Draw It or Lose It

CS 230 Project Software Design Template

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	09/18/22	Kevin Schmelter	Completed Software Design Template

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.

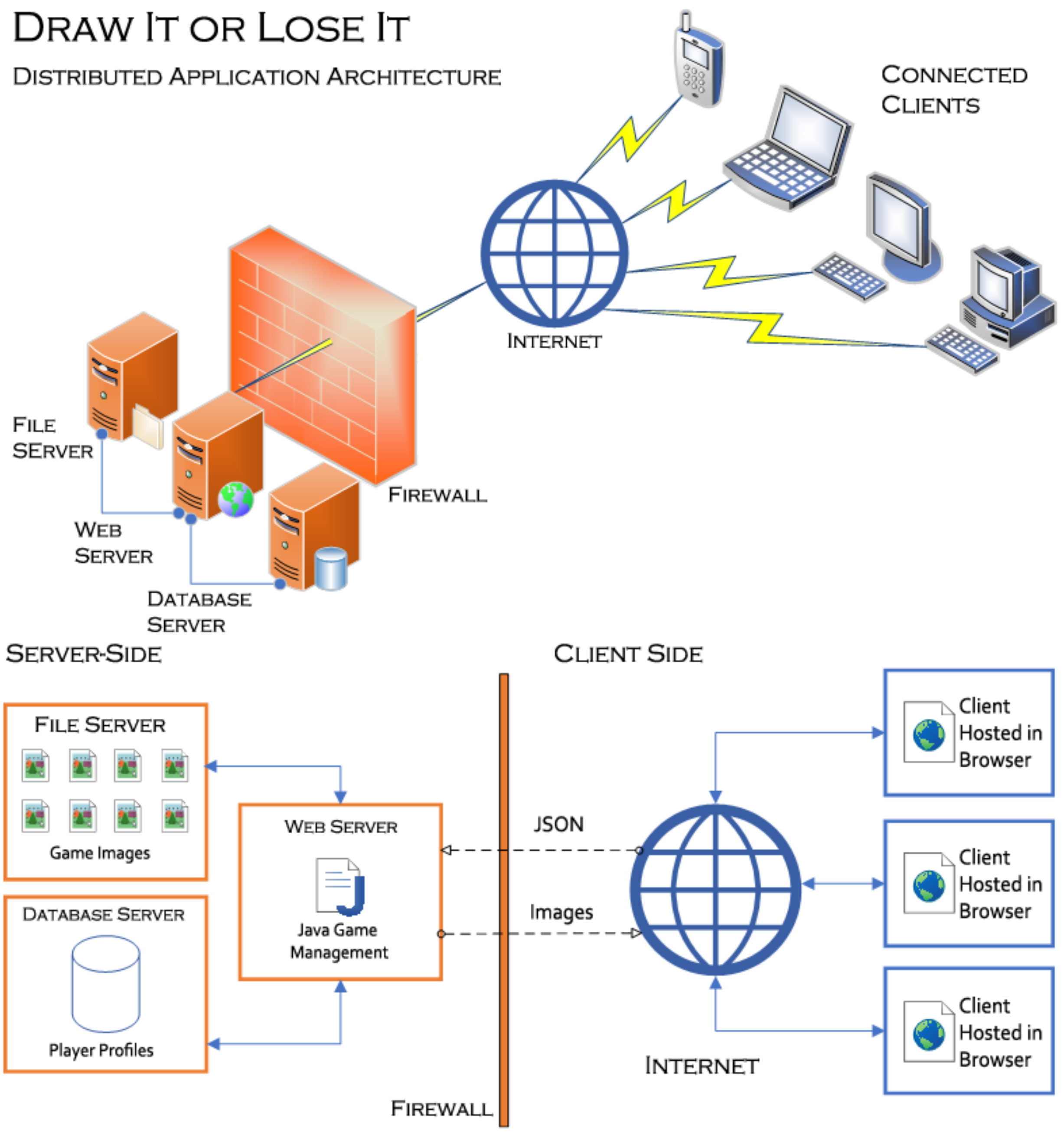
Executive Summary

The Game Room wants to create a Web based game in a distributed environment, that resembles the old game Win, Lose or Draw. The system will have to be able to handle multiple users playing separate games at the same time, that can add teams, and players.

Design Constraints

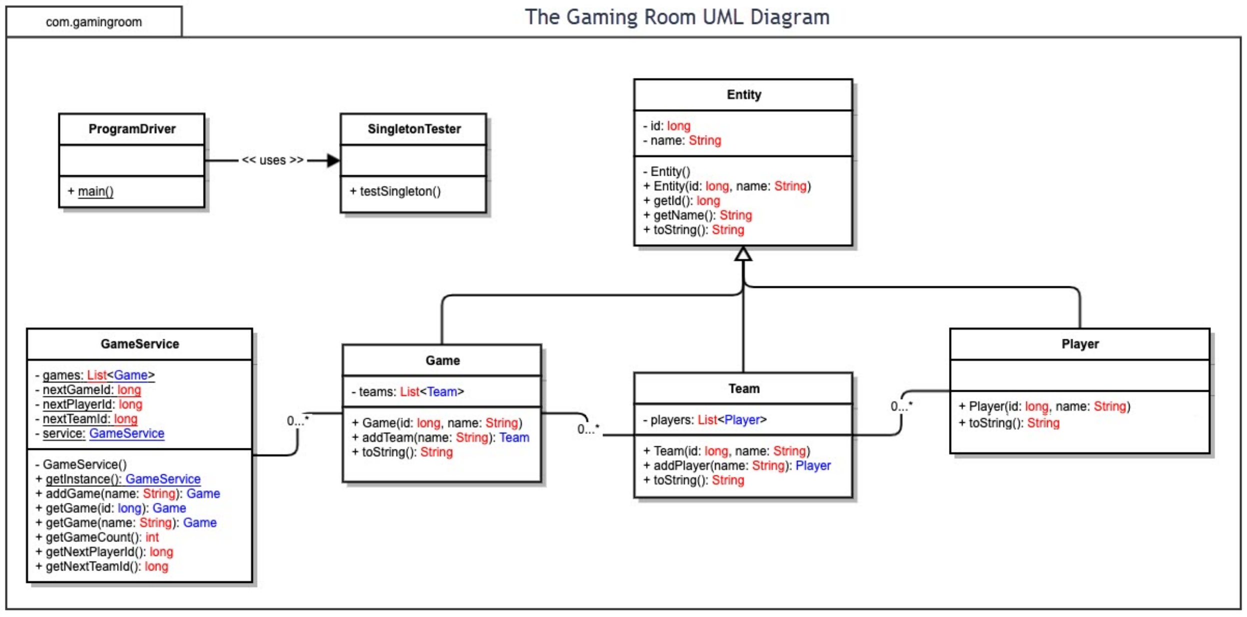
In order for the environment to be a web based game handling multiple users playing at the same time, there needs to be a single entity that manages the games, teams and players that are added to the system. The main program is going to need to keep track of the game, team, and player instances so there will be unique ID's created for each game, and teams of players associated with those games. The web architecture will need to be distributed, so there will be need of a hardware or cloud based server and API for the clients to connect with.

System Architecture View



Domain Model

The Entity class has two attributes, an id and a name. It also has methods such as a private and public constructor that sets the attributes, getters for the attributes, and a toString function that formats the print statement. The Player class extends from the Entity class and overrides the toString function. The Team class extends from the Entity class, and has an array list of players as an attribute. The methods include a constructor for the id and name, a function to add players to the team list, and an overridden toString function. The Game class extends the Entity class and has an array list of teams as an attribute. The methods include a constructor that sets id and name, a function to add the teams to the array list, and an overridden toString function. The GameService class is a singleton class that has an array list of games, a next game, team, and player id, and a GameService object. The methods include a private constructor, a function to create a single GameService object, a function to add games to the array list, functions to get the game based on id and name, a function that gets the amount of games, and two functions that get the next team and player ids. The Program driver class has a main method that handles the GameService, game, team and player objects. The SingletonTester class has a test singleton function to test if the GameService object has already been created.



<u>**Evaluation**</u>

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	+ Good for Mac on Mac use + Unix OS + Open source servers + User and file access control + Support LDAP and ADP - Docker support is virtual only - Difficult to create server only environment - Expensive to implement	+ Open source system + Minimal up-keep + Allows for apache use + Allows for MPMs for multiple connections. - Bit of a learning curve - Mostly open source so not the best documentation	+ Very popular + Has Windows server platform + Uses a windows file system + Users and groups allow for easy security - Requires more up-keep than unix - Does not allow for virtualization	+ Easier to develop using a serverless architecture + Use MDM for securing mobile devices and access controls - Need to cater to multiple device types. - Might need to create separate architecture for mobile if integrating with web app.
Client Side	+ Allows for browser to use game + Simple UI for easy usage + Supports Windows file systems + Can run windows through bootcamp or parallel - Very expensive - Windows cannot read Mac file system without 3rd party	+ Allows for browser to use game + Secure + Allows for lots of tools for like command line and package managers - Requires time and energy to learn the system - Not compatible with all software like Microsoft Suite apps	+ Allows for browser to use game + Most applications handle for windows + Not too expensive - Viruses are more likely on windows - Have to pay for licensing	+ Allows for browser to use game + Users can use the web app or download the app + Only needs a wifi connection - Needs to have layout be sized specifically for mobile devices
Development Tools	+ Allows for lots of languages + Has Xcode + Main language is objective C + Can use visual studio suite + Can use React Natve - Not the greatest with developing and running c++	+ Allows for lots of languages + Comes with compilers, bashes and shells + System is customizable + Uses eclipse netbeans and atom - Requires time	+ Windows is mainly written and runs smoothly with c++ + Has whole visual studio suite available + Can use shell	+ Can use Xcode for wireframes + Has options for development on both iOS and android using react native - Requires knowledge of phone system in order to work on mobile

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**: Since we are developing to handle users of different devices in a web based environment, windows would be the best option. It has many options for mobile development such as Xamarin and React, and has the capacity to use emulators to handle testing and development on other platforms.

2. **Operating Systems Architectures**: The Windows system splits the operating system up between user processes and the kernel processes. The user processes is mainly what the user interacts with, and the Kernel works on lower level system processes like memory management, I/O processes and so forth. Windows uses a directory like file system to store data so it is easily accessible. It also modularizes the hardware and uses multiprocessing so it can be customized to the users specifications.

3. **Storage Management**: For a web based environment using Microsoft's cloud based system, Azure would be the best option for storage management because:
    - It is very cost effective
    - There is a small learning curve
    - It allows for scaling based on the user base
    - Has customer support for the platform
    - Always updated my Microsoft

4. **Memory Management**: Windows 10 is the best latest version of the windows system and can use a 32 bit or 64 bit architecture to manage its main memory. Each process on a 64 bit windows allows for 8 terabytes of space for address space, where 32 allows for 4 gigabytes. The threads of processes are encapsulated as to not be allowed to view the address space of other threads.

5. **Distributed Systems and Networks**: Since we chose Azure before it is also a great choice for handling development in a distributed system environment. It would allow for process monitoring on each user session including games, teams, and players that are created and used. It allows for most of the networking to be done by the Azure platform allowing more thought towards application development.

6. **Security**: Again since we are using Azure there are lots of security implementations that can be used through the platform. For each user the platform can securely handle the data from each session to make sure no other user can access their data using:
    - VPNs storage option for the users
    - Configurations using the IP so that users info is separate
    - Database could be secured through an IP whitelist access system or password