

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 | 09/19/21 | Hunter Webster | Project 1 Submission |
| 2.0 | 10/03/21 | Hunter Webster | Project 2 Submission |
| 3.0 | 10/17/21 | Hunter Webster | Project 3 Submission |

**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)

Our new client, The Gaming Room, wants our team to develop a web-based platform for their game “Draw It or Lose It” which is currently available exclusively on Android devices. Since the aim of the project is to expand the game’s potential audience, The Gaming Room wants the new web-app to be accessible from other capable devices (e.g. iOS, Windows, Linux). The game works by giving one team of players the chance to guess a puzzle over a 30 second period, after which the other teams will have a 15 second period to make a single guess. Each puzzle is a stock image pulled from a large library that steadily renders over the guessing period.

## [Design Constraints](#_2et92p0)

This project will be written in java based on the template code that was modified in Milestone 1. A key concern throughout the development process should be ensuring that the application works on as many of the major web browsers as possible. Since the platforms that users access the game from will vary, the design team will need to be mindful of varying display resolutions so that all essential content is able to be fit on-screen. Additionally, it will need to be determined whether the stock images used for the puzzles should be stored in a proprietary database or pulled from directly from the image provider’s online database each round.

At this point, no specific business constraints have been identified.

## [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 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.**

**Game Service:**

* Singleton Class
* Has a zero-to-many relationship with the Game class.

**Entity:**

* Parent class for the Game, Team, and Player classes.
* Holds basic attributes: id (long), name (String)

**Game:**

* Has a many-to-zero relationship with the GameService class
* Has a zero-to-many relationship with the Team class
* Inherits methods and attributes from the Entity class
* Unique Attributes: teams (list)

**Team:**

* Has a many-to-zero relationship with the Game class
* Has a zero-to-many relationship with the Player class
* Inherits methods and attributes from the Entity class
* Unique Attributes: players (list)

**Player:**

* Has a many-to-zero relationship with the Game class
* Inherits methods and attributes from the Entity class
* Unique Attributes: none

**Program Driver:**

* Tests all objects described above
* Calls the Singleton Drive.

**Singleton Driver:**

* Called by Program Driver

## [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** | * Offers server-based deployment method * Has professional grade server tools that cost a fee. Open-source tools also exist * great for server configuration. * versatile terminal that enables easy server design, access, and modification. | * Offers server-based deployment method * Has many open-source server tools that are free to use. Some services may charge. * Has a command shell that enables simple server configuration similar to Mac. * Some distributions are strongly suited for server design. | * Offers server-based deployment method * Has professional grade server tools that cost a fee. Open-source tools also exist. * Offers the strongest server security. * Most common OS * High level of support * Probably the most expensive server-side * Uses command prompt | * No effective means of deployment due to hardware capability * Mobile devices have widely varying hardware specs, making it hard to target a specific level of performance universally. * Operating systems will vary strongly between android versions and iOS. |
| **Client Side** | * Web browser compatibility can be best endured by using HTML, CSS, JavaScript, and a frontend framework like React or Angular * Most approachable OS for new users. * Wide install base, common for specific industries. * Uniformity of install base makes it easier to develop for. * Many developer tools available | * Web browser compatibility can be best endured by using HTML, CSS, JavaScript, and a frontend framework like React or Angular * Least uniform OS. Many distributions exist for varying purposes. * High level of expertise and time needed for Linux Development * Linux users need the most time with the OS to be considered “proficient” | * Web browser compatibility can be best endured by using HTML, CSS, JavaScript, and a frontend framework like React or Angular * Most popular OS, designing any web-app will require accounting for Windows users. * Many development tools available thanks to popularity * Moderate expertise needed to develop for. | * On mobile devices, the most important factor to consider is varying display sizes. * Similar to the non-mobile platforms, using HTML, CSS, and JavaScript with mobile constraints in mind is the best way to ensure that the app is mobile compatible. * Hardest platform to develop for due to the severe differences in the many distributions of Android. * High level of expertise and time needed for development. * Most difficult platform to implement the system on. |
| **Development Tools** | * Languages may include: HTML, CSS, JavaScript, Node.js, Ruby * Will use supporting libraries to develop the front end of the product. * No notable licensing costs unless a chosen library has a fee. * IDEs: AppCode, IntelliJ IDEA, GoLand, PyCharm, Eclipse | * Languages may include: HTML, CSS, JavaScript, C, Node.js, Ruby * Will use supporting libraries to develop the front end of the product. * No notable licensing costs unless a chosen library has a fee. * IDEs: Eclipse, bluefish, Atom, Sublime | * Languages may include: HTML, CSS, JavaScript, Node.js, Ruby * Will use supporting libraries to develop the front end of the product. * No notable licensing costs unless a chosen library has a fee. * IDEs: Visual Studio, IntelliJ IDEA, Atom, Eclipse, PyCharm | * Languages may include: HTML, CSS, Java, PHP, JavaScript, Swift, Objective-C * Will use supporting libraries to develop the front end of the product. * If an app is released, in app sales result in a commission being taken by Apple/Google * IDEs: Android Studio, Xcode, Visual Studio, IntelliJ |

## 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**: For Draw it or lose it, I would recommend Windows Server OS for its expansion capability and performance. Windows Server OS has protected and supervisor modes designed to ensure that the companies servers perform well and reliably. Additionally, I would recommend Windows Server OS for its flexibility to operate in different roles ranging from file, application, web, mail, and database servers. Another key factor to consider is that Windows Server receives constant support from Microsoft, and as such will be reliably stable.
2. **Operating Systems Architectures**: Like many server OS’s, Windows server architecture has many tools designed to optimize performance in many contexts, such as specific tools for memory and file management designed to allow for optimal allocation of computing resources. Windows server utilizes multi-processor scheduling to allow for greater app performance, particularly on server comprised of many processors. These sorts of tools allow for smarter distribution of computing resources that may prove to be key when operating a web-based game running many instances. Lastly, while it may be a bit of a personal preference, I am partial towards using PowerShell for maintenance and optimization purposes.
3. **Storage Management**: Windows server offers a broad selection of tools designed to allow for routine file and storage management. The server OS is capable of relocating large segments of stored files to solid state and alternative storage options in order to free up space that may be necessary for any updates or maintenance without running into disk space issues. Like many server OS’s, Windows is also capable of allowing company employees or designated personnel to pull files from the server to personal work devices should the need arise.
4. **Memory Management**: Windows server contains Random Access Memory (RAM) comprised of physical memory and virtual memory space of up to 4 GB. Virtual memory allows the system to behave as though it had significantly more memory space than the hardware specs indicate. This virtual memory is capable of quickly being moved from virtual addresses to physical disk space in the system, freeing up extra RAM for the system to delegate as needed.
5. **Distributed Systems and Networks**: For this type of application, including networking support for a distributed system is fantastic way to optimize performance and mitigate any downtime during maintenance or server issues. Distributed systems allow for users to potentially communicate with multiple servers designed for specific tasks and allow for a significant increase in both latency and computing efficiency. In this instance where multiple users will be connected to a single game instance, users should see an increase in performance when a distributed system is in place.
6. **Security**: Windows Server OS comes packed with security features designed to fend off any malicious external actors. A type of security commonly employed in Windows servers are shielded virtual machines designed to defend against rootkits, bootkits, and kernel level malware that might expose crucial user or company information. With these sorts of VMs, unauthorized access to protected data is strongly deterred as defined by the system’s administrators. Another security tool found on Windows Server is Windows Defender Application Control, which is designed to tightly manage which applications are allowed to run on the machine. Similarly, Windows Defender constantly scans the system for any form of recognized malware.