

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/19/23 | Dominic Caulfield Duverger | Draft |

**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)

Creative Technology Solutions (CTS) was requested to develop a web-based version of a game that serves multiple platforms by The Gaming Room. This game is Android based and is named Draw It or Lose It. The game has multiple teams with multiple players per team, and each game, team, and player should only occur a single time. An iterator will prevent teams and team members from conflicting and a singleton creation pattern is utilized.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

The business requires a web-based version of a pre-existing game based in Android. It requires multiple teams and multiple players per team.

## [Design Constraints](#_2et92p0)

The existing deployment of the game Draw It or Lose It is Android-based. We have been requested to extend the game to be web-based, so it needs to be deployed on the web as well. Java was chosen as it is a native language to Android, and will help with development and deployment. Only one instance of the a game can exist, each team should support multiple players, as well as game and team names should be unique to for verification.

## [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 ProgramDriver utilizes the SingletonTester class to verify one instance of a game is being played at a time. Entity is the superclass, and holds the common variables for Game, Team, and Player which are all subclasses. GameService has a reference to Game, Team does as well. Player has a reference to the Team class.

**"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** | Mac’s server environment is stable and reliable. It is well suited for web hosting as it is Unix based. It would be more expensive than the other options, and either the client would need to purchase a server or pay for a host. The lack of hosting options is a drawback | Linux provides a efficient and powerful platform for servers. It is an inexpensive option as hardware isn’t limited and hosts are widely available. It is popular and easily transitioned from/to Linux. It does require a deeper understanding to work in. | Windows is a well understood and a good middle ground. It is user friendly as it is GUI based. It is pretty popular and widely used, and costs land in between the other options and can vary significantly. Hosts are also widely available. It is less efficient in terms of performance in most case. | Mobile devices are widely available, but couldn’t really host them. To host, you would need to use separate services, likely cloud based. The advantages are lacking, as cloud based hosting services can be used on any platform and it is not an advantage specific to mobile devices. |
| **Client Side** | Clients using Mac to develop will need a Mac computer and will need to use the few tools available in Mac. As this is a web-based product, development wouldn’t be limited to any Mac-specific languages, but would be limited to the tools Mac provides. Cost, time, and expertise requirements aren’t very high. | Linux development isn’t as limited in tools as Mac, but requires a higher level of expertise as an adjustment period to get used to the OS. Cost and time are both relatively low, as it is similar to all the other options. | Windows, being widely used has the most tools available, and is well known. Most people are accustomed to windows as an OS. Cost and time are both relatively low, as well as the expertise required to work in windows. | Android is java based and Android Studio is the most highly utilized to develop. iOS utilizes Xcode to develop in. Both are relatively inexpensive but developing on a mobile device is not very common, while developing for a mobile device is one of the most common things to be done. It would take a fair amount of extra time, but be relatively inexpensive and would be similar in expertise required for the other platforms. |
| **Development Tools** | Xcode is the IDE most developers use while developing in Mac, and utilizes Objective-C and SWIFT but can be used for other languages, such as JavaScript, HTML, and CSS | Linux development can use most languages such as HTML, CSS, JavaScript, PHP and there are many different options for IDEs. | Windows development can use most languages, the same most common ones for web development as Linux, and has the largest variety in IDEs. | As developing on mobile devices is uncommon, not many options exist. There are various IDEs, and a wide array of languages. |

## 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**: Utilizing Linux would be the most effective solution for hosting the service. It is efficient and powerful and allows for a cost-effective method to host servers. Linux also provides good security. To keep it cost effective, purchasing hardware and self-hosting the servers, or by utilizing a cloud-based provider. Linux There are no licensing costs involved with most Linux distributions, and hardware is easy to access, as well as there are plenty of cloud-based server providers, which are also not overly expensive and allow for more scalability. Converting over to a cloud-based service would also not be too difficult in the future.
2. **Operating Systems Architectures**: Linux’s architecture is efficient and simplistic, and offers a architecture with few vulnerabilities. Linux is a monolithic architecture and has several primary components. The lowest level is the hardware layer, which manages the hardware components and includes the drivers, the kernel functions, cpu control, memory management, and I/O operations. The kernel would be the second, and main component of the Linux architecture. It controls the other hardware components, visualizes the hardware resources, and provides the processes with resources. It includes the process scheduler, the memory management unit, and the virtual file system. The next layer is the shell, where commands can be entered by the user or by the next layer, the system utility. This layer would enter automatic “commands” to complete specific core tasks.
3. **Storage Management**: Depending on the number of users, and preferred methods, hardware could be purchased and data could be stored locally, on a local server, where there would be no extra fees, just the cost of the hardware. Bulk storage can be relatively inexpensive, and fast storage isn’t necessary for the current application. Faster storage, such as solid state drives (SSDs) would likely be used as a buffer, while Hard Disk Drives (HDDs) would be utilized for bulk storage, there are also other alternatives, but they are more proprietary, meaning less repairable and more expensive. Backup drives would also be needed so data wouldn’t be lost in the event of a failure or corruption. This could also be done via a cloud-based service, while it may cost more as there are monthly costs, it is more scalable, and the service provider would need to manage the hardware side of it. This would also prevent down-time more effectively as 100% uptime is very difficult to achieve and can be costly when self-hosting. There are many different reasons for downtime, such as system updates, pushing updates to the game, and various other things.
4. **Memory Management**: To manage memory, the software would need to be optimized to allow for efficient usage, as if it is not it can cause memory leaks and further costs. The benefit to Linux is that it utilizes virtual memory, which combines both RAM and disk space. This method has been proven to be more secure than Windows, for example, as it uses dynamic memory linking, which while efficient is vulnerable to malware. These issues would still be existent even using a cloud-based provider.
5. **Distributed Systems and Networks**: Utilizing a cloud-based service would be the best option to avoid downtime. As the provider of said service relies on uptime, they have to keep the services up as close to 100% uptime as possible. They have backups set in place and need to provide sufficient connectivity to all the clients. The servers can be hosted in several locations allowing for both better connectivity and more backups. The hosting location can allow a user to connect to a closer server to them, allowing better connectivity. Backups of the data can be in several locations as well. If it is self-hosted, these backups wouldn’t exist, or the infrastructure would need to be created to provide good uptime and connectivity.
6. **Security**: Linux is a good system for security, and cloud-based services are also good for security. Linux is good as it is simplistic and has many people developing security measures for it as it is established as a commonly used server host. If there is a vulnerability, it is more out in the open than other operating systems, such as windows, whereas stated before, there are more major vulnerabilities. While these vulnerabilities are known and patched, there are more people looking to abuse these vulnerabilities. Applications that may contain a malware have a harder time getting deeper into the system when it comes to Linux. The data “The Gaming Room” stores may contain personal information regarding the users, while the users likely won’t experience any vulnerabilities caused by the game itself. There are also a vast number of expansions for security services, allowing for more advanced firewalls and controls. Cloud-based providers need to provide good security as they store data for many different businesses where the data stored may be even more personal, requiring the best security they can get. There is a downside to the cloud-based when it comes to security, which is that there are more people with access to the data.