

Sean Reid

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.4

## 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 | 3/16/2022 | Sean Reid | Executive Summary and Design Constraints |
| 1.1 | 3/21/2022 | Sean Reid | System, Arch, and Evaluation |
| 1.2 | 3/22/2022 | Sean Reid | OS Grid, Recommendations |
| 1.3 | 4/3/2022 | Sean Reid | Updated grid, recommendations |
| 1.4 | 4/18/2022 | Sean Reid | Update recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

The Gaming Room has a successful Andoid App. They would like to create the game in a multi-player online environment. CTS is proposing a Java based solution that will be hosted on a Linux server to make the game available to users via web browser. This will make the game available on multiple platforms without requiring the user to download any new software.

## [Design Constraints](#_2et92p0)

In re-designing the game to run in a browser, we will need to consider how the browser will communicate with our servers. We will need to be able to receive https requests and respond with in expected ways. CTS proposes building out an API that will serve the pages and game information to the browser. This will mean following some basic standards which will constrain our design in some ways but will offer us some well-established guardrails and direction.

## [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)

At the heart of our API will be the class structure shown in the diagram below. The GameService will be the interface for the API. The GameService will be able to manage a number of ongoing games with multiple teams of multiple players. As shown in the diagram, the GameService has three things it is tracking – Game, Team, Player – each of which are Entities (things that contain an ID and a Name). The GameService will ensure that only one instance of any game, team, and player exist.

By using Java, CTS can use the strengths of the language to ensure quality and security. Encapsulation lets us use the Singleton pattern, keeping users, teams, and games unique. Inheritance lets us easily recreate similar items by abstracting out identical properties to a parent form (in this case the Entity class). We are also able to use features like polymorphism as seen on the Entity class where we can create a new Entity in multiple ways (with a name and ID or with no parameters).

The relationships between classes are shown in the UML Diagram below.

"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)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | Expensive, not widely adopted. | Widely adopted, affordable, with large community support structure. | Expensive licensing, high quality and intuitive interface. | No simple way to use mobile as a server. |
| **Client Side** | Safari browser - possible | Popular browsers for this platform are Firefox and Chrome | Windows comes with the Edge browser | Need responsive browser experience |
| **Development Tools** | VSCode or JetBrains IntelliJ  Docker  Git version control | VSCode or JetBrains IntelliJ  Docker  Git version control | VSCode or JetBrains IntelliJ  Docker  Git version control | Hopefully none of our developers are trying to develop in a mobile environment. |

## Recommendations

1. **Operating Platform**: CTS recommends hosting the application on a modern platform that can run Linux CentOS server. We can start with a single large instance and add more if needed.
2. **Operating Systems Architectures**: CentOS 8 is a modern OS which can run on 32 or 64 bit platforms.
3. **Storage Management**: We will host user, game data, and metadata on a Postgres server with a central write and distributed read instances. Images will be hosted in object storage and made available in edge locations throughout the world for fast loading over the network.
4. **Memory Management**: We will store player, team, game, and image data on a Postgres database with a central write instance and multiple read instances. The API will be hosted on multiple servers using Docker containers, allowing us to increase servers as demand goes up and decrease servers when demand is low. Images will be hosted in object storage and made available in edge locations throughout the world for fast loading over the network.
5. **Distributed Systems and Networks**: All computers will use the standardized http methods to communicate. In this way, any machine with a browser will be able to communicate with Gaming Room servers. CTS recommends hosting these servers with a major cloud provider such as Amazon or Microsoft and having servers available in multiple regions and availability zones.
6. **Security**: Security is our first concern at CTS and to protect the integrity of the systems we will ensure that all data is encrypted in transit and at rest. To do this we will ensure that all traffic coming into the server and out of the server is over HTTPS and that all volumes are encrypted using RSA encryption with rotated encryption keys.