

The Gaming Room

# **CS 230 Project Software Design Template**

Version 1.1

## 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 |
| --- | --- | --- | --- |
| 1.1 | 07/15/2023 | Justin Swinney | Implementing new class |

## [Executive Summary](#_sbfa50wo7nsh)

Creative Technology Solutions has created an overview of The Gaming Room’s request for a cross platform gaming application for their current game Draw it or lose it that is currently in service on android devices only. The game needs consist of four rounds lasting one minute each with a drawing completion rate of 30 seconds. If 30 seconds are not met the puzzle expires and another team can solve the puzzle with 15 a second window.

## Requirements

* Multiple team involvement.
* Multiple players within a team.
* Game and team names must be unique.
* Check for existing team and player names are in use to avoid duplication.
* One instance of the game existing in memory at any given time.

## [Design Constraints](#_2et92p0)

* Screen size for multiple devices on different operating systems and phone and tablet sizes.
* Internet connectivity.
* Cross platform server connection.
* Budget.
* Time constraints
* Design system constraints regarding style of preexisting application will these system designs be interchangeable across multiple operating systems.

## [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 UML class diagram provides us with a clear view of an object-oriented principle called inheritance as the game, team and player class extends or inherits information from the entity class. These classes share the references name and id to eliminate repeat code and allow for future implementation of additional features that may inherit certain objects from classes later. Team references player, games references team, game service references games. Within these classes they share relationships such as game service and games while team and player share a has a relationship.

**"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** | **Characteristics**   * Unix based operating system that provides a secure, stable, and efficient foundation for hosting web applications. * MacOS has a built-in web server called “Apache” that is easy to set up and allows for local hosting for testing purposes. * MacOS is known for its security and stability of the OS that could be important when hosting web applications.     **Advantages**   * Security * MacOS is known for its friendly user interface that could benefit server management. * Server Stability * Server-side tools are open source and free of cost, but commercial grade server implementations can be purchased if necessary.   **Weakness**   * Mac hardware can be costly to replace and provide limited upgrades compared to a windows-based server. * Scalability, MacOS would not be an ideal choice for a large web-based application as scalability would not be accomplished easily as it would require a lot more computing power. | **Characteristics**   * Open source therefore its free to use. * Preferred choice for hosting due to its stability and reliability. * Security * Large software support * Great performance * Great scalability   **Advantages**   * Security * Cost effective. * Performance * Flexibility to adapt to server needs. * Reliability * Open source   **Weakness**   * Linux requires a large amount of knowledge to fully understand and utilize the potential of its hosting capabilities. * Hardware support may vary and produce better results on a different operating system. * Not beginner friendly for hosting. | **Characteristics**   * User friendly operating system * Support for Microsoft services * Wide range of development tools * Dedicated server operating system designed for web hosting.   **Advantages**   * Windows is widely used, giving an advantage to server management staff. * Support for Microsoft services such as Microsoft SQL server. * Large support system for issues. * Hardware upgrades   **Weakness**   * Licenses cost will be higher compared to Linux based. * Security * Resource intensive * Less flexibility compared to Linux based web server. | **Characteristics**   * Mobile internet connectivity * Portable * Equipped with multiple hardware features such as GPS.   **Advantages**   * Convenience * Accessibility * Better user experience tailored to touch screen mobile devices.   **Weakness**   * Limited processing power and memory * Battery life * Connectivity * More likely for downtime * Limited screen size * Not ideal for server hosting. * Scalability |
| **Client Side** | * License for deployment tools such as IDE’s * Cross platform testing equipment * Strong expertise required in building a responsive web application design, browser compatibility, UX design. * Time restraints for ensuring custom interfaces for individual platforms, cross browser testing and mobile devices optimization | * Cross browser testing tools can be costly. * Creating a responsive design that adapts to different operating systems will take time and cost. * Testing in general will take a chunk of time and having the resources and devices to test on will require money. * Linux requires a large amount of expertise to understand Linux specific items like configuring client-side web applications to ensure compatibility. * UX design and front-end development skills will also be required as well as mobile development knowledge. | * Cross browser testing will require tools and time. * Responsive design requires extensive testing to ensure the application will be optimized to multiple devices requiring multiple devices to be purchased and time. * Development tools will need to be purchased and frameworks should be cross compatible. * Front-end development skills will be required. * Knowledge of client-side frameworks like React or angular or Vue. * Knowledge of mobile development and window specific optimizations. | * Flexibility for updates at any location. * Device testing cost and time * Development time and tools will cost. * Testing to ensure optimization across platforms. * Front end development skills * Mobile server and application development knowledge. * Mobile specific features will need to be tested and implemented. |
| **Development Tools** | Swift  Xcode  Java script  HTML/CSS  Ruby  PHP  Python  Visual studio code  IntelliJ  Visual studio  React  Angular  Bootstrap  Some IDEs will require licenses, multiple development teams would improve the deadline of the application as agile principles could be followed and testing could be done on a regular basis as the application is developed. | Visual studio code  WebStorm  Atom  Java script  HTML/CSS  Python  PHP  Ruby  Java  Golang  Some IDEs will require licenses, but some are also free. Building a development team that is familiar with free programs or that can adapt quickly would be beneficial to cost. Multiple development teams containing several engineers, and testers and front-end developers would be preferred. | Visual studio code  Visual studio  WebStorm  IntelliJ  Eclipse  Java script  TypeScript  HTML/CSS  C#  Java  Python  PHP  Ruby  Multiple development teams with plenty of testers and engineers and front-end developers will be required as well as selecting preferred languages and IDE to use. | Visual studio Code  WebStorm  Adobe Dreamweaver  HTML/CSS  Java script  React  Angular  Vue  jQuery Mobile  Multiple Development teams would be preferred. |

## 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**: <Recommend an appropriate operating platform that will allow The Gaming Room to expand Draw It or Lose It to other computing environments.>
   1. A server-side operating platform The Gaming Room could use to expand Draw it or lose it to other computing environments could be a java-based platform, Java is supported on many operating systems and uses bytecode to compile the program, A java supports robust scalable server-side applications and would be a good fit for this task.
2. **Operating Systems Architectures**: <Describe the details of the chosen operating platform architectures.>
   1. Java uses JVM or java virtual machine architecture that allows java’s bytecode to be compiled and executed on many different operating systems. The JVM architecture uses a class loader subsystem that is made up of bootstrap class loaders that will load core java classes, an extension class loader that will load class from extension directories, and an application class loader that will load classes from an applications class path. The JVM architecture also includes runtime data areas like methods, heap, java stack, PC register and execution engine. Other features include direct memory access, security verifications.
3. **Storage Management**: <Identify an appropriate storage management system to be used with the recommended operating platform.>
   1. Given the usefulness of the cloud and the budget friendly scalability of cloud storage I would recommend using a cloud-based solution like Amazon S3 for storing user data and any requirements of the application, the scalability of cloud storage allows for a “use only what you need” method that can be billed based on usage and scaled up if needed.
4. **Memory Management**: <Explain how the recommended operating platform uses memory management techniques for the Draw It or Lose It software.>
   1. Java platforms manages memory using automatic garbage collection technique. The garbage collection feature automatically finds memory that is no longer in use and reclaims this memory to optimize resources. JVM heap memory allows for dynamically stored objects to balance memory, JVM uses memory pools for specific functions these memory pools are also handled by the garbage collection. The garbage collection can be tweaked to the applications specifications with the use of algorithms and configuration settings to ensure proper memory management based on the applications needs as well.
5. **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).>
   1. Using the java operating platform, we can integrate Restful API with a set of endpoints to communicate from the client to the server, implementing a cloud resource like load balancing would always ensure a stable connection for all users. Implementing an offline mode that would ensure users could continue playing and progress would be stored until connection to the server is restored. Using caching would help reduce the dependency on the server to improve connectivity and responsiveness. Error handling mechanisms could be implemented to notify the user of their current connectivity issues and retry the connection if an offline mode is not available. e
6. **Security**: <Security is a must-have for the client. Explain how to protect user information on and between various platforms. Consider the user protection and security capabilities of the recommended operating platform.>
   1. Implementation of user authentication and authorization would ensure that user information is secured within an individuals account using standard protocols like OAuth would ensure that user information is secured, and only registered users would be allowed to access resources. When creating the program following secure coding practices would help prevent any vulnerabilities or attacks like an SQL injection. Data encryption would help protect data when it is in transit from place to place. This can be accomplished by using encryption algorithms like AES. Rolling out patches and continuously monitoring the application searching for vulnerabilities would also improve security.