

Draw It of Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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)**-6**

**Recommendations 6-7**

JAVA Code\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7-11

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 6/12/24 | Matthew Haught | Completed Project Software Template |
| 1.1 | 6/21/24 | Matthew Haught | Revised Evaluation portion |
| 1.2 | 6/28/24 | Matthew Haught | Revised Recommendation portion |

**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, our newest client, has approached our firm and tasked our company with the development of a web based game, Draw It or Lose it, which is cross compatible across multiple operating platforms. The goal of this project is expand the operating system from the pre-existing Android to other popular operating platforms such as Linux, Windows, and MacOS software products. The game itself is structured around 4 rounds with each round lasting a duration of one minute. Two competing teams try to guess the pictures that are being chosen. The guesses that each team can give are also limited to a time amount.

## [Design Constraints](#_2et92p0)

* Cross-Platform: Android is the current operating system for the game. The app needs to be functional with other operating systems.
* User Interface: The interface needs to match as closely as possible to the interface on the Android system. If this is cannot be completed, the interface should be updated in order to have the most consistent user interface across platforms.
* Multiple Team: The game must be able to support at least two teams with several players for each team. The server must be designed to handle all of these aspects.
* Identifiers: Teams must have a identifying name with a single game being stored in memory. Identifiers must be used to distinguish between games, players, and teams. Memory management needs to be practiced to determine which memory method will work the best across the different platforms.
* Copyright and Pictures: Since the game resolves around images, the proper licensure and permissions will be needed for the images used in the game. Image resolution and sizing will need to be monitored adequately to ensure that the images look the same across multiple devices.

## [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)

The visual design of the game will be developed using the Unified Model Language diagram. From the diagram, we can see that the player, team, and game classes derive a relationship from the entity class. Arrows denote inheritance among attributes from the super class. The diagram also includes different development characteristics including methods, variables, and classes. The programDriver points to the singletonTester which is used to evaluate the code. The game service is the method used to structure the framework of the game. Each class represents the game, team, and player attributes with the lines representing the association between classes. The numbers denote how many associations are within the classes. The diagram is referenced throughout the project to develop the game that meets the requirements of the client.

**"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** | -Mac is commonly used for desktop development, but can also be used for web application hosting.  Advantages:  -Mac is user friendly which makes user more receptive to product deployment.  - Mac products have excellent security.  -Development tools are readily available for web design and development.  Weaknesses:  -Licensing costs are higher.  -Products are not used as frequently as Linux or Windows. | Linux offers robust server-based deployment options and is widely used for hosting web applications.  Advantages:  -Linux is customizable for different types of servers.  -Licensing costs are cheaper compared to others  Weaknesses:  -Linux requires more training to run effectively.  -User interface is difficult for those inexperienced with the system | Windows is commonly used for hosting we applications with a user friendly interface and extensive web development resources.  Advantages:  -Windows has numerous server management resources.  -Windows is compatible with other Microsoft products.  -Windows can utilize development resources.  Weaknesses:  -Licensing is more expensive than Linux counterparts.  -Windows has more maintenance requirements compared to Linux | Oracle manages mobile devices and manages their servers. Mobile applications can be developed using hybrid design techniques to allow functionality across multiple platforms.  Advantages:  -There are multiple resources that enable code to be written once and then deployed across multiple device platforms.  Weaknesses:  -Extensive testing is required to ensured that all functionality transfers across devices.  -Design may take longer to consider platform specific features. |
| **Client Side** | The cost associated for the client is relatively cheaper. The language requires much training in order to program effectively. | Linux is a cheaper option. Howver, extensive training and expertise is required to configure the systems successfully. | The price for a web based app can vary. It depends on how extensive the software is for the client. Windows based apps are fairly easy to use. | The price for these devices are around the same price across the different operating platforms. Each has their own unique features and functionality, but are easy to use. |
| **Development Tools**  **\*Considerations** | Objective C  \*Mulitple teams with extensive expertise may be required .  \*Thorough testing may be needed to ensure cross compatibility  \*Licensing costs may be higher | Qt Creator  Aptana  Eclipse  Git  NetBeans  Visual Studio  Docker  \*Technical Requirements may influence the composition of the development team  \*Testing to ensure cross compatibility  \*Licensing fees may be incurred. | Visual Studio  PyScriptor  Apache NetBeans  IntelliJ Design  HTML  CSS  WebStorm  Atom  \*Technical Requirements may influence the composition of the development team  \*Licensing costs may be incurred depending on the tool used | HTML5  ReactNative  Flutter  Java  Kotlin  \*Testing on multiple devices is required to make sure functions are working properly  \*Mobile platforms require additional considerations and specific features for user experience  \*Multiple teams may be required.  \*Client may need licensing. |

## 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**: After analyzing the needs and requirements of the client, Linux would be the best operating system for their game development due it characteristics of being open source, having the support of numerous development tools and it’s customizable nature to meet any future requirement needs.
2. **Operating Systems Architectures**: Operating systems are the intermediary between computer resources and the end user. The construction of an operating system is comprised of the hardware, kernel, and shell. The hardware is made of I/O, cpu, and the memory. The kernel is the link between data processing and applications. The shell is the interface of the end user of the operating system.
3. **Storage Management**: Since the game will be web based, a cloud storage unit would work the best. The cloud will only have enough storage to maintain running the game. As the game expands, more storage can be added at a later date. If the client decides less functionality is needed, the premium can be adjusted to match the required functionality requirements. The storage holds long term data such as user information, account information, and passwords.
4. **Memory Management**: Java is the recommended language for development since storage is implemented automatically. Since this is done automatically, it will help save time in development. Memory is responsible for holding short term data. End users will be able to save their progress in a game to resume at a later time.
5. **Distributed Systems and Networks**: The Google chrome browser would be an excellent client server distributing system since it is a readily available across many platforms. This would ensure that the clients requirement of cross compatibility would be met. Google offers a strong server network. The game allows multiple players to access and compete against one another. Having many players on a single server, such as Google, will help ensure that the game runs smoothly without compromising functionality.
6. **Security**: After analyzing the needs of the client, we recommend using the Google Cloud Services for security since Google will supply the systems and hardware. Google will also provide any updates to fix any security or bug issues. Using the Google cloud automatically encrypts the data to ensure sensitive information such as account information and passwords are protected.

Java Code

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated