

Draw It or Lose It

# **CS 230 Project Software Design Template**

Version 1.2

## 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 | 05/17/23 | Jared Bickler | Initial revision |
| 1.1 | 05/17/23 | Jared Bickler | Design constraints, domain model, evaluation, recommendations |
| 1.2 | 06/03/23 | Jared Bickler | Updated evaluation |
| 1.3 | 06/15/23 | Jared Bickler | Updated recommendations |

## [Executive Summary](#_sbfa50wo7nsh)

*The Gaming Room* wants to develop a game that can run on multiple different platforms, namely, iOS and other platforms. Currently, only Android is supported. The game is called “Draw It or Lose It.” The object of the game is to guess what an image pulled from a stack in a certain time limit (15 seconds).

## Requirements

The game must support multiple teams, each with multiple people. The game must also support multiple platforms and can only run one instance of the game at a time.

## [Design Constraints](#_2et92p0)

* Multiple teams
* Multiple people per team
* Unique team names
* One game instance at a time
* Supports multiple platforms

The game should run on all mobile devices, along with desktop platforms such as Windows, macOS, and Linux. This means that each platform will need parallel development bandwidth. This will likely require multiple languages and software architectures to be properly implemented.

## [Domain Model](#_8h2ehzxfam4o)

Game, Player, and Team have a relationship via Entity, and they all inherit attributes from it. Team and Player are a “has a” type, whereas Game has a Team and GameService has Games.

**"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** | * Popular for web hosting * Upgradeable, supports different requirements * Very configurable using the terminal   **Update:**   * Running an HTML websocket can reduce the startup costs | * Cost-effective * Secure * Preferred for web hosting * Difficult to find supported applications   **Update:**   * Linux is best hardened for security by using multi-factor authentication and secure keygen | * Most available software support * Dominant platform * Susceptible to viruses * High resource requirements   **Update:**   * Windows Server is a fairly inexpensive option for setting up a local node.js with RESTful authentication. The system is also scalable and can start small then expand | * Wide array of use * Lack of upgradeability * Multiple differed kernels and platforms |
| **Client Side** | * Expertise required * Similar cost to Windows   **Update:**   * An app could be created which is hybrid instead of native, which would increase portability and reduce time to market | * Minimum cost * Lots of expertise required   **Update:**   * React-native web can be implemented to support android, iOS, and web, this solution is best for front-end clients | * Minimum expertise needed * Similar cost to macOS   **Update:**   * Visual Studio Code can be used to work on a react-native web codebase | * Some expertise needed * Low cost   **Update:**   * React can be used to scale the interface for mobile use |
| **Development Tools** | * Swift is most popular, though Macs can run all languages * HTML * CSS * JavaScript * Python * Ruby * PHP * Java   **Update:**   * Flask * Node.js * Websockets * mySQL | * Works with Visual Studio, Eclipse * HTML * CSS * JavaScript * Java * Python * Ruby * PHP   **Update:**   * Firebase * Amazon * AWS * Heroku | * Same languages as Mac and Linux | * Same languages as previous |

## 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 recommend using Windows as a base to start with, due to the high level of support and low cost and expertise required to implement the solution. Additionally, Windows supports many IDEs and other development environments.
   1. Cross-platform functionality can be accomplished afterwards, focusing on macOS, Linux, and mobile platforms.
2. **Operating Systems Architectures**: Graphical User Interface (GUI), along with other multimedia platforms. I recommend using Flask on a macOS-based server, deployed in a virtual environment. This will ensure consistency and reliability.
3. **Storage Management**: Windows includes a storage management feature called storage sense which helps keep storage to a minimum. It also supports cloud storage via OneDrive and Google Drive. In addition, an off-site backup, containing a secure mirror of the server code and compressed user data will be available.
4. **Memory Management**: Windows has great memory allocation and many IDEs support file management.
5. **Distributed Systems and Networks**: An IDE that can run on multiple platforms, like Eclipse or Visual Studio. Then, when the game is finished, it can be exported to the web. The servers will need to be able to support large player volumes.
6. **Security**: Windows defender is an excellent built-in security solution. Critical user data should be encrypted on the database server. The principle of least privilege should be implemented for user accesses, and multi-factor authentication used for validation of users.