docker for quick deployment and management.

## [Design Constraints](#_2et92p0)

The design objective is to develop the game application in a web-based distributed environment. Two technical constraints here are determining the appropriate language and framework to use to develop the application and determining the best storage solution to use since no specific details were given as to how large the library of stock drawings is expected to be. There are a variety of languages and frameworks available that may be used to develop this application (Javascript with React, VueJS or Svelte frameworks; Python with Django, Ruby on Rails). Each of these combinations highlights specific strengths and weaknesses. Detailed expectations regarding expected performance and security are needed to best identify best language and framework to use. Additional details are also needed to determine the amount of storage needed to host the expected image library. Decisions will need to be made regarding where this image library will be hosted, on a local server or in a cloud infrastructure like an S3 AWS bucket.

A third constraint is the project budget. No specific monetary value was given regarding any budget limitations for this project. With the possibility of needing to purchase new equipment and hiring new personnel; it is vital that the exact budget be disclosed prior to beginning the project.

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

The GameService, Game, Team, and Player classes all have association “HAS-A” relationships with multiplicity meaning that an instance of one class can have multiple references to another class. A single GameService can have multiple Game references, a single Game can have multiple Team references, and a single Team can have multiple Player references.

The Game, Team, and Player classes have an “IS-A” relationship as they inherit specific attributes from the parent class Entity. Using inheritance allows for code reusability, in this case being the need to generate and access both an id and name for every game, team, and player created without having to repeat same code over and over again.

****

## [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** | Configuring a web server, accessing the server, and making changes to the server is easily attainable given flexible built-in commands. MacOS terminal application is super responsive and pretty much capable of handling any command. Cost can be expensive. | Configuring a web server, accessing the server, and making changes to the server is easily attainable given built-in commands in a very cost-effective environment. Linux is great at minimizing installation of bloatware keeping OS very lightweight and maximize resources. May need to be familiar with Linux in order to correctly set up environment. | Unlike Linux and MacOS, more server-side software exists for Windows that make it easy to navigate server configuration through GUI. Such applications make it easier for user to configure needed tasks but may hog memory resources. Like Mac, cost to implement can be pretty expensive. | Although many options exist to handle server processes involved with authentication, database, management, and other features for mobile applications, configuring a server manually for an application would most likely be hosted on an environment running Windows, Mac, or Linux. Cost to implement would fall under same scope. |
| **Client Side** | Moderate amount of time and level of expertise is required to be able to configure needed tools to create/run an application. Cost to implement can be an expensive cost for users. | Maximum amount of time and level of expertise is required to configure needed tools to create/run an application. Cost to implement is low. Most likely the most cost-effective option and most consistently updated platform. | Minimal amount of time and level of expertise is required to configure needed resources to create/run web applications. Cost to implement is similar to implementation on MacOS. Does not require a lot of time to understand setup requirements due GUI friendliness. | Developing web application on any OS provides equal flexibility to make necessary enhancements to ensure rendering of application on mobile devices is feasible. Additional framework libraries might be needed but none that require any additional expertise. |
| **Development Tools** | Relevant languages used to develop this type of software would be HTML/CSS/JS and any needed frameworks to support user frontend. Other potential languages that may be used are Java, Python, PHP, and Ruby on Rails.  Tools used to support languages listed include: PyCharm, Visual Studio, Eclipse, Git repository, Kubernetes, and YAML. If using a database, potential options are MongoDB and MySQL. | Relevant languages used to develop this type of software would be HTML/CSS/JS and any needed frameworks to support user frontend. Other potential languages that may be used are Java, Python, PHP, and Ruby on Rails.  Tools used to support languages listed include: PyCharm, Visual Studio, Eclipse, Git repository, Kubernetes, and YAML. If using a database, potential options are MongoDB and MySQL. | Relevant languages used to develop this type of software would be HTML/CSS/JS and any needed frameworks to support user frontend. Other potential languages that may be used are Java, Python, PHP, and Ruby on Rails.  Tools used to support languages listed include: PyCharm, Visual Studio, Eclipse, Git repository, Kubernetes, and YAML. If using a database, potential options are MongoDB and MySQL. | Relevant languages used to develop this type of software would be HTML/CSS/JS and any needed frameworks to support user frontend.  Additional libraries  needed would  include: BootStrap,  Quasar, Flutter,  React Native, VueJS. |

## 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**: My recommendation for operating platform would be Linux which can efficiently run Docker containers managed by Kubernetes. Such infrastructure provides a great environment that can be easily replicated and provides optimal portability and scalability. This type of environment can be set up on Windows and Mac as well, but Linux’s ability to limit un-needed software overhead provides the best platform.
2. **Operating Systems Architectures**: The Linux OS consists of three primary parts: Kernel, System Library and System Utility. The kernel is the core part of Linux that is responsible for all operating system tasks. The kernel interacts directly with the underlying hardware. The system libraries are special functions used to implement operating system functionality. System utility provides the functionalities of the operating system to the user.
3. **Storage Management**: Without knowing how large the image library can become, the recommendation to host images on a cloud service like an Amazon S3 bucket would help ensure that server running the web application doesn’t end up using too much memory serving images.
4. **Memory Management**: For efficiency, Linux manages memory in what it calls pages (4KB Buffers) in order to dynamically accommodate memory needed for process execution and data input. If decision is made to serve images directly from application, paging will help ensure that application can continue to run processes required without bringing application to a complete halt.
5. **Distributed Systems and Networks**: Web application will most likely be accessible cross platform via a web browser so communication between various platforms requires no additional setup. Using containers managed by Kubernetes to setup a server, a client, and database (if needed) allows us take advantage of CI/CD (Continuous Integration / Continuous Delivery) tools that can help ensure that the application has very little to no downtime at any stage. Kubernetes will assist with ensuring all needed containers are communicating accordingly and run self-heal services amongst containers if any issues were to occur. If a container were to crash, Kubernetes can re-deploy needed image from specified Docker image within seconds.
6. **Security**: Web application will most likely be accessible cross platform via the browser. Most browsers have integrated security features that help in protecting users from malware or viruses. Linux, MacOS, and Windows can all run similar browsers like Google Chrome and Firefox. Being an open-source OS, a large community of developers review its code for vulnerabilities or threats on a consistent basis which helps with developing needed security patches to keep platform as secure as possible.