

The Gaming Room

# **CS 230 Project Software Design**

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.tyjcwt)

[**Requirements 3**](#_heading=h.3dy6vkm)

[**Design Constraints 3**](#_heading=h.4d34og8)

[**System Architecture View 3**](#_heading=h.2s8eyo1)

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

[**Evaluation 4**](#_heading=h.lnxbz9)

[**Recommendations 5**](#_heading=h.1ksv4uv)

## [Document Revision History](#_heading=h.44sinio)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 9/13/2023 | Joy-Kalinowsky | Filled in the blank sections of the document |

**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.2jxsxqh)

The software design problem is to create a game application that handles how the players of Draw It or Lose It, an online game in development, can start multiple games and organize themselves into teams. It needs to be able to allow the creation of multiple game instances, as well as create teams and assign players to those teams. Game and team names and ids need to be unique, so they can be distinguishable from each other. Only one instance of the game is allowed to exist in memory during the playthrough. This solution will create one GameService object, which can create multiple Game, Team, and Player objects, simulating the teams players are assigned to in each game.

## Requirements

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_heading=h.z337ya)

The design constraints for developing the game application are:

- Must be coded in Java

- Must be able to be run on an online website

- Must not use up memory by allowing multiple instances of the game to exist at once

The design constraints must be worked with in an application environment by using the design constraints to make a game application can run without bugs or crashes in its designated environment.

## [System Architecture View](#_heading=h.3j2qqm3)

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.1y810tw)

"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.

**The UML Class Diagram contains multiple classes that are used to sort players into teams in every game of Draw It Or Lose It they play.**

**The GameService class can create multiple instances of the Game class, which get added to an ArrayList called games, meant to keep track of all existing Game objects. The Game class can create multiple instances of the Team class, which get added to an ArrayList called teams, meant to keep track of all existing Team objects. The Teams class can create multiple instances of the Player class, which get added to an ArrayList called players, meant to keep track of all existing Player objects.**

**The Game, Team, and Player Classes inherit from the Entity class. They override the getID, getName, and toString methods of the Entity class.**

**Finally, there is the ProgramDriver class. The ProgramDriver class contains the main() function, which can have statements added to it to create new Game, Team, and Player objects. It can also create a GameService object, but only one GameService object can be created. It uses the Singleton Tester function to ensure that only one GameServive object exists at a time, and that Game objects that have been created can have their information printed.**

## [Evaluation](#_heading=h.4i7ojhp)

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 does offer a server based deployment method in the form of the Apple Business Manager software. An advantage of Mac for hosting a web-based software application is that its software is very secure, meaning your software on a Mac device has a very low chance of being hacked. A weakness of Mac for hosting a web-based software application is that it requires more RAM due to the Mac OS’s high RAM consumption. | Linux does not offer a server based deployment system. You will have to install the server software yourself.  An advantage of Linux for hosting a web-based software application is hosting is free on Linux. A weakness of Linux for hosting a web-based software application is that it does not have as much proprietary software as Windows to work with in developing web-based software applications. | Windows does offer a software based deployment method in the form of their Windows Deployment Services. It comes with the Windows Server 2008 edition. An advantage of Windows for hosting a web-based software application is that only it can run several common proprietary applications, such as .NET, Microsoft Access, or MSSQL databases. A weakness of Windows for hosting a web-based software application is that it can lag when running high-CPU applications. | Most Mobile Devices do not offer server based deployment methods. To deploy to mobile devices, third party software will need to be used.  An advantage of Mobile Devices for hosting a web-based software application is that since the application is presumably on multiple devices, if one copy of the application is lost, it can simply be restored from another device.  A weakness of Mobile Devices for hosting a web-based software application is that it requires a constant internet connection to operate. |
| **Client Side** | The software development considerations that are necessary for supporting multiple types of clients as they pertain to Mac are cost, time, and expertise. The cost of deploying software applications on Mac is $24.99 a month for an employee with three devices with Apple Business Manager. | The software development considerations that are necessary for supporting multiple types of clients as they pertain to Linux are cost, time, and expertise.  The cost is the best part of deploying software applications on Linux. It is completely free to deploy software to Linux devices.  Deploying the software application to Linux will likely be the most time consuming deployment, since the workings of the Linux OS are not as well documented as the workings for Windows or Mac OS.  For expertise, since Linux is a more obscure OS, it will likely take a more skilled programmer to build applications for it. | The software development considerations that are necessary for supporting multiple types of clients as they pertain to Windows are cost, time, and expertise.  The cost of deploying software applications on Windows is around $20 per month with Windows Server 2008 software.  It will likely take very little extra time to make the software application deployable on Windows, since that is already the default operating system to be deployed to.  Windows likely requires the least expertise to deploy, since it is a very familiar operating system to most people and so has the most resources for help. | The software development considerations that are necessary for supporting multiple types of clients as they pertain to Mobile Devices are cost, time, and expertise.  The cost of deploying software applications on Mobile Devices depends on the third party software of app used to deploy the application, since most mobile devices do not offer built in deployment services. It will likely take the longest time to make the software application deployable on mobile devices, since mobile devices have different user interfaces than their desktop counterparts, especially considering the reliance on touchscreen input. |
| **Development Tools** | Deployment to a Mac OS, while likely not as familiar as Windows, is likely very simple. Only one team will be needed for this deployment.  Some of the relevant programming languages for deploying on Mac are XCode, Netbeans, and Tower. Some Mac programming tools that can be used to build this type of software for deploying are iTerm 2 and MacDown. | Deployment to a Linux OS is likely more complicated than on the two more popular operating systems. There may need to be a specialized deployment team with knowledge for this particular operating system.  Some of the relevant programming languages for deploying on Linux are Sublime Text, Atom, and IntelliJ IDEA. Some Linux programming tools that can be used to build this type of software are Gedit and Quanta Plus. | Deployment to a Windows OS is likely the simplest of them all. Only one team will be needed for this deployment. Some of the relevant programming languages for deploying on Windows are Microsoft Visual Studio, Ecliipse, and Netbeans. Some Windows programming tools that can be used to build this type of software are RadStudio and Delphi. | Deployment to the OS’s of different Mobile Devices will be much more varied. Multiple teams, each devoted to deploying to a specific Mobile Device OS, are recommended for Mobile Device deployments.  Some of the relevant programming languages for deploying on Mobile Devices are Android Studio, Xamarin, and AIDE. Some Mobile device programming tools that can be used to build this type of software are Kobiton and Testim. |

## 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**:

An appropriate operating server platform that will allow The Gaming Room to expand Draw It or Lose It to other platforms is a Linux OS such as Ubuntu. Linux is a free operating system, and Ubuntu is the Linux OS with the most support for any software issues that may occur, since Ubuntu is also the most popular Linux OS. The Linux OS is also the most compatible - it has been ported to more platforms than any other operating system. This goes in line with one of the main objectives of Draw It or Lose it - to enable the greatest number of players to access the game. Linux has a support lifecycle of 10 years, so the Draw It or Lose It team can be assured they will receive support for their server if any problems arise.

1. **Operating Systems Architectures**:

The architecture of the Linux OS consists of several layers. At the very core of the Linux OS is the hardware which consists of physical machinery such as CPU and RAM. Building on that layer is the kernel. The kernel manages the actions of the Linux OS and interacts with the hardware. Building on the kernel layer is the shell. The shell takes commands from the user and runs them through the kernel. Building upon the kernel layer are the applications and utilities which are the everyday software applications used in everyday life and the utilities which keep the application and the operating system running smoothly.

1. **Storage Management**:

An appropriate storage management system to be used with the operating system is a Linux Server. Linux Servers are free-to-use, compatible with a wide variety of operating systems, such as x86-64 and ARM v7, and support many software tools such as HP, Apache2, GCC, and Python.

1. **Memory Management**:

The Linux OS uses memory management techniques for the Draw It or Lose It software such as paging on demand, memory allocation, and mapping of files into address spaces. Paging is when in the absence of free main memory, processes are stored in the form of pages into secondary memory. Memory is allocated to store different processes that are used in the Draw It or Lose It software so they can be easily accessed when required. The Linux OS maps files into address spaces by storing partial and entire files into the address space for processes. This allows the OS to use memory sharing efficiently between processes.

1. **Distributed Systems and Networks**:

Linux OS is well known for its compatibility with a wide variety of platforms. This platform compatibility can be combined with distributed software to enable users to play the game remotely on their own personal computers. Application programming interfaces (APIs), such as the REST API make it possible for user computers to communicate with the Linux Server upon which Draw It or Lose It is located. Since the Draw It Or Lose It game uses the REST API, under normal working conditions, the users should be able to access the game through a website containing the game that is hosted by the Linux server. The REST API also allows multiple users to play the game and communicate through its multiplayer functionality. However, in order to host the game and provide all of this functionality, the server needs to have a constant and reliable stream of power so that the game does not go offline. This can be ensured with a backup power source, such as an Uninterrupted Power Supply (UPS), for the server to get its power from, so even if there is a power outage, the server can remain online until the power returns.

1. **Security**:

One of the main benefits of the Linux OS server is its security features. The latest release offers two factor authentication and AppArmor3, which is a mandatory access control system. These features create very robust security capabilities for the Linux OS. They create user protection by keeping user information secure and not easily accessible. The user information can be protected on and between various platforms through the use of encryption and authentication. In order to access the game data of a particular user file, a user must input the matching credentials to the file. Only then will the user gain access to their game file information.