

# Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 03/19/23 | Stephanie Douglas | Wrote Executive Summary, Design Constraints, and Domain Model. Requirements section is incomplete. |
| 1.1 | 04/9/23 | Stephanie Douglas | Wrote Evaluation section |
| 1.2 | 04/16/23 | Stephanie Douglas | Wrote Recommendations section |

**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 has requested a web-based version of their game Draw It or Lose it. This game should access images from a library which will act as clues to the answer of a puzzle. These images will appear steadily within 30 seconds of each round. There are four rounds and each are 1 minute long. If the puzzle is not solved, then the other teams will have 15 seconds to guess. To develop this web-based application, a web environment must exist, the game code must be created, and the game code should be implemented on the website.

## Requirements

Technical requirements:

* Ability to have more than one team
* Ability for each team to have multiple players
* Only one instance of the game should exist at any time
* Team and game names are unique
* Timers for game round
* Library of images
* Web host, domain, database, and SSL certificate

## [Design Constraints](#_2et92p0)

To create a web-based distributed environment essentials for a website are needed: host, domain, database, SSL certification, and a web developer. In preparation for the game implementation, the web developer creates the website using HTML, CSS, JavaScript, and Java. The game is coded in Java and includes game, team, and player classes. Game, team, and player instances are lists that contain details regarding multiple games, teams, and players. To ensure only one game exist in memory at any given time, unique identifiers for game, team, and player are created. Checks should exist when creating team and game names to avoid duplicates. A library of images is added to the database and the game GUI interface is created. Timers must exist, including 1 minute rounds, 30 seconds of images appearing, and 15 seconds for other teams to solve puzzle. Once the game is coded and tested it should be embedded on the website.

## [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 following UML diagram has a base class named Entity and three inherited classes, including Game, Team, and Player. The Entity class holds the attributes and behaviors that the three inheritance classes can reference. GameService class is used to create a singleton, meaning only one instance of the game can exist at any time. The UML shows that GameService class can have zero to many games, the Game class can have zero to many teams, and the Team class can have zero to many players. The addGame method in the GameService class ensures only unique game names can be added to the game, while the addTeam method in the Game class ensures that only unique team names can be added. When the ProgramDriver class is executed, it will initialize data and use the SingletonTester class to obtain the singleton instance from GameService.

**"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)

Consider the similarities and differences between operating system and server/client side deployments. In addition, compare the some of the development tools per operating system.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | -More secure than Client Side  -Easily create features for Mac clients  -Easy administation  -Intuitive GUI  -Requires Apple hardware to run  -Not cost efficient | -Secure  -Less security threats than Windows Server Side.  -Free to use  -Open source | -More secure than Client Side  -More security threats than Linux Server Side  -Easy to use.  Requires paid license  -Closed source | -More secure than Client Side.  -More cost effective than client-side mobile apps.  -Dependent on internet.  -Reduced speed. |
| **Client Side** | -Less secure than Server Side.  -App store ease of use.  -Fast updates  -Development cost is costly. | -Secure  -Many Windows and Apple apps do not work on Linux.  -Many games are not supported by Linux.  -Open source | -Less secure than Server Side.  -Less costly than Mac. | -Less secure than Server Side mobile apps.  -Less cost effective than Server Side mobile apps.  -Independent on internet.  -Faster speed. |
| **Development Tools** | Programming language: Swift  IDE: Visual Studio, Eclipse, NetBeans  Dev Tools: FlexiHub, Homebrew, Xcode, Sublime Text | Programming language: C  IDE: Visual Studio, Sublime Text, Atom, Eclipse  Dev Tools: Seamonkey, Quanta, Bluegriffon | Programming language: C, C#, C++, Asm  IDE: Visual Studio, NetBeans, JetBrains Rider, IntelliJ IDEA  Dev Tools: Seamonkey, Quanta, Bluegriffon | Programming language: Objective-C, Swift, Java  IDE: Android Studio, Qt, Xcode, Eclipse  Dev Tools: React Native, Flutter, Xamarin, Ionic |

## Recommendations

1. **Operating Platform**:

It is recommended that The Gaming Room use a cloud-based service as the operating platform for Draw It or Lose It. By using a cloud-based environment it does not require server management and allows The Gaming Room to focus on what is important, the game itself.

1. **Operating Systems Architectures**:

A cloud-based platform architecture is managed by the service provider. While it does come at a cost, it will likely be more cost efficient than managing your own server. The web-based game Draw It or Lose It will be easily accessible from various operating systems through the web.

1. **Storage Management**:

One advantage of using a cloud-based platform is storage accessibility. Most IaaS (Infrastructure as a Service) providers have tiered subscription based on storage. After launching the game the team can better gauge how much storage is truly required based on the popularity of the game.

1. **Memory Management**:

Cloud-based platforms offer data analytics to monitor memory and usage. If it is determined that Draw It or Lose It runs slow, or sessions are dropped due to being overloaded, increasing the available RAM from the service provider is available. In addition, the code can be reviewed to ensure it is polished and there is no unnecessary code that is contributing to the lag.

1. **Distributed Systems and Networks**:

Considering Draw It or Lose It is a web-based game it can be accessed through various operating systems via the web. Access to the game is dependent on the network connection or internet access. The game will need to be tested on all web browsers to ensure there are no bugs. There is a risk of server outages in which case this would be escalated to the cloud service provider.

1. **Security**:

Infrastructure security is handled by the cloud service provider, however The Gaming Room should uphold security regarding sensitive information about their users in the databases. This means that user information in the databases should only be accessible by designated employees and the sensitive information should be encrypted on the databases.