

The Gaming Room

# **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 | 10/4/2020 | Ben Wornom | Initial Assignment |
| 2.0 | 10/20/2020 | Ben Wornom | Software Recommendations |
| 3.0 | 10/25/2020 | Ben Wornom | Final Recommendations |

**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)

The Game Room is requesting a software development environment complete with all the necessary tools for coding, deployment, hosting, and data storage as well as the actual development of the Draw It or Lose It game. The game itself needs to be web-based with multiplayer capability(Users will log in as players and join teams of other players). The game will draw from a large pool of stored images to include in the gameplay. Draw It or Loose It will be available to users on Desktop(Mac, Windows,Linux) as well as mobile devices. The solution presented will feature a Java backend and an outsourced front end design team. The images will be stored in AWS, while team/player/user/game data will be stored in a SQL database.

## [Design Constraints](#_2et92p0)

Technical Constraints:

Once the development environment is set up and the product is running, the staff at The Game Room will need to learn how to maintain and support the solution.

Appropriate design patterns to ensure only one instance of the game can exist in memory at any given time.

Deployment mechanism to push patches and bug fixes to testing/production environment.

The hosting platform must be fast enough to keep up with real-time gameplay and strong enough to facilitate large teams.

The user facing side of the game must be secured from malicious attacks.

Business Constraints:

Costs for hosting, AWS image repository.

The game must be made available on a wide range of devices and published on each appropriate app store or game store.

## [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)

Three classes (Game, Team, and Player) are the key objects for running games. These three classes inherit from the base Entity class which contains generic variables and functions. The GameService class has a 0 to many relationship with the Game class, the Game class has a 0 to many relationship with the Team class, which has a 0 to many relationship with the Player class. The GameService class is the controller class that ensures only one instance of a game is available at a time.



## [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 server is an option, however there is less support and upgrades being developed by Apple in recent years. Mac has fallen behind considerably in the server game. Hosting providers offer limited options for Mac servers, and costs are higher compared to Windows and Linux. | Linux has a wide range of hosting providers(GoDaddy, InMotion) for inexpensive rates. Security is good, Linux servers are not targeted as often compared to Mac,Windows anyways. The open source model of Linux offers lots of control and options for server design. Performance stays strong for extended times. Server costs are lower than any other OS can offer. Due to the open source nature of Linux server software however, configuration processes can be more complicated. | Windows Server hosting capability strengths are speed, cost, and professionalism. ASP.NET framework has unmatched performance. User interfaces for server administrators are high end and professional. Server costs are more expensive than Linux, but a wide array of offerings are affordable. | Using a mobile device to host a web-based software application would be impractical due to the limited number of options designed for mobile devices to work as servers. Mobile devices generally lack the computing power to host a web-based software application effectively. |
| **Client Side** | Developers must create an executable program for Mac. Safari is a common browser used by Mac users, so testing will need to be done to ensure all the front-end features display properly for Mac Clients. | Linux clients are not common gaming platforms. Expertise in supporting various Linux clients would be rare. Users would be less numerous. The game would work well functionally, but getting access and support to users would consume more time and money. | Windows clients are widely used for online gaming, so extra expertise will not be necessary. The development team can release the game in the Windows Play store. Some Windows Clients use Microsoft Edge or Internet Explorer, so some extra time will be needed to ensure the frontend renders properly on these browsers. | Developers will have to consider which app stores to make the game available and the costs for the various distributions. For each mobile device(Android or Apple, Phone or Tablet), time will need to be taken to ensure the screen size and inputs are solid. Mobile developers will need to be hired in addition to regular software developers. React Native Framework |
| **Development Tools** | Development on Mac machines would be most logical for the team. Sublime Text Editor is an excellent IDE for development on Mac. In addition to Python, Golang is a newer programming language with strong Mac support for backend development. Deploy Studio is a free deployment software for Mac specifically. | PHP integrates very well on Linux environments for developing the frontend. Python and Ruby are strong performing languages for developing the backend, and both languages have complete IDE’s for Linux. SQL database has widely documented features compatible with Linux OS. Symantec Ghost Solution Suit is a Linux compatible deployment solution | In addition to Sublime, Visual Studio is a strong, complete IDE that helps users write and release code and manage an array of environment tools to help support the application, such as  MSSQL. C# is a seamlessly integrated language for this environment. SmartDeploy is a top deployment solution for Windows. | The React Native Framework is highly tailored to help developers create mobile applications for Mac, Windows, and Linux. Appy Pie is a popular development tool for mobile devices as well. |

## 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**: This recommendation is for a Linux Ubuntu server for connecting a network of users and giving multiplayer access to Draw It or Lose It in a AWS Cloud environment.
2. **Operating Systems Architectures**: The Ubuntu version of Linux is designed to support several thousand users. It provides a shell interface for users to run commands through the hierarchal file structure and other OS components. Linux ensures encrypted data and password protection. The Linux kernel has structures for handling networking packets(TCP/IP protocol) to connect to various clients.
3. **Storage Management**:

AWS Cloud storage is recommended because Draw It or Lose it has a relatively small amount of data it needs stored(around 1.5 GB images, 3 GB User data). This is the most cost effective storage management system. The front end application should be configured to run on different instance than the backend application, so that if the frontend goes down, the backend storage will still be functional. Developers should code to fetch and cache large memory requests before the start of the game. Keep offline copy of images on two devices. Implement Content Delivery System to deliver content to distant users. Store images in different part of the Ext4 file system from user data. MySql database installed on the Linux server will provide fast data retrieval for the live games.

1. **Memory Management**:

Linux OS makes use of virtual memory to allow more storage than the physical hardware contains. There is more virtual memory than physical, so it uses a ‘demand paging’ to properly manage the physical memory it does use. This ensures that only the immediately executing program is loaded into memory at a given moment. Linux gives each process a unique virtual address space to protect from overlap. For the game software, this means a task that queries the database for a user will not need to load the entire database program into memory, but only the relevant query. On a programmatic level, developers will write memory efficient code. Ensuring allocated memory is cleaned up after games finish(garbage collection), as well as the Singleton design pattern are some ways to do this.

1. **Distributed Systems and Networks**: React Native is a framework for creating a cross-platform front end. Developers will be able to write native code for various clients, and use Reacts API to wrap it into a synchronized solution. The backend logic will communicate through an API to this frontend. Using REST methodology, the app will connect the same on any device. To troubleshoot various clients, the Securi Global Performance Test tool will be part of the developers toolkit.
2. **Security**:

To ensure the Linux server platform is secure, the Cloud provider will protect the actual server instance. Assumedly, the maintenance team will keep up with server updates and patches as they are released for the active version of Linux. The system firewall will be on at all times as well. Attacks can be thwarted by limiting unnecessary outward facing surfaces, disabling extra root accounts, and scanning server logs for suspicious activity. Users with accounts to play the game will have strong password requirements, and the application code will use authorization tools prevent underprivileged users from executing sensitive methods. The webpages will be secured with SSH.