

<Draw it Or Lose It>

# **CS 230 Project Software Design Template**

Version 3.0

## Table of Contents

[**CS 230 Project Software Design Template**](#_l6ti7uoag22u)1

[**Table of Contents**](#_30j0zll)2

[**Document Revision History**](#_grjogdjh5fi8)2

[**Executive Summary**](#_sbfa50wo7nsh)3

[**Design Constraints**](#_2et92p0)3

[**System Architecture View**](#_ilbxbyevv6b6)3

[**Domain Model**](#_8h2ehzxfam4o)3

[**Evaluation**](#_2o15spng8stw)3

[**Recommendations**](#_m8aleynsvzvc)5

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 3.0 | <10/20/2024> | <William Neal> | <Updated code in multiple classes to startup a web based game>  < The goal of this documentation is to inform the reader about the overall summary of the software lifecycle. Including user features and implementation. As well as potential roadblocks and design constraints.> |

**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](#_sbfa50wo7nsh)

<The gaming Room is looking to develop and implement a cross platform web-based game. Game is only available on Android currently.>

<Currently The Gaming Room has a stable Android exclusive game called Draw It or Lose It. TGR is looking to expand its player count by expanding into the multi-platform market. The proposal surrounds a centralized game engine. We also already see in the code base a use of Singleton patterns so that only a single instance of the game exists at a time. The game could also implement a multiteam – multiplayer architecture via unique id’s for players, teams, and individual games. Again there will need to be a centralized architecture for the engine, a unique ID generator and a way to manage the ID’s, and various gaming mechanics. Some things TGR should consider in cross platform playability is that currently TGR is only seeking to enter the game as a web based application so that it can be ran from a desktop or mobile device, how will this scale long term? Will new features and user enhancements be able to scale alongside the current architecture? Will there be individual apps for Apple, Android, Steam, Xbox, PS5, and other devices? Currently an epic might look like As a player, I need a menu interface that can be used on various screen sizes and resolutions so that my user experience is seamless regardless of hardware. Or as a player, I want the game to be accessible from both mobile device browsers and my desktop browsers so that I can choose which device to use. We plan to implement robust architecture to meet requirements while considering long term scalability.>

## [Design Constraints](#_2et92p0)

<Needs multiple cross functional teams. Agile based, and proper product owners. Needs to run on multiple platforms for maximum user experience>

<The single largest issue this project will run into design wise will be the cross platform compatibility. The UI is going to need to be responsive to basically any screen size and will need to adapt to multiple screen resolutions. This is going to require tedious architecture and skilled labor. Another problem that will need to be addressed is unique identifiers for players, teams, and single games. I think this has been tackled across several games including several online games, but again we will need time and skilled agile teams. As I mentioned in a previous assignment we should also start addressing scalability. How is the company going to manage overnight success? Notice even simple games like Palworld, Among Us, Fall Guys, Hades and other small indie developed games can go from not being known to becoming culturally relevant sensations. Can Draw It or Lose It scale up to meet spikes in demand and scale down for when demand pulls back?>

## [System Architecture View](#_ilbxbyevv6b6)

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](#_8h2ehzxfam4o)

<Entity extends confirms relationship for multiple classes. Entity is a superclass, “name” and “id” are inherited in multiple classes such as Game, Player, and Team. >

<Alright the ProgramDriver is using the SingletonTesterClass, while ProgramDriver has our main() method, SingletonTesterClass has our testSingleton() method. Our Entity class has a couple of important methods, which are our constructor and our getters for id, name and to string. This is where game, team and player are going to get their attributes. This is also a great example of encapsulation, because these items will be kept private. You are also seeing that entity is the base class from which player, team and game are going to inherit from. Our GameService class has a list of games, our next id’s for game, player and team. It also has an implementation of the singleton instance of game service. There’s a ton of methods the getInstance ensures that there’s only one instance of the game service I just mentioned. Add game is simply adding a new game while we have several getters like game by name or id, get count, and a generator for unique id’s for the team and player. Lastly our game, team and player class ever have a constructor. They all have a specific add (team, game, player) and the ability to present them as a string. There’s multiple associations as well game service is servicing 0 or more games, and a game can have 0 or more teams and teams can have 0 or more players.>

**"The Gaming Room UML diagram. The top of the diagram is labeled as com dot gamingroom. Test boxes are placed in two layers. The first layer has three text boxes and the second layer has four of them. In the first layer, the 'ProgramDriver' textbox points to 'SingletonTester' textbox. The 'ProgramDriver' textbox contains the text 'asterisk main round brackets.' The 'SingletonTester' textbox contains the text 'asterisk testSingleton round brackets.' The arrow between these two text boxes are labeled 'open two angle brackets uses close two angle brackets'. In the second layer, there are 'GameService', 'Game', 'Team', and 'Player' text boxes. The 'GameService' textbox has texts arranged in two layers. The first layer contains games colon List open angle bracket Game close angle bracket, nextGamesId colon long, nextPlayer Id colon long, nextTeamId colon long, and service colon GameService. The second layer contains GameService round brackets, getinstance round brackets colon GameService, addGame open parenthesis name colon String close parenthesis colon Game, getGame open parenthesis id colon long close open parenthesis colon Game, getGame open open parenthesis name colon String close open parenthesis colon Game, getGameCount round brackets colon int, getNextPlayerID round brackets colon long, and getNextTeamId round brackets colon long. The 'GameService' box is connected with the 'Game' textbox with a line labeled 'zero dot dt dot asterisk'.  The 'Game' textbox also contains text in two layers. The first layers contains the text teams colon List open angle bracket Team close angle bracket. The second layer has Game open round bracket id colon long comma name colon String close parenthesis, addTeam open parenthesis name colon String close parenthesis Team, toString round brackets colon String. The 'Game' textbox is connected with the 'Team' textbox with a line labeled 'zero dot dt dot asterisk'. The 'Team' textbox also contains text in two layers. The first layers contains the text players colon List open angle bracket Player close angle bracket. The second layer has Team open parenthesis id colon long comma name colon String close parenthesis, addPlayer open parenthesis name colon String close parenthesis colon Player, and toString round brackets colon String. The 'Team' textbox is connected with the 'Player' textbox with a line labeled 'zero dot dt dot asterisk'. It contains the text Player open parenthesis id colon long comma name colon String close parenthesis and toString round brackets colon String. The 'Game', the 'Team, and the 'Player' boxes point to the 'Entity' textbox in first layer. The 'Entity' textbox contains text in two layers. The first layer has the text id colon long and name colon String. The second layer has Entity round brackets, Entity open parenthesis id colon long comma name colon String close parenthesis, getId round brackets colon long, getName round brackets colon String, toString round brackets colon String.**

## [Evaluation](#_2o15spng8stw)

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** | < NA >  < So, I failed to recognize that macOS has some good Unix based environments and great compatibility with Apache and Nginx. I think when I originally wrote this I had very little experience with macOS. I also think I overlooked how easy cross platform development with Xcode/Swift is. The downsides are almost nobody specifically hosts on prod with macOS. > | < Secure, free, cost effective, open source >  < Everything I said above is still somewhat accurate. I think I would also mention now that Linux makes things easily customizable. I also think the drawback of hard to learn obsolete seeing as there’s several GUI’s now like GNOME and KDE. If anything, I would argue against it only because of lack of compatibility from micro services and third part vendors. > | < More secure than linux, expensive and not as cost effective, but more support >  < Massive enterprise support. Having way more experience in the actual tech field now I can accurately and anecdotally say that my experience with Microsoft is they have swaths of documentation and amazing support. The issue I have noticed is that they are expensive. It’s a massive fight to get licensing on particular tools at work. The other thing I have noticed is scaling isn’t cost effective compared to other options on the market > | < NA >  < I still don’t know anyone who’s tried this or if it’s even possible. I just don’t understand how you expect to meet processing power via an android phone. Like is there even hosting for a mobile device? I thought maybe there is someone out there who has tried this, and to my surprise there is. Using postmarketOS you can get Alpine Linux on your mobile device and then grab some necessary apps. There’s even this guy (<https://android.thin.computer/>) who successfully did it. Surely isn’t built for optimization nor would I ever suggest this, but it is possible. > |
| **Client Side** | < SCRUM / Agile  Multiple Devs  Could lower cost with off short teams  Dev Team specifically for API’s >  < I think I failed to mention that Mac is very expensive to work with. The native hardware tends to be some of the most expensive on the market. Also note that the goal is to be cross platform and Mac/Apple is the least willing to play with the others, so, you’re going to need to have a high skill set team. Time spent will probably be slightly longer because of this. > | < Least Costly  Lots of Open Source information  SCRUM Teams  Dev Team specifically for API’s >  < Again this is the least expensive route to take. No licensing fees and it’s open source. Linux likes to run on whatever hardware you got and they play well with other frameworks. Here’s the hang up. If you are building this out you’re going to need someone who knows their way around a shell and they’re going to need to know full stack and I mean full stack. JavaScript, Vue.js, Angular, React, Node, SQL, Java, Python. And to be frank probably some DevOp’s stuff too. While GUI’s have improved package managing hasn’t and the skillset laid out is hard to find as it is. One Angular upgrade can break everything now multiply that by dozens due to dependencies like Ubuntu and Debian. I would say the time spent will be longer than the others. > | < SCRUM Teams  Again Multiple Devs  Common Languages and IDE’s less costly than Mobile  Dev Team specifically for API’s >  < The biggest downfall of a Windows/Microsoft based project is the cost. I personally love VS and VS code. I have a For-Loop in C# tattoo’d on my arm. I have always had an infinity towards this environment. The issue is, its extremely expensive and C# can be complex for the lay-developer. On the flip side of the coin individuals who are familiar with .NET frameworks are in my opinion more specialized. What I mean is that because .NET is so Microsoft/Windows dependent, individuals devote themselves to the platform. Where in my opinion someone who writes Python (which is a scripting language not a programming language, yes this is a hill I’d die on) will be more flippy-floppy and probably more versatile in other frameworks and languages. I would argue that this project would be the least time consuming but most expensive. > | < Most Expensive and hardest to get talent for.  Multiple Teams  Performance might be better  Updates are harder  Can be opened anywhere  Run locally on device >  < So I am not sure if in 2024 this is the hardest and most expensive. I say this because over the past 4 years since I took this class pretty much every company I know has some mobile app now that they want you to download. Even SNHU has a mobile class where you help build out an Android App. I think the expense here is odd because you will need Apple hardware for an iOS app and Window/Linux/Other for your Android App. Cost overall though will be minimal due to very minimal licensing fees. And also very minimal expertise. It seems like Kotlin and Swift are very easy to pick up on. As well as Python which a troglodyte could understand. This ease to learn might also explain why the local movie theater, the DMV, your doctors office, your insurance company, your local clothing retailer and Dunkin Donuts all now have mobile apps they’re begging you to download. > |
| **Development Tools** | < Objective C or Swift, maybe even some Java  IDE-Xcode or Eclipse >  < I pretty much standby what I wrote in here. It also doesn’t really specify whether we are building the app here or for here. I think maybe the only thing I would add is Cocoa, which I had the displeasure of encountering in a side project. > | < C and maybe some python?  Eclipse? >  < So, as you can see, I was rather unfamiliar and instead of looking it up or asking questions I threw something at the wall and hoped it stuck. With Linux you have some create freedom. So C is still your standard and C++ will be right behind it. You can really build out with Java and python if you wanted to as well. GNU would be your go and you should also be familiar with Bash/Shells. Most IDEs will be compatible with Linux. > | < C# or Visual basic  IDE Visual Studio or Visual Studio Code >  < This is personally my favorite environment and framework. With Windows I can write python, to Java, Java Script, C++. I can use pretty much any IDE (though we all know VS is the best and you should never want to use anything else but I digress). PyCharm, NetBeans, Xcode, Atom, JetBrains (IntelliJ and Rider), Eclipse, Android Studio and several more are all Windows compatible. Windows PowerShell is my preferred shell but Git-Bash is a close second. Both run on Windows. .NET and .NET core and C# were my first framework(s) and language so I am clearly very biased but the fact remains that in Windows you have a lot of flexibility > | < Android studio with Java or Kotlin  XCode with Swift >  < I wasn’t wrong with my first statement. Swift and Objective C with Xcode. As well as that awful Coca framework. Java and Kotlin for Android. Android Studio for your IDE. I have seen people use Flutter with Dart but I am really unfamiliar with both. > |

**< \*\*\*\* Special notes. Web apps are run on the server(PHP usually and MySQL), the client facing app is a POM. UI Testing is done in the QA environments and POM is usually taken by happy pathing, relative or absolute paths of various xpaths. Web app UI and API can be easily tested with automation (cucumber, karate, selenium, gherkin, etc.) This makes testing (Unit, Component, Smoke, Regression, and Performance) much easier than testing on Mobile Apps. On a special note, certain Web Apps need to have specific Accessibility for the seeing/hearing impaired. Web based apps frameworks include REACT, Angular, Bootstrap (if you hate your life), Vue.JS and JQuery. Most common languages for Web Apps would be JavaScript, HTML, Bootstrap and CSS. With Mobile you basically get stuck with Swift. Not sure if this covers a complete break down, but it was an attempt. \*\*\*\*\*\*\*\* >**

**< Yeah, not sure what this was? While not wrong I am not sure why I included this explanation. I did have a point though; we don’t seem to cover much database wise and I might add that information in for project three when I go back and refactor. Also, not sure why I was only speaking about iOS? I can see why I earned a C last time I took this class. >**

## 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**: <Would highly suggest windows. C#.NET, C++, with Visual Studio/Visual Studio Code being preferred IDE’s.> <Because the Gaming Room wants to be cross platform and we should think about scalability now, instead of later. I would think we should use something like React for the actual application. I would not suggest Unity after their debacle in leadership with John Riccitiello. The game/app itself needs containerized for development purposes. Look into Docker, Docker Compose, and Docker Scout. Alphine has amazing debugging capabilities in Scout, just don’t shard. Now base platform should probably be cloud based. Would highly suggest AWS or even Azure if you want to be a little more cost effective and not have as many bells and whistles.>
2. **Operating Systems Architectures**: <Windows GUI can grab any necessary data via Window’s user accounts.> <We could honestly get away with 32-bit, but I would suggest 64-bit just for scalability and let’s say the graphics scale up. I wouldn’t want to be in a position where we have the revenue to improve graphics, the talent to do so but we are stuck in a 32-bit architecture. Also just from experience here at SNHU, the graphics class makes you run OpenGL on your machine. I personally use Windows and Linux. OpenGL projects are not optimized for either. So when building out this game we should be paying attention to how we effectively optimize for each operating system.>
3. **Storage Management**: <Preferably SQL database, need to be aware of any potential memory leaks with C++.> <This is a case of cloud storage. There is no way in good faith I can say we should hire an IT department, purchase real estate with the capacity to host servers in a completely temperature controlled environment, then buy and build out the servers, as well as pay the electric bill for them. I would highly suggest speaking to a solutions expert at AWS. We could probably use a mixture of S3, EC2/Block Storage, Glacier, Snowball, and maybe Gateway. Yes will we need some in house expertise in proper formatting, of course. But I do not think we need local hosting. Any database we might use could also be cloud based. While I personally detest DynamoDB it might be a solid solution here. We could also use MongoDB or MySQL. Oracle could be another choice; they just tend to be expensive at the enterprise level.>
4. **Memory Management**: < C++ allows the allocation of memory of various variable or an array in run time. Or better known as dynamic memory allocation. C#.NET the compiler would manage and allocation of memory to variables. > <In my first answer I think I was headed down the right road but missed the mark. I was correct, we need to worry about memory leaks. We need to ensure that we are collecting the garbage, especially if we are using C++ or even Python for development. The last thing we want is poor RAM usage. Nowadays lazy loading is done in almost every game. I say this as someone who avidly plays video games in their little spare time. When I load up Silent Hill 2, 40k War-Hammer SM2, Mario Party Jamboree, or any modern game for that matter. It only renders what I am actually interacting with at that moment. There is no reason to render the whole game or the level when I might only be interacting with, 2% of the environment in any given moment. We should be applying that same principle here. We load and render every single image in the game, why not just render what the player is currently seeing. Lastly, as I mentioned before we need to take into consideration now that these base operating systems have different architectures and requirements. We should be thinking about how to optimize things like OpenGL, DirectX, etc. on a variety of environments.>
5. **Distributed Systems and Networks**: <Physical servers and possibly virtual machines. Would highly suggest a Three-tier distribution system for the web-based application .> <Since I am directing this to be a cloud-based application that has cross platform multiplayer capabilities I would think we need to have automatic load balancing. I think AWS with S3 would be able to handle this. I think they offer services like Elastic Beanstalk and ELB (Elastic Load Balancing). As far as scaling these services, AWS offers ECS and EC2. We could use WebSocket’s for better communication between players to players, and players to our server. I would suggest something like AWS API Gateway since we are already using their cloud-based environment. Or maybe something like Socket.IO. I only suggest Socket.IO because I know the library has some really amazing tools for integration and its cross platform friendly.>
6. **Security**: <Windows built in security. > <We should investigate data encryption microservices like Sophos or Thales group. There are also plenty of built in encryption services built into AWS but for an added layer of cloud security I would recommend a microservice. Yes we should be using AES-256, but the more layers of protection we have the better. I would rather be in a tank and only need a shield, than to need a tank and only have a shield. I think AWS Shield would protect us from any DDoS attacks. In light of recent news we should probably avoid Cloudflare. And I would suggest Helix for user authentication services for the development team and I would suggest Clerk, AWS Cognito, or Auth0 for user authentication. Either way we should be using some form of JWT or oAuth2 for users and developers. While I don’t think we are a primary target, these are just basic industry practices.>
7. As I look back on my answers in red, I realize I had no idea what I was talking about 4 years ago. I don’t even think I was sure what the question was really asking. I think I am still really clueless but now I know how to think about what the question is asking and if I don’t have an idea for the answer, how to find it. I am not sure if I was ready for this course so early in the program. It might have helped me to take some other CS courses before this one in retrospect. Also, I don’t mean to sound like an AWS fanboy, they just have a lot of solutions for cloud based applications. I have been seeing them more as the Walmart of microservices and solutions. While they might not be the best solution, it’s convenient and sometimes extremely affordable if you’re already in their environment. I tried to include some other solutions just to give some flexibility but ultimately I wouldn’t imagine this product having a massive user base. I say this and forget that Minecraft started off as a small crappy indie game, same as Cuphead and Among Us. So maybe it’s better to build like you’re going to be the next Palworld. As a last note I hope I have shown some improvement and growth over the past couple of months, I can without a doubt say I see growth over the past 4 years (Like seriously a 4-word answer on security? I don’t think I even answered the operating system architecture question. What was I thinking?). I look forward to any feedback and hope you have a wonderful week.