

Project One(3)

# **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 | 12/09/2021 | Melanie Probst | The Player, Game and Team classes have been modified to inherit basic attributes and common methods from a newly created class called Entity. |

**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)

<Write a summary to introduce the software design problem and present a solution. Be sure to provide the client with any critical information they must know in order to proceed with the process you are proposing.>

The client needs help streamlining the development of a new game called Draw It or Lose It. Specifically, they need the game to have one or more teams involved, and multiple players assigned to each team. This can be done by creating functions within the classes that reference the objects to be contained within them. In other words, the Team class will contain a function to create and add Players to the team, while the actual method of creating a Player will be defined in another class. All game, team and player names must be unique. This can be done by adding all the relevant objects to a list, then checking that list for any matching names. Games, Teams and Platers will all have a separate list to check for duplicate names. Finally, only one instance of the game may exist at a time. This can be accomplished with a Singleton design, where the Game and its subsequent components will be tied to a single separate object in another class that cannot be created more than once.

## [Design Constraints](#_2et92p0)

<Identify the design constraints for developing the game application in a web-based distributed environment and explain the implications of the design constraints on application development.>

A web-based game application requires a constant internet connection; therefore, the application must account for possible buffering, dips in internet speeds and communication between devices. In order to ensure the application runs properly, there needs to be resource management implemented within the application, and there should be a number of servers so that every player’s game can run smoothly and without performance issues. The program should have quick execution times and a good handle on memory management in order to accomplish this goal.

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>.

The ProgramDriver class is used to demonstrate and implement the code written in the rest of the program, and the SingletonTester class is meant to demonstrate the Singleton pattern used in the GameService class. The ProgramDriver class specifically implements the use of the Singleton pattern by creating a single instance of the object created in the GameService class so that a game, teams and players may be created. The GameService class contains a function to add a game, check for a duplicate game name, and fetch data from Game objects, such as the name and id of the specified game. The Game, Team and Player classes all inherit from the Entity class. The Entity class holds the functions for getting the name and id data, while also holding those variables. It also contains a protected default constructor to prevent empty instances from being created and a regular constructor that is modeled for use by the Game, Team and Player classes. The Game, Team and Player classes all contain an overridden version of the constructor contained in the Entity class so that each class may have a unique identifier. The Player class contains just a constructor and a function for formatting. The Team class contains a list for Players associated with the team as well as a separate function to add them. The Team class also has a constructor and a formatting function. The Game class, much like the Team class, contains a list for the Teams that are added and a function to create them, along with a constructor and formatting function. The Game and Team classes implement a way to check for duplicate names in their functions for adding 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](#_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 web-hosting requires Mac hardware to run, and their products can be pricey. Mac is the most secure option when it comes to security, since the hardware is limited, the software is proprietary and the code is complex. This will make it harder for hackers to attack the server. Mac is also the most reliable, as there are hardly ever hardware or software errors that may crash the site. Mac is by far the most expensive option, but it is the most secure. | Linux requires less resources to run, and can be run on lower-end machines. Linux is open-source and is hardly ever updated, so the company will need to implement their own security measures in order to keep the server secure and stable. Linux is the cheapest option, but there are consequences to the lower price tag, such as the responsibility of securing the server manually. | Windows will need a higher end hardware because of its tendency to crash if the CPU is not fast enough or there is not enough RAM. Though more expensive hardware is needed than with Linux, it is still cheaper than a Mac. Windows security needs frequent updates, and there will be times where the server is down due to the updates. Windows is a solid option that satisfies a lower budget and higher security needs, but has the potential to crash the server for updates often. | Mobile apps are made for specific platforms, such as the iOS and Android platforms. They are designed to live and run on the mobile device they are installed on. Mobile apps are more expensive to develop than web apps since they are platform specific. The apps have access to system resources, so they offer greater functionality. Apps do have to be approved by the app store, which makes them safe and secure, but it may also be difficult to gain approval. |
| **Client Side** | Mac has multiple built in features to ensure a smooth experience on the client side, one of which being the secure server. However, it is important to consider that only someone with a Mac can use a Mac exclusive program. The customer may have to create their own Mac emulator if they want to play the game. | Linux servers tend to perform better than Windows servers. Linux web servers don’t experience memory leaks and the up-times are often much better than other servers. Not only is Linux cheaper when it comes to setting up the server, but it is cheap and user-friendly for customers as well. | Like Mac and mobile applications, Windows software can only be used on a Windows platform. Windows, however, is the world’s most widely used computer operating system, accounting for 71.06% of the console, desktop and tablet OS market in 2021. As mentioned above, Windows servers need constant updates, so there will be many periods where clients will not be able to access the server while it is updating. | Mobile apps suffer from the same problems as Mac applications do: they cannot run on a system they are not designed for. If there is to be a mobile app on both Android and Apple devices, the apps must be approved by both app stores and they must be individually built for those systems. In other words, the application needs to be built in 2 different ways if it is to run on both systems. |
| **Development Tools** | Mac supports and encourages the use of the Apache web server, which also includes architecture for deploying Java applications. Many tools for deploying a web-based application on Mac are included as part of the web hosting package. It includes features such as OpenSSL, which is integrated with the Apache web-server, that allows for public key infrastructure (PKI) authentication. There are many, many more features included, but some of the notable ones relevant to this application include inline scripting using HTML code, JavaServer Pages to embed Java source code into HTML pages, database integration, and remote management and monitoring using Server Admin. | Linux is open-source, which means it can be altered and freely available for anyone to use and share. In this way, Linux is easy to use. Software for Linux can be developed using languages such as Perl, Ruby, Python, PHP, Java and .Net. | In order to run a web-based application on Windows, there needs to be an Internet Information Services (IIS) server to run the application. The application can be developed in a variety of languages using Microsoft Visual Studio, such as C#, Visual Basic, C++, J#, Python, and more. Microsoft also offers Microsoft Azure, which is the company’s own cloud computing service. Having the application built on Microsoft Azure means the application will be securely built, multi-lingual and globally available. | Mobile apps for the Android are built using Android Studio or Eclipse IDE. IOS apps are built using Objective-C or Swift. Both Apple and Google also offer their own software development kits, or SDKs, for developers to make their own apps. |

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>
2. The best operating platform for expanding Draw It or Lose It to multiple computing environments would be a web-based application in a Mac operating platform. With a web-based application, anyone with an internet connection can use the application and Mac offers the best security, both on the client side and the server side.
3. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>
4. The Core OS of Mac is built on a Unix core known as Darwin and is based on the Berkeley Software Distribution version of Unix. Mac has a graphics subsystem, an application subsystem, and a user interface. The graphics subsystem includes three main components: Quartz Extreme, OpenGL and QuickTime. OpenGL provides 3D graphics support for 3D applications. It is industry standard so it is compatible with Windows and Unix systems. The application subsystem includes three application development environments: Carbon, Cocoa and Java. With the availability of Java application development, the web-based application can be easily created.
5. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>
6. Darwin has a Virtual File System design that supports several file systems including Mac OS Extended Format, Mac OS Standard Format and UFS, the standard file system for Unix systems. In addition to the file structures available for use, additional computers would be helpful for extra storage management.
7. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>
8. The heart of the Darwin core is called Mach. Mach provides a separate, protected memory area for each application to run in. Since the applications run in their own space, other applications won’t be affected if one of them crashes. Mac OS X also has automatic RAM management, and automatically allocates RAM to applications that need it.
9. **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>
10. Darwin supports major network file protocols including Network File System (NFS) client, which is the dominant file-sharing platform on Unix platforms. Additionally, Mac OS X provides support for Windows-based network protocols, which means you can interact with Windows machines just as easily as you could other Macs. Mac OS X also uses loadable bundles, such as plug-ins used in web browsers.
11. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>

Because Mac software is proprietary and the code is complicated, it is more difficult for any would-be hackers to launch an attack. Mac is so secure, it is the OS of choice for the United States military. Additional security measures, such as requiring a username and password for user access to the application and a locked facility containing all the servers should be taken. Education for the employees on good security practices would also be beneficial in addition to a hierarchy of access to the sensitive information, such as adding admin privileges.