

Draw It or Lose It!

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## 

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/14/2023 | DEONNE LUDWIG | **In the Game Service class, use the iterator pattern to complete the coding for the addGame() and both getGame() methods.** |
| 1.1 | 01/21/2023 | DEONNE LUDWIG | Create base class called Entity to hold common attributes and behaviors of, refactor Game class to inherit new Entity class, and complete code for Player and Team classes. |
| 1.2 | 02/05/2023 | DEONNE LUDWIG | Complete evaluation of the characteristics, advantages, and weaknesses of each operating platform (Linux, Mac, and Windows) as well as mobile devices. |
| 1.3 | 02/18/2023 | DEONNE LUDWIG | Analyze the characteristics of and techniques specific to various systems architectures and make recommendations. |
|  |  |  |  |
|  |  |  |  |

**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)

We are serving as a Technology Consultant for Creative Technology Solutions (CTS) and have been asked to prepare a software design document and begin developing the game application, addressing their software requirements.

CTS has recently taken on a new client, The Gaming Room. 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. Draw It or Lose It is loosely similar to the 1980s television game *Win, Lose or Draw*, where teams compete to guess what is being drawn. Rather than a player drawing images on an easel to help team members guess the puzzle (a phrase, title, or thing), the application will render images from a large library of stock drawings as clues. A game consists of four rounds of play lasting one minute 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.

## Requirements

*Please note: While this section is not being assessed, it will support your outline of the design constraints below. In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.*

## [Design Constraints](#_2et92p0)

* Must be in a web-based format and serve multiple platforms.
* A game must 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.

## [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 singleton pattern is used to adapt an ordinary class,** so only one instance of the GameService class can exist in memory at any given time by creating unique identifiers for each instance of a game, team, or player.

The Entity class utilizes polymorphism by holding and overriding the common attributes and behaviors (name and id protected by encapsulation). The Entity class is also inherited by Game, Team and Player.

The Game class extends to Entity and ensures only one instance of the game can exist. It also has a relation to the Team class so a game can the have the ability to have one or more unique teams involved. The Player class extends Entity and has a relation to the Team class so that each team can have multiple unique players assigned to it.

**"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** | High cost. Closed platform which will add to costs. Licensing costs.  More hardware resources needed (CPU, RAM, etc.).  Proprietary hardware/software needed increasing cost but also providing better security and less errors.  Can run on Apache servers like Linux | Low cost. Open-source platform can cut costs.  Can be run on low-end or older servers. Infrequent updates and security will have to be maintained to achieve stability. Less cost and can run on Apache servers. | Closed platform which will add to cost. Licensing costs. More hardware resources needed(CPU, RAM, etc.).  Frequent updates will require more downtime. Less stable. Proprietary hardware/software is needed but offers ASP, .NET, Access, and MSSQL. | Open-source platform. Less common. Reduced speed. Limited browser support. |
| **Client Side** | Mac offers very user-friendly applications for development that may save costs in time but can only be used on Mac machines that cost more. | Fewer restraints in development may save in costs but it may cost more for the development and expertise needed to implement. | Very accessible development solutions but more involved to set up and establish stable environments. Well-documented application development support capabilities. | Cost may be more for expertise in creating added UX/UI, integration, responsiveness, etc. Speed, power, and connectivity may be problematic. |
| **Development Tools** | Swift, JavaScript, SQL (or SQL Server), Java, C#, PHP, Python, C++, C, js, AngularJS, Ruby, Objective-C. Visual Studio, Xcode, CLion, AppCode, PhpStorm  Possible multiple development teams needed and licensing for tools. | Visual Studio, Sublime Text, Atom, Brackets, Komodo Edit, Geany.  IntelliJ IDEA, Eclipse, Apache NetBeans, Bluefish  Extra development for cross-platform compatibility. | Visual Studio, NetBeans, JetBrains Rider, IntelliJ IDEA, Android Studio.  Possible multiple development teams needed and licensing for tools. | Android Studio, Qt IDE, Xcode, Eclipse IDE, JetBrains Rider, Visual Studio Code, DroidScript, Swift  Extra development for cross-platform compatibility. |

## 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**: I would recommend using the Windows operating platform to allow The Gaming Room to expand Draw It or Lose It to other computing environments since it has the largest user demographic.
2. **Operating Systems Architectures**: Windows is a rectangular area which provides an environment to run many programs using a Graphical User Interface (GUI). Windows Explorer is a tool that allows you to browse, view, copy and manage files. Since it has very accessible, well-documented, development solutions, programming and gaming support, it would be a good choice to start development.
3. **Storage Management**: We would need to at least consider the known variable: will have 200 high-definition image files to choose from, each one approximately 8 megabytes in size.

The Gaming room has expressed that it will offer 200 images so we at least know that we will need room to house the software files themselves, the images, and other components needed for the functionality of the game and records, player info, etc. The storage solution chosen would need to preserve the capabilities of growth for the future. Multiple external drives or cloud storage solutions could be used or a combination of the two.

1. **Memory Management**: We would need to consider the following:

1.) must have the ability to run multiple instances of the game at once, each having players and teams. 2.) will have 200 high-definition image files to choose from, each one approximately 8 megabytes in size 3.) the game application will need to render and display pictures at a fixed rapid rate for up to 4 minutes for each game

Since the new implementation of the game would be on various OSs of the user’s computer (and their memory, resources, etc.) you could set the game to have a minimum RAM, etc. requirement for access. Also, you could use a memory debugger to see the game’s memory usage and the impact it will have on the system. Physical and allocated memory used, the memory footprint, memory growth, memory graphs, etc. would need to be analyzed and addressed. From there you could find ways to reduce usage for example reducing the file size of the images. Overall working to reduce memory while preserving quality.

1. **Distributed Systems and Networks**: Since Windows is so commonly utilized it would offer an opportunity and capability of a stable server environment and network to support the higher traffic volume that the game predicts to have. Regular updates and patches will work to keep systems up to date and outweigh the added instability issues of the platform.
2. **Security**: The client-server pattern will allow better security and scalability. REST API could be utilized so that when the request is made from the client all of the information needed for the server to respond should be included and will be returned, giving the state back to the client and cached. Uniform interfaces allow the server to easily communicate with the client. The messages between the server and the client will include layers and optionally return the code to the client. Since each security principal object is represented by a unique identifier it allows authorization and authentication that can add security by implemented annotations. The JSON formatting used allows the interchange between all of the programming languages also offering security and encryption. Ample server/connection resources, security, and error fail-safes are needed.