

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 | 1/20/2025 | Michael Duke | Made adjustments to the cover page, Document Revision History table, Design constraints, Domain Model and Headers. |
| 1.1 | 2/9/2025 | Michael Duke | Made adjustments to evaluation portion of template |

**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 client, The Gaming Room, has developed a game called “Draw It or Lose It”, which is based on a 1980s television game. The game is currently only available on an Andriod app. The client is looking to develop a web-based game that can be used on multiple platforms based on the current game. The rules of the game are to have multiple teams, with unique names, compete in 4 1-minute rounds to attempt to guess a puzzle before the timer expires. Over 30 seconds, a drawing is rendered at a stready rate, if the team is unable to guess the puzzle within time, the remaining teams can offer one guess each to attempt to solve the puzzle, they are each given 15 seconds to submit their guess.

## [Design Constraints](#_2et92p0)

* Must allow for multiple teams with unique names, consisting of potentially multiple players.
* It must be able to run on the existing platform but also a web-based version of the game, that is compatible on different devices.
* Only one instance of the game can exist at any given time.
* The timer for each round must display accurately and execute the next portion of the game at the appropriate time and execute in the correct order, concluding the game after 4 1-minute rounds.
* If the guessing team is unable to guess the puzzle within time, the other teams must be given 15 seconds each to submit one guess for the puzzle.

These design constraints will have to be met by the developers so the game can run properly.

## [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)

This UML diagram uses seven classes. Classes Game, Team, and Player have a generalization relationship that references the same general class Entity, this example of multiple inheritances. These three classes, along with the GameService class, share a direct association and multiplicity, shown in the curly brackets that say our classes might share zero to many objects. Multiple players can be added to a team, each identified with an id and name. Multiple teams can be added to a game, each also identified with a name. The GamerService class has a list of games and also the singleton method called to service. This class also has references from the game class. We also have a class SingletonTester that is testing if a single occurrence of the game is running at a time associated with the ProgramDriver class. Within the ProgramDriver class, the main method is stored for the terms of use.

**"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.**

* ProgramDriver: The purpose of this class is to serve as a starting point for the program, it contains the main() function.
* SingletonTester: The purpose of this class is to test for the Singleton within GaneService. Relates directly to the ProgramDriver.
* Entity: This class serves as the parent class for the Game, Team and Player classes, and serves as a getter for the id and name. It is a base class introduced to hold common attributes, as well as, behaviors.
* Game: The purpose of this class is to manage the game, it contains the list of teams and can manipulate the team data as needed. Has a zero to many relationship with the Team Class.
* Team: The purpose of this class is similar to the Game Class, except it contains the list of players and can manipulate the player data as needed. Has a zero to many relationship with the Player class.
* Player: Stores the players ID/name to be used by the parent classes as needed.
* GameService: The purpose of this class is for managing the games operations. The class manages the Game, Team, and Player classes while utilizing a Singleton pattern to ensure that only one instance of GameService exists at any given time. It has a relationship to each of the previously listed classes, connected by a series of zero to many relationships from the classes, Game, to Team, to Player.

## [Evaluation](#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Some of the advantages of Mac, when compared to other operating systems is that it has strong security due to its smaller user base. Additionally, it is highly compatible with Apple products which may encourage apple product fans to use this ios. Mac is also known for being a very stable operating system and for being user friendly. The largest con of the Mac ios is that the servers are more expensive compared to other ios. Another con would be the limited amount of products this ios is available for, since it is only available for apple products. | One notable advantage of Linux when compared to other ios system is it’s cost effectiveness. The system also offers high flexibility for its software and fairly strong server security. It also has an open-sourced nature, which allows for additional customization. Some cons include, the fact it is more difficult to learn than other ios, and the potential of incompatibility issues. | Windows offers a user-friendly interface with high compatibility with other third-party applications. Additionally, it is a very popular ios therefore it has much higher potential in terms of management capabilities. The largest con seems to be its security vulnerabilities due to its high popularity. It also has a higher potential for licensing costs and the fact that its resource-intensive. | The mobile devices ios offers improved performance by offloading its processing to the server, it also handles sensitive data on the backend which enhances the security of the operating system and its servers, as well as allowing more user protections, it also allows for more flexibility to allow for more diverse content updates. One clear con is that the servers can take larger data loads due to its over reliance on the servers processing of data. It also is more complex in development since it has extensive backend logic requirements. |
| **Client Side** | One major advantage of the Mac system is its user-friendly interface. Also, as previously stated, it offers strong performance power and it’s compatibility with Apple devices. It also is considered to be reliable which offers users and potential users comfort of use. Again, the largest con is the systems costs compared to other PCs. It also has limited hardware upgradeability when compared to PCs, and it has a limited range of compatible software for things like gaming. | Just like on the server side, some pros include, it’s open sourced nature and customization options. It also offers security features for its user. Again, one large con is the learning curve associated with this system. It also has incompatibility issues, as previously stated, with less compatible software when compared to Windows or Mac. | One pro, is that the applications can run directly from the Windows computer, opposed to using a remote server. This allows for faster responsiveness from the application and provides less stress for the systems servers. However, this comes with the con of potential security risks and will require the user to have hardware that supports the extra processing requirements. | Since processing of data is done directly on the device, it allows for faster responsiveness and offers immediate user interaction. It also has offline capabilities and access to multiple device features. However, one major con is the fact user data can be stored directly on the device which jeopardizes the security of the data. Malicious code and potential for viruses is also a constant concern. |
| **Development Tools** | Mac’s development tools primarily focus around Xcode, which offers significant advantages for operating system app development, a user-friendly interface, and offers powerful debugging features. Some cons include, the large learning curve for people unfamiliar with Mac systems or produces. It also has limited cross platform capabilities and could result in performance issues for larger projects and applications. | Linux is an open-source operating system. It allows high customization and is a stable system. It also offers powerful server-side development capabilities. Like other systems, there is a sizable learning curve. It also has a sizable developer community and offers a command line interface. | Windows offers extensive libraries, strong integration with windows products and systems, has a large developer community. Some cons include, platform dependance, potentially higher learning curve for non-microsoft environments, and licensing costs. | Mobile device development tools offer extensive cross compatibility, code reusability and faster development time. It also allows app developers to launch apps more quickly. Some cons include, limited access for native features, limited performance capabilities, access limitations and compatabilty issues, for complex apps especially. |

## 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 the operating platform, I would recommend Microsoft. Microsoft offers many features that make it an easy choice for our application. Primarily, it has an extensive cloud storage system that uses OneDrive. This will assist in things like version updates, as well as file sharing for internal purposes. Additionally, Microsoft offers extensive cross-compatibility thanks to their .NET MAUI framework. This allows for a single code base for the application so that it can run on various platforms like macOS and Android. Additionally, Microsoft offers a user-friendly interface and top-of-the-line security features as well.

1. **Operating Systems Architectures**:

Microsoft’s operating systems architecture is structured in layers. It contains a Kernel mode, User mode and the network architecture. Windows can also have various environment subsystems, which can utilize multiple API sets. It was designed with the intent of supporting applications that were written for multiple operating systems. It is also worth noting that the User mode does not allow direct access to any of the hardware. Access to hardware functions is possible by calling into the Kernel mode routines.

1. **Storage Management**:

Regarding storage management, Microsoft’s “Storage Sense” offers several benefits. First, it offers effective disk usage by offering a build-in disk management that features a user-friendly interface to help manage partitions, volumes, and storage pools. This removes the need for any third-party software. Additionally, it offers data protection components, including storage spaces. It also has the ability to manage storage of multiple drivers. Essentially, Microsoft offers a cost-effective storage management system.

1. **Memory Management**:

Microsoft memory management offers many capabilities including keeping track of available memory space, determining when to switch between RAM and the hard drive to optimize the performance, and allocating some of the memory to run processes. Additionally, it utilizes virtual memory by using part of the hard drive as extra memory when RAM is full.

1. **Distributed Systems and Networks**.

Thanks to Microsoft’s operating system, communication between various platforms is easy. As previously mentioned, Microsoft uses the .NET MAUI framework which allows for a single code base to be used to run an application on multiple platforms, including macOS, IOS and Android. Since the service would be provided by Microsoft, things like outages would then be handled by them, while this could create risk when considering other operating systems, Microsoft is a well established and reliable company so outages shouldn’t be a major concern. However, the Kernal and User mode also assists us in keeping connected, yet secure. Speaking of connecting, Microsoft offers other services like Microsoft team to keep everyone connected with access to a centralized space. This is yet another way Microsoft can be used to cut out third-party software and help promote cost-effectiveness.

1. **Security**:

Security is essential regardless of what operating system is being used. Fortunately, Microsoft offers multiple security tools including Microsoft Defender Antivirus, Microsoft Sentinel, Microsoft Azure Key Vault, Microsoft SmartScreen and others. I would personally recommend a multi-factor authentication system; these are becoming more and more common and offer an effective and relatively simply way to help ensure user and administrative security. Additionally, I would suggest setting up conditional access policies, which allows us to set rules for users based on their location, device type, and user status. Data encryption is also an important and effective way to protect data in the event of theft or loss. For this, I would recommend BitLocker Encryption. Windows Firewall is used to filter incoming and outgoing traffic to the network, this helps to prevent unauthorized access. Naturally, I think we should use Microsoft Defender as well since it can monitor cloud applications for various security risks and flag them for our attention, the software also has the authority to enforce their policies as necessary.