

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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)4

[**Recommendations**](#_m8aleynsvzvc)8

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 3/20/2022 | Chris Wills | Initial design and summary. |
| 1.1 | 4/3/2022 | Chris Wills | Updated and additions made to Evaluation section for licensing cost for server deployment, client side compatibility requirements, and IDEs available for their respective costs. |
| 1.2 | 4/17/2022 | Chris Wills | Revised Recommendations section after gaining more understanding of the aspects included in said section. |

**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)

Gaming Room wants to create a web-based game that can be used across multiple platforms. Game name is "Draw It or Lose It" and it's available on Android currently. The objective of the game is to have teams contain a few people who go four rounds per minute. A team guesses a picture every time it is taken from the library. All opposition party members receive a response within 15 seconds if not answered.

## [Design Constraints](#_2et92p0)

Several groups must be involved.

Names of games and teams must be unique.

There can only be one game event at a time.

Most platforms should be compatible with it.

Users must properly authenticate in order to play the game.

Security measures must be in place to secure user and server data.

## [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)

Entities create relationships between games, teams, and player classes. This means that they all inherit or get information from Entity. Therefore, each class will share public references such as "name" and "id". This makes the entity a superclass. When we look at their relationships, we see that teams and players are "has a" type. While Game has a team, GameService has games. When viewing the diagram, GameService has a reference of Games, Games has a reference of Team, and Team has a reference of player.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Various terminal commands to access the server, configure it, or make changes.  Characteristics It is popular for hosting websites  Advantages It can be upgraded, has many options for different web hosting requirements  Disadvantages It is less popular for hosting websites.  Mac does offer server-based deployment.  While MacOS had a standalone installation available for server licenses, they currently offer is as an add on package that are installed on top of existing MacOS installations. | Similar compared to the Mac plus it is more cost-effective  Characteristics Secured, most preferred.  Advantages Security flaws are caught before they become an issue, it is a favored hosting platform  Disadvantages It is hard to find applications that work with Macs Disadvantages It is more difficult to find applications that work with Linux.  Linux offers server-based deployment as well.  Linux, being open-sourced, can be implemented for free on the server side. | Software available is more than that of other platforms.  Characteristics It is dominant over its rivals. Closed platform.  Advantages It is faster, less prone to bugs, and more comfortable to use.  Disadvantages It is vulnerable to viruses and has poor tech support.  Windows offers server-based deployment.  License price for Windows 2022 Standard for on site server has an MSRP of $1069 through resellers. They offer cloud server options starting at $0.10 an hour for their Standard Azure. | Characteristics More popular; it is highly portable.  Advantages Have broad reach, better compatibility; it is cost-effective  Disadvantages It is only compatible with a limited number of smart mobile devices Poor security.  Server-based deployment is available here as well.  Since mobile apps don’t have a specific server OS, they often use existing platforms like Azure, AWS, or IBM. If a universally used language is implemented, we should be able to use desktop class server to host the content. |
| **Client Side** | The time and expertise required are moderate. The cost is comparable to Windows.  While using the most up to date version of HTML to deliver the experience, we want to ensure the most compatible version is used to maximize the available target audience. | The highest level of expertise and time is required. The least amount of money is needed.  HTML experience is still applicable here. | It requires little expertise and time, and it is similar in price to Mac.  HTML experience is applicable here. | The ability to view updates at any point provides flexibility to clients and even developers. Slightly more difficult to implement compared to other devices.  HTML experience is applicable here, especially. |
| **Development Tools** | For languages running on macs, Swift is the most popular option. Xcode is available as an IDE for MacOS. Notepad++ is a nice tool to mix in as well. Although Macs are capable of running all languages. In addition HTML/CSS/JavaScript libraries support general-purpose languages and libraries for the frontend. Among these are Java, Python, PHP, and Ruby.  Combining with iOS may reduce the need for time expansion, but if using the in-system language Swift, we may need a separate team compared to Windows and Linux.  If using Xcode, it is available to download for free. | The Linux operating system can be used with Visual Studio, Eclipse, and notepad++. You can use similar libraries found in Mac such as HTML/CSS/JavaScript. You can also use Java, Python, PHP, and Ruby.  Windows and Linux have some shared IDE’s and languages available, so the need for a different time may be reduced.  Eclipse for use with Java or C/C++ is free and open source. | Easy to use but able to perform the same tasks as Linux. So visual studio, eclipse to name a few of the many IDEs. You can generally find the same library and language usage found in Linux and Mac, with the addition of C and its variants.  If using Visual Studio, a professional subscription is available for $45 a month. | Android and Swift can be used to create countless apps. Both languages and software can be run on all three machines. Due to the advancement found in modern mobile devices, you can generally have the same library and language support that you’d find in desktop class OS.  iOS apps can use Swift as well, so a team can cover both MacOS and iOS. If the team is familiar with Java, then we can have continued Android development set.  Eclipse for Java and Xcode for Swift is free to download. |

## 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**: While hosting on a Linux based server would be the least expensive route of all the server options, my recommendation would be using a server based on Windows due to its ubiquitous nature and widespread use among users.
2. **Operating Systems Architectures**: The more common architecture found when it comes to the Windows platform is the X86 platform. X86 provides much more power and utility, and is generally utilized in instances where high performance and stability matters. ARM, while on the rise notably from Apple’s implementation of their M1 chips, are less complex, more power efficient, but at times more difficult to code for without the use of software translation.
3. **Storage Management**: While HDDs will provide the most amount of space for storing our data, I would recommend using SSDs for our storage as it will reduce the need for paging in the chance that our RAM allotment is exceeded. This will increase cost compared to HDDs, but SATA SSDs have come down in price significantly over the years. NVME drives have started to come down as well, but the price for incredibly large NVME storage may still be prohibitive.
4. **Memory Management:** In terms of cache available for the fastest memory a CPU can use, it will depend on the budget and performance of the X86 processor(s) we select for the server. Higher end chips generally will have more cache available compared to the entry level CPUs. RAM is one of the easier components to upgrade down the line, but it’s important to gauge the public’s desire for the game so we can implement a good amount of RAM at the very start to provide the best possible user experience.
5. **Distributed Systems and Networks**: Since operating systems differ, it is vitally important to use a language and database that has wide support across devices. With the game being played by the user through a web browser, using languages such as HTML and JavaScript would be highly recommended, as most browsers nowadays will support. JavaScript runs directly within a web page, so for the sake of simplicity for the user this would be the best course to go.
6. **Security**: Security is incredibly important, especially when handling user data. Implementing an authentication system, as we have learned when practicing using the REST API. Giving the option to users for MFA/2FA authentication will greatly reduce the risk of unauthorized intrusions. It would be wise to ensure that the data traveling through the network is encrypted to keep prying eyes from being able to determine what the data entails. On the server side, while Windows is one of the most targeted systems due to its widespread use around the world, we can implement stricter authorization for admins and encrypt sensitive data being stored in our storage devices. Depending on how the server is hosted, many hosting companies also offer enhanced protections from particular attacks, such as DDoS attacks.