

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 4**

[**Evaluation 5**](#_Toc115077325)

[**Recommendations 7**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/13/2023 | Thomas Weinhold | Drafted initial entries in each field. Pending manager approval |

**[Executive Summary](#_sbfa50wo7nsh)**

## The client The Gaming Room has decided to outsource a project to our consulting firm. They require the development of a web-based application that serves multiple platforms based on their current game, Draw It or Lose It, which is only currently available in an Android application. We must develop software that involves either one or more teams with multiple players in each game. The program will allow players to determine if game and team names are available for use. Unique identifiers for each instance of any object must be created to ensure only one instance of any one game exists at a time.

## Requirements

* 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](#_2et92p0)

Requirements Constraint:

There is a need from the client for our services to meet the specified requirements.

Style Constraint: There is a need fro+m the client to maintain a specific style that represents the application’s current success. Every iteration of the application needs to maintain a uniformity that reflects the style presented in the original application.

Usability Constraints:

There is a need from our company for the framework standards and development language to be consistent for our developers to be able to easily access and debug the system.

Principle Constraints:

There is a need from both our company and the client to uphold the respective principles of design that each company holds.

Integration Constraints:

There is a need from the client for the application to be integrated into multiple platforms since it is currently only available through the android platform.

## [Domain Model](#_8h2ehzxfam4o)

The UML diagram below shows how the program uses multiple classes to help developers to understand how the different objects interact with each other. Software that is easily understood by developers can be easily fixed by developers. The program driver operates both the Singleton tester and the game service classes. The game service class pulls information from the game, team, player, and entity classes to create distinct individual games with either one or multiple distinct teams made up of multiple distinct players.

The diagram shows how the game, team, and player classes each inherit information from the entity class. This is shown by the arrow which points to the entity class from multiple branches originating for each child class. This system helps a developer see which objects belong to specific classes and which objects can be used by all classes.

The diagram shows the private and public members which are identified by the minus and plus symbols respectively. This ensures the users only interact with the objects that apply to them while the developer can still interact with the whole system.

Within the game service class there are multiple methods for getting a game with different parameters. This ensures the user can retrieve the specific information they require from inputs into the same source.

**"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

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | The terminal commands for Mac are more flexible to configure the server than other operating platforms. It’s also easier to access and make changes. The biggest weakness to Mac is the cost of owning Apple products. | The Linux operating platform also has flexible terminal commands but is much more cost efficient than owning Apple products. However, Linux is generally not readily available to most users. | Terminal commands are not as flexible as Mac or Linux. However, Windows is way more readily available to the average user than Linux or Mac. | Having the server immobile is a better option than trying to create a server on a mobile device so that the client can track the server to a single place. Mobile devices don’t generally have correct specifications to host servers. |
| **Client Side** | Cost will depend on the team’s pay rate, length of the project, and licensing costs associated with development. Mac can be more expensive than other platforms due to the cost of Apple products and need for specialized software.  Time required will depend on the complexity of the application and number of clients to be supported. Using tools like Electron and React Native can reduce development time.  The expertise required to support multiple clients on Mac include a knowledge of languages like Swift or Objective-C. The developer should also be familiar with the Mac OS and tools like Xcode. | Cost will depend on the team’s pay rate, length of the project, and licensing costs associated with development. However, development on Linux is generally less expensive since the Platform and its associated tools are mostly open source and freely available.  Like Mac, time is heavily dependent on the task at hand. However, time can be reduced by using cross-platform frameworks.  Expertise in Linux development includes languages like C, C++, or Python. Experience in tools like GCC can be useful as well. | Cost to develop on Windows devices are generally middle of the road. There may be specialized software license that would be required. However, the hardware is normally less expensive than Apple products.  Time requirements will be very similar the that of mac since the user friendliness of Windows is about the same as Mac.  Windows development will require expertise in special languages like C# as well as familiarity with the operating system itself. | The requirement to test on multiple devices and operating systems through a mobile device can cause the cost to be quite a bit more than using a stationary system. The licensing and hardware necessary can be quite expensive as well.  Using cross platform frameworks like flutter can reduce time, but developing on a mobile device will always take more time than using a traditional computer.  Expertise in languages like Swift and Kotlin will be necessary as well as knowledge in platforms like Xcode and Android Studio. |
| **Development Tools** | Programming Languages available to build web applications for Mac: JavaScript, HTML, CSS, PHP, and Ruby  IDEs:  Visual Studio (Code), Sublime Text, Atom, Xcode  Other tools: Git, Node.js, Apache, and MySQL. | Programming Languages available to build web applications for Linux: JavaScript, HTML, CSS, PHP, Python, and Ruby  IDEs:  Visual Studio (Code), Sublime Text, Atom, and Eclipse  Other tools: Git, Apache, Nginx, and MySQL. | Programming Languages available to build web applications for Windows: JavaScript, HTML, CSS, PHP, Python, and C#  IDEs:  Visual Studio (Code), Sublime Text, Atom, and Eclipse  Other tools: Git, IIS, and SQL Server. | Programming Languages available to build web applications for mobile devices: JavaScript, HTML, CSS, Java, Kotlin, Swift, and Objective-C  IDEs:  Visual Studio (Code), Android Studio, Xcode, and Eclipse  Other tools: Git, React native, Flutter, and Iconic. |

**Recommendations**

1. **Operating Platform**:

The three major operating systems available are all fully compatible with expanding a web-based game. While not as popular as Mac or Windows, Linux is generally favored as an operating system by tech savvy developers. However, Windows is used by many people. If the budget is available, any operating system can be used as long as it’s capable of running a cross platform game engine. Windows is the recommended operating system due to its availability and widespread use.

1. **Operating Systems Architectures**:

Windows application programming interface provides a set of functions and services that allow developers to create applications that run on Windows. Web-based games can be developed in this API using web technologies such as HTML5 or JavaScript. Microsoft has also created a framework called .NET which provides libraries and tools for development on Windows devices. Another set of APIs available for development on Windows is known as DirectX. This is commonly used for the development of graphics and audio in video games. Finally, the Windows store is a built-in platform where games can be sold and distributed.

1. **Storage Management**:

There are many options for storage management involving web-based applications. The recommended option is the relational database management system. The benefits of this system include scalability. RDBMSs can handle large volumes of data and can grow with the increase of storage needs. This system also ensures data integrity by enforcing consistency and rules for data. These rules include primary keys and foreign keys. RDBMSs can improve the performance of the application by optimizing query execution and indexing.

1. **Memory Management**:

Windows uses virtual memory to give the game application the impression that it has more memory than it does. This technique enables the game application to use more memory than what is physically available. When the memory runs out, the operating system swaps some of the data in memory to disk to free up physical memory. The swap file, also known as a pagefile, is a reserved section on the hard drive that Windows uses to store data that does not fit into the physical memory. Windows uses memory caching to store frequently accessed data in memory. When the game application requires data that has been previously accessed, it retrieves the data from memory instead of the hard disk, which speeds up the application's performance. Windows may use memory compression to optimize memory usage. This technique compresses memory pages that are not being actively used and stores them in memory. When the game application requires compressed data, Windows decompresses it and makes it available to the game application. Windows prioritizes memory allocation to the game application. This technique ensures that the game application has access to the necessary memory resources to render the images over time.

1. **Distributed Systems and Networks**:

A common approach for web-based games is to use a client-server architecture. The game runs on a server, and the clients connect to the server to play the game. The server can be located anywhere in the world and can handle multiple connections simultaneously. The server manages the game's state and communicates with the clients to send game data and receive user input. The clients can be running on various platforms, including desktops, laptops, or mobile devices.

To ensure that the server can handle multiple connections simultaneously, load balancing can be used. Load balancing distributes the workload across multiple servers to improve the system's performance and availability. This technique helps to avoid bottlenecks that can occur when a single server is overloaded. To communicate between the different components of the distributed system, various network protocols can be used. For example, TCP/IP can be used to transfer data between the client and the server, while HTTP can be used to request and deliver web pages.

1. **Security**:

Users should be required to use strong passwords that are difficult to guess. The game should recommend users to not reuse passwords across multiple platforms. Windows provides various password protection features, such as password complexity requirements and account lockout policies, to help protect user passwords. Encryption can be used to protect user data, such as files and emails, on the device and while transmitting data over the network. Windows provides various encryption technologies, such as BitLocker and Windows Defender Device Encryption, to encrypt data on the device, and Transport Layer Security (TLS) to encrypt data transmitted over the network. Two-factor authentication can also be used to enhance user account security. Windows does provide built-in antivirus and antimalware features, such as Windows Security, to help protect user devices. The game should provide available education about security best practices, such as not clicking on suspicious links or downloading unknown software.