

<Draw It or Lose It>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_heading=h.gjdgxs)1

[**Table of Contents**](#_heading=h.30j0zll)2

[**Document Revision History**](#_heading=h.3znysh7)2

[**Executive Summary**](#_heading=h.2et92p0)3

[**Design Constraints**](#_heading=h.tyjcwt)3

[**System Architecture View**](#_heading=h.3dy6vkm)3

[**Domain Model**](#_heading=h.1t3h5sf)3

[**Evaluation**](#_heading=h.2s8eyo1)3

[**Recommendations**](#_heading=h.3rdcrjn)5

## [Document Revision History](#_heading=h.3znysh7)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/21/2022 | Tyler Johnson | Second Draft |

**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](#_heading=h.2et92p0)

The client wants to create a web based game called Draw It or Lose It. The game works as follows: there will be multiple teams, one image from a larger library will start to render over a 30 second window, during that time the first team will try to guess what the image is, if they fail to do so the other team will have 15 seconds to guess before time is expired. The game already exists for android but it needs to be created for a web browser using Java. Some software requirement that were already laid out are the following: Multiple teams allowed, multiple players per team, team name and game names should be unique, only one version of the game should exist in memory at any given time

## [Design Constraints](#_heading=h.tyjcwt)

*Web Based, the client needs their application to be a web based application using java. This will limit the technologies we will be able to use and we also need to be able to run the application through different browsers and operating systems.*

*Time, there will be a time constraint on when the application needs to be done.*

## [System Architecture View](#_heading=h.3dy6vkm)

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](#_heading=h.1t3h5sf)

The UML diagram consists of an entity class that has three classes that inherit from it, game, team, player. This is a strong example of inheritance in OOP. Each of those three classes are also related, there are many players per team and many teams per game. The game services class is a singleton which will mean there is only one instance running in memory at any given time. There is also a program driver and singleton tester class, the first of which serves as the entry point into the application and second is a class to test the game services class. There is also encapsulation shown in this uml diagram as each individual class is self-contained with its own logic.

**"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](#_heading=h.2s8eyo1)

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** | Most secure option because of Apple's strong security, works well and seamlessly with a lot of commonly used technologies. GUI available, as well as command shell | Less secure option but has a strong number of options which will not come with a huge price tag. Command shell | Best used to host asp.net applications and other microsoft technologies but is an overall good option somewhat pricey with strong emphasis on being user friendly | <Evaluate Mobile Devices for their characteristics, advantages, and weaknesses for hosting a web-based software application.> |
| **Client Side** | Mac is an fairly easy option that is user friendly but is more expensive than the other options, moderate time required | Linux is the most difficult option, but the least expensive. Without knowing the platform it is the most time consuming | Middle option for price and also the easiest option. Most users and developers most familiar and comfortable with this option | Mobile is a difficult option and requires a lot of work to set up |
| **Development Tools** | For this application we can use HTML CSS javascript and java eclipse is available as well | For this application we can use HTML CSS javascript and java eclipse is available as well | For this application we can use HTML CSS javascript and java eclipse is available as well | For this application we can use HTML CSS javascript and java eclipse is available as well,there are also other options such as Swift and microsofts mobile development tools |

## 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 recommend Windows for the operating system, this is because of its strong usability and the general familiarity that comes with windows. Windows requires the least expertise and includes all the necessary things to create our application. The time and price requirements will also be lower than some of the other options.
2. **Operating Systems Architectures**: Windows offers a wide selection of available systems such as file management, memory management, security, ect. Windows is a powerful operating system that is more than capable of handling our application needs.
3. **Storage Management**: For storage management I would recommend Azure, a fairly expensive option however for added security and the benefit of it being maintained and constantly improved by microsoft is in my opinion a better option than AWS or others for this application.
4. **Memory Management**: Paging is used to retrieve information/data from a hard drive to be used in the main memory. It is useful to allow the program to exceed the size of available main memory. It is done by breaking up physical memory into frames and logical memory into pages. Then pages are then loaded into the frames to allow access. There is also swapping which is a technique that allows a process to be temporarily moved to a hard drive to free up space in the main memory.
5. **Distributed Systems and Networks**: Distributed Systems will allow us to be able to share basic resources between the different operating systems such as the images and potentially even the users as well.
6. **Security**: We will use user authentication and authorization as a basic way to secure our users information, Authentication is the ability for users to login to our application and authorization is the ability for them to access certain parts of our application based on who they are.