

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

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

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_1fob9te)2

[**Executive Summary**](#_3znysh7)3

[**Design Constraints**](#_2et92p0)3

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

[**Domain Model**](#_3dy6vkm)3

[**Evaluation**](#_4d34og8)3

[**Recommendations**](#_17dp8vu)5

## [Document Revision History](#_1fob9te)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/23/2022 | Charles Breuer | Executive Summary, Design Constraints, and Domain Model were updated |

**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](#_3znysh7)

A game development company ‘The Game Room’ wants to expand the supported platforms for their Android based game titled ‘Draw it or Lose it’. The application needs a new framework to support the variety of new platforms and create a user experience similar to their original game.

## [Design Constraints](#_2et92p0)

**The application needs to be supported in the many different screen resolutions of applicable devices.** This is a technical constraint that dictates decisions about user interface, testing, and techniques to ensure compatibility. To ensure that this constraint does not cause issues for the client, decisions about the scope of the project and technology that supports inclusiveness need to be made effectively.

**The application needs to support multiple connections for concurrent gameplay.** The application relies on it's ability to provide seamless connections to game instances and teams. Issues with connections and server reliability create unpleasant experiences for users. A reliable and efficient method for providing connections to the application are an implicit expectation for user experience.

**The application needs a programming language that runs efficiently on multiple platforms.** Many programming languages are built to run well on specific operating systems. Certain languages are run on virtual machines that allow compilation to be independent of the system architecture. These considerations need to be made for an application that aims to provide services for a large breath of platforms.

**The application needs to support technology for rendering images at a steady rate.** Decisions on software design, API, frameworks, and supporting databases factor in to providing the performance required for a smooth rendering of images. Rendering images at a steady rate is a core requirement for enjoyable gameplay.

## [System Architecture View](#_tyjcwt)

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](#_3dy6vkm)

The UML diagram consists of a singleton GameService class which manages the various Entities that include the Game, Team, and Player classes. The ProgramDriver functions as the client entry point for the program and the SingletonTester is used by the ProgramDriver to ensure the GameService class is a singleton. The GameService class organizes the various games in progress and ensures all games, teams, and players are unique by providing unique identifiers for each Entity. The Entity class serves as a superclass for the Game, Team, and Player classes to ensure all subclasses contain a unique id and name during creation. The program utilizes inheritance (with the Entity class) to provide the functionality of unique games, teams, and players. The program uses encapsulation by providing an ArrayList inside of the GameService, Game, and Team classes that are only accessible through public methods that check if new entries have unique names. These ArrayLists allow multiple games, teams, and players to be organized inside their respective objects.

"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](#_4d34og8)

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 web hosting provides software that is secure due to Apple’s high level of security. Mac is accessible with a decent GUI and server configuration options. Mac OS web hosting tends to be the most expensive since they have less demand. Most of the software can also be hosted on Linux for less money. The Mac OS web hosting software was discontinued in April of 2022, making it not a good option. | Linux is Unix-based, making configurations for servers easy with default deployment methods available.  Linux is an open source operating system, where licenses can be obtained for free. Different Distro versions make learning/navigating the platform difficult. Frameworks such as Apache are available for deployment of servers. | Licensing a Windows server is expensive, but not as expensive as a Mac OS server.  Frameworks such as Apache are available for deployment of servers. Software available to support development, including IDEs and frameworks are heavily supported on Windows. Windows includes a user friendly shell and GUI. | The number of different devices available make hosting difficult.  Hosting a web-server on a mobile device is possible, but not a viable option. Deployment methods exist, but are inconsistent. |
| **Client Side** | The developmental process needs to ensure the application runs through an API (Preferably RESTful) that delivers content uniformly with HTML, CSS, and JavaScript in common Mac Desktop form factors. Developmental costs/expertise are similar for Mac, Linux, and Windows with similar browser support. Desktop browsers are becoming uniform in their support for web-based applications, but tackle situations differently. | The developmental process needs to ensure the application runs through an API (Preferably RESTful) that delivers content uniformly with HTML, CSS, and JavaScript in common Linux Desktop form factors. Developmental costs/expertise are similar for Mac, Linux, and Windows with similar browser support. | The developmental process needs to ensure the application runs through an API (Preferably RESTful) that delivers content uniformly with HTML, CSS, and JavaScript in common Windows Desktop form factors. Developmental costs/expertise are similar for Mac, Linux, and Windows with similar browser support. | The developmental process needs to ensure the application runs through an API (Preferably RESTful) that delivers content uniformly with HTML, CSS, and JavaScript in common Mobile aspect ratios. This requirement is considerably more difficult than the other options with varying screen sizes in Android and IOS markets. The browsers that are support are similar to the desktop options, but take different strategies for deployment. Mobile devices require the most expertise and cost to deploy to. |
| **Development Tools** | IDE’s that are well supported on Mac include simple text editors to popular IDE’s such as XCode, CLion, and VisualStudio. Similar options are available for Linux and Windows except for XCode.  Mac supports common server side programming languages such as Java, Python, Ruby, and PHP. Client side frameworks are supported such as REACT or Angular JS for creating user environments in HTML/CSS/JavaScript. | Linux supports similar large scale IDE’s with only small differences to Windows and Mac. These include Eclipse, Visual Studio, and notepad++. Similar front-end and back-end languages and frameworks are supported for Linux and Windows. | Popular IDE’s including Visual Studio Code, Eclipse, Notepad++, Netbeans, and many others are available on Windows.  Many frameworks are available for frontend and backend that support HTML/CSS/JavaScript on the client side and common server side languages such as PHP, Ruby, Python, and newly Node.js.  Windows is a well suited environment for development. | For server development, Android devices support languages such as Java, and Kotlin and have the Android Studio IDE. The main IOS development language is Swift and the IDE is XCode. The client side would use the typical stack of HTML, CSS, and JavaScript with possible frameworks of REACT or Angular JS. Mobile Devices are not well supported for creating software, even though progress is being made. |

## 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 best operating platform option for expanding the Draw it or Lose it application to a web-based service is Windows. Windows is a common and user friendly option for producing software. The cost of licensing a server for Windows is fairly cheap and the operating system supports common technologies needed for server/client side development including common IDE’s, web-browsers, compilers, and programming languages. The Windows operating system is built to run on multiple different types of devices/specifications.

1. **Operating Systems Architectures**:

Windows provides an easy to use graphical interface That gives the users access to their file directory and applications. Windows includes a command prompt for navigating the system and executing application/files. Windows also includes a number of system applications that's provides built-in functionality such as a web browser, firewall, Internet services, and virus protection. Windows gives the user access to system calls where kernel processes can be used for application refinement. Windows also has optimization for process scheduling that utilizes multiple processors easily. Windows comes with a few preinstalled compilers for essential languages such as C++, C#, and Java. The application will use a client/server style architecture.

1. **Storage Management**:

During development, software engineers have access to multiple storage management programs including disk management, storage sense, and disk cleanup to manage the server. The data will be stored on the server as a MySQL database. The client side of the application does not need to contain a storage solution. All relevant information about the game will be stored in memory. An appropriate storage management system for the images on the server side would be either a hash table or an array list. An algorithm can choose the images to be associated with a particular game and use their ID's to choose those images. The data that describes users can be stored has attributes for their instance of a particular class.

1. **Memory Management**:

For the application itself, the clients memory will Represents the images that are in use during a particular game. The remainder of the files stay on the server side . At the end of a game, the client's memory is cleared of any resources from a previous game. The server will hold all of the images in memory for easy access on the client side. Since there is only 1.6 gigabytes of approximate storage for the images, That's good the ram should be able to handle the load. Information about the user interface will be downloaded into memory on the client side before the application begins. User data and game analytics will be stored inside of the database until it needs to be used. Windows servers allow access to physical and virtual addresses of space up to 4 gigabytes of memory for use in the application.

1. **Distributed Systems and Networks**:

To prevent issues such as outages or loss of connectivity servers need to be managed for scalability and ensuring individual server devices are competently allocated. The application will use RPC protocols to communicate through JSON ensuring that all interactions are uniform and stateless. The application itself is web-based and uses common technologies such as JavaScript, HTML, and CSS to create the web pages. Creating the client side of the application needs to be mindful of the memory and CPU performance of the devices that are being targeted. Communication between the client and the server will utilize restful API to ensure communication is uniform.

1. **Security**:

The Windows operating system comes with built-in virus protection and firewalls. This will help with protecting the databases and ensuring that development of the software is safe. Windows also comes with automatic updates for the operating system and supporting software. This prevents issues that can arise from an updated software. To ensure data is safe inside of the database we will implement an encryption system particularly for user data and game data. Windows also provides a number of features that can be used to ensure the system is safe including windows defender, threat analytics, and host/administration privileges. Specifically on the client side of the application, data involving the customer will be encrypted before it is moved to our databases. There is an authenticator and credential software features on the client side to verify user rules.