

Creative Technology Solutions

Gabriel Sequeira-Bacher 2022

# **CS 230 Project Software Design**

Version 1.1

## 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 | 05/18/2022 | Gabe Sequeira-Bacher | Added classes |
| 1.1 | 07/01/2022 | Gabe Sequeira-Bacher | Memory, security and storage information updated |
| 1.2 | 07/12/2022 | Gabe Sequeira-Bacher | Version Three |

## [Executive Summary](#_sbfa50wo7nsh)

Our client, The Gaming Room, wants to expand their user base and increase revenue by developing a game for web browser and base it around the existing game for the Android Store called Draw It or Lose It. Myself and the staff will be working on setting up the development environment. I propose that we borrow at least some of the functionality of the Android app and refactor the code to be compatible with web browser software.

## [Design Constraints](#_2et92p0)

Since this game is created to run on the web, constraints will include (for obvious reasons) security as well as the time it takes for the program to render. One way in which we might mitigate security risks of a web-based application (passwords being guessed, for instance) is by using a token-based approach in which the user, after successfully entering login credentials is given an encrypted token to be used by an asynchronous API. Another way is by using a level of authentication that goes by certificate in which a digital signature is produced which can proves that the person requesting access permission has the right key.

One or more teams must be able to compete in games.

Players and teams must use names/identifiers unique to themselves.

Team must consist of multiple players.

Only one instance of the game can exist at any given time.

Program can run on a variety of platforms.

Will be coded using the Java programming language.

It’s important that this version at least somewhat resembles the game already available on android. A certain amount of development time should be allocated to refactoring the existing Android code to work for multiple platforms.

## [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)

In this program, there is a single driver class that creates individual games with corresponding players and teams. The class that’s called GameService is primarily responsible for the creation of single instances of these. Since it is using the singleton pattern of software development, only one instance of GameService is allowed to exist in the computer’s memory at any given point in time. In this description of the Domain Model, I will elaborate on the various classes which allow this singleton pattern to work.

It is only thorugh calling the getInstance() method that a GameService instance can be created. A definitive feature of getInstance() is that it can check whether an instance exists within the computer’s memory and only instantiate one if that isn’t the case. By setting GameService’s constructor to private, we prevent an instance from being created except by calling the method getInstance().

This part will be devoted to the method addGame(). The aforementioned method can be called by the driver as soon as there is an instance of GameService running. One feature of this method is its use of the iterator pattern to keep games from being added that share a name in common or mostly in common.

Next, the program allows for the creation of a team by way of the addTeam() method. This uses the iterator method as well, and likewise will add the new team to the teams list.

addPlayer() does the same as addTeam() or addGame() but with the distinction that, rather than add a new game or new team, this method will actually go ahead and add a new Player object which is then automatically added to the list of players. This also uses the iterator method to keep similarly named Players from being created.

The one thing that each of these object types share in common is that each one is a subclass of the Entity superclass. The protected attributes of Entity are *name* and *id*, as well as the default constructor.

As outlined in the UML diagram, many of the object-oriented principles are at work in this program. Specifically, inheritance is used in the creation of multiple subclasses (Team and player) from the superclass Entity. Polymorphism is present in how constructors are able to be overloaded. Abstraction and Encapsulation allow for a user to access the program, even to create teams and utilize all of its functionality without knowing exactly which of the methods are doing the adding, so they don’t have to know everything about how the code works to in order to operate it. It just works, which is good for our human brains.

**"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)

Linux, Windows, Mac, and mobile devices could all be deployed for this development project.

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | While it is possible to host our application on Mac, it might be expensive seeing as the client may have to buy their own Mac hardware device(s). | Linux uses opensource software, so it costs much less to use than its counterparts in Windows or Mac OS. | Windows has a variety of applications, most of which can be used for work and in the server as well. Licensing expenses tend to be high for these models, however. And Linux may have more hosting platforms. | One thing about mobile devices is, they are not great for sharing a server among multiple users. |
| **Client Side** | Developers might have to already know about using mac hardware beforehand. An advantage is it can run applications for mac, windows and Linux OS’. Updates are less frequent, however. | This is the most niche OS and the expertise would have to be greater for this than for windows or mac. Aside from Mac, however, it is the most secure system to use. It’s the cheapest to use, but there is more room for error with Linux. | This would take less expertise to be able to use client-side. Viruses could become an issue if protection is not already put in place. This is also the best for gaming in general. Price range is like mac as well. | Anyone can basically develop a mobile apps, there are even apps now where you can make your own app. It can be very cost-effective. It may require some degree of expertise to develop between different mobile systems. In this case it is not cost-effective. |
| **Development Tools** | Development tools could consist of Javascript, CSS and HTML. It gives a greater range of control over the mac system via the console alone than windows PCs. IDEs might be Eclipse, Visual Studio suite and notepad++. | Linux is astonishingly similar to Mac in terms of development tools/IDEs. | Runs similar IDEs to Linux operating systems. But Windows has more tools available for it. | Programs would be developed in Swift for Android, and Objective C or Java for Apple. |

## 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**:

The OS I recommend most to use is Windows. This will be for hosting servers as well as playing the game Draw it or Lose it. I say this mainly because of the tremendous amount of support that’s provided for it through its various tools. Most video games are hosted by Windows, which just makes it the best way to draw people in to the game that are predisposed to play it. Windows provides a great deal of adjacent applications to help with information storage (Microsoft SQL being one), and it’s a good enough platform to support the needs of a large user base. Also unlike Linux, it’s not necessary to make changes to everything in the infrastructure all the time.

1. **Operating Systems Architectures**:

The application of a web-based videogame is hosted on a Windows NT server, which can support files for Macintosh systems, host up to 256 remote clients, and supports up to 32 processors. The structure of NT is layered, with each layer performing a specific level of task. This actually makes the application quite modular, and it’s easy to debug certain parts of the application as a result of this. The two modes are User and Kernel mode.

1. **Storage Management**: Windows allows you to manage your own files on your drive and the amount of space you can use. You can also choose where to save files to. The use of cloud computing software could cut back on cost and runtimes. It could be a good idea to use some form of caching of frames to allow for the game animations to be rendered smoothly. The storage should be dealt with by SSD or HDD. As an alternative, if MSSQL were used, it would have a lot of database storage available. Both are viable options for this.
2. **Memory Management**:

Memory management is very important to consider when developing web-based games. A good 4 gigabytes of RAM should be available on the user’s computer for them to play the game at a regular speed. The memory is automatically managed by Windows for a game such as this. A paging scheme captures data from a secondary storage location to use for the main memory storage. Data can also be cached to the disk for later use with Windows. This is favorable for access and speed.

1. **Distributed Systems and Networks**: It’s best to use a tool that can cover a range of OS systems. Unity is one-such game creation engine. It is also the most cost-effective. A server that mostly handles games would be ideal for a project such as this one.

Asynchronous, RESTful APIs will be able to allow the front and back ends to interface with one another. This allows for the front end to see all communication between the client and server.

1. **Security**: The best security option for this is to use MSSQL to store not only the user’s login credentials but also data specific to their user profile. Captcha technology can be used to make sure it’s an actual person playing the game, and not an AI. Encrypted data will store the user’s information.

Sources

<https://www.geeksforgeeks.org/windows-memory-managment/>

<https://docs.microsoft.com/en-us/sql/sql-server/maximum-capacity-specifications-for-sql-server?view=sql-server-ver16>

<https://www.minitool.com/lib/windows-nt.html>