

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)** 33

**[Evaluation](#_2o15spng8stw)** 3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 11/13/22 | Karl Konz | Technical documentation to accompany implementation of Entity, Team, and Player classes. |
| 1.1 | 11/27/22 | Karl Konz | Update Evaluation of app development |
| 1.2 | 12/11/2022 | Karl Konz | Recommendation Submission |

**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, The Gaming Room, will be extending their code base for a game, Draw It or Lose It, to facilitate web-based functionality as a platform agnostic application. In addition to the web-based environmental aspects of this development task, the game will also be updated to incorporate handling multiple teams of players.

## [Design Constraints](#_2et92p0)

* Retain current code base for a Java backend to a React application for UI, due to its support multiple devices
* A game will 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 users to check whether a name is in use when choosing a team name.
* Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

## [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 is a parent to the Team, Player, and Game classes. This means that the child classes can derive logic from the Entity class to access and mutate attributes of an object such as the entity id and name.

The Player, Team, Game, and GameService all have a chain of “has-a” relationships.

The Team class has a list of zero or more Player class objects. The Game class has a list of zero or more Team class objects. Additionally, the GameService class has a list of zero or more Game class objects.

**"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** | For Development purposes, a local install of a Tomcat Server can be used for the back-end java code. Installation can be done via **brew** with additional configuration as needed. Since the application will be deployed with React Native and Kubernetes, it will be OS agnostic.  **Mac is not available as an OS to host the EKS deployment.**  Cost may be a deterrent as macs are typically more expensive than other pcs. | For Development purposes, a local install of a Tomcat Server can be used for the back-end java code. Installation can be done via **wget** with additional configuration as need. Since the application will be deployed with React Native and Kubernetes, it will be OS agnostic.  Control plane for container orchestration layer can only run on Linux.  Currently AWS offers Linux OS for hosting a managed Kubernetes cluster.  Linux is generally cheaper to run in AWS than Windows.  Cheaper than mac, but some developers may prefer a mac OS | For Development purposes, a local install of a Tomcat Server can be used for the back-end java code. Installation can be done via **CURL** installation. Since the application will be deployed with React Native and Kubernetes, it will be OS agnostic.  To run Kubernetes for Windows containers, the Kubernetes cluster must include multiple operating systems since the control plane for orchestration can only run on Linux.  Windows is generally more expensive to run in AWS than Linux.  Currently AWS offers Windows OS for hosting a managed Kubernetes cluster. | Not feasible/ reasonable |
| **Client Side** | Front-end will be developed into a React Native application. A mac-based device can be tested via an emulator within the IDE of choice.  Browser:   * Chrome * Safari   Cost/Time:  Cost of macs tend to be more expensive than other pcs.  No substantial impact on time | Front-end will be developed into a React Native application. A Linux based device can be tested via an emulator within the IDE of choice.  Browser:   * Chrome * Firefox   Cost/Time:  Cheaper than mac, but some developers may prefer a mac OS  No substantial impact on time | Front-end will be developed into a React Native application. A Windows based device can be tested via an emulator within the IDE of choice.  Browser:   * Chrome * Edge   Cost/Time:  Cheaper than mac, but some developers may prefer a mac OS  No substantial impact on time | Application will be written using a mobile first design. Same code base will be used and tested for various devices through emulations in the IDE of choice  Browser:   * Chrome   Cost/Time:  NA |
| **Development Tools** | * VS Code * Eclipse * XCode | * VS Code * Eclipse | * VS Code * Eclipse | Not feasible/reasonable |

## 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**: The application will be developed on local PCs by developers and version controlled through GIT and GitHub. The application will be built with a Jenkins pipeline into a docker container. The deployment of the application will be done through AWS EKS (Elastic Kubernetes Service), via a helm chart which is a package manager for Kubernetes. The application will be provided as a SAAS, software as a service, offering.
2. **Operating Systems Architectures**: Since the application will be running through a docker container it will virtualize the OS. This allows for the container to share the OS from the AWS Kubernetes cluster, which are available in either Linux or Windows. Windows is typically more expensive than Linux and running Kubernetes with Windows requires multiple operating systems. Even though this application will be running through a managed Kubernetes service, it is still recommended that a simpler OS architecture is used. For these reasons of cost and simplicity, running the Kubernetes cluster on Linux is recommended.
3. **Storage Management**: Long term storage will be achieved through an AWS DynamoDB. This option is typically cheaper than using an RDS database. Since this application is a transactional system and does not require multi-indexing and expensive queries, it should not require the overhead of an RDS database.
4. **Memory Management**: Memory management will occur at multiple levels in this application. The backend will manage memory through Java. Heap and thread stack configuration can be handled within Java with the -Xmx and -Xss commands respectively. From there, memory management is largely automated by the built in Java Garbage Collector. This will automatically delete objects from the program memory Heap when they are no longer being utilized. In addition to the Java level memory management, Kubernetes has its own memory management. Kubernetes uses NUMA, non-uniform memory access, which is a method of configuring a cluster of microprocessors in a multiprocessing system so they can share memory locally. The way that Kubernetes implements memory management guarantees QoS class, which is a way to classify pods in a cluster to make decisions about scheduling and eviction.
5. **Distributed Systems and Networks**: Kubernetes has a HorizonalPodAutoscaler, which automatically updates a workload process to meet demand with deploying more pods. This works both with scaling up and down to match demand respectively. Additionally, with this cluster, AWS Load Balancer Controller will be implemented on the Kubernetes cluster to help with load balancing and simplification of management of the cluster and application.
6. **Security**: Security settings can be configured in AWS along with Identity Access Management controls. Within this, users will be assigned to security groups and resources will be restrict to ports with specific access control lists. Gaming users of the application will have the option of using single sign-on. Additional access will be given for developers and administrators which will only be able to be accessed through a Virtual Private Network and will require 2 factor authentication with SSO. Additionally, a web service for the application backend will be available through routing and service endpoints to the front end which will give extra controls to the backend resources.