

<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 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0  1.1  1.2 | 05/25/2024  06/08/2024  06/23/2024 | Liel Simon | Design document filled out. Executive summary, Requirements, Design Constraints, and UML Diagram description completed.  Development requirements of Mac, Linux, Windows, and mobile devices added including benefits and drawbacks of the various platforms.  Operating Platform, Operating Systems Architectures, Storage Management, Memory Management, Distributed Systems and Networks, and Security recommendations added. |

**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 needs help developing a web-based version of their gaming application. The game “Draw It or Lose It” has competing teams guess what is being drawn. The application will render the images from it’s library of stock drawings. Each game will have 4 rounds and last 1 minute each round where the drawings and steadily rendered until completion at 30 seconds. If a team is unable to guess the puzzle the remaining teams can offer one guess within 15 seconds.

## Requirements

Game must include 1 or many teams.

Each team can have multiple players

Team names are unique

Game is unique and only 1 instance of the game can exist in memory at a time

Must support a library of stock images

## [Design Constraints](#_2et92p0)

Game, team, and player must have unique identifiers. This means the classes will need to use the singleton pattern when coded. This will also allow team name to check if a name is already in use. Iterator pattern in addTeam and addPlayer methods must be used.

Entity base class must be created with private variables id, name and methods -Entity, +Entity, getId, getName, toString.

## [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)

Based on the UML diagram, Entity base class is to be added. Entity utilizes abstraction for its attributes. Those attributes are private variables id: long, and name: String with private method Entity, and public method Entity, getId and getName to initialize the entities id and name. Public method toString to return the entities id and name. This is an example of encapsulation where variables and methods are grouped together. Inheritance is also featured in the code where Game, Team, and Player will inherit from Entity, namely id and name.

Game will store a list of teams, initialize a Game with id and name, include a method to add a team or teams to the game and check for uniqueness. Game will also have a toString method to return game with id and name.

The Team class while similar to the Game class will create a list to hold the players. A constructor to initialize team and id. A method to add players to a team which utilizes abstraction so no other class can add players. As well as a toString that returns the Team with id and name included.

**"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)

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** | Well known for its stability, security, being user friendly, and excellent development tools. High cost and low flexibility. | Due to linux’s open-source nature linux is generally has a lower operating cost. Highly customizable and know for its security and stability when used in servers. High complexity can be difficult for users. | Commonly used with by normal users and enterprise environments. Excellent support and highly compatible. High licensing cost and resource demanding. Thought to be less secure then other platforms. | Allows access from anywhere. A very large user base. iOS and Android being the 2 most popular mobile device OS’s will need to be developed separately. High performance variance between different devices. Easily affected by network disruptions. |
| **Client Side** | As with most Apple products MacOS devices incur a large cost on the user. Mac App Store charges developers an annual subscription fee. Apple has a “Human Interface Guidelines” that are meant to be followed so users will feel comfortable with the user interface. | Linux is free and open source making the cost to the user only come from the hardware itself. Highly customizable can be great for some users and overwhelming for others. Support for multiple distributions may need to be considered which will include development time and testing time. Common and popular programming languages like C++ or Python are used allowing familiarity for developers. | High licensing cost but generally flexible hardware costs. In order to publish an application to Microsoft Store a developer account is required which has a fee. Many different hardware configurations and Windows OS’s in use could demand extensive testing to ensure incompatibilities do not arise. Visual studio fluency and knowledge of the Windows development ecosystem are important. | Low cost for mobile OS development as Xcode for iOS and Android Studio for Android devices is free. Apple App Store requires a subscription for publishers and Google Play Store has a one time fee. With so many different phones with different hardware capabilities, OS’s, and screen size. This can introduce large development and testing phases. Different programming languages will be necessary for different OS’s of the mobile devices. |
| **Development Tools** | MacOS primary programming languages include Apple’s in-house languages Swift and Objectice-C. Python, C++, C, and java are also commonly used. The IDE Xcode can be used for Swift and Objective-C | Common programming languages for Linux include C++, C, Python, Java, and even GNU Bash which offers shell scripting. Linux offers a wide variety of IDE’s for developers to choose from. This includes, Visual Studio, Eclipse, IntelliJ, NetBeans. With most of these supporting C, C++, Java, and Python. | Some common programming languages for Windows include C++, C#, JavaScript, TypeScript, Visual Basic, Python and many more showing that developing apps for Windows offers a large range of choices to choose from. Windows also supports many IDE’s including personal favorites from JetBrains such as IntelliJ IDEA, PyCharm, Rider and more. Visual Studio is also one of the most popular IDE’s for Windows software development. | With mobile devices having different operating systems development for all intended platforms may require separate tools. iOS primary programming languages will be Swift and Objective-C which could use the IDE Xcode. Android programming languages include Java, Kotlin, and C. Android Studio being the official IDE for Android app development. However, Eclipse, Visual Studio and others can be used. Cross-Platform development may be possible with C# and the IDE Xamarin, as well as JavaScript or TypeScript using the IDE React Native. |

## 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**:

Using Linux for server-side hosting would offer many benefits. If Linux were to be used it would likely be the most cost-effective choice due to lower resource requirements and no licensing cost. Linux is also highly flexible allowing many different tools and languages to be used. Linux also allows for scalability if needed. This will also allow cross-platform for client-side development.

1. **Operating Systems Architectures**:

Cross-platform development may be possible if using tools like React Native, Flutter, Kotlin Multiplatform, Ionic, .NET MAUI, and NativeScript. Doing so would cut down on development cost, time and maintence. This would also allow for simultaneous releases allowing Draw it or Lose it to be available to the most people quickly.

1. **Storage Management**:

Using a cloud-based storage solution may be a good option. Doing so would solve storage management needs quickly. Cloud-based solutions are highly scalable and reliable with many options to choose from. While a cloud-based solution may have a lower startup cost it may end up costing more in the long term then using an in-house Linux server for storage management.

1. **Memory Management**:

A cloud-based solution may also be useful here for memory management. It would again offer great scalability that could be achieved relatively quickly should the need occur in the case of a sudden large influx of users. The upfront cost of a cloud-based solution is initially lower than an in-house server setup. Cloud providers are generally reliable and may offer a better experience for users since cloud providers usually have multiple cloud servers within a region reducing user latency. A cloud solution would also allow for a high standard of security. However, the cost over time should also be a consideration.

1. **Distributed Systems and Networks**:

Using a distributed system will allow for increased reliability and communication. While it could also improve computation speedup and resource sharing it is my opinion that the reliability and communication benefits would be more important for this game. If one site were to fail within a distributed system, other sites would be able to take the place of the failed site. This could prevent loss of connection to the game and loss of game data. People would still be able to sign into and start game sessions even if one site were to go offline. The improved communication would also be essential. It would allow users from across the globe to play with each other. It could also reduce latency between the players allowing for a much smoother experience while playing together.

When using a distributed system, it also allows for much easier repairs and maintenance of sites. This is because a site can be taken down without causing complete failure of the network. Due to the simple nature of Draw it or Lose it (not graphically intensive, currently no voip, no heavy calculations, extensive scripting, or game logic) I do not foresee a need for computation speedup, load sharing, and process migration feature integration within the distributed system.

1. **Security**:

It is essential to protect user information regardless of platform. As with most implementations of security we can begin with authentication. To protect user information all users must be required to have a password attached to their username. Multi-Factor Authentication can be an optional feature should the user want such protection. Passwords should also include the standard requirements such as requiring a special character, a minimum of 8 characters, and a temporary lockout should a user enter the wrong password too many times to prevent brute forcing of a user’s password.

Access Control Lists (ACLs) should also be utilized to manage permissions, ensuring that users only have access to the resources they need. This should remain in line with the concepts of the principle of least privilege where users should only have access to what is essential to them in order to properly use the program. For data protection, the application should leverage the operating system’s encrypting services for both data in transit and at rest. Virtual Private Networks can also be utilized to protect users during sessions. This will help keep data secure by passing the data through an encrypted “tunnel” which would remain unreadable even if intercepted by a malicious actor. An intrusion detection system (IDS) can also be created to automatically identify suspicious network activity. When an alert is detected, the IDS can be configured to either alert an administrator and/or automatically block the threat.