

“Draw It or Lose It”

# **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 | 11/12/2021 | Siriwat Vongtip | Update to sample template provided. |
| 1.1 | 11/26/2021 | Siriwat Vongtip | Revisions to week 5. |
| 1.2 | 12/9/2021 | Siriwat Vongtip | Revision in support of Learning Module 7. |

**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 is seeking to create a game loosely based on the 1980’s television game “Win, Lose or Draw”. The application will render images from a large library of stock drawings and will consist of four rands lasting one minute each. Drawings are rendered at a steady rate and are completed at 30 seconds, if the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess within 15 seconds.

## [Design Constraints](#_2et92p0)

The game will need to have the ability to have one or more teams involved as well as teams with multiple players. 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.

## [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)

Game, Team, and Player are all child classes of the parent class Entity. The classes all receive the attributes from entity. Game service is an association of game and the three other classes. The ProgramDriver class is an association of Singleton tester which is just a class that tests the program. The driver is the one class that runs that entire app, as it creates instances of the objects of the other classes to play. The 0…\* between the classes can be zero to many games. The game service object manages the games that are played, only one instance of the game can be played at a time. The singleton pattern helps us accomplish this task. It only allows one instance of the game to be run. The service can still be utilized by other parts of the app.

**"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** | MacOS and Linux share the same “operating system ancestor” so by developing on a mac you will benefit from Mac’s management tool “Homebrew”  In order to get an application submitted, apple has created a guide on how to distribute it worldwide or by doing it ad-hoc, rules for worldwide distribution are found here <https://developer.apple.com/app-store/review/guidelines/> As they cover everything from parental restrictions to not stealing other developer’s work and punishment for such an infraction. | As an open-source rich OS, we find that the vast number of choices can be incredibly large. If given the proficiency and designated tools this could prove to be an efficient option.  Linux is a great choice when utilizing PHP, Perl, Python, Ruby, or MySQL. Linux still rein true as they offer numerous open-source applications for hosting, including ready to use homepage solutions, applications, and content management systems. | One of the most widely used OS among the federal government as well as many municipal offices you’d be hard pressed to find someone that hasn’t touched a windows OS.  Windows utilizes Windows Deployment services and allow for deployment of Windows operating systems over the network.  Windows has licensing cost per user and supports most programs and has long term support for all versions. | Due to the nature of mobile phones most of the heavy lifting will need to be done server side due to the limitations of such small devices. This would also possibly require server-side software to help ensure functionality.  Update and maintenance efforts are multiplied, and not only do they have to be updated on the regular, but they will need to go through a vetting process each time.  Marketing pressure would also drive-up costs to get the application to reach the intended audiences. Not only would you need a URL for your apple app, but you would also need one for your google play app. |
| **Client Side** | Mac can make the client side incredibly intuitive. With the constraints of ensure products pass the App store test of quality control.  Utilizing frameworks like bootstrap and foundation will help making cross compatibility easier. | Dependent on the codec being utilized by the application these programs are optimized for both low bandwidth consumption as well as high bandwidth connections.  Looking for the correct parameters of Linux devices provides ease to the developer to how they would like to develop their application. Most people that use Linux use it for the purpose of non-binding licensing agreements. Staying within the realm of open-source software. | The wide distribution of the Windows OS also come with the perks of supporting most if not all major web browsers and cross platform software.  Users can access many model-driven apps with Microsoft Edge, Chrome, Firefox, and safari. Internet Explorer and Microsoft Edge Legacy support has ended.  Utilizing a cloud-based platform allows you to perform cross browser testing on browsers installed on real devices. | As a mobile device you are faced with low processing power. All devices will have to have heavy lifters on the Server side for these applications to produce similar results to their computer counterparts.  Identifying the needs of the customer’s requirements is also incredibly important. One of the biggest downfalls is having difficulty with utilizing some API methods on a mobile platform or anything that essentially requires the use of Adobe flash. |
| **Development Tools** | X-code is an incredibly powerful developer tool that can run programs with ease. For lighter testing and quick patching, visual studio code has also presented itself as a usable program for the OS.  OSX’s preferred language is Java, this is due to in part that Java is the general-purpose language all in one language. Cross compatibility with different platforms makes this quite the choice.  Directly from the JAVA website " The new license permits certain uses, such as personal use and development use, at no cost -- but other uses authorized under prior Oracle Java licenses may no longer be available.” | Sea monkey Composer is one of the easiest website builders which works well with Mozilla. Visual Studio code is also used as an option for this OS.  Having multiple developers with the team is always the preferred method when launching any project, but you will always be restricted to the constraint that holds as true as time. Monetary constraints. Linux has no licensing requirement although they have costs for distribution. | C# is the dominant language here. Other languages are supported, and the widespread usage of this OS will help with this project.  As mentioned previously Windows does require a license per user. A development team is always advised as the scope of certain applications can be overwhelming for one person to handle. | iOS and Android prefer two different programming languages. To support both these mobile platforms, developers will need to have proficiency with both languages. Or team that can be dedicated to work on each language. Of course, you could go the easy route and utilize Java, which was initially meant for android development, but has been now crowned as an all-around use programming language. |

## 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 suggestion is to employ Linux as the preferred operating system as it has the widest variety to help it creatively become what it needs to become as you arrive at the final product. Once reaching that point and by utilizing JAVA you can easily deploy this to other OS’s. Using this OS is the easiest way to avoid malware. The only way system settings and configuration settings can be altered is if the user is logged in as the root user and most users do not log in as the root. With the nature of an open-source OS Linux is able to be viewed by a large number of developers.
2. **Operating Systems Architectures**: The Linux OS operates only on core/bare bones components making this a lightweight OS to navigate through. Linux is also able to call on different functions through the numerous standard libraries that are available to the OS. The system is also very stable and not prone to crashes. The uptime for Linux servers is also very high and the availability is around 99.9%.
3. **Storage Management**: As many things currently, cloud-based platforms are king when it come to protection, management, and user accessibility. Though with a cost per month, the cloud will allow us to focus on the program and not worry about facilitating local storage on a server. Utilizing Linux utilizes ext4 which has been optimized to store more for less space. We also have the added benefit of the storage defragging itself on the go which ultimately optimizes speed of the system.
4. **Memory Management**: Linux utilizes partial memory execution by utilizing virtual memory and demand paging. This reduces physical memory size constraints and allows multiple programs to run in parallel. Partial memory also loads the app faster, which improves overall performance.
5. **Distributed Systems and Networks**: As the application scales, it will need to run across multiple servers to be able to handle client demand. Having a solid data loss prevention plan is also key to ensure we maintain the customer base in the event of a disaster. In a distributed system, the database can be run across multiple machines and must be continuously synched to appear as a single instance.
6. **Security**: Two factor authentication seems to be the norm in the current trend. Having the program send a code to an app on your phone as well as having some sort of smart card that you would insert into the computer, this further increases the security of the program. Use of the lightweight Linux system would enable users to utilize only the apps they need to log in and authenticate on whatever servers they need to access to play the game.