

Draw It or Lose It

# **CS 230 Project Software Design Document**

Version 3.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_heading=h.gjdgxs)1

[**Table of Contents**](#_heading=h.30j0zll)2

[**Document Revision History**](#_heading=h.3znysh7)2

[**Executive Summary**](#_heading=h.2et92p0)3

[**Design Constraints**](#_heading=h.tyjcwt)3

[**System Architecture View**](#_heading=h.3dy6vkm)3

[**Domain Model**](#_heading=h.1t3h5sf)3-4

[**Evaluation**](#_heading=h.2s8eyo1)4-5

[**Recommendations**](#_heading=h.3rdcrjn)6

## [Document Revision History](#_heading=h.3znysh7)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 9/18/2022 | Avery Cross | Completed paragraph outlines for executive summary, requirements, design constraints, domain model. |
| 2.0 | 10/2/2022 | Avery Cross | Added additional information based on feedback and worked on evaluations. |
| 3.0 | 10/16/2022 | Avery Cross | Continued to work based on feedback and completed recommendations. |

## [Executive Summary](#_heading=h.2et92p0)

Creative Technology Solutions, known as CTS, would like to create a web-based game similar to their existing app which is only available on the Android OS at the moment. Their game must be designed to connect players online in games with multiple teams and players. In teams, the players must be able to interact with a rounds system which is based on guessing stock drawings pulled from a library of stock drawings. CTS also wants to ensure that their game can be played on multiple platforms. However, currently their game application is only available on android. The game will be developed with an engine which allows it to be published in a web-based HTML5 build so it can be played on as many supported platforms as possible.

## [Design Constraints](#_heading=h.tyjcwt)

Requiring support for a larger variety of platforms and a web-based output will constrain the available game engines and optimal operating system. HTML5 requires that a browser be run through a separate application, usually a browser, which limits the amount of processing power most users will have allocated. While games run locally can utilize the full processing power of the device, in situations like a computer browser it will limit the graphical capabilities of the game. Luckily, the game being requested is not particularly graphically intense.

## [System Architecture View](#_heading=h.3dy6vkm)

**[N/A]**

## [Domain Model](#_heading=h.1t3h5sf)

By looking at the function of the program through the main method we can see the basic outline of the program execution. The ProgramDriver contains the utilization of the Entity class and its subclasses. ProgramDriver also uses the SingletonTester to run a test of the singletons created to ensure they are distinct instances of the program. The entity class itself contains three subclasses: Game, Team, and Player. Game contains a reference to the Team class as does Team to the Player class. On the other hand, the GameService class can utilize the Game class but is not a subclass of the Entity class. GameService contains functions to store and manage data within the entity created for managing individual instances.

**"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](#_heading=h.2s8eyo1)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | The Mac operating system is very user friendly and has the full capability of running a browser based application but many applications which are not designed for it will not be compatible with Linux or Windows when developed for Mac. | Linux is a very lightweight operating system which can be run on many different systems and allows for easy maintaining of a server but would not be suited for this project as dedicated servers are not planned. | Windows is the game development standard but is not used for as many server side applications. MySQL and others can be used to manage databases within Windows. For hosting, Windows is the second most popular. | Mobile devices are deployment environments which require different methods dependent on their operating system. Luckily, an HTML5 browser solution allows easy deployment. |
| **Client Side** | Computers which can run the mac operating system are much more limited and proprietary than Linux or Windows. This means that development will cost more and require more specific support. | Game development is much less common for Linux users as it is a much smaller user base. Fewer developers are familiar with development directly in Linux which means development on a different OS with deployment on Linux is the better option. | Most users are comfortable developing and playing games on Windows. Publishing to windows is an important platform to maximize revenue. When using windows it is important to use the .Net framework and many lightweight languages like C++. | Mobile devices are lower end hardware and can cause issues but the application will already need to be very lightweight in order to run on more devices, especially browsers. |
| **Development Tools** | While not native to the environment Mac does have some applications like VS Code which is very lightweight. However, it also has a built in IDE called Xcode and apps such as CLion. Fewer game engines are present with Mac support but | Linux has very limited options specifically related to game development but still has some options. Unity and Godot both have versions which function in Ubuntu however both are also designed primarily for Windows. | Windows contains the most variety in development tools and engines available for game development. Visual Studio, NetBeans and JetBrains Rider provide a variety of coding environments. While engines like Unity and Unreal Engine dominate the development space. | Popular game engines such as Unity come with browser output through their web players which allow HTML players to play their games. Additionally, Unity allows deployment to Android or iOS devices. |

Recommendations

1. **Operating Platform**:

I would recommend using Windows for development of Draw It or Lose It. Both of the most prominent game engines, Unreal Engine and Unity, which support HTML5 output run natively. While there are solutions to run them on other operating systems they do not receive the same level of support.

1. **Operating Systems Architectures**:

HTML5 allows presenting content through a web interface and can be combined with other languages in order to expand functionality. This limits the platforms possible if a language is incompatible. The most common HTML5 engines work with JavaScript for game functionality and allows deployment of software on many platforms.

1. **Storage Management**:

To operate on the most platforms possible data should be stored online to remain accessible from any device the user chooses. This hosting can be done through third-party hosts in order to avoid server upkeep costs at a smaller size.

1. **Memory Management**:

Memory management is extra important in this scenario because we are developing an application designed to run in browsers and on a wide variety of devices. In order to provide the client with an application which runs on as many platforms as possible it must have low system requirements. The largest usage of data memory will be through the generation of the images. By streaming the loading images to the user after they are generated and also generating the necessary data prior to a game’s start will help alleviate the device’s demands.

1. **Distributed Systems and Networks**:

It is important that the necessary user databases for Draw It or Lose It can communicate between every platform which is expanded to. Data distribution needs are handled through MySQL databases which can be accessed on any platform necessary through hosted servers. Servers hosted through Google or Amazon can be efficiently used to minimize costs.

1. **Security**:

Since our application is designed to be an HTML5 application it contains basic security through HTTPS connection security. In order to protect the individual data of players, however, a login system will be utilized with optional cookies only by user request for password recall. Additionally, to ensure the game is fair and balanced the image shown to players will never be stored on the users device.