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	06/07/20	Michael Palatta	This is the original template document prior to any revisions being made.

Executive Summary

For ***Draw It or Lose It!***, The Gaming Room has proactively decided to expand the product's accessibility from web browser to mobile app in order to increase consumer base and revenue. The aim of this project is to create a functional, user intuitive app that is available to users via both iPhone and Android app stores. In order to convert and/or extend the current application from web browser to mobile app, different programming languages will be used to develop the app and it must be done within the parameters set forth by both iPhone and Android app store policies.

Design Constraints

The following design constraints must be addressed in creating the ***Draw It or Lose It!*** mobile app:

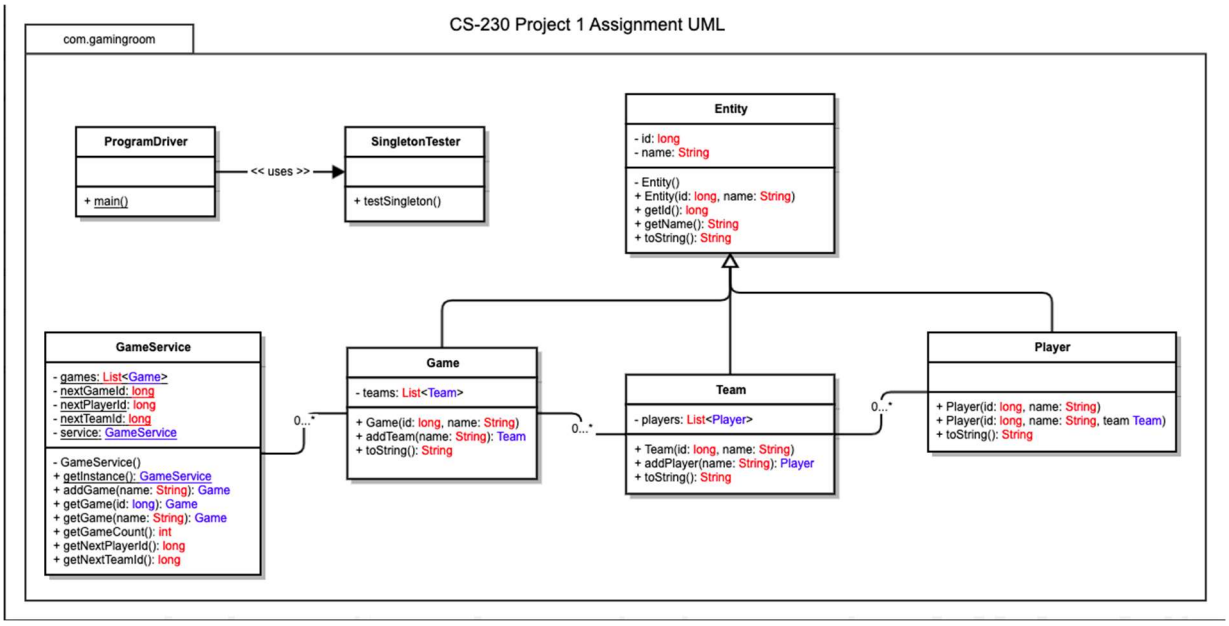
- Coding the user interface in a different programming language than the existing one
- Assigning an unbiased group of testers to provide constructive feedback on the mobile app using devices that are not native to the current web app
- Developing a generic and easily readable document so that programmers may correctly interpret the current app's logic and reproduce that in both OSs for the mobile app

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 ProgramDriver *main*() method is the driver that will call methods from each of the other classes in order to access the program, while the *testSingleton*() method is the tool that the driver uses to do so. The Entity class is the parent, from which the other classes inherit attributes. This class initiates the basic *id* and *name* attributes that will identify all objects created and used throughout the program. The Game, Team, and Player subclasses are the mode detailed objects that first takes the attributes from the Entity class, and then adds more attributes specific to that subclass. Also, each subclass will also have an attribute linking it to another. For example, an object of the Player class contains identification linking it to a specific Team in the *PlayerID* attribute. Once an Entity object is created and further detailed by subclass attributes, it may then be called and used by the methods within the GameService class.



Evaluation

Development Requirements	Mac	Linux	Windows	Mobile Devices
Server Side	-Apache server -PHP code structure -Memory, HDD/SSD -Data redundancy -Load balancing	-Apache server -C++ code structure -Memory, HDD/SSD -Data redundancy -Load balancing	-Active Directory authentication -.NET framework structure -Memory, HDD/SSD -Data redundancy -Load balancing	-iOS, Android, Windows Mobile OS supported code structure -Memory, HDD/SSD -Data redundancy -Load balancing
Client Side	-iPhone SDK -iTunes App Store supported -Adobe development software	-Adobe development software	-Windows Mobile SDK -Windows Store & Marketplace supported -Adobe development software	-Android SDK -Google Play App Store supported -Adobe development software
Development Tools	-Objective C -Swift -GoLand -Eclipse -Atom -Visual Studio -Java -IntelliJ IDEA	-Godot -C++ -CLion -Adobe -Proton	-C# -Visual Studio -ReSharper -CLion -HTML5, CSS3, JavaScript	-Java -Kotlin -Visual Studio/Code -IntelliJ IDEA -WebStorm -Eclipse

Recommendations

1. **Operating Platform**: The new app should be developed starting on the most current version of Windows OS. The template should be constructed according to the .NET Framework so that it may eventually be adapted to operate on both Android OS and iOS. This way, the app will be available across multiple platforms via Windows Store/Marketplace, iTunes Store, and Google Play App Store.

2. **Operating Systems Architectures**: This will provide access to both C++ and VB Net libraries and the primary program may contain either .EXE or .DLL extensions. They can be compiled into the intermediate language MS-IL. The code will ultimately be compiled into native code specific to the platform being used and its CPU.

3. **Storage Management**: The most appropriate system for app maintenance and usage is a tree-structured directory structure.

4. **Memory Management**: The platform will make use of both main memory with HDD/SDD systems as well as virtual memory. Date will be secured through a three-way mirror in order to provide redundancy. On-site servers will house the memory and backups allowing to account restoration when needed.

5. **Distributed Systems and Networks**: Networking for cross-platform operations will be wired and wireless. Both ways, connectivity will be subject to downtime depending on the ISP used. TCP/IP will be followed and may slow down connectivity in order to ensure authentication at times when the web servers are communicating back and forth with the clients.

6. **Security**: The platform recommended provides Active Directory implementation and control over group policies and user accounts that will be the basis for security in the app. The users device credentials will be tied to their AD authentication information to allow for the type of multi-factor authentication their particular device requires (biometrics, alpha-numeric passwords, security tokens, etc.) Additionally, resting data in the servers' databases will be encrypted and TLS/SSL protocols will be followed.