

Draw It or Lose It

# **CS 230 Project Software Design**

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 | 11/10/2021 | Joann Carter | Constraints, model, evaluation |

**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)

Draw It or Lose has the potential to be a successful game after the initial planning is done. From my understanding, this needs to be a web-based application that consists of one or more team(s) with multiple players. There will be a set of images stored and only one image will be selected and sketched to the user. Only one image will be drawn at a time and this should take less than 30 seconds to reveal. If the time is expired, the next team has 15 seconds to solve the puzzle. In order for this application to run smoothly, a few things need to be addressed. Starting with the actual game play, would you like this to run on only one computer or multiple computers at once? Second, the images need to be equal in draw time length for the reveal to work correctly. Or, you can have increasingly harder images as the game progresses. Additionally, the order in which the teams play will need to be established. After all of this is completed, there will need to be testing done to ensure 15 seconds is enough time for the other team to type in their answer. Also, do you plan to account for accessibility in this game? Accessibility will broaden your audience. Examples include color blindness and/or keyboard limitations. Lastly, a budget and time frame need to be determined for clarity on all parts of this project.

## [Design Constraints](#_2et92p0)

|  |  |
| --- | --- |
| 1. Size of game board 2. Complexity of images 3. Selection Algorithm 4. Storage | 1. User feedback 2. Budget 3. Accessibility 4. Time-line |

|  |  |
| --- | --- |
| **Constraint** | **Rationale** |
| Size of game board | If this game will be rendered on multiple computers, zoom and re-sizing of the game board need to be accounted for. This may distort the images if it is rendered on a web browser from a mobile phone. |
| Complexity of Images | In order for the game to be fair and consistent with the 30 second draw time, images of the same draw consistency need to be chosen. |
| Selection Algorithm | The selection of the images needs to be determined. Will it be random or will it go in order from simple to complex images? |
| Storage | The storage of high scores and images needs to be determined. Storing the images within the game is an option. |
| User Feedback | Testing this game with multiple age ranges and devices is a must. Leaving time to account for user feedback is important. |
| Budget | A budget needs to be determined and approved. |
| Accessibility | Color blindness, motor movement and deafness need to be kept in mind for this game to appeal to a larger audience. Using black for the images, sound and display for the timer, and an option of auto completion on the guess words can solve these problems. |
| Time-line | A time frame of completion needs to be established. Keep in mind there needs to be a phase one time frame and allocated time to test with additional time to fix any user feedback that wasn't found prior to phase one deployment. |

## [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)

Object oriented principles are found in this project and can be easily seen in the diagram below. Game, Team and Player are all subclasses that use inheritance from the superclass Entity.Polymorphism is seen by the reusable methods in the Entity class. Encapsulation is found in the private variables in Entity, GameService, Game, and Team Classes. Abstraction is also used in these classes by hiding the id from the user.

GameService uses a one to many relationship with Game, Game uses a one to many relationship with Team, and Team uses a one to many relationship with Player.

**"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.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac is the most expensive option. However, it is a more secure option than Windows. | Linux is the cheapest option because it is open source. It is also more secure because of this. Another advantage is the touch screen functionality. However, it is also has the least amount of users due to its tech-savvy environment. | Windows has the largest user base. The One drive cloud storage has good reviews and it is less expensive then Mac. Windows can also run on external  Hardware. The inconsistent software options and versions needs to be considered. | There is less storage on a phone then on a PC and the screen sizes vary tremendously. Even though most people has a mobile device, there are many inconsistencies like the screen size and service while the user is mobile. |
| **Client Side** | The best feature is the mobile integration. The accessibility could be addressed through the use of a tablet as a side car to inherit touch screen capability. | The general public is not familiar with Linux. This will limit the customer base that the game will appeal to. | Windows has the largest user base. It has full-touch screen support, voice assistance, stylus input, and touch gesture support. It is also great for gaming because of the graphics cards. | In order for mobile devices to be considered, all OS must be written. This will take the most time. Also, screen size is a huge variability that will take a lot of testing to ensure the game board is rendered correctly for the various phones out now. |
| **Development Tools** | Swift is a programming language created by apple for building MacOS programs. Xcode is a free app that can be used to build apps. It has an interface builder built in to save time and it highlights issues in the code as you go. his includes the Xcode IDE, Swift, and C/C++/Objective-C compilers  Postman can be used for API development and it provides support. It has a free version and a 15$/month option. | C is used in Linux. Monkeyshine is a top recommended code editor. It also has website builder capabilities that comes with a preview tab to view live changes to the website. | Visual Studio code and .NET are popular on Windows. C, C#, C++ are popular languages used on this OS. | Ionic is a framework that is cross-platform and easy to use. Combined with Apache Cordova, UI and native phone API’s can be created for iPhone and android apps. Jquery mobile is widely used to develop mobile apps. However, it would not include the native API’s needed for accessibility. |

## 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**: Windows is the recommended Operating Platform for this game design. Windows has a large customer base and a large assortment of accessibility features. These characteristics will ensure that the application appeals to a wide range of users. It should be noted that the attention to security features included in the updates of Windows has improved. With Draw It or Lose It’s success in mind, accessibility and security were the main deciding factors of this conclusion.
2. **Operating Systems Architectures**: The Windows NT kernel consists of a simple kernel, hardware abstraction layer, and drivers. Run-time optimization will be another focus to ensure a seamless user experience. The game and all images will be stored on the Windows storage unit. Only the user data, authentication and scores will be stored on the cloud server.
3. **Storage Management**: Windows Storage Management uses an API for C/C++ languages. The game and all the images will be stored here. All the user authentication will be validated through the data received from the connection to the server. Only the local variables and one image will be stored in memory.
4. **Memory Management**: Windows 10 comes in 32 and 64 bit. Both the 32 and 64 bit Windows assignable memory space is shared between active applications and the kernel. The default 32-bit configuration provides up to 4 GB. The 64-bit provides up to 16 TB of memory space. Only 8 MB are needed to store a single image. For run-time optimization, and to avoid fragmentation, only one image will be stored in memory at a time. Before trying to show the image, the entire image has to be loaded from the hard drive to the memory. Completing the load before trying to render the content to the user will avoid buffering.
5. **Distributed Systems and Networks**: Accessing this application across multiple platforms will need to rely on a cloud storage space for cost effectiveness. This means a reliable internet connection is imperative. The timer will start only after the picture has loaded to maintain a positive user experience. Once it is the next players turn, they will receive a notification and upon opening, the 15 second timer will start.
6. **Security**: To ensure security, multi-factor authorization and authentication will be used. Hiding the API key for an encrypted password, and using a two layer authentication for user creation are examples of what can be done. Also, the use of firewalls, validations, privileges, intrusion detection, protection kernels, and authentication from Windows will be implemented. Without this, there could be a security breach into the application. Having a securely generated game code, instead of a simple encrypted user name and password, will help protect the users’ identity.