

<Name-of-Software-Application>

# **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 | <09/15/2023> | <Amarah Kirkman > | <Project 1> |

**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)

Our company has taken on a new client, The Gaming Room. They want to develop a web-based game that is based on their original game, Draw It or Lose It. Draw It or Lose it is currently only available on Android app only, and it consists of teams competing to guess what images will be drawn. The company is requesting that the new game will be able to run on other operating systems such as Mac, Linus, Windows, etc.

## Requirements

,

*<* Please note: While this section is not being assessed, it will support your outline of the design constraints below. *In your summary, identify each of the client’s business and technical requirements in a clear and concise manner.>*

## [Design Constraints](#_2et92p0)

The design constraints for the application consist of:

* Cross-platforms: Enabling the app to be accessible from other operating systems
* Multiplayer features: Within the game each team should be able to have multiple players assigned to it.
* Multiteam features: The game should have the ability to have one or more teams involved
* Unique user-ids: Game and team tags must be unique to allow users to know whether a name is already in use
* Unique Identifiers: Only one instance of the game can exist in memory at a time

## [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   efficiently.> The Unified Modeling Language diagram is utilized in this project to provide a visual of the game system for both the client and the team. In this diagram the entity class creates a relationship between the game, team, and player classes. The arrow represents the classes inheriting attributes from a super class. The “ProgramDriver” class is pointing to the “SingletonTester” which shows us that the “ProgramDriver” will use the “SingletonTester” to test the code. This is developed so we can test the restriction of having one instance where the game can exist in memory. The “GameService” class will hold all the complex methods that make up the game and its functionality. This diagram will be used during the development process to create a final product and some adjustments can be made along the way.

**"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** | Advantages include flexible terminal commands, provides security and stability, Unix-based environment, and a great graphic interface.  Some weaknesses for a Mac server include that it may be the expensive option, limited hardware options with Apple company | Linux is a somewhat popular choice that offers many advantages and some weaknesses. Some advantages include a free and open source, offering a wide range of server software options, good security, and being cost-effective. A possible weakness depending on the client is that it might require more in-depth technical expertise | Windows is a very commonly used platform, especially using Microsoft technologies. Advantages of Windows could include pre-supported integration with Microsoft tools, more software available, and easy server-based setup. Some weakness include closed source software, updates through Microsoft, and it being a less secure option. | Pertaining to mobile devices, servers are used to serve mobile app data and can use the previous mentioned platforms. Advantages include easier portability and optimized data collection. Possible weaknesses include lower specs in hardware, and the changes that are made can be on a large scale differing greatly model to model. |
| **Client Side** | While developing for clients using Mac we want to take into consideration that they have expensive hardware and systems, the compatibility is limited to apple products, and its going to require some in depth learning to learn how to navigate the coding tools. | The development considerations necessary when discussing Linux are user freedoms, rapid development, and costs are always typically low. | Developing for clients using Windows we can consider that there is minimum expertise and time required compared to other platforms. The costs can vary and in terms of IDEs clients often involve Visual Studios. | Developing for mobile devices involve platform specific development. Costs can very with the mobile devices with both expensive and cost-effective options. The mobile devices provide flexibility to clients and developers to be updated. |
| **Development Tools** | The primary integrated development environment for Mac applications is Xcode, which supports Swift, C++, and C. Languages could include Javascript and HTML. PyCharm, GitHub, and Visual Studios can also be used. | Linux supports a wide range of programming languages and popular IDEs includes Visual Studios, IDEA, and Eclipse. | In terms of programming languages and tools, Visual Studio is the primary IDE for Windows, with supporting languages like C# and C++. | For mobile devices programming languages and tools are looked at in terms of IOS, Android, and cross-platform tools. |

## 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**: Windows is a very user-friendly system and is most diverse in development. Windows has the most popular GUI and it very compatible with many emulators.
2. **Operating Systems Architectures**: Windows has a great graphic operating system that allows the flexibility for software, files, code, and many other features to be integrated and interchanged.
3. **Storage Management**: Windows offers Windows 10 and Windows 11, who use storage sense to calculate how much storage is being used and when it is almost full.
4. **Memory Management**: The memory management is usually up to the user’s preference, but Windows does group similar data together in an easy-use interface.
5. **Distributed Systems and Networks**: Multi-integration systems that are network based bring original designs for player communications and data sharing.
6. **Security**: Windows excels in security, having integrated security measures.