

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 | 09/16/22 | Shawn Way | Initial definition |
| 1.1 | 09/30/22 | Shawn Way | Multiple platform requirement |
| 1.2 | 10/11/22 | Shawn Way | Recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wishes to develop a web-based version of their Android game, Draw It or Lose It. Players will have short periods of time to guess the answers to image puzzles rendered from a large stock drawing library. One or more teams can play the game at a time. Teams will contain multiple players. Game, team, and player names must be unique.

## [Design Constraints](#_2et92p0)

Game functionality will be written in Java.

Code must be written to function on any platform that can run a web browser.

Server access will be necessary for this game to run as requested.

All below software requirements will be adhered to:

* A game will have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* 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. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [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)

Draw It or Lose It will be written to the below UML diagram. The Entity class will be the base from which the Game, Team, and Player classes will inherit. This will adhere to the object-oriented principle of inheritance, which will simplify development and lead to more readable and maintainable code. The GameService will use the singleton pattern to ensure there that 1 or less instances of it can exist. There can be any number of Games per GameService, Teams per Game, and Players per Team. The ProgramDriver will serve to run the program and use a SingletonTester to verify that no more than 1 instance of the GameService can exist. Each class will have private attributes, public methods, and some will have private methods. This follows the object-oriented principle of encapsulation, which will hide unnecessary details from user. It also follows the priniciple of abstraction by providing a set of methods for interaction with the objects without the need to understand the inner-workings of each object.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Highly stable OS. Simple and user-friendly interface. The Mac OS license is quite expensive. Mac is not customizable, for the most part. | Most Linux OSs do are free or a one-time payment. Linux is an inherently more secure OS. Requires more system knowledge and effort to set up. | Windows provides automatic driver setup. Windows is highly adaptable and optimizable. Windows security suffers due to its prevalence in the business world. Windows OS license is not free. | Most user-friendly interface. Allows for mobility. Severely limited in storage and processing capability. Would result in very low availability. |
| **Client Side** | The client side would be developed to run inside the web browser. This would allow for a platform-agnostic user interface. Most of the development concerning the UI on differing devices would require an adaptable use of screen space for the many sizes of screens. | The client side would be developed to run inside the web browser. This would allow for a platform-agnostic user interface. Most of the development concerning the UI on differing devices would require an adaptable use of screen space for the many sizes of screens. | The client side would be developed to run inside the web browser. This would allow for a platform-agnostic user interface. Most of the development concerning the UI on differing devices would require an adaptable use of screen space for the many sizes of screens. | The client side would be developed to run inside the web browser. This would allow for a platform-agnostic user interface. Most of the development concerning the UI on differing devices would require an adaptable use of screen space for the many sizes of screens. |
| **Development Tools** | Java can be used on all platforms. Eclipse can be installed and ran on Mac for server development. The Eclipse Web Development Tool Kit will be used to develop the client. There will need to be at a team for the client side and for the server side. | Java can be used on all platforms. Eclipse can be installed and ran on Linux for server development. The Eclipse Web Development Tool Kit will be used to develop the client. There will need to be at a team for the client side and for the server side. | Java can be used on all platforms. Eclipse can be installed and ran on Windows for server development. The Eclipse Web Development Tool Kit will be used to develop the client. There will need to be at a team for the client side and for the server side. | Development for a server on a mobile device will need to be done on a desktop. Eclipse plugins can be used to emulate a mobile device during development. Eclipse Web Dev Tool for client. Two teams to develop client and server side. |

## 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**: Linux will suit all of the needs of the game well. It is a free OS, it’s inherently secure, and it is much more lightweight than the other options and will therefore use up less space for the OS.
2. **Operating Systems Architectures**: Linux has layers to handle all operating system related tasks. Starting from the deepest layer out, the hardware layer is fairly self-explanatory. It is the hardware and devices like the CPU, disks, and printers. The next layer is the kernel, which is the heart of Linux and interacts with the hardware layer. The shell is the layer that allows user interaction with the kernel. This is where the command-line or graphical interface interprets user commands and executes kernel functions. The last components of the Linux OS are the system library, which contain functions necessary to the OS, and the system utilities, which do specialized tasks.
3. **Storage Management**: Linux’s stock file system, EXT4, is optimal for this use. It is the most recent version of the EXT file systems and has greatly improved performance and capacity. It is more likely to store files contiguously, which maximizes storage space. It is also more reliable than past versions.
4. **Memory Management**: Linux uses virtual memory to abstract the details of physical memory. This allows the programs to interact more smoothly with an otherwise complex matter of direct memory access. It also uses a cache in order to speed up access to memory. Linux also has the ability to reclaim previously used memory.
5. **Distributed Systems and Networks**: The client will run within the user’s browser, which will allow it to be platform agnostic. The server will act as the backend, storing all game data and resources necessary to the game. HTTP requests made through the REST API will be made by the client to the server to read and write data when necessary. The client will need to be developed in such a way that it is compatible with the majority of currently used browsers. Updates will be applied on the website for the client and on the server for the backend. This will simplify maintenance and upgrades. The common failure point will be the server. Any outage of the backend will affect all users. Server uptime and loading will need to be optimized to match user traffic during every hour of the day and during each day of the week. This will ensure maximum availability while minimizing server overhead.
6. **Security**: Linux is an inherently safe OS due to the built-in security features which do not allow installation of files or running of scripts without user permission. Passwords should be required to be sufficiently complex. Failure attempts should be limited, and password resets will require verification of user data, like security questions, to perform. The server will need a firewall that maximizes performance without sacrificing security.