

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 | 03/14/2022 | Gregory Dionisio | Updates to executive summary, design constraints, description of the domain model, evaluation, and recommendations. |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room is requesting a web-based game to run on any operating system. The game is currently available as an android app and the name is “Draw It or Lost it”. The game will consist of 4 rounds lasting one minute each in which a picture is rendered slowly over a 30 second interval and teams guess what is being shown. If a team fails to guess the image the opposing teams get to answer for 15 seconds until time runs out.

## [Design Constraints](#_2et92p0)

* Needs one or more teams involved
* Each team has multiple people
* Game and Team names must be unique to allow users to check whether the name is in use or free
* Only one instance of the game can exist at any time.
* Must run on multiple platforms

These are the requirements of the game. For the purposes of application development, the Gaming Room would like this to run on all devices. With this application being built in a web-based environment it will be much easier to build for all platforms as they all support html experiences and building from html into apps is not overly difficult and html 5 is much stronger than previous versions at working cross browser. We already have the app on android but need to work it into another mobile device. Along with machines like Windows, Linux, and Apple. Code can be either re-written for Apple devices or existing code to be run on other devices by inheriting other languages.

## [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 class entity builds a relationship with Game, Team, and Player classes. Entity is the parent class and the other classes share the same traits. In UML designs this is called inheritance, each class will share references like “name” and “id”. Team and Player have a “has a” type. While Game has a Team and Game Service has Games. “Has a” refers to an an instance of a class and has a reference to the instance of a different class. Game Service has a reference of Games, Games a reference of Team, and Team 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)

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** | Strong terminal for configuring access or changes to server.  Advantages: upgradeable, can change web hosting requirements.  Disadvantages: Not often user for hosting. | Cost friendly option as its open source.  Advantage:  Security, good at catching flaws before problem occurs, most often used for hosting.  Disadvantages: Difficult to find application to support. | Advantages: Most software available. High resource requirements, faster loading, comfortable to use.  Disadvantages:  Easily corrupted by virus, bad support | Advantages: most popular, good compatibility, portable  Disadvantages: poor security |
| **Client Side** | Macs require a medium amount of time and expertise. Cost is like the other base systems like windows. Building programs for mac typically require the use of swift for programming which is uncommon to other languages. | Linux require a large amount of time and expertise. Cost is cheaper to the other base systems because it is open source. | Windows requires the least amount of expertise and time required. | Requires the most amount of time and cost. Difficult to implement. |
| **Development Tools** | Macs typically use swift for programming. However, macs can run most languages such as html, javascript and css (common web languages) as well as java, python and php that are used for web application as well. Some common ide’s don’t work well on mac. | Slightly difficult to use. Most IDE’s and languages work well with linux. Visual studio, eclipse, java, python, html, css, javascript. | Easier than most to use. Most IDE’s and languages work well with linux. Visual studio, eclipse, java, python, html, css, javascript. | The majority of apps built using android and swift which can be run on the other operating systems as well. Html, css, javascript, python, java, python, php all work well. |
| **Web based** |  |  |  |  |
| **Server Side** | You can run a web-based application through mac os server however it is not very common. And is deprecated technology. So we will not be hosting the gaming app through a mac. | Characteristic: Open Source  Advantages: Secure, Free, Cost Effective  Weakness: Bad Support  Hosting the game on a linux server is great choice. It is open source and free. | Characteristics: Proprietary  Advantages: Most Secure, Best Support  Weaknesses: Paid, Expensive  Hosting the game on a windows server is possible and a good choice. Windows server costs between $300-600. | Servers are very rarely done through a mobile device and this is not a viable option the this game application. |
| **Client Side** | To work across all web browsers the application must use html to display information to the user, typically this is done with javascript, and css to aid in a robust application. There will also be another programming language that handles the communications with files and databases.  Typically, when developing an application you would need to test the application on all types of devices to make sure that it is working well with all three. | | | |
| **Tools Required to build web based application** | Xcode is used as the development tool.  The main programming language is swift. | The most popular development tool on linux is eclipse.  C programming languages is also most popular | Development too is visual studio.  The default programming language is visual basic. | Game development is not done on mobile devices. |

## 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**: A good place for The Gaming Room to start is windows due its ease of use, relatively low cost, and large availability of IDE’s.
2. **Operating Systems Architectures**: Windows is an operating system that uses a graphical user interface to run applications that can access system resources. Most applications have availability on windows due to its popularity.
3. **Storage Management**: Windows makes working with files very easy. For purposes of storage a cloud-based solution can be accessed by all the operating system and depending on how the backup of the data will work can be very safe.
4. **Memory Management**: Draw it or lose it requires high memory usage due to the number of pictures that will need to be stored. This will likely be stored in an SQL database that will be delivered to the users through the web browser.
5. **Distributed Systems and Networks**: Finding a game creation platform that is compatible with all operating systems would be ideal. The servers that will hold the game information will need to be accessible by all types of operating systems. As mentioned above the server will serve most of the data to the consumer no matter which operating system they are using. There will be back up servers and potentially if there is enough clients servers set up strategically throughout the us to service specific areas.
6. **Security**: The amount of security application available for windows makes this a good choice for security. Windows comes with a security protection, but a stronger application of malware, virus and security threats can easily be downloaded to ensure the safety of the data.