

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 2.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc77528021)

[Table of Contents 2](#_Toc77528022)

[Document Revision History 2](#_Toc77528023)

[Executive Summary 3](#_Toc77528024)

[Design Constraints 3](#_Toc77528025)

[System Architecture View 3](#_Toc77528026)

[Domain Model 4](#_Toc77528027)

[Evaluation 5](#_Toc77528028)

[Recommendations 7](#_Toc77528029)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <07/17/2021> | Vincent Del Vecchio | Added Executive Summary, Design Constraints, and Domain Model description. |
| 2.0 | <08/06/2021> | Vincent Del Vecchio | Completed Evaluation of Operating Platforms regarding server side, client side, and development tools. |
| 3.0 | <8/19/2021> | Vincent Del Vecchio | Completed Recommendations portion of the document. |

## 

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wants to expand upon its current Android app Draw It or Lose It and develop it into a multi-platform web-based game. This document will serve to help facilitate the development of a web-based version of the existing app and streamline development by addressing the client’s software requirements.

## [Design Constraints](#_2et92p0)

**Technical**

*Programming Language - Java*The Program must be written in Java to leverage and reuse existing code from the existing Android Application.

*Web Browser and Operating Support*

The Client request that the program be web-based and support multiple platforms. Great attention and testing must be put forth towards ensuring that the application is supported by the variety of different environments that the program is expected to run in.

## [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 domain model is built upon the foundations of OOP by displaying several classes that represent entities within the game system. The Entity class serves as a base class that servers as the foundation of its subclasses which all share the attributes and methods found within the base class.

The Game, Team, and Player classes all inherit from the Entity class. This is shown throw the open Arrow pointing from each of the three classes to the Entity class.

Each of the four classes on the bottom are in an associative relationship where each class is made up of the class on the right with a zero to many, multiplicative relationship. The Game Service objects serves as a manager for the games that are currently being played. Only one instance of a game may be active at any given time, so the game service uses a singleton pattern to ensure that only one service exist at any time with the ability to create games. The Game service host many games, which each can contain many teams, which in turn can each contain many players.

The Program Driver class is the main point of entry for the application, it uses the SingletonTester class as shown in the model to validate that only one instance of the Game Service exists.

All these classes are contained within the gamingroom package.

****

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Due to licensing cost, Mac is an expensive platform to host a server on.  Contains many software packages, however open-source alternatives are available on other platforms.  Server interface is graphical. | Well equipped and widely used for server hosting with no licensing fees required.  Contains numerous open-source libraries and utilities.  Depending on distribution, interfacing with the system is largely done via command line. | Like Mac, Windows has server hosting capability, but at an expensive licensing cost.  Most widely used operating system allows freedom of choice from a plethora of libraries and utilities.  Server interface is also Graphical. | Lack of power for high end and heavily used servers leaves mobile devices at an extreme disadvantage for server hosting compared to traditional operating systems. |
| **Client Side** | Requires a developer skilled in Swift to develop a native application.  Development can only occur on Apple products which increases cost due to their high price point. | Development time would be high due to supporting a smaller number of file types leading to increased programming overhead. | Developers would most likely need to be skilled with the .Net framework.  Supports all major web browsers. | Special care needs to be taken so that the application delivers an optimal user experience across the varying form factors of mobile devices.  No support for executable files which forces all logic to remain server sided. |

| **Development Tools** | There are a variety of IDEs that can be used to develop the application with most of the coding done in Apples Programming language Swift | Linux also supports multiple IDEs and programming languages that are left up to the choice of the developer. Languages that are not interpreted are favored as they do not limit the scalability or design of the system. | Windows shares the same characteristics as Linux in this regard. Favored IDEs among window developers include Visual Studio and Visual Studio Code. | Development for android requires use of Android Studio and favors Java as its native language.  Apple devices share the same requirements as Mac desktops.  Development can also be done using a third party engine such as Unity which would then allow you to compile the app for both mobile devices without needing to develop separate applications in different languages for both devices. |
| --- | --- | --- | --- | --- |

## 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 using a serverless platform due to it reducing inefficiencies and complexity of the project by transferring maintenance, and ownership costs to company that specializes in offering favorable prices and the ability to easily be scaled based on the growth of the project. Services can be accessed over HTTP and API request allowing it to be accessed by any internet capable device with a web browser. A platform running on Linux OS would be a good fit for the project.

1. **Operating Systems Architectures**:

Serverless platforms build upon the traditional three tier approach by modularizing the middle layer, allowing our app to include several robust features such as business logic functions, content delivery, and security services. Linux OS is also constructed in a modular fashion, separating modules that allow expansion of services from the core components necessary for the OS to function which allows it to be lightweight and occupy less disk space. Linux also offers additional security due to it’s use of system libraries that do not have direct access to the kernel.

1. **Storage Management**:

Linux’s file system reduces effective access time of memory due to it’s use of multi-tiered indexed allocation. Each file in Linux contains a unique identifier, and by storing files on a block close to its respective index, Linux can reduce the seek time of each file. Indexed allocation also helps prevent wasted space in memory due to fragmentation from contiguously allocated memory.

1. **Memory Management**:

Linux helps eliminate size constraints and allows multiple programs to run in parallel due to its use of virtual memory and demand paging. This would allow for more instances of games to be run at once for the application. Linux also employs the use of the Least Recently Used algorithm for handling page swapping which helps keep page thrashing low and maintain good performance for the users.

1. **Distributed Systems and Networks**:

To communicate between various platforms, the app should make use of REST APIs which will allow any internet capable device to be serviced due to REST portable and uniform nature. REST presents different method types which carry out specific functions within our application. Resources that are necessary to supply to the user are easily mapped to the individual functions that serverless architecture makes use of.

Distributed systems help the app maintain connectivity and prevent outages by balancing the load of user request across several different servers and offering security measures that help prevent malicious attacks on our application such as DDoS. Several instances of the Databases are also run simultaneously across a distributed system helping to maintain service should anything happen to the database during system uptime. The redundancy offered by a distributed systems also allows for the continual deployment of new code to our application without any down time by sequentially deploying code to servers always leaving functional servers for the users preventing any downtime.

1. **Security**:

To secure our application, it is important that we make use of the role-based access control that is offered to us through the REST API. By ensuring that players are assigned roles that give players the least privileges necessary to use the application helps prevent them from accessing functions of our application reserved for administrators. Encryption of user information in the databases is also a must. Linux also helps secure our app due to its process independence. Processes have their own virtual space and are prevented from affecting each other. Linux also enforces least privilege automatically and limits the damage that can be done via requiring authentication to use the OS and protecting it’s file directories through it’s use of discretionary access control.