

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 | 02/20/2022 | Tyler Hollingsworth | -Added a summary of the projects’ tasks and goals as per the client’s request  -Added possible technical and business constraints for the project  -Gives insight into the UML diagram characteristics and relationships  -added an evaluation of operation platforms using ('+' as an advantage and '-' for a disadvantage): |

**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 client Creative Technology Solutions is looking to expand their android app game called draw it or lose it to a web-based version that serves multiple platforms. This game consists of a four round game and allows one team to try guessing what an image is as it is rendered into perspective over a 30 second interval. If the team does not guess correctly within a one-minute time frame, then the opposing team has 15 seconds to try and guess the image. The construct of this game consists of having one single game instance in memory at any given time that consists of one or more teams. Each team will be made up of multiple players and both the team and player identifiers must be unique. To accomplish this task different platforms must be acknowledged separately due to each platform having different benefits and constraints that could help or hinder game development. To be able to do this we have to create unique identifiers for each game, team, and player instance. We must also implement a design that can verify these unique identifiers upon creation by comparing the identifier to all existing identifiers.

## [Design Constraints](#_2et92p0)

Technical

-creating the game to be run on multiple platforms

-Game, Team, and Player identifiers must be validated for uniqueness

-One instance of the game can exist at a time

Business

-Multiple teams needed to develop different platform aspects of the game

-Time frame to develop multiple versions of the same game could be varying

-offering an agreeable budget for the client

The technical constraints of the development lifecycle are that the game will need to be adjusted so that it can run on multiple platforms requiring experienced programmers in each of these platforms. Correctly creating unique identifiers and validation methods for those identifiers need to be done in a way that is not memory or time consuming to the overall program. The program must also create and validate that one instance of a game exists at a given time which may require many checks that can add to the constraints of memory and time management. The business aspects of these constraints look more toward the financial development of the program. To create a multi-platform program there would need to be multiple teams, each of which specialize in a different platform that can both tie up dedicated programmers and create added costs for the use of those teams. Lastly, setting up and agreeing upon a budget with these constraints in mind may be more costly than originally expected.

## [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 unified modeling language (UML) diagram below shows the classes and their relationships in the gaming room application. These classes show both the private and public functions and variables within the application as well. This diagram shows that the Game, Team, and Player classes are all inheriting from the Entity class meaning that they inherit Entitys’ attributes and methods. It also shows that the ProgramDriver class has an association with SingletonTester by the ProgramDriver class using the SingletonTester class. The diagram shows a relationship between the GameService class having a zero to many relationship with the Game class as well. This zero to many relationship shows up two other times between the Game class to the Team class and between the Team class to the Player class. The OOP principle of encapsulation is shown in the program by setting some attributes and methods as private so that users or other classes cannot directly access them, but the program can. For example, the game, team, and player lists cannot be accessed by a user, but the program can work with these lists to alter and add new instances or get a count of the current instances within the lists by using public methods that can access and alter them. We can also see polymorphism through the Entity class with the Game, Team, and Player class taking from Entity but using their own iteration. For example, the toString function is a method within the Entity super class but each child class (aka Game, Team, and Player) have their own iteration of the method by using @overide.

**"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** | + Unix OS + Open-Source servers + Secure + User and file access controls + Support LDAP and ADP - Docker support virtual only - Difficult to implement server-only environment | + Unix OS + Open-source servers + Secure + User and file access controls + Support LDAP and ADP + Docker support - Need to decide on vendor support; Redhat or Ubuntu + Large online/cloud support, the internet runs on Linux | + Secure + User and file access controls + Support ADP - Need to use Server version - Limited LDAP - Poor open-source server support | + iOS developer tools + iOS multi-device support (iPhone, iPad, watch, TV) + Android developer tools - Android poor multi-device support |
| **Client Side** | + Support Safari browser + Firefox + Chrome - No IE - Small install of client desktops | + Support Firefox + Chrome - No IE - No Safari - Small install of client desktops | + Support Large installed base of desktop clients + Firefox + Chrome + IE - no Safari | + iOS: Firefox, Chrome, Safari + Android: Firefox, Chrome |
| **Development Tools** | + iOS native and Android development supported - Small pool of developers + High-productivity environment - Difficult /impossible to test IE browsers - Few IDEs, but high quality + Large language support including Swift and objective-C + Java support + Unix shell scripting | - Android dev support only: native + Large developer pool + New technology first platform + Largest set of IDEs with quality but not always support + Large language support including Swift and objective-C + .net support + Java support + Unix shell scripting + PowerShell | + Large set of IDEs with support and quality - Android dev support only: native + Large language support including C# + .net support + PowerShell + Java support | You do not really develop on a mobile platform. |

**1**

## 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**: I would recommend Linux as the operating platform to expand the Draw It or Lose It game to other computing environments.
2. **Operating Systems Architectures**: The Linux operating system consists of the layer’s hardware, kernel, shell, and applications all broken down into separate modules that interact with the module lower than itself all the way down to the hardware. This hardware layer of the system consists of the peripheral devices such as the RAM, HDD, and CPU. The kernel layer is responsible for all the major activities in the operating system and hides the low-level details from the other modules. The shell layer is an interface between the user and kernel such as command line interface or graphical user interface and executes those user commands in the kernel. The application layer consists of the applications installed on the device and is considered the user level.
3. **Storage Management**: Linux uses disk partitions to separate the storage drive into smaller blocks using either MBR (master boot record) or GPT (GUID partition table). The Linux system then formats that information into a file system so that file-operations can be used on those files. Linux supports a wide variety of file systems so there are multiple options to be chosen from. The Linux system also uses a different style of directories than other platforms for example it is case sensitive.
4. **Memory Management**: Linux uses virtual memory and demand paging to manage its memory usage strategically even allowing page caching. This platform can allocate memory for the kernel and the user space as well as map files to an address space.
5. **Distributed Systems and Networks**: Linux can use either Ubuntu or RedHat for support to establish a distributed system and network. RedHat for example is an open-source software company that uses a hybrid cloud approach to bridge diverse cloud environments.
6. **Security**: Linux is considered the most secure operating platform available and is used extensively. This is mainly due to Linux being open-source software and has had many years for programmers and developers to search for and correct bugs. Linux uses user ID’s and passwords to verify users and each of their access rights. It also uses firewalls, secure boot firmware, among many other security measures built into Linux.

**References**

Silberschatz, A., Galvin, P. B., & Gagne, G. (2009). *Operating System Concepts, 8th edition*. O'Reilly Online Learning. Retrieved February 12, 2022, from https://learning.oreilly.com/library/view/operating-system-concepts/9780470128725/silb\_9780470128725\_oeb\_c10\_r1.html#h3