

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 |
| --- | --- | --- | --- |
| 2.0 | 11/28/2021 | Mohamad Darwiche | Analyze the operating system for the client program "Draw It or Lose It." |

**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](#_sbfa50wo7nsh)

The Gaming Room wants to develop a web-based game that serves multiple platforms based on their current game, Draw It or Lose It, which is currently available in an Android app only. to help team members guess the puzzle, the application will render images from a large library of stock drawings as clues. Each game has four rounds of play lasting 1 min each. Drawings are rendered at a steady rate and are fully complete at the 30-second mark. If the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess each to solve the puzzle with a 15-second time limit.

## [Design Constraints](#_2et92p0)

* Currently runs on Android and wish to develop a web-based game that serves multiple platforms.
* 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.

Each operating system makes use of a unique platform that must be developed separately. To develop its applications, all platforms use different coding languages, requiring the deployment of multiple development teams. All applications will not be identical and must be developed in accordance with the platform's functionalities.

## [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)

Entity has an Is-A relationship with the Game, Team, and Player. Game, Team, and Player are all inherited from the Entity. All three, game, team, and entity, share common attributes such as id and name. As a result, it is defined as Super class Entity.

A Team consists of Players, a Game consists of Teams, and a GamesService consists of Games. Aggregation can be used to represent it in UML. A Has-A relationship indicates that one instance of one class refers to another instance of another class. In the diagram above, GamesService is a reference to Games, Games is a reference to Team, and Team is a reference to Player. One GameService can have multiple Games, each of which can have multiple Teams and each of which can have multiple Players.

**"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** | It is a less popular choice to run a server on Mac OS. One major disadvantage of Mac OS is that you will need someone who is familiar with it to get the most out of it. Having someone dedicated to running the server may raise the prices of operation and maintenance as well as the licensing fees for the operating system. | Navigating the platform is difficult. Linux, like Mac but its low cost and an open source. Not many people are familiar with it. | Because Windows is most likely the most popular operating system, finding users to run a Windows server would be much easier and cheaper. More software available compared to other OS. | It's better if the server is immobile and can be tracked at a single place. Specifications are better in other devices. |
| **Client Side** | Expertise and time are required. Costs about the same as windows. | Expertise and time are both fully required possible. The least expensive option. | Minimal knowledge and time are required. Costs the same as a Mac. | Allows clients or even developers to view updates from any location. Implementation is slightly more difficult than with other devices. |
| **Development Tools** | JavaScript, HTML, CSS, Python, and other IDEs for Mac systems are available. Visual Studio, Eclipse, and online development tools are among the tools available on Mac systems. | Java, Python, Ruby on Rails, HTML, and other IDEs are available for Linux systems. Github, Repl.it, nodejs, Visual Studio, and other development tools are available for these systems. | Python, C++, HTML, and other languages are used in Windows OS. Eclipse, command prompt, Repl.it, and other developer tools are available. Many of the tools that are supported on other OS. | HTML, PHP, C++, Python, and other programming languages are examples. Github, Visual Studio, command prompt, Repl.it, and other development tools are available. |