

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template** 1](#_Toc115077317)

[**Table of Contents 2**](#_Toc115077318)

[**Document Revision History 2**](#_Toc115077319)

[**Executive Summary 3**](#_Toc115077320)

[**Requirements 3**](#_Toc115077321)

[**Design Constraints 3**](#_Toc115077322)

[**System Architecture View 3**](#_Toc115077323)

[**Domain Model 3**](#_Toc115077324)

[**Evaluation 4**](#_Toc115077325)

[**Recommendations 5**](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 03/24/2024 | Roderick Flowers | Made changes to the cover page, document revision history, executive summary, design constraints, system architecture view, domain model and recommendation. |
| 1.1 | 04/12/2024 | Roderick Flowers | Made some changes to recommendation reasoning |
| 1.2 | 04/22/2024 | Roderick Flowers | Made changes to the 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 Gaming Room wants help setting up the environment for developing their web-based gaming app. They need support in streamlining the development process to ensure efficient progress. To do this, I need to make a comprehensive software design document and initial development of the gaming application are necessary to meet their specific software requirements.

## Requirements

The client needs assistance in setting up the environment for developing their web-based gaming application to ensure a streamlined development process. Only one instance of the game should exist in memory at any given time, achievable through the creation of unique identifiers for each instance of a game, team, or player. The game application must have the capability to involve one or more teams, each team within the game must be able to have multiple players assigned to it, and game and team names must be unique to enable users to verify name availability when selecting a team name.

## [Design Constraints](#_2et92p0)

* Each game must have the ability to have one or more teams involved.
* Each team will have multiple players assigned to it.
* Game and team names must be unique to allow the user to check if a name is in use
* There should only be one instance of the game at a time.
* Must be able to run on multiple platforms.
* Language Java
* Framework API

## [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)

Classes Game, Team, and Player have a generalization relationship that references the same general class Entity. These three classes, along with the GameService class, share a direct association and multiplicity, which is shown in the curly brackets that says our classes might share zero to many objects. Multiple players can be added to a team. Multiple teams can be added to a game. The GameService class has a list of games and the singleton method called to service. This class also has references from the game class. We also have a class SingletonTester that is testing if a single occurrence of the game is running at a time associated with the ProgramDriver class. Within the ProgramDriver class, the main method is stored for the terms of use.

**"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** | Mac has both easy server configurability and accessibility. The graphic user interface is easy to use. | Inexpensive, but the platform is difficult to navigate. | The server side is expensive. The graphic user interface is easy to use. It also includes a command prompt. | These specifications differ from device to device. |
| **Client Side** | Expensive. Needs time to understand how to navigate the operating system. | A lot of expertise needed to navigate the OS. Big time investment to learning it. | More expensive than Linux, but easy to learn and navigate. | Very flexible for clients. Easy for the client to learn. |
| **Development Tools** | Can use basic languages such as Java, JavaScript, and the like. Can use IDEs like Pycharm, Visual Studios, etc | Can use basic languages such as Java, JavaScript, and the like. Can use IDEs like Pycharm, Visual Studios, etc | Can use basic languages such as Java, JavaScript, and the like. Can use IDEs like Pycharm, Visual Studios, etc | Uses languages like Go, Swift, and Objective-C as well as the languages like Java, JavaScript, Python, etc |

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’ve changed my recommendation to a Linux operating system to expand Draw It or Lose It to other computing environments. I chose this because of its high level of customization, stability, flexibility, and security. The dedicated community of developers and users provides valuable support and resources to the learning process Linux requires since it isn’t as user-friendly as say Mac or Windows. The cost of using Linux is minimal when compared to the aforementioned operating systems as well.
2. Operating Systems Architecture: Linux has modular architecture with three main components: the kernel, the system libraries, and the user space. The kernel manages resources the system uses like the CPU, memory, and input/output devices. The system libraries give essential functions for the OS and applications. The user space is where the applications that the user sees are run. Since Linux has a layered architecture, it provides customization and flexibility to its build. This makes it a suitable choice for our game.
3. Storage Management: Even though all the operating systems have great storage capabilities, I think Linux has a big advantage over the other two main systems. The file system that Linux has gives a hierarchical structure for organizing files and directories. It also supports many different types of storage devices. Linux offers a variety of storage management systems like the Logical Volume Manager and the Redundant Array of Independent Disks. Linux also has support for cloud-based storage systems.
4. Memory Management: Linux uses an efficient memory management technique known as demand paging that the other two operating systems do not use. Demand paging is a method where pages are only loaded into memory as they are needed. This helps reduce memory usage and improves the overall system performance.
5. Distributed Systems and Networks: Due to its extensive support for various network protocols and interfaces, I recommend that Linux be used for Draw It or Lose It. Linux can establish and maintain communication between different devices and platforms far better that Mac and Windows can. Linux also provides an array of tools and technologies that can manage dependencies and ensure high availability and reliability.
6. Security: When comparing all three operating systems, Linux has certain advantages that put it ahead of the competition. Linux has built-in security features that provide mandatory access control and application sandboxing. These features help prevent unauthorized access to sensitive data.