

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 3.0

## Table of Contents

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

[Table of Contents](#bookmark1)**2**

[Document Revision History](#bookmark2)**2**

[Executive Summary](#bookmark3)**3**

[Requirements](#bookmark4)**3**

[Design Constraints](#bookmark5)**3**

[System Architecture View](#bookmark6)**3**

[Domain Model](#bookmark7)**2-3**

[Evaluation](#bookmark8)**4-6**

[Recommendations](#bookmark9)**6-7**

## [Document Revision History](#bookmark10)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | <01/29/23> | Kenneth Wilkerson | Developed initial software design report. |
| 2.0 | <02/05/23> | Kenneth Wilkerson | Updated system architecture and development requirements evaluation section. |
| 3.0 | <02/19/23> | Kenneth Wilkerson | Updated recommendations section. |

## [Executive Summary](#bookmark11)

Our Client, The Gaming Room has tasked us with updating their current Android app game *Draw* *It* *or* *Lose* *It*. *Draw* *It* *or* *Lose* *It* puts a modern spin on the classic 80’s TV game show called *Win, Lose or Draw*. This is a team oriented game and the object is for your team to score the most points by correctly guessing a series of puzzles through using visual clues. The game features 4 one minute rounds for added excitement. Our update of the current form of this game involves creating a web-based version of this game so that it serves a greater customer base across multiple operating systems/platforms.

## Requirements

The Gaming Room has four software requirements that need to be met through our project:

1) The game has the capacity to include one or more teams

2) A team will be composed of multiple players

3) Names for games and teams must not already exist. Also, users should be able to check to see if their chosen name is unique and available.

4) Only one instance of the game can be saved at a time. Each game, team and player should have unique identifiers

## [Design Constraints](#bookmark12)

The following constraints could impact the overall project:

1) Team Maximum Capacity- we will need to set a limit in regards to the maximum number of players that can exist on a team. This decision will impact how methods and functions are defined in our program. This could also increase or decrease the total time playing the game. If the cap is too low, we could end up with too many puzzle rounds due to having more teams which could decrease user satisfaction. On the other hand, if the cap is too high, the game could move too fast because there are less teams which could also potentially have negative impacts on the user experience.

2) Copyright Issues- Since this game will be using a large library of stock drawings as clues, it will be important to avoid potential lawsuits due to violating copyright law. Only images that fall under the fair use policy should be used unless we receive permission from the copyright holder. This will protect The Gaming Room’s overall revenue and their reputation.

3)Compatibility - Draw It or Lose It already exists as an Android app. We will need to ensure that our web based game can serve most operating systems. This will involve creating software that is compatible with a variety of internet browsers such as Google Chrome or Firefox.

## [System Architecture View](#bookmark13)

The game app utilizes a client-server architectural pattern. On the server side, user data and game data is stored. User data includes team, player and game ID numbers. Game data includes the game instance that currently exists in memory. The program utilizes the singleton design which only allows for one game instance to exist at a time. Game data also involves picture files that are used as clues in the game. The server side of the game utilizes REST API principles that ensures the portability and maintainability of the application. Authentication measures are in place to validate user information to create a more secure application. The client side of the application is designed to be playable across operating systems such as Windows, MacOS and Linux. Internet browsers communicate with the server in order to gain access to the game and enjoy the gameplay.

## [Domain Model](#bookmark14)

The Entity class is a base class that is extended by the Game, Team and Player classes. This is inheritance because there is a parent class and multiple child classes. This allows for the child classes to write fields, methods and functions that are relevant to the subclass. One thing that all four classes have in common is the toString method. This represents polymorphism because one method takes on many forms depending on the class being used. The Game, Team and Player classes override the toString method in order to create unique method outputs for their class. In the GameService, Entity, Game, and Team classes, they all have private fields. This represents encapsulation because certain data is protected from the outward interactions of the program. There is a zero to many relationship that exists between GameService-Game, Game-Team and Team-Player classes. The ProgramDriver class is where main() is and therefore the core functionality of the program. The ProgramDriver class uses the SingletonTester class to test for the existence of only one object in memory.

**"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](#bookmark15)

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** | Has exclusive software in a similar way Windows does. More secure than other operating systems. Very user friendly. New hardware has M1 or M2 chip which has strong data protection mechanisms such as automatic encryption of personal data. Licensing costs may be higher due to using external server operations. | Open source operating system.Compatible with common web browsers such as Firefox and Chrome. Licensing costs will be lowest with this operating system. Licensing costs could be high due to using external server operations. | Popular and well-known operating system. They have a monopoly on some types of software. Large infrastructure capacity which would makes a stable host. Windows has a built in firewall and antivirus software. Potential licensing costs with Microsoft cloud servers. Higher user base potential | Less memory, graphics and processing capacity than with PCs. Storage will be more limited and costly to host a server. Could be more prone to security attacks. High licensing costs to accounts for the variety of potential mobile devices. |
| **Client Side** | MacOS has a clean and simple UI.The cost on average will be more because the pricing of Apple hardware is more high-end. The expertise needed is moderate because it is a common and well-known operating system. There is a lot of compatibility due to the variety of web browsers such as Google Chrome or Safari that can be used. More support options than other systems. Application will need to use a language like Java that is portable. | There is a high skill cap as far as working with Linux. Since the software is open source, there is a greater responsibility to fix bugs or make improvements. This would lead to more time spent developing. OS is free to use however the cost is higher because of time investment needed to problem solve.  Application will need to use a language like Java that is portable. | Moderate expertise and cost (Windows pricing is mid-high end). Well-known operating system and user friendly. High compatibility with a variety of web browsers such as Google Chrome. Microsoft has their own browser with Microsoft Edge that could be used. More support options offered than other systems.  Application will need to use a language like Java that is portable. | Some webpages have limited functionality on mobile devices. High cost due to graphic and memory demands. Moderate expertise. Touch screen and non touch screen devices and difference in programming controls. More limited with types of internet browsers than can be used. Application will need to use a language like Java that is portable. |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Development Tools** | Relevant IDE’s include Visual Studios and Eclipse. Programming languages such as Javascript, C++, Python or Java. Two developer team should be deployed, one for front-end and one for back-end. Additional licensing costs will be added if not using open source IDE. | Relavant IDE’s include PyCharm, Eclipse, and Visual Studios. Programming languages such as Javascript, C++, Python or Java. Two developer teams should be deployed. One for front-end and one for back-end. Additional costs may be added if not using an open source IDE. | Relavant IDE’s include PyCharm, Eclipse, and Visual Studios. Programming languages such as Javascript, C++, Python or Java. Two developer teams should be deployed. One for front-end and one for back-end. Additional costs will be added if not using an open source IDE. | Relavant IDE’s include Visual Studio and Android Studio. Programming languages such as Javascript, C++, or Java. Two developer teams should be deployed. One for front-end and one for back-end. Additional costs may be added if not using an open source development tool. |

## 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**: I would highly recommend Windows due to the variety of web browsers and IDE’s that are available on this platform. Historically, this operating platform is the most conducive to gaming and has a proven track record. Windows offers a more stable and consistent user experience which will be important when it comes to increasing the user base of the application. They are also the most well-known and popular operating system. It would be beneficial to leverage these strengths for the Draw It or Lose It application.
2. **Operating Systems Architectures**: The Windows API supplies services to the Draw It or Lose It application. Windows Service Manager shows all of the critical services that the system uses. Windows uses threads and processes. A process is a program that contains one or more threads. Threads are what execute the code. Applications can be controlled through the Control Panel folder. Windows has a wide range of web browsers that are shared with other operating systems. I would recommend using Google Chrome as the primary browser. This would lead *Draw It or Lose It* to have a greater reach.
3. **Storage Management**: In order to ensure more stable connectivity, I recommend having multiple servers in different regions so that the application can serve a global user base. There is also potential in the future for cloud gaming through Azure. On the server side, I would recommend using linked allocation in order to avoid external fragmentation. It will be important to maximize storage space and storage usage as the game continues to grow.
4. **Memory Management**: Windows has 32-bit and 64-bit capabilities. I would recommend using 64-bit due to increased RAM capacity. The application will need RAM in order to move game data from storage so that it can be quickly and more easily used during a game instance called on from the client side. One important feature of Windows memory management is its usage of virtual addresses. This operating system supports good frame rates while reducing lag. Both the server and client benefit in terms of memory management through using Windows.
5. **Distributed Systems and Networks**: In order to create a web based version of Draw It or Lose It, I recommend using a client/server design through using REST API. The benefits include having a uniform interface that identifies the resources that are used on both sides. Furthermore, the interface and data are separate which makes the game portable. Also, having a stateless interaction ensures that our clients will need to start a new request in order to gain access to the game. Since the system is layered, the behavior of the application will be constrained.
6. **Security**: Windows has strong security features based in their built in features. They have their own firewall and antivirus software called Windows Defender. Windows also utilizes access tokens which identifies and authenticates users. Using a RBAC system (role based access control) pairs well with this operating system. There are also access control features that ensure authorized users can access resources utilized by the application. In terms of the client/server setup, the server side will need to validate user information from the client side request in order to provide access to the application. Clients will need to provide a ID number that is associated with a player, team or game instance since only one instance of the game will exist in memory through the singleton design. Lastly, I recommend adding a two-way authentication feature for added security for the client side.