``

<Name-of-Software-Application>

# **CS 230 Project Software Design Template**

Version 3.0

## 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 | 07/28/2024 | Joseph Eaton | Wrote down the executive summary, requirements, and design restraints. |
| 2.0 | 8/10/2024 | Joseph Eaton | Evaluated and took down the pros and cons of various platforms. |
| 3.0 | 8/23/2024 | Joseph Eaton | Analyze the characteristics of and techniques specific to various systems architectures |

**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 Gaming room is seeking to develop a web-based game that is available on multiple platforms based on their current game called “Draw It or Lose It”. Currently the game is only available on android. The purpose of the game is for teams of 2 or more people. In which an image is drawn at a rate of being completed in 30-secounds and if one team does not guess it the opposing team will get to guess it in 15 seconds.

## Requirements

* *The game must include multiple teams.*
* *Each team must be able to have multiple players.*
* *Each Team must have unique naming*
* *Only one instance of the game can exist at any given moment in memory.*
* *Each round will be 1 minute.*

## [Design Constraints](#_2et92p0)

* The game must be web-based.
* The game and team names will need to be unique. Meaning there must be a way to check for unique names
* Since the game must run on multiple platforms it must be compatible with multiple OS
* *The game must include multiple teams.*

Alot of our design constraints are also requirements for the game so there will not be a lot of give when it comes to acceptations. Having to make the game functional on multiple OS means that what we develop must work for everything. We will also ensure we have multiple teams since this is a team game, meaning we need to be able to accept two instances of teams. Having the game be web-based will change how we go about coding and developing how it works since it will run out of browsers.

## [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)

<Describe the UML class diagram provided below. Explain how the classes relate to each other. Identify any object-oriented programming principles that are demonstrated in the diagram and how they are used to fulfill the software requirements efficiently.>

**When looking at the uml diagram we can see that the Entity class shares a relationship with the Game Class, Tean class, and Player class. That relationship being the fact that all three of them inherit attributes and methods from the Entity class. The GameService class and Game class relate with a solid line and a 0... meaning there is a one-to-many relationship showing that one instance of Game Service may be related to many instances of Game Class. Likewise, the Game class is the same as Team class showing that there is a one-to-many relationship between the two showing that for every GameClass there are multiple instances of teams. And this is the same for Team class to player class.**

**The final connection is the ProgramDriver and SingletonTester class which uses a solid straight line with a closed arrowhead meaning that the Program Driver class uses the SingletonTester class to run the program to design specifications.**

**"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 allows for a reliable and stable server environment Since its Unix based which gives it a robust amount of security features allowing it to be well suited for web hosting with the drawback of having some limited server options compared to Linux which is more common. The cost tends to be higher due to needing Mac OS machines to be able to work with it and the licensing cost tends to be higher. | Linux is a great open-source platform that is extremely efficient when it comes to the server side of things. With it being open source, the maintenance and licenses costs will be cheaper than other closed Oses. It requires more expertise compared to the other platforms. The con of Linux is that it is not user friendly and requires a lot of prior knowledge. | Windows servers come with the pleasure of being GUI based which makes it more user friendly. It has a lot of familiarity with other windows-based machines allowing for an easier understanding. Windows will also have a lot more supporting software for it since it is the most common OS. For that same reason security can be an issue since again it is the most common OS meaning attacks on it are more common. License costs are extremely high comparatively to Linux and has less hosting options compared to Linux. | The advantages that would come with mobile devices is that they can all access web-based apps but when it comes to hosting, we would have to go through cloud hosting while nice because of how scalable it is, would then be hard to control the server hosting environment. |
| **Client Side** | When looking at Mac a person with moderate expertise would be able to do the work if there was prior knowledge of Mac's programming languages of Swift and Objective-C. In terms of time a person with knowledge of the Mac systems should not be an issue. And Cost will only be an issue since a Mac device would be required. | When looking at linux the cost will be the lowest of the three main OS due to most of the IDEs and Build Systems and libraries being all open source. The expertise will be higher due to Linux not being super user friendly and a lot of prior knowledge will be needed to ensure the security and development through Linux. Time will also be quite consuming due to needing extensive testing for ensured compatibility. | Windows will be the go-to in terms of not needing a super high expertise and cost on the client side due to all the support and understanding that comes with working on a Windows based machine since there is so many supporting materials and knowledge with Windows is quite common meaning the time aspect will also be shorter. | When developing for mobile devices the time will be a crucial factor since if there will be multiplatform support acros iOS and Android it will lead to more time and a higher cost in terms of wanting to be able to test compatibility across multiple devices. Having someone with a high expertise in mobile development is necessary since they both will have their specific languages and IDEs that are needed to perfect and optimize for multiple devices. |
| **Development Tools** | Some languages that would be used are HTML, JavaScript and CSS, Swift one of Apples primary languages, Objective-C and older language that is still used in some of the older macOS applications, C/C++ and Python for scripting. The IDES that would commonly be used would be Xcode which is apples official IDE, Visual Studio Since it so versatile, and AppCode which was created by JetBrains for macOS development. | The relevant languages that will be used for Linux will be c/c++ python, java, Go, Rust, and JavaScript. In terms of IDEs used will be Visual Studio Code, Eclipse, CLion and NetBeans. | The relevant languages used with windows development will be similar to Linux but will include C# as it is the primary language used with windows. Common IDEs Will include Visual Studio, Visual studio Code, JetBrains Rider for C# Eclipse and PyCharm. | The main languages that will be used will be Swift and objective-C for iOS. For android it will be Java and Kotlin. With Dart being used with Flutter to allow for Cross-Platform development. The main IDEs will be Xcode, Android Studio, with Visual Studio Code, Eclipse and being used for the Cross-Platform IDEs. |

## 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**: When looking at the potential operating platforms to choose from the one that should be best fit for the client is windows. This can be said because of how user friendly the platform is, allowing for a lower expertise requirement to work with it unlike Linux. The platform also has a large amount of support for software compatibility and different IDEs to choose from. With windows being a common platform there are a ton of options to choose from in terms of developers since most developers will be well versed in windows.
2. **Operating Systems Architectures**: When looking at the Windows architecture I would recommend using Windows NT kernel. This is considered a hybrid kernel, allows for better security and is more manageable because they utilize a layered approach.
3. **Storage Management**: Since the client is looking to make a web-based app. The type of storage management that can be recommended would have to be a cloud-based storage system. For this reason, I will be recommending Microsoft Azure since it allows the client to scale the storage system to their needs as the game continues to grow.
4. **Memory Management**: Since we are going with windows there is a variety of choices when looking at memory and storage management tools. Going back to Azure services and Visual studio, the client can use these to manage the codebase versions. As long as the industry best practices are used, we can help maximize the efficacy of our memory.
5. **Distributed Systems and Networks**: When taking a look at distributed systems, and networks, cloud technology allows for servers to run with the companies databases which will allow for used to call the application on whatever platform they are using as long as they are able to access a web based browser as that is what the clients app will be running on. Cloud technologies also allow for the ability to configure different types of requests and respond with different instances of the game depending on what version of operating platform the user is using.
6. **Security**: When looking at security windows allows for the implementation of firewalls/encryption through creating personalized databases and infrastructures Along with dedicated teams at Microsoft that can provide support to the client given anything thing does happen in terms of a security breach. Since the client is looking for the application to be multi-platform, going with something that can provide security over those platforms is necessary. For that reason, Aura would be an excellent choice to go with while be cost heavy the support that is able to be given across Windows, Mac, Linux, IOS and Android is immense and would allow the application to be protected over multiple platforms.