The Gaming Room

CS 230 Project Software Design Template

Version 1.0

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	03/01/21	Kevin Schroeder	Layout software design and requirements

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

This software design was developed to support a game where multiple teams compete to guess a word based on stock images. The application will render images from a large library of stock drawings as clues. A game consists of four rounds of play lasting one minute each. Drawings are rendered at a steady rate and are fully complete at the 30-second mark. If the team does not guess the puzzle before time expires, the remaining teams have an opportunity to offer one guess each to solve the puzzle with a 15-second time limit.

Additionally, the software must meet the following requirements:

- A game will have the ability to have one or more teams involved.
- Each team will have multiple players assigned to it.
- Game and team names must be unique to allow users to check whether a name is in use when choosing a team name.
- Only one instance of the game can exist in memory at any given time. This can be accomplished by creating unique identifiers for each instance of a game, team, or player.

The scope of this document is specific to the design of the virtual infrastructure and the supporting components.

Design Constraints

Language - The language that an application is developed in will impact the application and the development process. If there is a required language that doesn't match up with what the development team has experience in, the project will take extra time to account for the learning curve. Some languages are typically considered to have a small learning curve (like PHP and Ruby) and some have a steeper learning curve (like C++ and Java). The programming language will also affect the application when it comes to security and run-time efficiency.

Framework - Using a framework has benefits and drawbacks. A framework can save a lot of time, but could also come with a learning curve and performance issues.

Database - The type of database or data storage we use will also come with performance effects. Additionally, the type of database we choose will limit our options on how we structure the data. For example, NoSQL databases allow different data structures than SQL databases. There may be a learning curve to this as well.

Timescales - The expected timeline of this project will influence technology decisions and whether or not we add nice-to-have features that are not essential to the application's functionality.

Testing - The way that we test the application will affect the development timeline and the resources needed. If we are planning to use a QA team, we will have to find a few resources to help us. If we are relying on automated testing, the application will take longer to develop and we need to use a programming language that has a good automated testing suite. A combination of a QA team and automated tests would be the best option in terms of preventing bugs but is also our most expensive and time-consuming option.

Bug Fixing - The process that we use for bug fixing and QAing the site will affect our timeline.

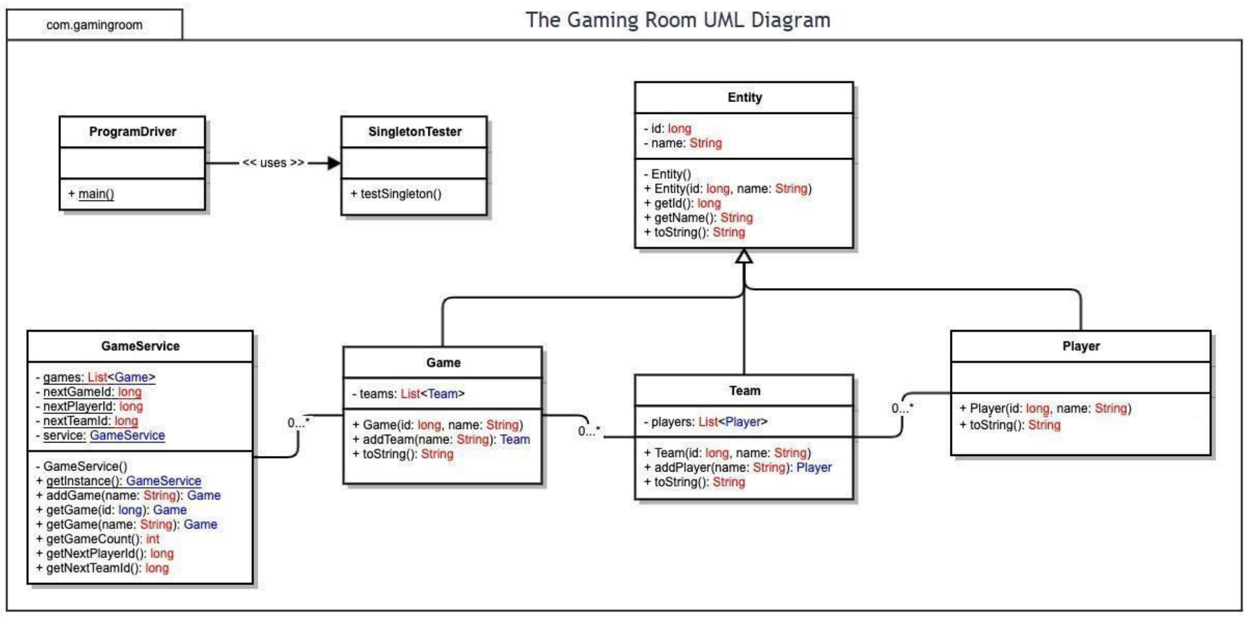
System Architecture View

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

Below are the essential components and descriptions of our UML diagram:

- The Entity class is the parent class to the Game, Team, and Player classes, which means that these three classes can inherit any public method from the Entity class.
- The GameService and the Games classes are associated where a GameService can have zero to many Games.
- The Game and Teams classes are associated where a Game can have zero to many Teams.
- The Teams and Players classes are associated where a Team can have zero to many Players.
- The Game class has a private list attribute called teams that will contain all of the teams for an instance of the Game class. The public addTeam method in the Game class will add teams to this list.
- The Team class has a private list attribute called players that will contain all of the players for an instance of the Team class. The public addPlayer method in the Team class will add players to this list.
- The GameService class has a private list attribute called games that will contain all of the games for an instance of the GameService class. The public addGame method in the GameService class will add games to this list.
- The Game, Team, Player, and Entity classes all have a public constructor that takes in an id and a name as parameters. The id field for the Game class will be used to ensure that only one instance of the game can exist in memory at any given time.
- The name and id fields in the Game and Team classes will ensure that games and teams are unique. Users will be able to check if their desired team name is used in their game using these fields.
- The GameService class creates an instance of itself and saves it in a private attribute called service. The constructor for this class is private, so only one instance of the GameService is allowed. This class also has a public getInstance method that will return the only instance. This type of software design is called a Singleton.
- The nextGameId, nextPlayerId, and nextTeamId private attributes in the GameService class will allow us to iterate over the games, players, and teams lists.



<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	Mac's server-side hosting is a good option for security and stability. The developers at Apple focus on creating complex code on their servers that have proven to enhance security. These servers are also known to rarely error to the point where the web application goes	Linux servers are open-sourced, which means that anyone can make updates. Though updates are not common, they are free. Linux is also the second most secure and reliable option only to Mac. These servers allow you to download custom security software that can enhance	Microsoft owns the licensing for all Windows servers, which means, like Mac, users need to pay for upgrades to the server. Windows servers do benefit from the flexibility of allowing users to develop from any Windows hardware. Like Linux, you can find cost-effective	Since the specifications state that this app will run within the browser of the mobile device, there are no additional server-side requirements beyond what has been listed for the other operating systems.

	down. However, benefits come at a price. Mac is the most expensive option when it comes to hardware and licensing. Apple requires additional fees for upgrading its servers. Mac servers require Mac hardware which starts at $999. We recommend using version 5.7 of macOS Server since Apple made significant changes to what they bundle on the server for it to run more efficiently. Licensing can be bought on the Apple App Store for $19.99 and requires 2 GB of ram and 10 GB of disk space.	the server's security. The hardware requirements are very flexible. You can run a Linux server off of a three-hundred dollar laptop or a multi-thousand dollar computer. Overall, Linux servers are the least expensive option. Linux Operating Systems are assembled under the model of Free and Open Source Software. This means that a license is not required to run a Linux server. However, there are several commercial distributions such as Red Hat's Enterprise Linux, Oracle's Linux and SUSE's Linux Enterprise Server operate under the open source license model what associated costs.	hardware that uses the Windows operating system. However, these servers are known to require frequent security updates that can crash the entire application. We'd categorize Windows servers as the least reliable and secure option Windows 2019 servers follow a core-based licensing with a minimum of 8 cores per processor and 16 cores per server. All physical cores on the system must be licensed. To set up a microservices architecture that ensures that the app can run on all popular operating systems, we recommend buying licensing for the Datacenter edition. This server is ideal for highly virtualized datacenters and cloud environments and the licensing cost is $6,155.	Hosting a mobile-responsive app is more of a challenge to the developers than the server. This is a highly recommended approach to building any sort of application in modern times as mobile usage is increasing rapidly. One additional consideration is that the increase in traffic and users may drive the need to increase the server size and processing speed.
Client Side	The Mac OS X Server An	Most of the common Linux	Applications using an asp.NET server	The front-end engineers on this

	industrial-strength server platform that supports Mac, Windows, UNIX, and Linux clients out of the box. The server will also require browser testing to ensure that our application is working properly. Lambda Test and Browser Stack are two good options that make browser testing easier and range in price from $15/mo to $25/mo. This browser testing software will require at least one QA employee to know how to use the software and to know how to thoroughly run user tests.	servers like Ubuntu or Red Hat are compatible with major operating systems. The server will require browser testing to ensure that our application is working properly. Lambda Test and Browser Stack are two good options that make browser testing easier and range in price from $15/mo to $25/mo. This browser testing software will require at least one QA employee to know how to use the software and to know how to thoroughly run user tests.	will run on Linux and Mac. Developing with an asp.NET server on a Mac device will cause some issues. Mac users will need to set up a virtual environment on their computer that allows them to work in the Windows OS. This might take a considerable time for the Engineering team to work out. The server will also require browser testing to ensure that our application is working properly. Lambda Test and Browser Stack are two good options that make browser testing easier and range in price from $15/mo to $25/mo. This browser testing software will require at least one QA employee to know how to use the software and to know how to thoroughly run user tests.	project will have to know how to make mobile-responsive apps. The back end will be the same as the desktop version of the game, but the engineers will have to know how to deal with multiple API calls at one time. Furthermore, we might have to consider hiring engineers that have experience in modern, mobile-responsive front-end frameworks such as React, Vu.js, or Angular. These frameworks are ideal for developers making user-friendly mobile sites. The server will also require browser testing to ensure that our application is working properly. Lambda Test and Browser Stack are two good options that make browser testing easier and range in price from $15/mo to $25/mo. This browser testing software will require at least one QA employee to know how to

				use the software and to know how to thoroughly run user tests.
Development Tools	Apple's main programming language that they support is Objective-C. This language is known to be moderately difficult to learn and rare to find someone with a lot of experience in it. However, Mac servers will allow you to develop in most of the major programming languages (Java, Javascript, Python, etc.). Using a non-Objective-C language allows flexibility when it comes to team skill sets, but it may come with slower performance. Most popular IDE's are available for MAC including Sublime, Eclipse, Visual Studio Code, and Atom. Visual Studio, which is one of the most popular IDEs, only allows Mac users to develop apps and games for iOS, Android, and web using . NET.	Linux is very flexible when it comes to programming language requirements. Almost all of the most popular languages can be run on Linux servers. Some languages require extra work to run on Linux servers. For example, C/C++ will require a GCC compiler to run on Linux. Also, Ruby and Ruby on Rails require Phusion Passenger to run on Apache HTTP or Nginx servers. Most popular IDE's are available for Linux development including Sublime, Eclipse, Visual Studio Code, Visual Studio, and Atom.	Microsoft's Windows kernel is developed mostly in C which means that C-derived languages tend to perform the fastest on Windows servers. C-derived languages have very large communities, so it won't be hard to find employees with the required expertise. Most popular IDE's are available for Windows development including Sublime, Eclipse, Visual Studio Code, Visual Studio, and Atom. Windows 2019 servers follow a core-based licensing with a minimum of 8 cores per processor and 16 cores per server. All physical cores on the system must be licensed.	As mentioned previously, we will have to select a programming language that supports mobile-responsiveness. We recommend considering React, Vu.js, and Angular. These frameworks may involve a learning curve for the engineering team. The frameworks can be developed on all of the major IDEs such as Sublime, Eclipse, Visual Studio Code, Visual Studio, and Atom.

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**: Linux offers the best bang for your buck. There is no licensing cost to start a project on Linux and there is a lot of flexibility when it comes to IDE, programming language, and OS of the devices that your developers are working on. Linux is open-source and has a large community of developers that will create packages that can enhance the server's performance. Best of all, all of the benefits come without performance decreases. For flexibility and future scalability, we recommend Linux.

2. **Operating Systems Architectures**: We recommend using a microservices architectural pattern. This pattern breaks a system into several smaller services, or microservices, that have limited interdependencies. Since each part of the game will have its own microservice, you will be able to reuse, update, or remove one part of the game without anything else being affected. Each microservice can be coded in a different language, which gives the development flexibility on how the technology that they use. If one microservice goes down, it's much easier for the developers to troubleshoot when they don't have to look through an entire monolithic application, which will lead to faster debugging. This architectural pattern is also ideal for scaling because you can add additional microservices to support the microservices that are used the most without having to update the entire application.

3. **Storage Management**: The Gaming Room will have 200 high-definition image files that we will have to create a storage management strategy for. First of all, we can get a baseline for the amount of storage that we need by taking the average size of the image file (8 megabytes) and multiplying it by 200 to get 1600 total megabytes, which converts into 1.6 gigabytes. We highly recommend utilizing a cloud storage strategy because they are cost-effective, easy to set up, and easy to scale. We recommend the Amazon Simple Storage Service (S3) because it is the most durable, highly performant, and secure cloud storage service on the market. It will handle storage for companies of all sizes and will scale with you. It even offers analytics that can help further the business. The cost is $0.023 per gigabyte per month for the first 50 TBs, so we are looking at a total cost of $0.0368

4. **Memory Management**: With an application that needs to run many different services at the same time like Draw it or Lose it, combining demand paging and segmentation will result in the most efficient memory management system. This strategy ensures that only the essential parts of the application are being stored in memory and prevents memory from being wasted. Virtual memory management also allows us to use the disk drive when the RAM is full. This will slow down the application, but at least it will continue running. One con of this strategy is that the overhead cost can be significant, but it will lead to a better product in the end.

5. **Distributed Systems and Networks**: With a distributed system, each of the software/hardware nodes will be able to communicate with each other to fulfill a task. The nodes run independently and do not all have to run on the same operating system or platform. However, the nodes will run on virtual machines communicating over the client-server model, which allows the processes contained on each node to run the same on all platforms. Each node will have resources that can be accessed by any other node on the distributed system. These resources could be images, files,

data, or anything that would be important to share between nodes. A middleware connects all of the nodes to the client, making the application seem like it's running on one big server. The middleware handles outages by preventing the entire app from crashing because of one bug on a node. The distributed systems will also manage the concurrency of multiple requests happening at one time and it manages the coordination of multiple requests for one resource at the same time.

Draw It or Lose It could use a node to handle the timer, a node to handle retrieving the images and the associated word, a node to handle the guesses, and a node that handles keeping score and ending the game when appropriate. These nodes can handle multiple requests, allowing for multiple games to be played at one time. If there is an outage to one node, the application will keep running thanks to the middleware. However, these nodes will be heavily dependent on each other to exchange resources for the game to run properly. We might want to consider adding another node that can receive a message about an error on one of the other nodes, display a site maintenance screen on the front end, and notify the development team of the dysfunctioning node.

6. **Security**: There are many industry best-practice measures that we will take to ensure that our Linux server is secure. The first is that the manager of the server should update the server, operating system, and installed applications regularly to implement the latest security patches. To protect the server, we recommend adding users to the server when necessary and giving the appropriate privileges. No one should be logging in as the root user. This helps prevent mistakes and deactivating the root user makes the server less susceptible to attacks.

   Additionally, we recommend requiring an SSH key to log into the server and adding IP exclusions. The SSH key will only be available to the people who need access to the server and makes the server much harder for outsiders to access.

   We also recommend installing and maintaining a firewall on the server. A firewall stands between the internet and the server to filter out unwanted traffic. Two examples of free, easy-to-use Linux firewalls with graphical interfaces are OpenWrt and IPFire.

   Fail2ban is another free Linux application that scans the log field on the server and bans IPs that show malicious signs. Common malicious signs include password failures and seeking for exploits. You can access the list of IPs that the program deems malicious and make edits if need be.

   On the development side, all PII should be encrypted in the database. The developers should not have access to any personal information. There are many open-source packages, depending on the programming language, that will encrypt data in the database.

   Finally, if The Game Room is planning to use a username and password authentication system for the game, we recommend adding password requirements such as length, special characters, and 2-factor authentication.