

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc160383373)

[Table of Contents 2](#_Toc160383374)

[Document Revision History 2](#_Toc160383375)

[Executive Summary 3](#_Toc160383376)

[Requirements 3](#_Toc160383377)

[Design Constraints 3](#_Toc160383378)

[System Architecture View 3](#_Toc160383379)

[Domain Model 3](#_Toc160383380)

[Evaluation 4](#_Toc160383381)

[Recommendations 8](#_Toc160383382)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 3.0 | 3/3/2024 | Jhariel Almonte Ventura | The document has been updated from the previous version. Recommendations based on the findings has been completed. |

**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)

Upon reviewing the code and implementing some features, I noticed some issues with how the different classes interact with each other. Currently, Game, Team, and Player classes are child classes of Entity; however, this “relationship” does not reflects with how they interact with the GameService class as instead of handling the objects representing each class (Game, Team, Players), it holds values for both the name and ID of each object.

A solution to this issue is to create data structures to hold Game objects, which contain teams, and each team has players. The structure would work as follows.

* GameService may contain multiple games.
  + Game has multiple teams.
    - Team
      * Players

## Requirements

As “Draw it or Lose It” will rely on rendered images, the organization must ensure that these images are from a trusted source to avoid any copyright-related issues, and to ensure that these images are suitable for the audience the organization is catering to.

## [Design Constraints](#_2et92p0)

A design constrains we must keep in mind is the connection between the applications, the APIs to be more specific, as we would be developing a web-based application.

## [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)

**As stated previously in the summary section, the structure of the application currently works as follows: We have three child classes (Game, Team, Player) that extend the base class (Entity) which contains a name and an ID as unique identifiers. In the other hand, we have the GameService class which brings everything together, although this implementation is quite unreliable when taking scalability into consideration, as the class does not hold a reference to the objects of the child classes, but a value (game name) referencing these objects.**

**"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** | Developers have multiple server-based Mac deployment tools at their disposal, these include build-in tools such as macOS server which allows developers to deploy and host servers using Apache and Nginx. Additional third-party tools are available such as Docker, could services such as AWS and even the competition’s deployment options such as AWS, Azure and Google Cloud. However, despite the available deployment platforms and tools, Mac is not as suitable for large-scare servers as the other platforms, or even cloud-based servers to ensure scalability and performance. Additionally, it is important to keep in mind that macOS server requires a license, which may be the same case for third-party tools. | Linux’s strongest characteristic is how customizable, hence, scalable it can be. Linux-based servers are commonly used to host platform for a large-scale websites and applications as it is know to be able to handle HTTP request quite well, for its database versatility, security features, and its ability to manage the servers through the built-in command line, although this requires extensive knowledge about the OS and its commands. Although the OS itself is open-source and free, including some built-in tools such as Ubuntu, Debian and CentOS, we must consider other factors such as third-party tools which may require licensing or fees such as support, tools and cloud services. | Windows mostly utilizes the Internet Information Services (IIS) tool for server deployment over the .NET framework. This software supports tools such as PHP, and ASP.NET, database management systems such as SQL. Windows also has cloud-hosting capabilities through Azure. However, Windows is not as flexible as other platforms such as Linux when it comes to cost effectiveness as it requires licensing for the tools and platforms. | Mobile devices are not commonly used to host servers due the limited capabilities, however, there are tools available for this practice. However, server development frameworks and programming languages can be used for this purpose. This includes Python, Ruby, Node.js, Java and .NET. As mentioned above, mobile devices are not typically used for server hosting, the utilization of third-party tools may require licensing. |
| **Client Side** | To ensure that the application is compatible with all web browsers, we must emphasize scalability and compatibility through each browser’s guidelines, structures, and cross-platform tools such as Electron and React. This also entails that the application’s UI is compatible with these browsers. | The following factors need to be considered while providing support for Linux clients. Cross-platform compatibility, design constrains, accessibility. As Linux is known to be very flexible in terms of customization, it is required to have extensive knowledge about the platform, which may reflect as extended development time as testing and debugging may be challenging, although this would depend on the developer’s knowledge. | In comparison to Linux, Windows is known to have more restrictions and constrains in terms of cost effectiveness as licenses are required, and the OS is not as customizable as Linux. To ensure cross-platform compatibility for Windows-based applications, it is crucial to ensure web browsers updates are compatible with the Windows version. This means continuous updates and testing to ensure a responsive, cross-browser compatibility. | In the case of mobile devices, these are limited by the device’s capability individually speaking. However, as the previous platforms, the crucial factor to ensure completability across al browsers, is cross-platform frameworks to achieve said compatibility, in addition to accessibility, performance optimization, and constrain-based UI design. |
| **Development Tools** | Multiple development tools are available for Mac-based applications such standard programming languages and frameworks including JavaScript, HTML and CSS, react, Node.js, Ruby, Angular, among many others all of which can be used on Visual Studio Code or their build it IDE Xcode. | There is a variety of tools and programming languages at developer’s disposition such as frameworks, IDEs, text-editors, libraries and programming languages. These include Atom, Visual Studio Code, and PyCharm for IDEs which host the following programming languages and frameworks: Angular, Node.js, JavaScript, HTML, CSS, React, etc. | Windows has built-in tools for development, deployment and database management, although these may require licensing. Microsoft Azure and IIS for server application deployment and hosting. Additionally, there are database management systems such as MySQL. | As the previous platforms, mobile devices rely on the standard HTML, CSS and JavaScript for these software. Platform specific tools like Mobile Analytics allow developers to track data about app usage, and the standard App Store and Google Play for application distribution for the respective platforms. Additionally, testing tools or simulators like Xcode and Android Studio. |

## 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**: It is needless to say how big of a population the mobile device user is. Therefore, from this point on we can break down the potential audiences based on the overall global population of users, which Android makes up roughly for a 70% and iOS for about 30%. As the goal is to expand on the audience, I would highly recommend you to tackle this wide market of potential users, and utilize the listed tools for the respective platforms (Android and iOS).
2. **Operating Systems Architectures**: Both Android and iOS have multiple tools at our disposal to create appealing applications for our potential users. This includes tools supporting aspects such as UI, Server, Database and project management tools. This downside to aiming for both Apple and Android users is that these platforms use different structures, therefore programming languages and guidelines based on the devices and their capabilities.

1. **Storage Management**: Regarding storage, both Apple and Google implement a subscription based service for additional storage in their Cloud-base storage system. Apple charges $1 a month for an additional 50GBs exclusively for iOS, MacOS and Windows. Whereas Google charges 2$ a month for 100GBs and provides cross-platform support.
2. **Memory Management**: Both iOS and Android devices have different amounts of RAM (Randomly Allocated Memory). This type of memory is required to allow both the OS and applications to run on the device, and essentially, the more RAM a device has, the more actions it can perform and applications It can run. Apple devices generally have less RAM than their Android counterpart, however, this does not means that one is better than the other, as the RAM requirement would be determined by the devices programming language and processor capabilities to compile the code for both the OS and applications, as these applications require different RAM values based on the OS.
3. **Distributed Systems and Networks**: When Java code is compiled, which is the programming language used to develop applications for Android devices, this code is not reliant on the device’s processor to run, in fact it is converted into “Bytecode”, this means that the program is translated into binary codes which can be read by any processor, allowing for cross-platform compatibility. Apple in the other hand, relies on the device’s processor to compile the code
4. **Security**: Security is a factor of software development that no one can afford to compromise. This also applies to tech behemoths such as Google and Apple. In the case of Android, these applications are more vulnerable to attacks as it is an open source OS, this means that anyone has access to it, compromising the security of both the OS and the applications build upon it. As per iOS in the other hand, every single application available in the App store is monitored by Apple. Additionally, the source code is closed and owned by Apple, which means that these applications are not permitted to access iOS main code.