Draw It or Lose It

CS 230 Project Software Design Template

Version 1.1

Table of Contents

CS 230 Project Software Design Template 1
Table of Contents 2
Document Revision History 2
Executive Summary 2
Design Constraints 2
System Architecture View 2
Domain Model 3
Evaluation 3
Recommendations 7

Document Revision History

Version	Date	Author	Comments
1.0	09/17/22	Jonathan Miller	Added content to Executive Summary, Design Constraints, and Domain Model.
1.1	10/02/22	Jonathan Miller	Updated the Executive Summary. Added content to the Evaluation Table.
1.3	10/16/22	Jonathan Miller	Added Content to the Recommendations.

Executive Summary

We at Creative Technology Solutions propose a solution to streamline the development of The Gaming Room's web-based version of its Android application Draw It or Lose It. We will use the JAVA programming language to handle the *Game Service* management backend for this web-based application. This *Game Service* will allow each *Game* to have multiple *Teams* and each team to have multiple *Players*. *Game* and *Team* names will be unique allowing users to check whether a name is in use while choosing a name for their team.

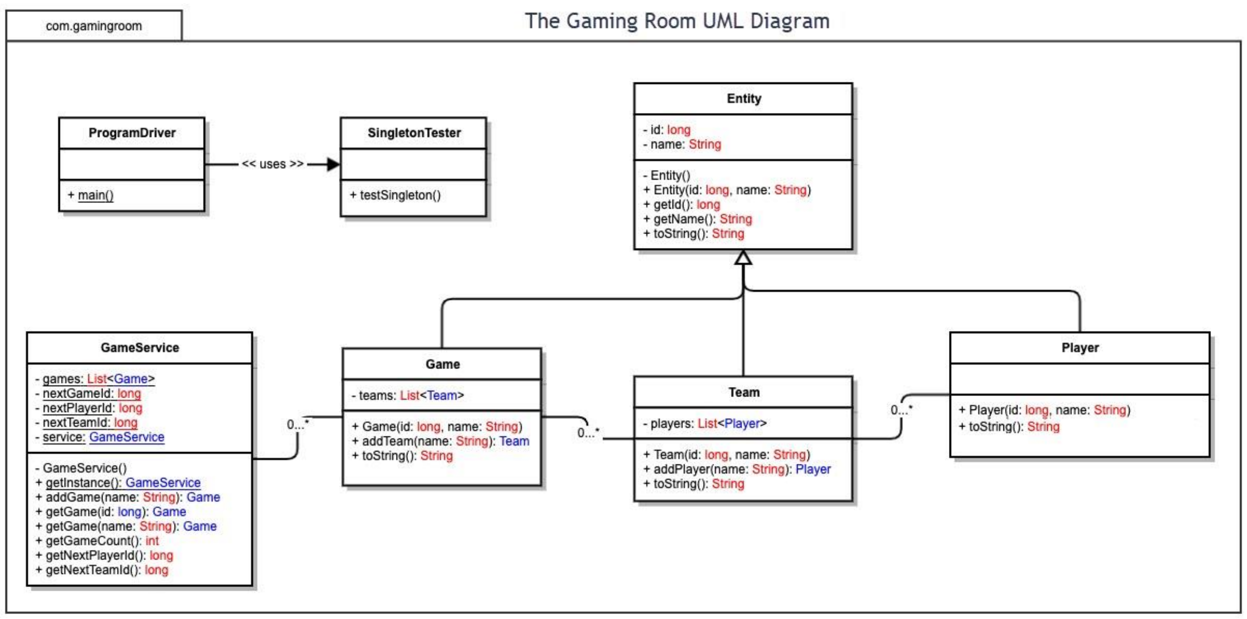
Design Constraints

This game application must be developed in an IDE compatible with the client-server software application model. The stack used to design the application must support the necessary tools to create server-hosted applications along with the proper testing tools. It must also be able to run on the most widely used web browsers such as Chrome, Firefox, and Safari. Therefore, multiple API's will need to be observed in order to ensure compatibility between different web browsers and operating systems. The application must also be developed to handle a variety of different monitor resolutions. Not everybody has the same monitor; therefore, the application must scale between different pixel resolutions. The data needed for the application, such as the images, must be able to fit in the allotted memory in the hosted application server. The images need to be compressed efficiently in order to take up the smallest amount of space while still being presentable.

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



The framework of this application consists of a GameService singleton class that is used to manage unique instances each game. The Game, Team, and Player classes all inherit from the same base Entity class. This Entity class provides the functionality needed to create unique instances of each child class. The GameService, Game, and Team classes will provide an iterator to efficiently traverse through the elements of the list defined in each class. The GameService class will hold a list of unique Game objects. The Game class will hold a list of unique Team objects. And, lastly, the Team class will hold a list of unique Player objects.

The structure of this program will allow for many unique Game, Team, and Player instances. Game and Team names will be unique as to allow users to check whether a name is in use when choosing a team name. This program structure uses encapsulation to store the attributes and methods needed for each different object type. Also, polymorphism is demonstrated through the overloaded toString methods that are in each child of the Entity class.

Evaluation

Development Requirements	**Mac**	**Linux**	**Windows**	**Mobile Devices**
Server Side	The Mac operating platform is more constrained than Linux and Windows when it comes to the hardware needed to host an application server. Mac Minis can be used in colocation data centers to provide the server hosting. The cost of the Mac Mini and colocation service plan will need to be considered. Server deployment can be achieved through the Jersey framework if hosting on a local machine. However, a more powerful Mac device may be needed without the assistance of a third-party data service. The Mac device will come with a version of the MacOS, so OS costs will be included in the device.	Linux is compatible with a wide range of hardware needed to host the application server. A dedicated server can be built with your choice of motherboard, CPU, RAM, storage device, and the network modem. In this case, the cost of each hardware component will need to be considered. Linux operating systems a free and open source. We recommend Debian. The costs for maintaining the server hardware will also need to be considered. Server deployment can be achieved through the Jersey framework. Alternatively, Amazon Lightsail could be used to host the application for a monthly fee.	Windows has a very wide range of server hardware compatibility. A dedicated server computer can be built with your choice of components. Networking hardware will need to be purchased as well. The costs of an enterprise version of Windows 10 or 11 will need to be considered. The cost of the server hardware and maintenance will need to be considered as well. Server deployment can be achieved through the Jersey framework. Various app hosting services can also be used. Amazon Lightsail is slightly more expensive for the Windows operating platform vs. Linux platforms however.	Mobile devices are generally not well-suited for web application hosting for thousands of clients. The hardware is not powerful enough to host an application server for thousands of clients. The network infrastructure is usually wireless in mobile devices, which does not provide enough bandwidth for data transmission to thousands of clients. Wireless connectivity is also not reliable enough for server hosting. Therefore, we do not recommend a mobile device for the application server hosting.

Client Side	Mac devices generally have very high-resolution displays; therefore, the client application will need to be readable on such high-resolution displays. The application will need to be compatible on the default Safari web browser. There are a limited number of MacOS devices available. Therefore, application testing will be less time consuming and costly. Security will be less of a concern in MacOS as the hardware is generally more secure than in other operating platforms.	Linux operating platforms support a very wide range of hardware configurations. Linux devices can consist of low-powered devices such as Raspberry Pi's or powerful desktop PC's. A wide range of display resolutions will need to be supported for the Linux operating platform. There are a wide range of Linux operating systems available, therefore application testing will be more tedious, costly, and time consuming. R&D towards the various available Linux distros will get costly and time-consuming. Linux is a secure operating platform; therefore, less time will be needed for securing the app for the Linux platform.	Windows operating platform support a very wide range of hardware configurations. The client application will need to work a very wide range of devices. Testing will be time consuming and costly due to the wide range of hardware that is supported on the Windows platform. Security will be more of a concern on devices running the Windows operating system. Therefore, more time and money will be needed in securing the app for Windows devices. There are a wide range of web browsers available for this operating system that need to be considered. Therefore, more R&D will be needed to ensure maximum compatibility for the web app.	There are two major mobile operating platforms to consider for the mobile web app. Android and iOS. Since the app is already natively on Android, less time and money will be required to get it functioning correctly as a web app on this platform. Android supports a very wide range of hardware. This includes very low-powered devices to powerful devices. iOS devices have a higher device power floor when it comes to the devices running the iOS. Therefore, less time will need invested for optimizing the code for performance on iOS vs. Android. Mobile operating platforms also have a very wide range of web browsers available to use. So, testing will be very time consuming and costly. Mobile operating platforms are very secure, so securing the web app will be less time-consuming.

Development Tools	MacOS has compatibility with many different IDEs. Apple provides the Xcode IDE that natively supports their Swift programming language. Visual Studio Code is a powerful IDE developed by Microsoft, and it works on MacOS. The Eclipse IDE is also compatible with MacOS. The programming language that is supported in the OS will depend on what IDE is chosen. Swift and Objective-C are the native programming languages for MacOS applications. However, Java, C++, C#, and others can be utilized with the right IDE for multiplatform app development. Xcode does have a license fee for uploading apps to the app store. This will not be needed for web apps. Multiple development teams may be needed for client and server apps.	There are a wide range of IDEs available to Linux operating systems. Some of these IDEs are IntelliJ IDEA, Eclipse, PyCharm, and Apache Netbeans, and Visual Studio Code. There are many compatible programming languages for Linux operating platforms. Some of these include Java, Python, C, C++, Perl. A more modern programming language to use would be the Kotlin programming language. Kotlin is cross-platform and can be used for both server-side and client-side web apps. There is also the Angular framework that can be integrated into Eclipse to develop the client-side of web applications. Dropwizard can be used in Eclipse to implement the server-side code, and Jersey can deploy the server.	Windows operating systems have a wide range of compatible IDEs. Windows is often used commercially to create complex web-based software. As such there are many options to choose from. Microsoft provides the Visual Studio IDEs as well as the Visual Studio Code IDE for usage on Windows platforms. These IDEs are compatible with many different operating languages, including C, C++, C#, Python, etc. There is also the ASP.NET web app deployment framework for use in Visual Studio. There are licensing fees with this approach, however. Alternatively, Eclipse and Dropwizard can be used to achieve the same thing for free. The Angular framework can be used to produce the client-side web app.	Mobile operating platforms are not a good environment for app development. Mobile operating systems are focused on portability and ease-of-use for touchscreen interfaces. Enterprise-level tools and IDEs to create multi-platform web applications are non-existent for the iOS and Android operating systems. Mobile operating systems can be useful for app asset generation. The native touchscreen interfaces of mobile operating platforms work well for creating art assets. For creating professional assets for use in applications, the Adobe software suite would be appropriate. There are licensing fees associated with this software, however.

<u>Recommendations</u>

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**:

   Creative Technology solutions recommends the Windows operating platform to host the server-side component of Draw It or Lose It. Microsoft Windows contains many useful server features for distributing, maintaining, and securing web applications. Specifically, we will be using Windows Server to host the Draw It or Lose It web application. This will allow us to use Azure App Service for deployment hosting of the Draw It or Lose It. This is a cloud-based hosting service for web applications that will allow focus to be steered towards the development and maintenance of the web application rather than the hosting network infrastructure. The cost will be less than that of building and maintaining a network of machines to host Draw It or Lose It. Instead, there will be an App Service Plan that will need to be subscribed to.

2. **Operating Systems Architectures**:

   Windows Server on Azure provides many services that will be used to host Draw It or Lose it on the cloud. The Windows NT kernel provides support for many different hardware configurations. This includes the Microsoft Azure cloud infrastructure, which runs on Windows Server. Azure includes many different architectures to provide most of the functionality that is needed for the Draw It or Lose It web app. These architectures include Azure App Service, Deployment Slots, Azure DNS, Azure Active Directory, Azure SQL Databases, and Azure Monitor.

   Azure App Service is a fully managed and configurable platform for deploying the web applications. Since Java is being used in the development of Draw It or Lose It, a Java web app will be deployed and hosted in Azure virtual machines based on the Windows Server. Multiple deployments slots can be used for development and testing, and then deployed to a production ready slot. Also, Azure DNS can be used to change the domain name of the Draw It or Lose It web application. Since this web app will need databases to hold information such as the various kinds of users and games in service, Azure SQL Database can be used to create and maintain each database needed for the app. Azure Active Directory will provide the security for Draw It or Lose It by providing user authentication and resource governance. And lastly, Azure Monitor can be used to detect and diagnose issues in the deployment of Draw It or Lose It. Azure Monitor can also be used for data collection of any monitored resources in the web app.

3. **Storage Management**:

   Since Draw It or Lose it is being hosted in a cloud-based environment, you only need to consider the cost of space required to host the application code and all resources involved. An Azure web app uses an NTFS file system at its core. The files are partitioned throughout the storage device in clusters of data in which the Windows Server operating system keeps track of. For the Draw It or Lose It web application, the solid-state storage option of Azure would be recommended. This would ensure rapid access of the game images from storage.

4. **Memory Management**:

   Draw It or Lose it will require an adequate amount of RAM on the server-side for continuous allocation and deallocation of resources related to the game for the thousands of potentially simultaneously connected users. This would mean that the server-side code will need to be in RAM for each instance of the web app that is in use. Game images will be transferred through TCP protocols in packets, and each packet will need to be processed through the systems main memory before it is sent to clients. In an Azure web app, the amount of memory and CPU processing power that is available to the web app instance is scaled up or down depending on the load.

5. **Distributed Systems and Networks**:

   Draw It or Lose It is using a client-server model. Since this is a web application, the processing resources is done on the server side. Each client will be connected to a web app instance in the Azure cloud. Clients will communicate to the web app instance through HTTP protocols using a RESTful API. Instances of each web app is spread out across many different machines in a network. Databases are used for each collection of resources needed for the application. The server-side web app instances will access information from each database, such as the collection of game images, games, players etc. Each instance will have its own defined virtual processor, storage, and ram capacity. Azure also has a fault-tolerant system in place to ensure that each web app instance will continue to operate normally if there is there is an unhealthy virtual machine in active use. If a web app instance in determined to be on a failing node, then it will be moved to another healthy physical node. Instances will be spread out across the various datacenters, so if a datacenter happens to fail, it will not affect the entire player database.

6. **Security**:

   For an Azure web app, all communication over the App Service connectivity features and REST APIs are encrypted. TLS/SSL certificates are used to secure domain names so that clients make secure connection to the domain. Clients are authorized through turn-key authentication from providers such as Azure Active Directory. Resources are also secured from outside networks as resource connections for the web app stay within the Azure network infrastructure. So, when a database is accessed through a web app instance, the connection will not be routed to outside networks. Azure Key Vault can be used to securely store user credentials by using regularly rotated encryption keys to encrypt the sensitive user information. Authorization is then done through role-based access control to ensure that only authorized users have access to the resources that they are requesting.