

Draw It or Lose It

# **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 | 9/25/2022 | Cade Bray | Created Executive Summary |
| 1.1 | 9/25/2022 | Cade Bray | Created Design Constraints |
| 1.2 | 9/25/2022 | Cade Bray | Explained the UML document and made connections to OOP paradigm |
| 1.3 | 9/25/2022 | Cade Bray | Completed evaluation for each applicable platform. Completed Recommendations. |
| 2.0 | 10/06/2022 | Cade Bray | Refined responses in Evaluation for each applicable platform and IDE’s for each platform. |
| 3.0 | 10/16/2022 | Cade Bray | Revised recommendations to client. |

**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 client needs to overcome the challenge of creating a development environment for a web-based version of their game. CTS will produce a design documents and develop source code for the project. Key points to accomplish include a game with one or more teams, multiple players per team, unique team names, unique game names, and only one instance of the game can exist in memory at any given time. These key points are defined in the software requirements by our client. To address a potential solution to the unique naming of various object of varying types we will be creating unique identifiers for each instance of each type of object. This will also allow us to keep only the necessary instances in memory.

## [Design Constraints](#_2et92p0)

To identify some design constraints while developing the game in a web-based environment we need to observe these technical and business constraints identified. It should be noted that pending software application decisions the technical manager will review progress and information with client.

Technical Constraints:

* The program should be developed in Java.
* Use the existing application for android as a guideline for the web-based application.
* Only one instance of the game can exist in memory at a time.
* Unique identifiers should be created to avoid reuse of existing team/player identifiers.
* Must create a connection between multiple players with a single team that is amongst multiple teams.
* Multiple platform support is required.
* Multiple browser support is required.
* Hardware requirements will come later and cannot be in cooperated in plans yet.

Business Constraints:

* There is no defined budget and may be subject to change throughout this project.
* There isn’t a specified due date for this project which may be set later.
* The platform must be accessible to all the gaming rooms clients on this web-based platform.

## [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)

Below is a UML diagram of com.gamingroom package. Throughout the explanation of this UML diagram, I will be relating the concepts of the object-oriented programming paradigm to objects and object relations. Starting from the left and move to the right we can see classes named ‘GameService’, ‘Game’, ‘Team’, and ‘Player’. These classes can each request an object of the class type to its left of the quantity zero ore more which is denoted as “0…” on the connection. Classes game, team, and player inherit the “Entity” class which has a foundation that each object should share. To adhere to the clients, request of ensuring only one instance of the object is active at a time there is a singleton design implemented.

**"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** | The cost of this platform for hosting is significantly higher due to a controlled nature on its hardware and licensing for the software. This platform does come with outstanding support through documentation and personable contacts typically. | Linux is extremely customizable and can be tailored to the clients needs if necessary with little overhead costs and overhead hardware requirements. The operating system is often free and known for its natural security and stability. | Windows consists of customizable hardware and slightly controlled licensing with its software. It does provide considerable support unlike its counterpart Linux. Hosting a server would be easily deployable and with great documentation. | Mobile devices would be difficult to deploy server side hosting on because of their controlled state in hardware and low resources because of their inherent nature of being portable. |
| **Client Side** | Client-side software development for Mac could take longer than Windows and Linux due to the controlled nature of licensing and software restrictions imposed by the manufacturer. The cost would also be considerably more because of licensing and time to move about the red tape. | The development for Linux shouldn’t be costly nor time consuming because it’s a widely used platform that is highly configurable and open source. There may be difficulties that arise depending on desired compatibility with distributions. | Windows is regarded as a standard development location for many applications and should have low cost excluding licensing, low time because of familiarity, and low expertise needed for familiarity as well. | Mobile platforms could pose consuming in cost because of licensing and working with many different variations of operating systems. The site itself would need more development time to adhere to varying screen sizes and could use expertise because of its unique nature from the other three platforms mentioned. |
| **Development Tools** | Common programming languages for this platform include but are not limited to Java, JavaScript, Objective – C and PHP. This should be typically across all platforms. We should consider using an SQL server connection if data is to be stored. A typically IDE that allows for flexibility across multiple languages is Visual Studio Code or otherwise known as VS Code. Some developers may prefer CLion or eclipse for their respective languages. | Common programming languages for this platform include but are not limited to Java, C, C++, Python, JavaScript, and PHP. This should be typically across all platforms. We should consider using an SQL server connection if data is to be stored. A typically IDE that allows for flexibility across multiple languages is Visual Studio Code or otherwise known as VS Code. Eclipse would remain the go to development tool for Java. | Common programming languages for this platform include but are not limited to Java, JavaScript, C++, and PHP. This should be typically across all platforms. We should consider using an SQL server connection if data is to be stored. A typically IDE that allows for flexibility across multiple languages is Visual Studio Code or otherwise known as VS Code. It should be noted that C++ is widely used on this platform before many other languages. | The Common programming language for this platform is Java. This should be typical across all mobile platforms. We should consider using an SQL server connection if data is to be stored. A typically IDE that allows for flexibility across multiple languages is Visual Studio Code or otherwise known as VS Code. Mobile development may encompass more languages, tools, or a different IDE due to its unique nature compared to other operating platforms. Eclipse would remain the go to development tool for Java and with mobile development hinging on Java so strongly you could expect this to be used often in combination with Android studios or XCode. |

## 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**: I would recommend the Linux operating system due to its flexibility and low overhead costs. This would allow for easy low-cost scaling that would be a perfect framework for a growing platform. Linux has support for various developments and may even come with desired tools built into the operating system depending on the distribution selected.
2. **Operating Systems Architectures**: Linux uses the fork system call which allows for duplication of a process. When fork is called the kernel will create a new task and copy its data to that task thus both tasks can be executed differently if needed from that point. Linux also supports the system call clone which allows for creating threads. When a task is cloned it shares the data with a pointer on the clone and allows for two tasks to run sharing the same data set.
3. **Storage Management**: Linux will operate with I/O storage and will make memory map system calls to build locations for data. Linux also has memory reserved for the kernel for requests that are less than a page and thus it cleans its data efficiently. The system runs with minimal processes out of the box so contesting resources of storage shouldn’t pose an issue with Linux unlike its competitors.
4. **Memory Management**: Using Linux memory management can be easy because the Linux operating system makes use of system calls to ‘mmap()’. Files between processes will simply have a shared memory map that allows them to access the same data located on the drive. Since we have a singleton design and no more than one instance of an object is active I don’t anticipate the full capabilities for memory mapping on Linux to be fully utilized.
5. **Distributed Systems and Networks**: With communication across multiple platforms and devices we will implement an approach where the server waits for a response from the client device. This initial communication is built by contact the server by its IP address and the server returns a port that they can continue passing information on securely. Once this connection is established data can be pass back and forth via an IP address concatenated with a port number. Since all ports below 1024 are considered well known we will respond with ports greater than 1024.
6. **Security**: Security can be achieved by authentication as the first communication via sockets and thus services and data are not rendered until authentication is complete. If we need to communicate between processes on Linux we can generate a pipe that allows two processes to pass information between each other. The operating system ensures that no other process can leak information from a pipe unless it’s a predetermined process. This is accomplished by a child process inheriting the parent processes pipe via the fork system call.