

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

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

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

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

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

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

[**Design Constraints 3**](#_heading=h.1t3h5sf)

[**System Architecture View 3**](#_heading=h.4d34og8)

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

[**Evaluation 4**](#_heading=h.17dp8vu)

[**Recommendations 5**](#_heading=h.26in1rg)

## [Document Revision History](#_heading=h.lnxbz9)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/21/24 | Kursheeka Milburn | Draw it or lose it is a game that allows players to choose a team where they have 30 seconds to guess the picture that is being created. If the team gets it incorrect the other teams have the opportunity to guess the image within 15 seconds. |

**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.35nkun2)

The gaming Room wants to create a web based version of the game which is only on Android app. The version we are looking to create will allow players to use other devices besides Android and will have teams competing against one another to guess the images. Software needs to be set up so it can manage the teams, players, and the game with unique names that will also be created.

## Requirements

The client’s business requirements are that the game should be able to be accessed on other devices other than Android, players should be able to team up and play against one another, and have/ create unique names. The technical requirement is that only one game is to be running at a time*.*

## [Design Constraints](#_heading=h.1ksv4uv)

The game must run on all platforms, only one instance may run at once, the game has to be able to accommodate multiple players and teams, each player, team, and game needs to have unique names.

## [System Architecture View](#_heading=h.44sinio)

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.2jxsxqh)

The diagram shows the main parts of the game which are the player- which represents individuals and keep track of their name and score, team- which is the groups of players and tracking their scores, and game- which is each game and keeping track of the number of rounds and the score. The entity class is the base which holds all of this shared information. The programDriver is where the program is executed and the Singleton Tester works with ProgramTester class to accomplish the requirements for one game to be allowed at one time only with multiple teams and players.

**"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.z337ya)

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** | User-friendly and secure but not ideal for hosting web apps. Has more software than other platforms. Can be costly and less effective than others. | Stable, customizable, and cost effective. Requires technical expertise. Has a lot of different web hosting options. | Most popular for web hosting services. Easy to use but can be more expensive. Able to run more software but also less secure than others. | Unable to host heavy apps directly but are amazing for user access and engagement. Highly compatible and cost effective but not very secure. |
| **Client Side** | Expertise and time. Development that is required can be pricey and time consuming for different client types. | Offers free tools and supports programming languages but has a steep learning curve. Low cost, open source OS. | Require the lowest time and expertise due to the market puncture of the platform. The development is straightforward with different tools but can add up. | Lowest cost platform but the development can be expensive. Easy to interface with and requires knowledge about the different platforms. |
| **Development Tools** | Great for making native Mac apps. Swift is used to write Mac apps. Git can be used for version control. | Able to use different programming languages like Python or Java. Many command line tools that may be helpful making it flexible which is why it is favored by developers. | Visual Studios can be used for IDE here with helpful features and Git to keep everything organized. This is very user friendly. | For IOS development Swift is important while for Android Java is used. Both have emulators to test the apps making it easy to see how they will actually work. |

## 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 using Windows to expand the game to other devices because a lot of people are already familiar with Windows and how it works. They are comfortable with navigating which makes it easier for players to enjoy the game.
2. **Operating Systems Architectures**: Windows has a user friendly design that helps create apps quickly and it supports different architectures. It can work on a lot of different computers which makes it flexible to reach players.
3. **Storage Management**: For storing the information for the game I would use MySQL because the database can hold players, team details, and the score information securely which makes things more efficient.
4. **Memory Management**: Everything is able to run amazingly because Windows use smart memory management techniques. Windows is able to manage many tasks at once making this important for this game. This game allows a lot of different players to play at once and this helps the responsiveness and fastness.
5. **Distributed Systems and Networks**:<Using a restful api which allows the game to remember the players if the internet crashes.
6. **Security**: Windows has really good security features and usually has regular updates to help protect and secure the data. I would recommend using SSL so everything would be encrypted.