

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 | 05/23/21 | Jonathan-Biedermann | In this revision, I create the entity class with a long id and string name. I then inherited from this class from the player class and team class. |

**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 wanted a game where only one instance of a team name and a player name can exist at any given time. The game also needs to have the ability to support multiple teams at once with multiple players on each team. Also, each group needs to have the ability to create a new game after each individual session.

## [Design Constraints](#_2et92p0)

The program needs to be able to run in a web-based environment. Another design constraint is that team names have to be unique, as well as player names.

## [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 arrows pointing from the game, team and player classes extending to the Entity class show that these classes are inheriting their functionality from the Entity class. The line extending from the gameservice to the game class shows that there can be one gameservice but that gameserice is responsible for managing multiple instances of the game class.

****

## 

## [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** | **PROS**   * Terminal commands allow for higher flexibility * Open source   **CONS**   * Due to limited hardware, unforeseen production costs may be a large factor * Limited support for software | **PROS**   * Linux systems are known for their reliability * Terminal commands allow for higher flexibility * More cost efficient than MacOs due to being able to run on more types of hardware * Open source   **CONS**   * Lacks widespread driver support * Poor support for software | **PROS**   * Most software is made for Windows * Runs on most hardware   **CONS**   * Closed source * Since it is closed source, any issue with the Os requires you to work directly with Microsoft and their customer service is notoriously bad | **PROS**   * Android is open source * Driver support is relatively stable * The portability of the server is a huge factor   **CONS**   * Having a stationary server is preferable |
| **Client Side** | **PROS**   * Aesthetically pleasing * User friendly * Open source   **CONS**   * Support only for mac hardware * Cost | **PROS**   * Runs on most hardware * Most cost effective * You can change it as you see fit * More stable   **CONS**   * Highest skill level required * Lack of driver support | **PROS**   * Widest availability of hardware and programmers since it has the lowest requirement of skill level * High driver support available   **CONS**   * Closed source * Poor security | **PROS**   * The game is already developed for a mobile platform (android) so limited work would need to be done * The widest customer base out of any of the other operating systems   **CONS**   * Inherent design flaws in the hardware |
| **Development Tools** | Visual Studio Code is a popular IDE for MAC and other platforms which has an intuitive design for software development | Linux has the best support for the most programming languages. It is the go-to programming language for the programmer. | C++ is the primary programming language for making Windows programs. | Java is the most popular programming language for making programs that operate on mobile architectures |

## 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**

Windows OS

Windows game development is user friendly and highly compatible with android sdk/game stores. This choice would allow us to expand our target audience and easily migrate onto mobile, Xbox or other Microsoft devices and even smart iot devices that run on android as well such as raspberry pi, amazon products like the ECHO, and even Renault vehicles or smartwatches such as I am Watch.

1. **Operating Systems Architectures**:

X86\_64 bit as it is the industry standard. Optionally, it would be trivial to offer a 32-bit version as well, but this is no longer compatible with many devices, Windows and otherwise. Ever since ms\_dos Microsoft has been a “closed architecture” company, so we will not be planning on compatibility with any other processors than the intel family.

1. **Storage Management**:

We will use the built in Windows filesystem, the family of FAT filesystems, as it is industry standard and supported by most PC and other OSs. For delivering the game on a flash drive or SD card, exFAT32 version will be optimal and the user will be prompted to copy over all game files onto their native filesystem. We will use the standard four regions used in FAT specifications (reserved sectors, FAT region, root directory region, data region).

1. **Memory Management**:

We will use standard x86\_64 memory management techniques which is built on the concept of paging, virtual memory, and translation lookaside buffers (TLB). Paging for our architecture will consist of 4KB pages for both physical and virtual memory, with Windows 64 capping total address space at 16 TB. TLB allows us to achieve efficient memory caching by storing the mappings between physical and virtual memory allowing fast lookups for both data and instruction addresses.

1. **Distributed Systems and Networks**:

Each round of the game is highly dependent on an accurate timer across all players, a small delay intra-round is insignificant as the game is taking place in privacy over the web. We will depend on this to sync up play results only at the end of each round, the server will control a master clock that each game instance will read from frequently. Therefore, the network layout will be many-to-one connection to the server and all the game instances, limiting the number of potential issues of a player’s weak connectivity. A standard server/client TCP connection (BSD sockets in Java) for the latter will largely suffice for the timer sync heartbeat (e.g. every second) and for round results. If a game instance loses connectivity for a designated period, we can decide if the game should be paused of if that player simply will become a spectator until the end of the game. The results will need to be synchronized so that two players do not think that they won in a single round. We need not end the round once a player has made a correct submission but can simply display before the next round starts which player(s) made correct submissions and at what times they did so thus showing the winner of said round.

1. **Security**:

For security we will follow a defense through isolation strategy, the network enabled port used for server client communication will have minimum access to each player’s device, which can be achieved using strict socket permission and optionally a disk mount on the file system or even a virtual machine. For web communication security we will implement Secure WebSockets which is built on top of TCP and is support by Oracle Java with Spring. For server authenticity Spring Security comes with token-based authentication methods that will ensure each game instance is communicating with the game server and not a malicious user or portjacker.

The game itself will be delivered on a secure and trusted platform such as Steam to avoid viruses or other dangerous executables being part of the download zip or tar. Seeing that there is no PII or sensitive data needed by the game, encryption is not a major concern