

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

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

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

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

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

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

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

[**Domain Model**](#_heading=h.2s8eyo1) **4**

[**Evaluation**](#_heading=h.17dp8vu) **5**

[**Recommendations**](#_heading=h.26in1rg) **6**

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/16/23 | Cameron Thorp | <Brief description of changes in this revision> |

**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](#_heading=h.35nkun2)

The Gaming Room wants to develop a web-based application of their game Draw It or Lose It, which is already available on Android devices. Creative Technology Solutions will be responsible for constructing an environment to allow them to make their app accessible to a broader audience. This will involve helping The Game Room to manage their software requirements as well as the hardware requirements that will be necessary for the new application.

## Requirements

The following requirements have been identified:

* 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.

## [Design Constraints](#_heading=h.1ksv4uv)

The following design constraints have been identified:

* The web-application should be accessible on any common browser and adhere to SEO practices
* Multiple users must access the same instance simultaneously to allow for an interactive experience
* Server-side processing must be maintainable on constraints established by The Gaming Room
* Clients must maintain constant connection in order to receive images from the database

## [System Architecture View](#_heading=h.44sinio)

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](#_heading=h.2jxsxqh)

The ProgramDriver serves a dual purpose, it acts as an entrypoint for the user, and it verifies the functionality of the singleton that will be used to manage players. For the web-application’s back-end the singleton structure will be used to allow multiple players and teams to interact with a single instance. This instance is a GameService, which players all utilize simultaneously. Within GameService, players, teams and games are creatable and accessible. Each of these objects is an extension of entity, which defines common members for Game, Team, and Player 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](#_heading=h.z337ya)

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.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac does not come standard with the ability to host a server. A system could be developed, but this would be at added cost to The Game Room. | Linux can be light-weight and oftentimes contains minimal resource consumption. This makes it ideal for being used as a back-end host. | Traditional Windows is easy to manage and maintain, but is less efficient with resources and more expensive. A possible solution would be the server-specific Windows OS | Mobile devices are not designed to maintain the continuous load of running a server. Additionally, it would be more expensive to procure the necessary hardware. |
| **Client Side** | The use of web-browsers allow for the application to be independent of the user’s OS. | The use of web-browsers allow for the application to be independent of the user’s OS. | The use of web-browsers allow for the application to be independent of the user’s OS. | Web-applications on mobile devices require more testing and development to make them accessible on mobile devices. They are, however, more common and would open the market to more users. |
| **Development Tools** | Mac has many IDEs and often has streamlined processes for developers to build and release code. However, flexibility and the cost of repairs and maintenance could be higher. | Linux allows for a flexible environment for any language, however utilizing the JDK and JRE, this would be feasible on any supported OS. Environment maintenance could be delegated to a 3rd party for a price. | Windows has support for almost all popular IDEs and allows for access to the maximum amount of extensions and external tools. Enterprise level IDEs would require subscriptions to access advanced features. | Mobile devices are not optimized for development, but can be used in testing developed code. They can also be emulated in order to accelerate the development process. |

## 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**: CTS recommends using Windows Server. This will allow for the implementation of extensive management tools and management without incurring the cost of training personnel to maintain a Linux system or the cost of hiring a 3rd party to do so. Additionally, this will more easily allow for the expansion of The Gaming Room’s capabilities through cloud infrastructure in the future via Azure. Utilizing the support structure for a professional Windows license and potentially Azure will allow The Game Room to have a high rate of uptime with minimal costs due to training.
2. **Operating Systems Architectures**: On the server-side it is recommended that a universal back-end environment is established, to ensure development and maintenance can be performed efficiently. By implementing proper APIs and microservices to act as middleware, the client-side architectures can be non-standard so long as they are interacting with Draw It or Lose It servers through standard protocols.
3. **Storage Management**: Due to the size of high-resolution images, storage will primarily exist on the server-side. This will include back-end code, databases, and file-systems that will enable client-side users to access pieces of data from the server as needed. The only client-side storage required will be for specific binaries for the randomized pictures from the database, and details about the game-instance they are currently part of.
4. **Memory Management**: Memory management for the server-side will be capable of maintaining the instances for all currently running games, and will be able to handle requests for new instances and data transfers for profiles and images. Client-side memory management will consist of instance data for their game and the request information for images, generated by the game instance they are part of.
5. **Distributed Systems and Networks**: Because of the memory management approach, clients will have access to an instance of a game and all associated information and images once the game begins. All actions that require server interaction will require a connection, however image rendering and profile information will exist locally, and will not be reliant on a constant network connection. Additionally, players across different operating platforms will not encounter compatibility issues due to the use of microservices and proper middleware between the front-end and back-end of Draw It or Lose It.
6. **Security**: Security will be maintained on both the server and client-side via the use of strict permissions and protocol management. No external user will have the ability to change or access player information without a valid account that is linked to that specific information. No remote users will have access to modify permissions, or static data like image files and the archived profile database.