

<Draw It or Lose It>

# **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 | 11/12/2023 | Chase rush | Executive Summary |

**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 software design would be to develop a gaming application that in a web-based distributed environment.it has been asked that the application is created with java object-oriented programming principles. With our application this would include classes of the following Team,Game,GameService and our player class these would be interacting with each other to further manage our game,Team and Players with this design I hope to further leverage the entity class this would help with giving a common base for all entities including attributes like the name and ID.I am trying to achieve with this approach a more efficient application and better functioning game!

## [Design Constraints](#_2et92p0)

**Web-Based Distributed Environment:** With the game application needed to be developed in the web-based distributed environment this give us the issues implying that the application should be accessible over the internet and support multiple active users this would also need the application to handle latency and have some security design to the product.

**Java Programming Language:** With the software we should make sure that this is being developed only using Java programming language. This could be a big constraint because we limit the technology stack to our java-based frameworks, tools, libraries. This would also be adhering to java coding and best occupational standards.

**Scalability and performance:** The gaming program should have been designed to handle a growing number of teams and players. This should also be scalable to accommodate more active users increasing the community for the game bringing in more people. This constraint mean we would need to optimize the database and try to minimize resources as much as possible.

**Object-Oriented Design:** The application would need to follow the object-oriented design principles this is to further promote modularity and better maintainability. This constraint would require making classes with the needed encapsulation and polymorphism. It should also use and apply design patterns and abstraction techniques this is to further help for a flexible and extensible codebase.

**User Interface:** we should make the applications user interface the most user-friendly and be intuitive and responsive. We should also try to give a engaging time for the players giving them more to interact with in the games and teams. The constraints here would be making this responsive and appealing to the players using this web technology.

## [Domain Model](#_8h2ehzxfam4o)

The UML class diagram shows the domain model of our gaming application. This consists of different classes, that being GameService,Game,Player and Team this is all interconnect to fulfill our program requirements.

With the Entity class this serves as our base for all the entities within our application, this helps by proving common attribute like ID and name. this would also show the principle of inheritance, this also allows the other classes to take in these behaviors. This is by utilizing the Entity class.

With the GameService class this acts as the central service for our games management. This also maintains a list of our games by Name or ID and would further retrieve the total amount of games. This class attempts to follow a singleton design by making sure the single instance is accessible throught our program.

With our game class this would represent the game within the program. This also helps by showing a list of the current participating teams in our game also by providing methods to add to the team.with the player class this shows a individual player and gives a method of string representation

All of these classes show the principle of composition, this is by complex object being built to combine with simpler objects. With our game the class would use teams with each team this would have players. The structure allows for hierarchy to manage the games and players.

**"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** | Mac operating systems are knows for their stability and security. These systems would mostly by used for web development or hosting a web application. | Linux can be a popular choice for hosting a web-based application this is due to the open-source nature, reliability, and the scalability. | Windows servers would commonly be used for hosting web applications especially those built with Microsoft technologies like ASP. NET. Windows provides a very user friendly experience | Mobile device have a lot of limitations in terms of hosting a web-based software program directly. These would be more commonly used by clients to access the server hosting the application basically for these devices. |
| **Client Side** | Supporting clients with mac has considering factors such as development costs, time, this is because mac users are generally accustomed to a very seamless and visually appealing user experiences so developing for mac can take a lot of additional testing. | Linux users often like software that aligns with the open-source principles. Supporting Linux clients might be considering the diversity of the package management systems and development cost and this can be lower than expected | Windows users have a significant portion of the user base developing these means development cost and expertise may be a lot easier to find for this operating system the development tools for windows are very well documented and user friendly. | Support clients on mobile devices would be crucial for our games success. These is development consideration to intake but this can do a lot for the company providing the game |
| **Development Tools** | With Mac development, developers would commonly attempt to create this in Xcode the integrated development environment, this also supports languages such as Swift and Objective-c. | Linux development would usually involve popular text editors or IDE like Atom or eclipse, even visual studio. The choice of programming language can vary but python and c++ are common in linux | Windows development would often be done using Visual Studio being the primary IDE, this supports languages like c#,C++ and visual basics also Microsoft provides a range of tools and resources. | For the mobile program development this can support languages like swift and objective-c and android studio also android provides a range of tools for designing and coding. |

## 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**: I could only recommend Microsoft given the requirement to expand draw it or lose it to other computing environments, a web-based platform would be highly recommended. And Microsoft has many tool and resources to support the building of the program and can be very cost efficient.
2. **Operating Systems Architectures**: With the chosen operating system for our web-based program typically we would utilize a client-server architecture. This is because the client side is responsible for rendering the game interface and further handling user interaction the architecture would ensure scalability and further ease of maintenance.
3. **Storage Management**: with a web-based application, I feel a cloud-based storage would be recommended. Services like google could storage or Microsoft Azure are very scalable and reliable storage this can be for images game data and user information this could would allow for the user base to grow and provide less redundancy to ensure data integrity.
4. **Memory Management**: The recommended operating platform, being that our web-based relies on the client’s browser for the memory management. Modern browsers would handle memory efficiently, this could help developers optimize the client-side code and this would further lessen or minimize memory usage.
5. **Distributed Systems and Networks**: To facilitate the communication between these multiple platforms we should implement a (representational state transfer). This would allow for different components to communicate over the standard HTTP protocols. Also the Cloud-based services. This is such as AWS and Azure functions would be able to be further utilized for the serverless architecture enhancing the scalability and reducing the dependencies on a specific server.
6. **Security**: Security would be paramount as this holds the user information and this should never be just public. Along with best practices in the web development we should also include HTTPS to be able to encrypt and transfer the data. As well as user authentication and authorization mechanisms should be implemented and meeting industry standard protocol.