

Win It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115647140)

[Table of Contents 2](#_Toc115647141)

[Document Revision History 2](#_Toc115647142)

[Executive Summary 3](#_Toc115647143)

[Design Constraints 3](#_Toc115647144)

[System Architecture View 3](#_Toc115647145)

[Domain Model 3](#_Toc115647146)

[Evaluation 4](#_Toc115647147)

[Recommendations 6](#_Toc115647148)

[References 6](#_Toc115647149)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/18/22 | Brittany Elderts-Ajala | Initial Creation |
| 1.1 | 10/02/22 | Brittany Elderts-Ajala | Evaluation Updates |
| 1.2 | 10/16/22 | Brittany Elderts-Ajala | Recommendations Updates |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room wants to develop a multi-platform, web-based game based on their current game, Draw It or Lose It. Draw It or Lose it is currently only available in an Android app. The game consists of four, one-minute rounds where the application will render an image from a large library of stock drawings. Drawings are rendered at a steady rate and are fully complete at 30 seconds. Opposing teams will have a chance to offer a guess if the current team does not guess the answer before the drawing is complete. Opposing teams will have a 15-second time limit. The game must have the ability to have one or more teams involved and each team will have multiple players assigned to it. Game and team names must be unique, and users must be able to check if a name is in use when choosing team names. A unique identifier must be created for each instance of a game, team, or player in order to ensure only one instance of the game can exist in memory at any given time. The staff at The Gaming Room are unfamiliar with the programming environment and will need help streamlining the development of the game application.

## [Design Constraints](#_2et92p0)

A programming language suitable for multiple operating systems is necessary in order to ensure cross-platform functionality.

The web-based gaming application must be based on the currently available Android application version of Draw It or Lose It.

The staff at The Gaming Room are unfamiliar with the web-based programming environment and will need direction regarding development and maintenance.

The game must have the ability to have one or more teams with the ability to have multiple players assigned to each.

A unique identifier must be created for each instance of a game, team or player in order to ensure that only one instance of the game can exist in memory at any given time.

Game and team names must be unique and users must be able to check whether a name is available when choosing team names.

## [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 class diagram below illustrates key object-oriented programming principles by creating concise objects, encapsulation to protect, and purposeful inheritance. The diagram depicts encapsulation by maintaining private data while allowing public access to methods. The diagram depicts abstraction by breaking down the larger category of “Entity” into the categories of “Game”, “Team, and “Player”. Because these have been separated, we can make changes to the “Game” class without affecting the “Team” and “Player” classes or the broader class of “Entity”. The diagram illustrates inheritance because the “Game”, “Team”, and “Player” classes all contain the variables and methods of the “Entity” class as well as those defined in their own classes. Polymorphism is displayed here due to the fact that the “Game”, “Team”, and “Player” classes are all forms of the “Entity” Class, even though they are different classes with different characteristics. The fact that these classes are all split with separate variables means the creation of objects within those classes can be controlled and limited to one unique instance.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac is an excellent choice for hosting a web-based application. It is highly secure due to limited hardware, complex code, and proprietary software. (Long, 2020) However, the required Apple hardware and licensing makes Mac hosting the most expensive option. | Linux is the cheapest option since the software is free. Linux servers can be run using older, low-end servers and don’t require any special hardware. Linux is an open-source OS, but you can use additional security measures to make it more secure. | Windows also requires high-end hardware in order to maintain reliable service. Software must also be updated every release and this adds to the cost. Microsoft also has a variety of proprietary web applications available. Since Windows is the most widely used, it is often targeted by hackers. | While mobile devices are capable of being used to host web-applications as a server, they lack the ability to maintain high-end, reliable service. A better option would be cloud-based hosting. |
| **Client Side** | The main consideration for Mac is the required Apple hardware and coding expertise. This takes additional time and funding. | Linux is lacking in high-end software and security, but it is inexpensive and widely used. | Windows does require high-end equipment for optimal performance and frequent updates of software. | Since mobile applications are fundamentally different than web-based applications, it is necessary to find a developer with experience. |
| **Development Tools** | Mac uses Swift for building applications. Swift is easy to use and open-source and can be used with the XCode, CLion, AppCode, etc. IDEs. | Most development on Linux uses C, C++, Perl, or Python. The IDEs that can be used include Eclipse, CLion, NetBeans and others. | C, C#, .NET, JavaScript, etc can all be used in application development. Visual Studio can be used for these languages and others. | Languages that can be used on iOS and Android operating systems would be a good choice, like Objective-C, Swift, and Java. But for a cross-platform mobile app, it might be better to use HTML5, CSS3, or JavaScript. The QT, Jetbrains Rider, Emacs and Visual Studio are all good IDE options. |

## Recommendations

1. **Operating Platform**: While Linux, Windows, and Apple are all good options, Windows is probably the best choice. Windows is consistently maintained and updated and is the most popular operating system available. Apple is also consistently maintained and updated and has built in security options, but it is very expensive. Linux is free-to-use, but is not maintained or updated.
2. **Operating Systems Architectures**: Windows has a layered architecture design, a user mode and a kernel mode. This provides a layer of protection because user mode stipulates limitations in what the user can do, like the user cannot directly access the hardware or reference memory in user mode. This protects the system during crashes and can usually be restored in user mode.
3. **Storage Management**: The best storage management option for the Gaming Room would be cloud storage. This provides fast download delivery to customers and the least investment expenses for the company. This also allows for real-time updates to the game itself and stored user information.
4. **Memory Management**: The Windows operating system must be able to determine the optimal balance between allocating long-term storage and random-access memory (RAM). Virtual memory management or heap memory management can be used to organize and streamline data retrieval and maintain high processing speeds. A small number of images would need to be stored in RAM, but the other images could be stored in the hard drive until needed, decreasing the necessary amount of RAM memory needed for the application to run efficiently.
5. **Distributed Systems and Networks**: Again, cloud-based hosting is the best way to ensure that the necessary hardware is available for multiple platforms. These services protect against downed servers and power outages by rerouting to other local stations to prevent user interruptions.
6. **Security**: Most cloud-based hosting have basic security options available, but additional security can be integrated to protect user information and prohibit unauthorized access. Role-based access, two-factor authentication and firewalls are all viable security options

## References

Long, R. (2020, Dec 13). Windows Hosting VS Linux Hosting. OmegaWeb. https://www.omegaweb.com/windows-v-s-linux-v-s-mac-web-hosting/