

The Gaming Room

# **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 | 07/15/2022 | Tyler Primas | Filled out all sections of design document up to executive summary, design constraints, and domain model |
| 2.0 | 07/30/2022 | Tyler Primas | Completed evaluation chart for Mac, Windows, Linus, and mobile devices on server side, client side, and development tools |
| 3.0 | 8/12/2022 | Tyler Primas | Completed recommendations and made finishing touches to the entire document |

**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 have an application that will render images from a library of stock drawings and have teams of people guess what is being drawn in a specific time frame. The Gaming Room is unsure of how to set the environment up, but it is based on their Android only app. They want to take their existing app and create a web-based game out of it that can serve multiple platforms.

## [Design Constraints](#_2et92p0)

* The game should be web-based that can serve multiple platforms.
* The game should be based on the Android app Draw It or Lose It.
* A game consists of four rounds of play, each lasting one minute.
* Drawings are rendered at a steady rate and are complete at the 30 second mark.
* If the team does not correctly guess the answer, the remaining teams have 15 seconds to solve.
* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow users to check whether a name is in use when choosing a team name
* Only one instance of the game can exist in memory at any given time which can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [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)

Most classes in the UML diagram stem from the Entity class. The Game, Team, and Player classes all inherit from the parent class of Entity. The GameService class contains everything needed to create games and find the id’s of games, players, and teams. The entity class is responsible for holding common attributes and behaviors. The ProgramDriver class is the class used to run the program and use all of the created classes in unison. The SingletonTester class is intended to make sure the game only runs one instance at a time so it does not use up too much memory.

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [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** | **Advantages:**  Top of the line hardware, software is secure with very little chance of data being hacked, easy accessibility and server configuration  **Disadvantages:**  A Mac is required to develop software on which can be expensive, can be expensive to run servers compared to other options, not nearly as common of an OS as other options | **Advantages:**  Linux servers are free, not often targeted by hackers, not very demanding on hardware, and rarely have security problems  **Disadvantages:**  It can be complex to operate, the update process can be complicated, not all versions end up being supported, and some programs do not work with Linux | **Advantages:**  Beginner friendly and intuitive through a GUI, supports a lot of third-party apps, easy system updates, guaranteed long term support  **Disadvantages:**  High licensing costs that increase with each user, susceptible to security problems, vulnerable to malware, resource intensive | **Advantages:**  Can have better speed and performance and better user experience  **Disadvantages:**  Development costs, Android apps and iOS apps must be developed separately in order to work on all desired platforms |
| **Client Side** | The costs would be relatively low after the initial purchase of a Mac device, and the time it would take to develop code would be about average, not too quick or too slow. The only issue would maybe be having to learn how to operate a Mac but after that the coding would be about the same. The expertise needed would probably be above average just because it is a less common OS. | The costs are probably some of the lowest when it comes to dev tools and time it takes to make a final product. Development is quick and the fact that Linux is open source helps to keep more options available while being cheaper. Linux requires a higher amount of expertise. It has some familiarity to it but many people have not come into much contact with it. | The costs would be high for licensing the products, but cost of develop would be pretty cheap because development can be fast. With Windows being the most common OS, I would say less expertise is needed to have a team fully put together a final product. Most people are familiar with Windows and know what to expect of it. | The costs cost potentially be high with having to code multiple apps. This also would affect the development time, making it take much longer to get apps that can work between iOS and Android. The expertise would also be pretty high with having to learn how to work with multiple operating systems to achieve the same task. |
| **Development Tools** | Objective-C is the primary language for writing software for OS X and iOS, but many coding languages can be used like Swift, JS, Java, C++, and Python. If going with Objective-C, Xcode is the IDE to use. Xcode is free to use but joining Apple’s developer program and uploading to the app store costs $99 per year | The main languages used for developing in Linux are Python, C++, and C. There are many IDEs that can be used for Linux, but from what I found Atom, Visual Studio Code, and Sublime Text were three that were constantly mentioned for developing with. Atom is free and Sublime is $80 so the options are some of the cheapest available. | According to the Microsoft website, the languages of choice are C++, C#, Microsoft Visual Basic, and JavaScript. I prefer either Microsoft Visual Studio or Eclipse when developing on Windows, but that may just be personal preference. There are many options for languages and IDEs because Windows is the most widely used OS. Microsoft Visual studio is free for individuals but larger organizations will pay between $45-$250 a month. | Depending on the mobile device, Swift is a good choice for iOS, Java for Android, and Objective-C can be used for both. Androids’ best IDE option is Android Studio and iOS is Xcode. Like all the other operating systems, there are many options that could be used, but these are the most common and easily accessible. |

## 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 server operating platform that is recommended is Linux. It is the cheapest of the available options, secure, and not resource intensive.
2. **Operating Systems Architectures**: The Linux OS is made of the kernel, the system library, system utilities, hardware, and the shell. The kernel is the core of the OS and is responsible for the actions of the OS. There are memory modules within that interact with the hardware. The system library can access the features of the kernel by using predefined programs. System utilities are programs responsible for tasks directed by the kernel and system libraries. The hardware consists of all the main pieces of the computer like RAM and CPU. The Shell is the interface between the user and the kernel and can provide either a command line interface or a graphical interface (Admin, 2019).

1. **Storage Management**: A serverless approach would be my recommendation for handling storage management. The application will be pulling images from a library that can be held on the cloud and loaded quickly from the chosen server. The cost of a serverless application is less than running a server by yourself and reduces the overall costs of keeping the application running. There are many different options for serverless like AWS Lambda, Azure Functions, and Google Cloud Functions which could all be great options (Bashir, 2019). I would recommend Google Cloud Functions based on Google’s reputation and reliability when it comes to almost everything they do.
2. **Memory Management**: Using direct access for memory management is the best way to run the application. Direct access can read and write quickly without having to be in any particular order, so it would work great with the planned system within the game. The game has to be able to render and display pictures quick enough that everyone playing the game can see them within a specified time limit. The images are loaded from a library and using direct access would be the fastest and most efficient method to utilize the memory of the system.
3. **Distributed Systems and Networks**: In order for the game to be able to communicate between various platforms, using a web browser to access the game would be the most straightforward way to accomplish this. I would recommend using Google Chrome or Mozilla Firefox in order to be able to run the application on various platforms. These browsers can work on Windows, iOS, Linux, and even on mobile devices. From what I have learned and observed throughout my life, Google Chrome generally has the least number of issues with running applications that other browsers sometimes face. The browsers provide easy access to being able to run the application without even having to download anything else other than the webpage.
4. **Security**: Linux is a very secure platform that will protect the application and the user. Linux uses authentication, authorization, and encryption to keep the user and their data safe from any unauthorized access like that from a hacker (Admin, 2019). Linux also have various other security features like firewalls and general virus protection. On top of that, Linux is not often the target of hackers, and it is harder to infect with viruses and worms which helps the security as well.

Works Cited

Admin. “Linux Server Architecture & Its Features.” *Blog*, https://www.snwntechsolution.com/blog/linux-server-architecture-its-features/.

Bashir, Faizan. “What Is Serverless Architecture? What Are Its Pros and Cons?” *FreeCodeCamp.org*, FreeCodeCamp.org, 13 July 2019, https://www.freecodecamp.org/news/what-is-serverless-architecture-what-are-its-pros-and-cons/.