

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 1.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_heading=h.gjdgxs)1[\_heading=h.gjdgxs](#_heading=h.gjdgxs)

[**Table of Contents**](#_heading=h.30j0zll)2[\_heading=h.30j0zll](#_heading=h.30j0zll)

[**Document Revision History**](#_heading=h.3znysh7)2[\_heading=h.3znysh7](#_heading=h.3znysh7)

[**Executive Summary**](#_heading=h.2et92p0)3[\_heading=h.2et92p0](#_heading=h.2et92p0)

[**Design Constraints**](#_heading=h.tyjcwt)3[\_heading=h.tyjcwt](#_heading=h.tyjcwt)

[**System Architecture View**](#_heading=h.3dy6vkm)3[\_heading=h.3dy6vkm](#_heading=h.3dy6vkm)

[**Domain Model**](#_heading=h.1t3h5sf)3[\_heading=h.1t3h5sf](#_heading=h.1t3h5sf)

[**Evaluation**](#_heading=h.2s8eyo1)3[\_heading=h.2s8eyo1](#_heading=h.2s8eyo1)

[**Recommendations**](#_heading=h.3rdcrjn)5

## [Document Revision History](#_heading=h.3znysh7)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 3.0 | 06/26/2022 | Nick Nealeigh | Final |

## [Executive Summary](#_heading=h.2et92p0)

The Gaming Room wants to create a game called Draw It or Lose It. The application should be ported from Android to a web based platform to reach a broader audience. The game should allow multiple teams and players with unique handles. The singleton pattern should be employed to ensure only 1 instance of the game can be open at a time.

## [Design Constraints](#_heading=h.tyjcwt)

* Web based application should function in any browser, regardless of OS.
* Employ Singleton pattern to ensure 1 instance of the game is open at a time.
* Game and Team names should be unique, with a method to check if a name is in use already.
* Games can house multiple teams, which can house multiple players.
* At least 2 teams of 1 player each to play.

## [System Architecture View](#_heading=h.3dy6vkm)

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](#_heading=h.1t3h5sf)

Entity links the game, team, and player classes together, with all 3 inheriting from Entity. GameService contains most of the code and manages how many instances of a game can be open(1). ProgramDriver is the main file and executes commands from GameService, with SingletonTester existing to ensure that the 1 instance per game is functioning. GameService references Game, which references Team, which references 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](#_heading=h.2s8eyo1)

| **Development Requirements** | **Mac** | **Linux** | **Windows** | **Mobile Devices** |
| --- | --- | --- | --- | --- |
| **Server Side** | As far as I am aware, macOS is not a popular platform for server hosting.  Can use Mac OS X Server for web hosting. | Linus is likely the most popular operating system for hosting web-based software applications. Linux is known as the most secure system.  Not user intuitive.  Server-Hosting methods exist via LAMP (Linux, Apache, MySQL, and PHP). | Windows is the most ‘in use’ operating system and likely offers the most compatibility for other systems. Many individuals are already comfortable working with Windows. It is less secure than Linux however.  Azure cloud server hosting. | A mobile OS is likely a poor choice for hosting an application.  Does not have server-based deployment method. (some applications can act as a server, but it is not the same as a true server) |
| **Server Side Costs** | Minimum 20 licenses at $29.99  Mac OS X Server at $49.99 | Free | Windows Server 2022 Edition Essentials  $501 |  |
| **Client Side** | More expensive than Windows. Requires more specialized familiarity with navigating MacOS. | Cheapest platform to work on. Requires the highest level of familiarity to work on efficiently. | Should be considered standard cost. Requires least level of familiarity to work with. | Flexible and easily accessible for use. |
| **Development Tools** | More limited than Linux and Windows. Likely fine for this purpose. Can use Swift, but this is more relevant for mobile applications.  HTML5/CSS/Javascript compatible. | Similar to Windows, Linux should be compatible with any language and most IDE. Requires more knowledge to navigate terminal.  HTML5/CSS/Javascript compatible. | Easiest to utilize various IDE on. Visual Studio preferred IDE for web development.    HTML5/CSS/Javascript compatible. | Likely more limited than other OS for development options. Netbeans can program in  HTML5/CSS/Javascript compatible on android 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:

* **Operating Platform**:
  + I’d like to recommend AWS serverless hosting for the purpose of Draw It or Lose It’s deployment. AWS is flexible and will be able to deploy the application on a variety of systems. Combining Amazon S3 for the code, then using Amazon API gateway to act as a trigger for AWS Lambda seems to be a streamlined approach that amazon has designed for web applications.
* **Operating Systems Architectures**:
  + By referring to the reference architecture diagram for AWS Modern Serverless Mobile/Web Application, there is a clear order of processes:
    - Users authenticate via Amazon Cognito user pools by retrieving the JWT token, and they then use those tokens to retrieve AWS credentials that allow their app to access other AWS services.
    - The web/mobile client interacts with AWS Amplify frameworks, which allow communication with backend services with iOS, Android, web, and React Native front ends.
    - The authenticated clients make API calls to AWS AppSync to perform GraphQL operations such as queries, mutations, and subscriptions.
    - The AWS Lambda resolvers communicate with Lambda with temporary AWS Identity and Access Management (IAM) credentials based on assumed IAM roles. A JWT token specific to the authenticated user is forwarded to Lambda for processing.
    - The HTTP resolver and the endpoints are protected with temporary IAM credentials based on assumed IAM roles. A JWT token specific to the authenticated user is forwarded to Amazon API Gateway.
    - The Amazon DynamoDB resolver enables connecting existing tables to a GraphQL schema by creating a data source to read, write, and subscribe to real-time data.
  + This process is setup in such a way that front end logic is handled on the client’s machine, while the back end logic, security, and the database are all handled through AWS Lambda and other services
  + <https://d1.awsstatic.com/architecture-diagrams/ArchitectureDiagrams/mobile-web-serverless-RA.pdf?did=wp_card&trk=wp_card>
  + https://www.gocd.org/2017/06/26/serverless-architecture-continuous-delivery/
* **Storage Management**:
  + App storage should also be handled through Amazon, with their Simple Storage Service, allowing The Gaming Room to scale their storage needs. As the application will be hosted using Amazon services, storing the data with Amazon S3 makes it accessible. Amazon S3 has different storage classes that allow for different speeds of data retrieval.

* **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>
  + AWS lambda allows you to control the memory allocated to a function, between 128MB and 10,240 MB. Depending on how memory intensive a function is, we can allocate either less or more memory to it.
    - <https://docs.aws.amazon.com/lambda/latest/operatorguide/computing-power.html>
  + Using Caching in the application, we can reduce latency. As we know that certain components of the application are going to be reused, we can cache that data and make it more easily accessible.
    - https://aws.amazon.com/caching/
* **Distributed Systems and Networks**: <Knowing that the client would like Draw It or Lose It to communicate between various platforms, explain how this may be accomplished with distributed software and the network that connects the devices. Consider the dependencies between the components within the distributed systems and networks (connectivity, outages, and so on).>
  + As the application is going to be a serverless, AWS app, most of the concerns such as outages and connectivity will be out of our hands. Client-side connectivity is dependent on the client’s machine and connection, while the bulk of the application would be handled server side. As this is a multiplayer “party” game, where players draw images, changes that one client makes, such as drawing their image, should be reflected on other player’s clients via the server.
* **Security**:
  + **Security is important**. On our side, we can use good programming practices such as encapsulation to minimize vulnerability within the code. The bulk of the security should come from AWS’s robust security systems. By combining methods such as Identity and access management, and data protection in the form of encryption, we can eliminate outside interference with the application.
    - https://docs.aws.amazon.com/lambda/latest/dg/security-iam.html