

The Gaming Room Web Service

# **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 | 03/21/22 | Michael Kennedy | Initial design document |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room Web Service has developed a highly successful Android experience with a native application titled, *Draw It or Lose It*. Seeking to expand on the current user base, the game will be developed to be hosted as a web application to reach the maximum number of users, and becoming available to multiple platforms via web browsers. This single version of the application in the form of a web application will focus efforts and limit the necessity for maintaining multiple versions of the application. Hosting the game will also allow for a unified experience where a single instance of the game can exist on the platform, and teams and players can compete with one another.

## [Design Constraints](#_2et92p0)

* The game must be hosted on web servers made available to the internet
* There must be the ability to host a single game instance in memory
* This game instance must host multiple teams with unique identifiers
* These teams must host multiple unique players also with unique identifiers
* Adequate security for a web-hosted application must be implemented and maintained

## [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)

Using the singleton pattern a single game service instance is maintained in memory. This single instance maintains a one-to-many relationship with games hosted on that service instance. There is an entity class maintaining common properties which are inherited by games, teams, and players. Each of the child classes then leverage an efficient iterator pattern to ensure that duplicate entries are not possible, but all members of the lists of objects are checked for an existing member matching the proposed attributes. In this UML diagram are also displayed a driver that uses a singleton tester to validate the expectation that a single instance is created and accessed instead of creating additional instances.

**"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** | macOS ships with web server software and has several tools made available to it, but the inclusion of unnecessary user experience desktop elements is unnecessary for operating in a client server environment. | Linux is an ideal option for web application hosting as there are many available web server technologies available to install, as well as the ability to maintain a minimal environment to limit competing resources. | Windows has web server software available to install, but as with other desktop environments includes resources unnecessary for an efficient web server hosting solution. | Mobile devices are ideal for a native application, but when the need here is for hosting, where mobile devices are not enabled with software to host web applications, and with typically limited networking capabilities are not ideal. |
| **Client Side** | macOS ships with a web-browser in Safari that uses modern web technologies common to browsers, which enables a stream-lined development for web that will be compatible for macOS. | Linux also has web browsers available based on distribution, which leverage modern web technologies that do not require additional efforts beyond that to enable all web applications. | Windows maintains compatibility with modern web technologies enabling the interactions from a web-browser which does not require additional efforts. | Mobile devices maintain compatibility with modern web technologies, and do not require alternative development. The primary consideration for mobile devices is their limited network connectivity being mobile as opposed to a wired network. |
| **Development Tools** | Java is an ideal programming language for hosting a modern web application and has several resources available for development. Java is a common language with many IDE available, with the preferred IDE being Eclipse for this project. | Java is an ideal programming language for hosting a modern web application and has several resources available for development. Java is a common language with many IDE available, with the preferred IDE being Eclipse for this project. | Java is an ideal programming language for hosting a modern web application and has several resources available for development. Java is a common language with many IDE available, with the preferred IDE being Eclipse for this project. | Java is an ideal programming language for hosting a modern web application and has several resources available for development. Java is a common language with many IDE available, with the preferred IDE being Eclipse for this project. |

## 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 recommended operating platform to host The Gaming Room to expand Draw It or Lose It is Linux. Linux is well established across the software development industry in providing a production-ready operating environment that is scalable and meets the needs of the broadest range of technologies. The open-source community is continually driving adoption of new technologies to support development efforts. Furthermore, this Linux environment will be maintained on a cloud-computing environment capable of scaling necessary resources as needed.
2. **Operating Systems Architectures**: The operating system architecture will leverage a client-server architecture where minimal development effort is requiring in maintaining the clients, while the server backend will be in a Linux environment, hosting a backend API that will facilitate requests and maintain an environment to host session details and the runtime environment for the games themselves. This architecture will rely heavily on quality network bandwidth availability, which will be addressed in a later section.
3. **Storage Management**: To support the storage requirements, our Linux backend API servers will have available a filesystem that will maintain persistent storage of assets and data that will be shared across servers as scaling is performed. This is a shared filesystem that is typically made available as an NFS mount, so that all systems can access and share the necessary data.
4. **Memory Management**: Each server will have sufficient memory to maintain the required session storage as the session and game instances will be maintained in memory on the servers which are running the application. As requests for new games are added, they will be added to the existing session. Each server then will be running an independent session instance providing scalability and efficiency.
5. **Distributed Systems and Networks**: The client-server pattern will rely heavily on network bandwidth and is therefore the network requirements will be that each client has a reliable connection to the backend servers. With so many services running in cloud-computing environments there are substantial networking resources committed to the services. The application will still be susceptible to outages within the cloud services, but with resiliency available across geographical locations, the impact will be limited, and services are typically restored in minutes. This provides the most efficient, resilient environment to run the distributed nature of the application across servers maintain session instances.
6. **Security**: Linux has several security implementations that are limiting the scope of access for each process to only that which should have access to. With the use of a cloud environment, physical access to the systems is nearly impossible, and the security mechanisms in place by the cloud provider will be maintained and further advanced during the lifetime of the application. For authentication and authorization, we will rely on the services made available by the cloud provider to efficiently provide these services and will have a dedicated service model to ensure improvement and stability of the security implementation.