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	09/20/20	Joshua Massey	Initial creation of the web application
2.0	10/8/20	Joshua Massey	Reviewed client, server, and development tool recommendations

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 Gaming Room would like a web application that imitates their Android game *Draw It or Lost It*. The lack of technical expertise to create the desired application is the biggest issue for this project. The resulting application developed by Creative Technology Solutions needs to address several requirements, including, but not limited to, the creation and maintenance of unique games and team names, as well as a limitation on the number of concurrent instances of the service running. The proposed solution is a Java application that will handle all of the backend processes.

Design Constraints

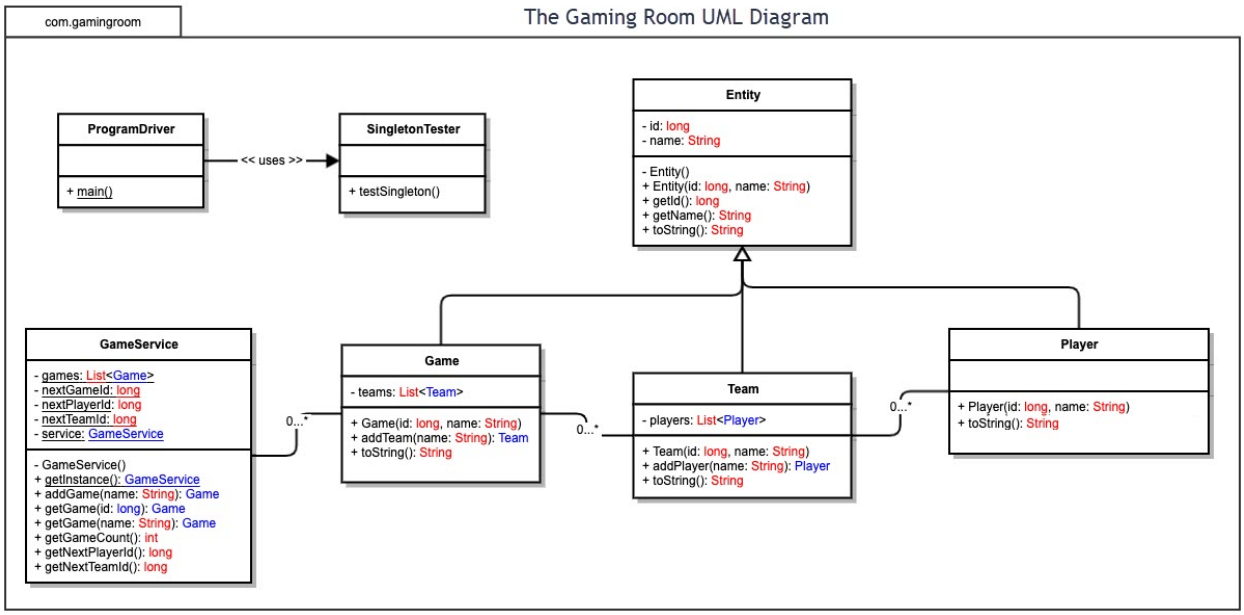
One major design constraint is that the application should only ever have one instance of the service running. This implies that all operations should be handled by the same service instance. Being a real-time web-based experience, users will expect requests to the service to be quick. This also implies that the development of the application should be written with to be highly efficient, especially in our searching algorithms.

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

In the diagram below, we can see the GameService class uses the singleton pattern by keeping its constructor private and ensure that the same instance is returned each time we want to use it. The service has the ability to host multiple games. Each Game can contain multiple Team objects, and each Team can contain multiple Player objects. The Game, Team, and Player classes all extend the Entity class, which contains an ID and name.



Evaluation

Development Requirements	Mac	Linux	Windows	Mobile Devices
Server Side	I would not recommend using the Mac operating system to manage the server due to its limited access to deeper system settings and a shortage of options for this kind of development.	Linux would be the best option to host/deploy the web server. Although there are many flavors of the operating system, they are built to be extremely lightweight. This means that more of the machine's resources can be exclusively dedicated to performance. Because it is the preferred OS for server side development, there are many powerful tools that already exist for the platform.	In terms of server side development, Windows is a better option compared to Mac. Windows gives users greater access system-level settings. But because it is a heavier OS, it needs a lot more RAM and processing power for sometimes basic operations.	Due to the limitations of both the software and hardware built for mobile platforms, it isn't possible to host any kind of server on these devices.
Client Side	In order to provide a good web experience for this mobile-first game, the web client should be written with a modern framework such as Angular. However, being relatively newer than many other options, there could be an issue of finding people who are skilled in this toolset.	In order to provide a good web experience for this mobile-first game, the web client should be written with a modern framework such as Angular. However, being relatively newer than many other options, there could be an issue of finding people who are skilled in this toolset. Additional expertise in Linux may be needed to solve issues with software compatibility.	In order to provide a good web experience for this mobile-first game, the web client should be written with a modern framework such as Angular. However, being relatively newer than many other options, there could be an issue of finding people who are skilled in this toolset.	Using frameworks such as Angular makes developing for a smaller screen much easier. However, accessing the application via an app requires knowledge of Swift (iOS) and Java, Kotlin, and Gradle (Android)
Development Tools	My recommended tech stack for this project includes Angular as the programming language, IntelliJ Idea as the IDE, Git or Github for version control, and Jenkins to handle deployments.	For Linux, the tech stack would include Angular as the programming language, IntelliJ Idea as the IDE, Git or Github for version control, and Jenkins to handle deployments. Though most tools have installers specific to Linux, I would recommend installation issues to be resolved using a Linux compatibility layer such as WineHQ.	My recommended tech stack for this project includes Angular as the programming language, IntelliJ Idea as the IDE, Git or Github for version control, and Jenkins to handle deployments.	The Android development tech stack includes Android Studio, Gradle, and either Java or Kotlin. The development tech stack for iOS includes Objective C or Swift

Recommendations

1. **Operating Platform**: To run this application I would recommend using a cloud-based platform. While there are several viable platforms in the cloud computing space, the offerings made by Google's Google Cloud Platform (GCP) are stand out for being both cost-effective and scalable. For this application, Google's Kubernetes Engine (KE) would be able to best handle its event-driven nature.

2. **Operating Systems Architectures**: Due to the nature of this game, I would recommend using an event-driven architecture. Since each game involves multiple people, there is a need to keep what's being on the screen updated. This requirements dictates the need for a system which handles a constant stream of data in being both received and sent by the client.

3. **Storage Management**: In addition to using a KE instance to host the application, I would also recommend the use of Google's cloud storage solution, Firestore. Firestore is a NoSQL cloud database solution which touts flexibility, scalability, and synchronization for client-server systems. Because it uses NoSQL to store the data, queries are extremely fast - especially when coupled with the native caching solution that it provides.

4. **Memory Management**: In terms of memory management, the use of the Kubernetes platform can integrate a caching solution such as Redis. When the application is deployed, you are able to control through various means how long the data is stored in the cache. By allowing this, the application can run efficiently as it won't be delayed by constant calls to the server. Cache management has the added benefit of allowing the application to discard information it doesn't need access to anymore. By doing this, the size of the cached dataset can remain relatively small which should have a positive impact on the performance of searches.

5. **Distributed Systems and Networks**: One of the biggest benefits of using a KE instance to run the application is its scalability. The engine can be set up to have a minimum number of instances which work in tangent to handle any additional load using what they call Pods. Additionally, multiple instances can be set up to handle requests and events with different origin points. This approach allows different clients a dedicated slice of the application to handle network traffic (ie separate services within the same KE for mobile and web traffic). The aforementioned Pods also act as a failsafe for the application itself. In the case of an outage, having multiple Pods for each instance allows for replication and backup of a service. Additionally Google's cloud infrastructure has the ability to bring itself back online in the case of a major incident in their cloud environments.

6. **Security**: In terms of security, GCP has several different layers that work in conjunction to protect both the application and the data it processes. The KE itself has an authentication layer in order to get into and manipulate the environment, and the Pods within a KE have their own set of additional security policies that have to be met in order to get access to the system from a client perspective. The KE can also perform security checks on the code itself. Depending on the policies set for the cluster, the platform can be either automatically or manually triggered to look at the client's code as well as the set security policies for the instance and runs it against the most common service attacks. As an extra layer, these code checks can be paired with the Source Code Analysis Tools provided by OWASP, a nonprofit organizations that works to help improve application security.