

<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 | 10/15/2023 | Israel Stillman | <Brief description of changes in this revision> |

**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 aims to create a cross-platform web game titled "Draw It or Lose It." Presently, it is exclusively accessible on Android. The objective of this game is to engage multiple teams, each composed of several participants, in four rounds, each lasting one minute. In a round, one team tries to guess a picture selected from a library of images before time expires. If they are unable to guess in time, each member of the opposing team has 15 seconds to provide an answer.

## Requirements

Software:

* The game will support the involvement of one or more teams.
* Each team will be comprised of multiple assigned players.
* To enable users to verify name availability when selecting a team name, both game and team names must be unique.
* To ensure only a single instance of the game is in memory at any given time, unique identifiers should be generated for each game, team, and player instance.

Business:

* The strategy for broadening the customer base involves reaching out to multiple platforms through a web-based application.

## [Design Constraints](#_2et92p0)

* Needs one or more teams involved
* Each team has multiple people
* Game and Team names must be unique to allow users to check whether the name is in use or free
* Only one instance of the game can exist at any time.
* Must run on multiple platforms

These are the guidelines to adhere to when developing the code and software. While we're presently focusing on the game aspect, we must also consider application development. The Gaming Room aims for cross-device compatibility, meaning it should run on various platforms, including Android, as well as on other mobile devices. Additionally, it should be compatible with operating systems such as Windows, Linux, and Apple. To achieve this, we'll need to explore options like either rewriting the code in Swift (for Apple devices) or finding a way to adapt the existing code to run on different devices by incorporating other programming languages. This involves the practice of integrating multiple programming languages to enhance the overall strength and functionality of the code.

## [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)

The Entity class establishes a connection between the Game, Team, and Player classes, allowing them to inherit or acquire information from Entity. This inheritance relationship can be visually represented using UML diagrams. Consequently, each class will possess shared attributes like "name" and "id", designating Entity as the superclass. Examining their associations, we observe that Team and Player exhibit a "has a" relationship. In contrast, Game possesses a Team, and GameService possesses Games. In UML terminology, this is referred to as aggregation, denoted as "HAS-A". When a user "has a", it signifies that an instance of one class holds a reference to an instance of another class. In this diagram, we note that GameService maintains a reference to Games, Games contains a reference to Team, and Team encompasses a reference to Player.

**"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** | Characteristics: commonly used for web hosting  Pros: upgradeable, availability of diverse options catering to various web hosting needs.  Cons: unfavorable services when web hosting | Characteristics: commonly used for web hosting, most economical  Pros: Early-detection security measures, top choice in web hosting services  Cons:  Locating suitable applications for required web hosting needs can be challenging. | Characteristics: Leading platform amongst other, proprietary platform  Pros: Fast loading, elevated user comfort  Cons: inadequate technical support, vulnerability to viruses | Characteristics: High popularity  Pros: excellent portability, broader accessibility, enhanced compatibility, economical  Cons: poor security measures, high selectivity for various smart mobile devices |
| **Client Side** | Comparable cost to Windows, necessary time investment needed, moderate proficiency | Minimal cost, necessary time investment, optimal proficiency | Comparable cost to Mac, necessary time investment, minimal proficiency | Increased implementation effort compared to other devices, flexible viewing of updates |
| **Development Tools** | Supports Swift & notepad ++, in addition to other languages such as CSS, JavaScript, & HTML. Encompasses general-purpose languages such as Python, Java & C++ | Supports visual studios, eclipse & notepad++, in addition to other languages such as CSS, JavaScript, and HTML. Encompasses general-purpose languages such as Python, Java & C++ | More user-friendly than Linux. Supports visual studios, eclipse & notepad++, in addition to other languages such as CSS, JavaScript, and HTML. Encompasses general-purpose languages such as Python, Java & C++ | Numerous app development using Swift & Android, which are both compatible with all three types of devices |

## 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**: When it comes to gaming platforms, there are distinctions between mobile and PC gaming in terms of usage, user interface, primary gameplay, game design, and even the mechanics of gameplay. It's well understood that all gaming platforms, whether on console, PC, or mobile, possess their unique features, each with their own strengths and weaknesses. These platforms contend for ratings to ensure their sustained popularity. To secure high ratings, all gaming platforms strive to create games of superior quality. These games play a pivotal role in defining the caliber of their respective platforms. Consequently, developers for each platform invest maximal effort in refining their games to compete effectively. Therefore, it is recommended to opt for Windows.
2. **Operating Systems Architectures**: ARM architecture, while not as potent in comparison to x86, does have its limitations. Its maximum potential primarily lies in the realm of mobile gaming.

On the other hand, the x86 architecture boasts significant power. It benefits from a robust development environment, making it easy to create games on platforms like Unity and Unreal. Additionally, x86 enjoys extensive hardware compatibility and support.

1. **Storage Management**: When it comes to storage options, there are a few choices available. HDDs, though reliable, are relatively slower and considered outdated. Even modern consoles are now using SSDs.

Therefore, SSDs are the recommended choice. However, within the realm of SSDs, there are further considerations, namely SATA and NVMe.

For those delving into the intricacies and seeking a system that's poised for future advancements, it's worth contemplating PCIe 4.

Taking both price and speed into account, the Q Cell NVMe SSD emerges as the recommended option.

1. **Memory Management**: As for the recommended operating system, Windows stands out. In a 32-bit Microsoft Windows environment, each process operates within its own virtual address space, allowing for the addressing of up to 4 gigabytes of memory. Conversely, on 64-bit Windows, each process enjoys a vast virtual address space of 8 terabytes. It's important to note that all threads within a process have access to this virtual address space. Nevertheless, they are restricted from accessing memory allocated to a different process. This safeguard prevents one process from inadvertently compromising another.
2. **Distributed Systems and Networks**: Network-based systems facilitating multi-user interactions, particularly in contexts like networked gaming, rely on a shared database accessed by players distributed across physical locations, all interacting through the network. Currently, developers of networked games face the task of building this shared database and establishing protocols for inter-player communication from the ground up.

The paper introduces Artery, a specialized distributed system architecture meticulously crafted to meet the unique demands of networked game applications. Artery achieves this by providing a sophisticated high-level application programming interface and by astutely leveraging application-specific semantics to fine-tune network performance. While the initial implementation of Artery may present challenges, the substantial advantages it offers to online gaming render the endeavor not only valuable but also highly rewarding.

1. **Security**: In terms of security, every game presents a significant risk that, in the heat of the moment, sensitive data might be inadvertently shared. Your IT infrastructure faces numerous threats. On the one hand, there's the danger of data loss or unauthorized access to your account. It could be even more severe if you incurred substantial financial losses. Whether you're gaming on a PC or smartphone, or engaging in online casino activities, there are proactive steps you can take to mitigate these risks.

Giving due consideration to data protection concerns during the development of game software can confer substantial competitive advantages. GDPR regulations mandate data controllers to minimize data collection and processing. Techniques like anonymization or pseudonymization of personal data should be employed wherever feasible.