

# **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  1.1  1.2 | 07/14/22  07/29/22  08/11/22 | Curtis Felsher  Curtis Felsher  Curtis Felsher | Outlined Executive Summary, Design Constraints, and Domain Model  Expanded on the development requirements section  Completed server recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

Creative Technology Solutions has been approached by a new client, The Gaming Room. The Gaming Room would like a web-based game developed based off their current game. The web-based game, Draw It or Lose It, is already available on android but the company would like it to be available on multiple platforms. The design of the game includes an application that will render images from a library of stock images, in a timed manner. The software design requires games to have one or multiple teams involved, multiple players on each team, game/team names using unique identifiers that are kept track of, as well as a single instance of the game existing in the memory at any time during gameplay.

## [Design Constraints](#_2et92p0)

-Multiple operating system/platform availability

-Multi-Person teams

-One or multiple teams per game session

-Unique identifiers for game/team names

-Must allow only one (1) instance of the game to be running in memory

-Network connection to each game instance should be included for cross-platform sessions

-Performance must be checked on multiple versions of hardware to ensure continuity

-Security of the application as personal details should be encrypted

Based off the requirements from the client, The Gaming Room, the constraints are listed above. Porting would perhaps be the most obvious solution as the application already exists on the android platform. This would take less time in theory as the application is already written for another platform. Each platform version needs to be checked on multiple versions of hardware as different pieces of hardware have different performance constraints/benefits. Security should also be high on priority if the game requires any personal information for account creation or linking.

## [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)

With the newly added entity class, this class shares aspects between itself, Game, Team, and the Player Classes. These classes share information such as name or id with each other through inheritance, which in turn creates a superclass. With all of the classes referencing each other, this type of programming is presented in the UML through aggregation. Off to the side are also the ProgramDriver and SingletonTester classes within the UML. The ProgramDriver class houses the main where any code execution takes place and the SingletonTester is used by the ProgramDriver class. These work together to ensure that only a single game exists at any moment within the memory.

**"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 server application  + In depth terminal commands  + LDAP support  + ADP support  + Cloud Support  - Current Arm transition can cause program incompatibility | + Open Source  + Terminal flexibility  + Inexpensive or free  + LDAP support  + ADP support  + Cloud Support  - Complex Operation  - Program incompatibility | + Beginner Friendly  + Tons of software support  + Guaranteed long term support  + LDAP Support  + ADP support  + Cloud support  - High licensing costs  - Vulnerable to malware if not handled correctly | + Mobile  + Cost  + LDAP support  + Cloud support  - Not much software support  - Weaker hardware |
| **Client Side** | + Shares similar costs with windows  + More than beginner expertise required  + Time is inverse to expertise; more expertise equals less development time | + Extremely low cost due to open-source nature  + Not a common OS, quite a lot of time and expertise are required  + LDAP Support | + Cost is similar to mac, on the higher side  + Development time and expertise are around the same as Mac since both are not open source | + Hard to implement as there is not much development support on the device itself  +Time and expertise would be high as it is extremely uncommon |
| **Development Tools** | + Visual studio  + VS Code  + Netbeans  + Xcode  + Atom  + Notepad++  + Android Studio  + Eclipse  + Swift  + Python  + Java  + C++ | + Eclipse  + Geany  + Atom  + Idle  + VSCode  + Notepad++  + Android Studio  + Python  + C++  + Java  + Ruby  + Swift | + Eclipse  + Intellij  + Visual Studio  + VS Code  + Notepad ++  + Pycharm  + Android Studio  + Python  + C++  + Java  + Swift | + App store IDES such as Pythonista or swift playgrounds  + Running the actual code device side is difficult without purpose-built 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**: Based on current data, my recommendation for The Gaming Room would be Linux. Linux has access to multiple IDEs as well as multiple language support. The open-source nature of the operating system allows incredible flexibility as well as low to no cost associated with using it for its server functionality. The operating system is slim and quick as well as stable if configured properly. Due to this, the hardware requirements are fairly low and do not require a high budget compared to other operating systems. The main reason I chose Linux though is due to the ease of modification. When the server is modified, it is quick/easy and does not require a restart after changes have been made unlike windows.
2. **Operating Systems Architectures**: My recommendation would be a multi-tier architecture. This allows scalability that is easy to implement as well as data integrity. By having a middle tier, this tier ensures that only important information is allowed to be updated and removes the risk of data being corrupted. Another advantage is scalability. If the game progresses to become something much larger than initially anticipated, this can be addressed easily and quickly.
3. **Storage Management**: With Linux being open source, storage management can be accessed at the lowest of levels. Since the company will have access to all partitions at a drive at any given time, this allows for greatly enhanced storage management and configuration. Recommendations for storage would be and onsite database configured in RAID 1 for data redundancy in case anything catastrophic were to happen and an offsite database configured in the same way. Redundancy is key for safe data storage.
4. **Memory Management**: Linux uses a buffer cache for memory management. If the application requires more memory, then the operating system will free up the buffers and cache to allow the application to have access to more memory on demand. Depending on how heavily the server is used, a recommendation would be 8gb minimum of ram with up to 64gb depending on server load for smooth operation and future proofing. Of course, less can always be used as Linux is incredibly flexible and can have memory requirements reduced greatly.
5. **Distributed Systems and Networks**: The implementation of the REST API is available on the Linux OS. Since the API itself is language agnostic, this ensures compatibility across multiple types of devices. This would be perfect for a web-based application that interfaces between a server and client as requests are managed through HTTP.
6. **Security**: Since Linux is Unix based, only users with the correct privileges have access to the kernel to execute malware or steal internal information. Through the concept of least privilege, Linux maintains a high level of security that other systems do not offer. With Linux being open source, many programmers can come together quickly to identify and patch vulnerabilities overnight. The server must be configured appropriately, however.