

The Gaming Room

# **CS 230 Project Two Software Design Document**

Version 1.2

## Table of Contents

[**CS 230 Project Two Software Design Document** 1](#_Toc48494249)

[Table of Contents 2](#_Toc48494250)

[Document Revision History 2](#_Toc48494251)

[Executive Summary 3](#_Toc48494252)

[Design Constraints 3](#_Toc48494253)

[Domain Model 4](#_Toc48494254)

[Evaluation 5](#_Toc48494255)

[Recommendations 6](#_Toc48494256)

## 

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 07/18/2020 | John Kiser | Created Software Design Document.  Added following sections:  Executive Summary, Design Constraints, Domain Model |
| 1.1 | 08/02/2020 | John Kiser | Added Evaluation section |
| 1.2 | 08/16/2020 | John Kiser | Added Recommendations Section |

## [Executive Summary](#_sbfa50wo7nsh)

The client, The Gaming Room, is seeking to develop a web-based game, Draw It or Lose It, based on their current Android application of the same name, in order to expand access to the game to multiple computing platforms.

## [Design Constraints](#_2et92p0)

One business constraint will be the proposed software solution will need to incorporate the currently existing software infrastructure belonging to the client company. This will be necessary to ensure data from the existing game will not be lost and that the game will function the same way on multiple platforms.

A technical constraint will be the proposed backend language for the application of Java. This will more easily allow reuse of the pre-existing code which is currently in use by the client’s Android application.

Another technical constraint will be the proposed software solution, being a web-based solution, will need to incorporate multiple technologies in order to be implemented properly across multiple platforms. These technologies include, but are not limited to, front-end development in HTML, CSS, JavaScript, in addition to the proposed backend language of Java.

Another business constraint, related to the necessary development languages/technologies, may include the requirement to hire additional long-term developer support in order to implement any necessary maintenance / updates to the software solution in the future.

## 

## [Domain Model](#_8h2ehzxfam4o)

The following UML Class Diagram visually depicts the class structure of the proposed software solution at this stage in development. It describes class attributes, methods, and relationships. The GameService class implements a Singleton Pattern in order to ensure the client’s requirement to only have one of any game service at one time. The GameService class has a zero to many relationship with the Game class, which in turn has a zero to many relationship with the Team class, which in turn has a zero to many relationship with the Player class. The Entity class is the parent/inherited class of the Game, Team and Player classes and contains the common class attributes among them. The common methods of the Game, Team and Player class utilize polymorphism as applicable by overriding the parent class methods. The ProgramDriver class contains the main function and provides the starting point for the program. The SingletonTester class exists for the purpose of testing the functionality of the Singleton pattern used within the GameService class.

****

## [Evaluation](file:///Z:\SNHU\CS-230\CS%20230%20Project%20Software%20Design%20Template.docx#_2o15spng8stw)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | The macOS server would be a viable option and could provide the hosting for a web-based application. However, the macOS server is designed primarily to interact best with other Apple operating systems and products and would not be the best option for this project. | A server running a Linux distribution (I would recommend RedHat Enterprise Linux or RHEL) would be an ideal solution for this web-based application. Linux offers a very stable platform for constant connectivity and provides maximum flexibility over the remote administration of the server system. | The Windows server would be a viable option and would provide the web-based site hosting acceptably. However, the Windows server is designed to interact best with Windows based client applications and systems for remote administration. | Any mobile device would not be very suitable for hosting a web-based application, due to the need for the server to be up and running constantly and require the ability to work on multiple cores for best effect. The majority of server systems are not set up as mobile devices and as a result are not running mobile operating systems. |
| **Client Side** | The software development considerations for Mac will require similar costs and expertise as other systems. The primary application used will be a web-browser, so a developer will need to test the web-site on a system running macOS at the very least. | The software development considerations for Linux will be similar to those involved in development for Mac and Windows based operating platforms, and the web-site will need to be accessed for testing from various browser applications from within Linux at the very least. | The software development considerations of cost, time and expertise will be similar for the Windows platform as the other major non-mobile operating platforms. A developer will need to test the web-site from a Windows operating system at the very least. | The software development considerations pertaining to Mobile devices will require more attention than the other primary operating systems due to the need to ensure compatibility of the web-browser-based client side usage of “mobile” versions of web-browsers being used. |
| **Development Tools** | For developing the server side application, the Swift language in the XCode IDE would be the ideal production solution for use. The client side will need to be tested at least within the Safari web-browsers application. | The server side application should be developed using the Java programming language using the Eclipse IDE for maximum productivity/flexibility. The client side should be tested using the FireFox browser application at least. | The recommended language for a Windows development of the server application would be C# and the recommended IDE would be Visual Studio. For the client side the application should be tested with both Internet Explorer and Edge browsers at the least. | The server side should not be developed on a mobile operating system. The client side will need to be designed with mobile web-browsers in mind for both iOS (Safari) and Android (Google Chrome and Samsung Internet). |

## Recommendations

1. **Operating Platform**: I recommend using traditional dedicated servers running Red Hat Enterprise Linux (RHEL) Operating Systems, as opposed to cloud-based server architecture or other dedicated server Operating Systems.
2. **Operating Systems Architectures**: The use of traditional dedicated servers will allow the greatest amount of control over the system administration/content and security of user/player data, which is typical of online game server architecture choices. The use of the RHEL server Operating System, being one of the most popular Operating Systems in use for traditional server architectures, will ensure maximum performance for users, and minimum expense in finding system administrators with experience with the system.
3. **Storage Management**: The storage management system recommended would be the use of the XFS file system which is the default file system used in the RHEL platform. This will ensure the highest amount of security and consistency of game and user/player data possible with the chosen platform, even in the event that data will need to be recovered due to any power outage or system crash. It also allows for variable block sizes and has support for delayed allocation, resulting in reduced fragmentation problems and increased performance. This ultimately provides the availability to utilize cheaper physical storage mediums (such as high capacity HDDs) and still achieve desired performance results.
4. **Memory Management**: The recommended RHEL operating platform will provide advantages in the area of Memory Management by its use of what are called “Huge Pages.” Normally, memory management is done though blocks that consist of 4096 bytes of data called “pages.” In order to efficiently increase the performance of its systems, the RHEL platform uses “Huge Pages,” pages ranging from 2MB to 1GB in size. These “Huge Pages” are made easier to manage by implementing an abstraction layer in RHEL called “Transparent Huge Pages (THP).” This will provide an efficient and effective method for ensuring delivery of game content to the various users/players.
5. **Distributed Systems and Networks**: The recommended platform will provide maximum interoperability for a web-based server set-up, since the connection to the server will through client web-browser software. RHEL is a leading platform in stability, minimizing down-time. The resistance to potential power/network outages will be slightly greater than if utilizing a cloud-based architecture, however, the performance, administrative and security gains of using a traditional dedicated server set-up outweigh those gains.
6. **Security**: The recommended platform of RHEL provides one of the most secure environments currently available in any server Operating System. The platform provides a secure architecture by very effectively separating system and user level processes, as well as providing numerous tools for system administrators to use to ensure maximum possible network security. Beyond this, the open source nature of the operating platform invariably results in a higher degree of security to a proprietary system which is not able to be tested to the same level by external security experts.