

Draw It or Loose It

# **CS 230 Project Software Design Template**

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 | 05/20/2022 | C. Franklin | <Brief description of changes in this revision> |
| 1.1 | 06/11/2022 | C. Franklin |  |

**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 currently has an Android based game called Draw It or Lose It. They wish to develop a web-based version of this game. In order to develop a web-based version of this game we will have to start from scratch as the functionality and user interface will operate differently. We will need to expand on the design to make use of computer functionality.

## [Design Constraints](#_2et92p0)

Having this application web-based will require a server or virtual server to run the game. We will need to determine what platform the game will be run on. We need to obtain a domain name or website the game will be run on.

## [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 GameService class to store basic information needed to run the program. The Entity class is the base class for user information. The Game, Team, and Player classes each inherit from the Entity class. They store and access the information for each game, team, and player in the program. The ProgramDriver class contains the main method and runs all functions of the application. The SingletonTester class serves to ensure the singleton is working properly.

**"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Mac is very powerful when editing servers. Mac is fairly versatile and easy to use. Mac has great security features. The costs associated with Mac can be very high. | Linux is very powerful yet very affordable. Has much or more of the flexibility of Mac but requires much more training and knowledge. Security threats are less than other OS' but security features are less. | Windows is by far the easiest to use. Costs are lower than with Mac but are more than with Linux. Good support but less adequate security and more security threats. | Mobile devices will require the least server resources. Android devices are higher risk for security issues. Apple devices have a much lower security risk. |
| **Client Side** | The costs associated with Mac can be very high. Time will be about the same as with Windows. Mac requires more expertise than windows however requires less than Linux. | Costs will be lowest available. Time will be cut due to better virtualization options and flexibility. Much more expertise is needed over other OS'. | Costs will be moderate. Time will be about the same as Mac. Windows requires less training and expertise than other OS'. | Cost will be moderate however higher level of expertise will be needed as many different variations of mobile devices exist. |
| **Development Tools** | Python, Java, Javascript, CSS, HTML, and PHP are common languages for web application development. IDE's include XCode, Eclipse, JetBrains IDE's, and NetBeans. Qt, Django, and Spring are some common frameworks. | Python, Java, Javascript, CSS, HTML, and PHP are common languages for web application development. IDE's include Eclipse, JetBrains IDE's, and NetBeans. Qt, Django, and Spring are some common frameworks. | Python, Java, Javascript, CSS, HTML, and PHP are common languages for web application development. IDE's include Visual Studio, Eclipse, JetBrains IDE's, and NetBeans. Qt, Django, and Spring are some common frameworks. | Python, Java, Javascript, CSS, HTML, and PHP are common languages for web application development. IDE's include Eclipse, JetBrains IDE's, and NetBeans. Qt, Django, and Spring are some common frameworks. |

## 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 recommend the use of Linux as the Operating Platform.
2. **Operating Systems Architectures**: Using multi-tier architecture will be very beneficial. This will allow the application to be developed and maintained independently. This will allow easier maintenance, adaption, and expiation of the program with less down time.
3. **Storage Management**: EXT4 file system is the best option as it has minimal fragmentation, quick reading and writing operations, and is very reliable. EXT4 is also the standard file system for Linux.
4. **Memory Management**: Linux uses various memory management techniques which are customizable in order to increase performance and decrease down time.
5. **Distributed Systems and Networks**: I recommend using a RESTful API which is available in Linux. Using a distributed system will allow the game to run on multiple systems at the same time.
6. **Security**: The program will contain many security features to prevent user data from being accessed. Linux also has many advanced security features which will help ensure the user data is protected from outside attacks.