

<Name-of-Software-Application>

**CS 230 Project Software Design Template**

Version 1.0

**Table of Contents**

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 11/20/24 | Gordon | Initial Software Design Document |

**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**

Our client, The Gaming Room, has consulted with us to facilitate the development of their web-based gaming app. They desire it to be web-based and to use a library of stock images for their game. We will use the singleton design pattern to fulfill their requirements.

**Requirements**

Web-based gaming app based on the 1980s television game "Win, Lose or Draw," and the requirements are: single instance game, unique identifiers, one or more teams per game, and multiple players per team.

**Design Constraints**

1. Ideal programming language: Javascript

2. How the library of stock images will be stored: TBD

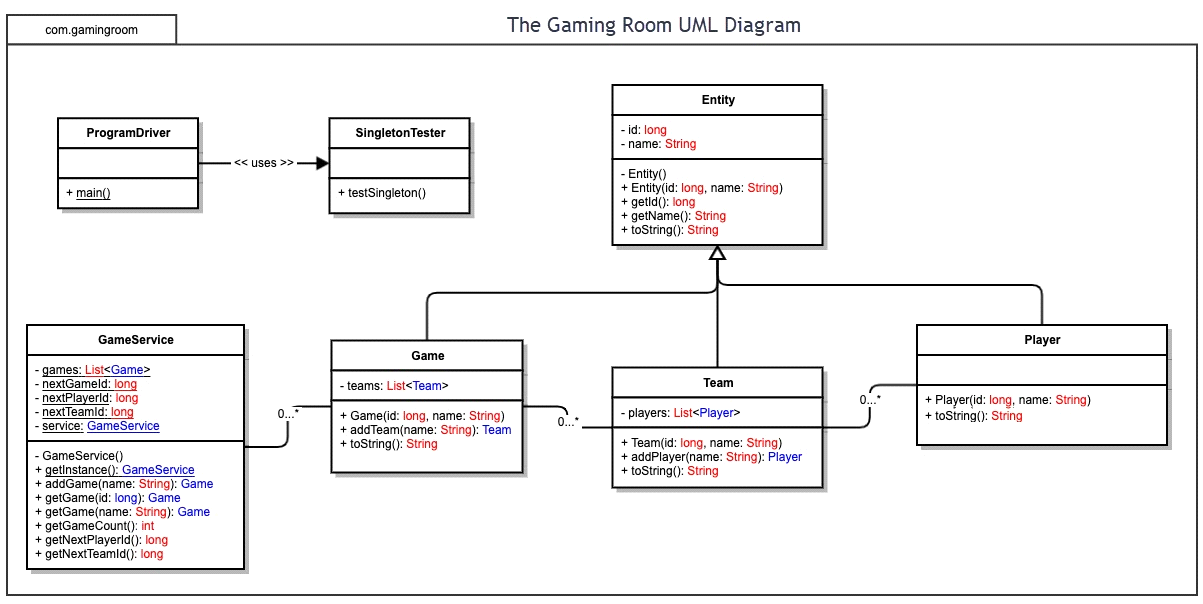
3. Game app requirements points towards the use of the singleton design pattern.

**System Architecture View**

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**

The three classes: Game, Team, and Player, all inherit the Entity base class. The Entity class ensures the objects created have a unique identifier, "id:long, and name: String." It also allows ease of validating for a unique identifier. The class GameService uses the singleton pattern to ensure a single instance of the game as per the requirements. The game class can contains a list of team objects, and the team class can contain a list of player objects. The driver for this program is "ProgramDriver."



**Evaluation**

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** | Proprietary to Apple. Strengths are in its stability, seamless integration with other Apple products, and aesthetic UI. Cons however are it’s very pricey and not many options. | Open source and free. Strength in its security and large number of tools. Requires extensive knowledge to get running from installation to maintenance. | Requires licensing. Not the strongest security. Paying for ease of use with its GUI. Due to Window’s large userbase, has a lot of tools available. | Web-based allows for application to be design for the browser. Not as secure for users. |
| **Client Side** | Developing for the macOS means it’ll work only for Apple products. Apple products are very expensive, but thankfully the operating system is priced into the hardware. | Free to use, and a plethora of tools. Linux servers, however, require more technical knowledge to get running and maintain. | Easier to use at the cost of having to pay for the licensing. | Development for browser rather than mobile app allows for less focus on one OS over another. Cheap to build a web-app. |
| **Development Tools** | C, Swift, C++. Xcode is the typical IDE. | A variety of IDEs are available to Linux. GCC, a collection of compilers, is also great for cross-platform development. | IDE such as Visual Studio. .NET Framework is also available to windows. | Javascript, CSS, HTML. Various IDE available, such as Visual Studio, Xcode, QT IDE |

**Recommendations**

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

* **Operating Platform**:

I recommend Linux as the operating platform.

* **Operating Systems Architectures**:

Linux’s system architecture is design to be modular and layered. It consist of the kernel, system libraries, system utilities, shell, and I/O management.

* The kernel is the core of the operating system, and is responsible for managing system calls, memory management, and process scheduling.
* The system libraries are essential libraries that provide the functions necessary for applications to interact with the kernel. Ex: glibc (GNU C Library) provides standard C library functions.
* The shell and system utilities consist of the command-line interface that allows users to interact with the Linux system by typing commands. The system utilities are composed of the basic tools and commands that users can enter into the shell to perform essential tasks such as file manipulation, process management, and system monitoring.
* The I/O management of the system uses device drivers. The device drivers manage the communication between the operating system and the hardware by translating high-level commands from the kernel into the low-level instruction for the hardware. The system also has Direct Memory Access (DMA) where it allows devices to transfer data directly without involving the CPU, which reduces the workload of the CPU and speeds up data transfer rates. Along with DMA, is the support of asynchronous I/O, which allows processes to initiate I/O operations and while waiting to complete, perform other task along side it.
* **Storage Management**:

For our storage devices, we can partition our storage in what’s called disk mirroring. Disk mirroring is when we replicate the data onto the separate partitions continuously in case of a malfunction. By using RAID (redundant array of independent disks) which generalizes disk mirroring, if one of the partitions/disk were to fail or malfunction, the content is restored. There are different levels of RAID configurations, but the common RAID configuration for a webserver is RAID 5 due to its good balance between data redundancy, performance, and cost-effectiveness. It allows for high read operations and data recovery in case of a single disk failure.

* **Memory Management**:

Linux has a memory hierarchy from top to bottom: physical memory, swap memory, kernel memory, and user-space memory. The kernel memory is composed of a dedicated chunk of the physical memory, and the user-space memory is used for user software and applications. One of the memory management techniques for the Linux operating system that is great for webservers is called swap spacing. Swap space is when the virtual memory is used by the operating system for overflow when the physical memory is running low. Another technique used is called page cache. Page cache is when the memory stores recently-accessed data, which is useful if certain request are frequent. A safety-measure called Out-Of-Memory or OOM is if the system runs out of memory, then the OOM is triggered, and as a protective measure, the system inspects currently running processes and determine which processes to be terminated so the system can free up memory to prevent a crash out.

* **Distributed Systems and Networks**:

If we’re trying to communicate between various platforms, we can accomplish this with the widely established DNS (domain-name system), protocol stacks, and communication protocols established in distributed systems and networks. Because we are trying to communicate between devices that do not share memory, they must go through communication lines to access the resources that the system provides.

* **Security**:

Through the use of cryptography, we can protect user information on our end. Encryption is the obfuscation of content from a sender that allows access only to the receiver with a specific key. And with user authentication, we require the authentication of our users with a username and password. This is a major security risk as our system would need to keep track of these passwords for the authentication. By encrypting the passwords, it makes it harder to crack and allows us to not have to keep a list of the passwords. The encryption is essentially a function designed to be difficult to invert, that hashes the password into a value that is encoded. And so only encoded passwords are stored.