THE GAMING ROOM

<Name-of-Software-Application>
CS 230 Project Software Design Template
Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	01/22/2025	Maxwell Sheehan	Developed our Software Design recommendations for our client.

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

Create a game software that allows multiple teams and players to share information at the same time. Only one instance of the game can occur at a time and all teams must have the same access to the information. Each team must be able to assign multiple players in varying quantities, team names must be unique entities that can be set by team members. Team members can also only belong to one team at a time. The gameplay will include drawings gradually being revealed, players on each team will be able to buzz in to guess the drawing as it develops. Once the drawing is complete each individaul team will have 15 seconds to submit a final guess as a team.

<u>Requirements</u>

< 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.*>

<u>Design Constraints</u>

- App must be cross platform to allow different players access to the mobile app.
- Must use significant resources to hire staff for creating apple mobile, samsung mobile and other app store and mobile software requirments.
- Game must have the ability to support multiple teams, with multiple players on a team.
- Team members can only be assigned to one team at a time.
- Team's must have a unique name.
- Game must be a sinlge entity, shared display, memory, and score accross a single game instance.
- Players must be able to guess while the image is rendering.
- A team can only guess once after the round has ended and has 15 seconds.
- The software must have a database of stock images to pull images from.

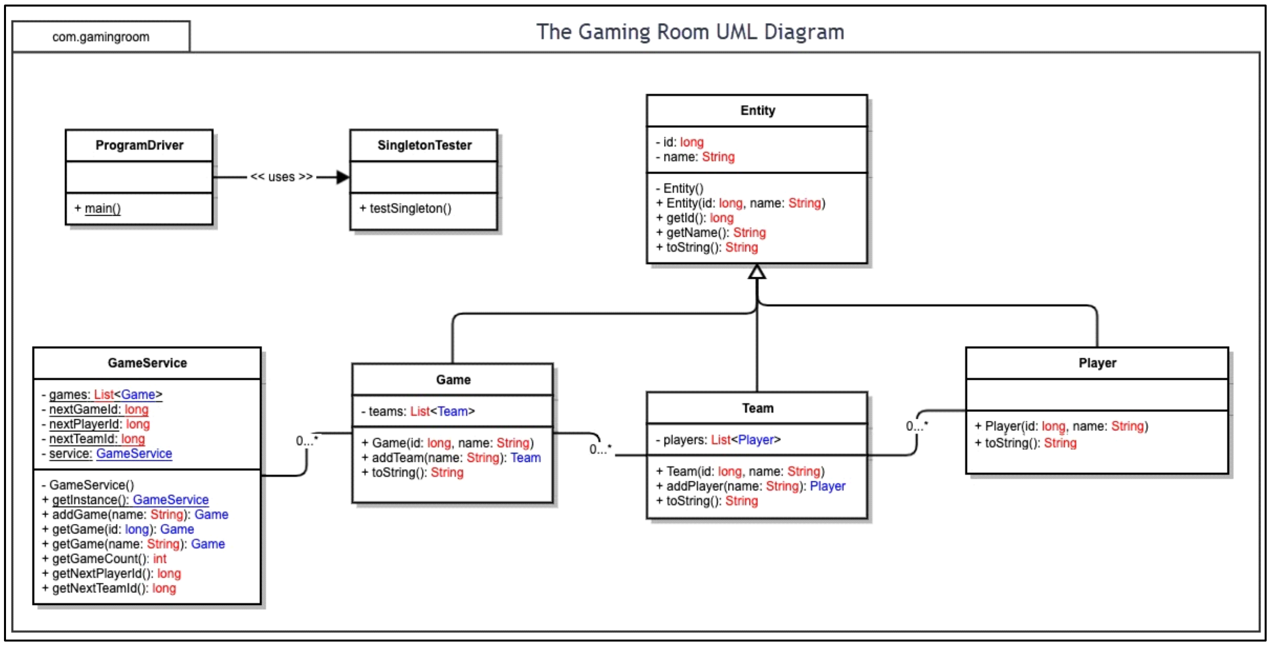
<u>System Architecture View</u>

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.

<u>Domain Model</u>

The Game, Team, and Player classes all inherit from Entity. These are an 'is-a' relationship. Furthermore Game has an aggregated list of Team objects, while Team has an aggregated list of Player objects. This is an 'has-a' relationship. Game Service is a static instance with the getInstance() method. This relationship is used and tested in the ProgramDriver class testing with SingletonTester to insure our GameService is working as intended.

OOP is represented in many ways. Encapsalation is shown in the private attributes marked by '-' within the uml diagram. Entity serves as a base class which provides inheritance to Game, Team and Player. Inherance is a core OOP principle that demonstrates reusable code and sustainability for further development. toString() is a good example of polymorpish, the function is used in Game,Team,and Player with unique implementations. This demonstrates the the handling of various objects in different types. Finally looking to abstraction, the Entity class has the attributes id and name which represent the basic identity of an object within it's system. This reduces redundancy and increases readability



Evaluation

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 is well supported with a large user base, it has less support then windows, but more then linux. Mac has a substantial userbase on mobile devices so developing it with this in mind would be potentially beneficial.	Linux is open source so it's likely we would find a bunch of free resources with a large quanitty of support. Linux would require more technical support as it is a CLI, it has more software compatability issues then mac or windows, and has a tough learning curve. These are all weaknesses Linux does have a plethera of webhosting services in open source, Apache for instance.	Well integrated with multiple microsoft tools, including SQL which will help with database managment. Super begineer freindly and widely used. The licensing fees can be expensive, and windows can be very resource consuming which is a weakness.	Mobile devices are portable, and very accessible which is an advantage. But for hosting a webserver they have very limited memory, processing power, and storage, which makes hosting super unreliable. Requires a power supply and network connection through wifi, overall lacks tool and features compared to standard OS.
Client Side	Mac requires confiquration of Json files and would require additional resources then soemthing like windows that is widly supported. The additional man power and support would have financial implicaitons.	Linux is open source so it's possibly we could save time and money if something was already developed for our needs. If not though we would need to hire linux devs to deploy and maintain the Linux system.	Windows is the most widly used system with the most support. It would probably be the cheapest as we hopefully would already have staff familar, the additoanl expertise would probably leave this as the cheapest and quickest option.	We need to develop the software in multiple app store and mobile OS enviornments. This will take multiple personale, and will be expensive. The app will also have to be mainteained on multiple services, we will need seperate data bases for each.

Development Tools	Unity is a game engine that works for all of our required platforms we could use. For our IDE visual studio code supports mac, linux, and windows. For the web application implementation, or app store implementation react is a common design tool used.	C# works on linux, and would be compatable with potential game engines like Unity. While also being avaliable on Visual Studio.	C# works Windows, and would be compatable with potential game engines like Unity. While also being avaliable on Visual Studio. Windows also has high potential with many languages like python, c++, java	Mobile devices can have software built on unity. We could also use Godot or unreal engine. Godot is open source which could potentially reduce financial burden.

__Recommendations__

Analyze the characteristics of and techniques specific to various systems architectures and make a recommendation to The Gaming Room. Specifically, address the following:

- **Operating Platform**: To save cost Linux would be the recommendation, it is open source and has a wide range of community supported features. If we have the budget windows in recommended. Windows has a lot of integration with core features, is wildly known, and will save labor costs. Overall though Linux has great security tools, the best memory management as well as access to databases such as MySql, all while being the cheapest.

- **Operating Systems Architectures**: Linux and windows are both hybrid operating systems that utilize kernels. Windows has a more restricted mode than linux, with a user mode. Linux can allow users to elevate their control but Windows is a lot safer and user friendly for most users.

- **Storage Management**: Sql is the most common database and is integrated into windows systems leaving SQL to be a great choice with the use of windows. Linux uses mySql as the open source alternative, it is very easy to scale and is compatible with cloud environments if the game takes off.

- **Memory Management**: Linux offers a lot of options on managing memory which makes it very resource efficient. Windows is much less efficient and can be very resource hungry. With proper optimization on our team it shouldn't' be anything to extensive but windows is notably much more memory intensive than linux can be.

- **Distributed Systems and Networks**: Windows and Linux both have access to a variety of API's that can offer network connection. These API's are often run by other distributors so outages can be independent of us. This could be both good and bad for the client as multiple parties could experience outages. We could store data locally in a cache to reduce network outages and maintain basic feature usability. But this is a similar process across both OS.

- **Security**: Security is a serious concern for any OS. Linux is more secure than Windows in some ways. It's open source and less utilized so less vulnerabilities are known and when a community member finds a vulnerability it will be patched very quickly. Open source offers some huge benefits for security. Windows is much more used and thus has a lot more known vulnerabilities and exploits. Frequent windows updates would be required for optimal security as windows 'patchs' out vulernabilites. This causes a lot of server downtime. Linux also has a lot more security and can be easier to manage permissions with a much more robust file system with tiers of access integrated. Open source security features also exist for Linux that can assist in protecting user data.