

<Name-of-Software-Application>

# CS 230 Project Software Design Template

Version 1.0

## Table of Contents

[CS 230 Project Software Design Template 1](#_Toc115077317)

[Table of Contents 2](#_Toc115077318)

[Document Revision History 2](#_Toc115077319)

[Executive Summary 3](#_Toc115077320)

[Requirements 3](#_Toc115077321)

[Design Constraints 3](#_Toc115077322)

[System Architecture View 3](#_Toc115077323)

[Domain Model 3](#_Toc115077324)

[Evaluation 4](#_Toc115077325)

[Recommendations 5](#_Toc115077326)

## [Document Revision History](#_grjogdjh5fi8)

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 01/26/2025 | <Bharat Sharma> | The changes are made to the cover pages with the document revision, with executive summary, design constraint, system architect, domain model and last recommendation. |
| 1.0 | 01/25/2025 | Bharat Sharma | The development requirements within the evaluation section according to their needs. |
| 1.0 | 01/24/2025 | Bharat Sharma | The change has requested to following section and change is made to:  Recommendation |

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 games we are developing is web-based and app-based games that serves multiple platforms with numerous operating platform such as PlayStation, Xbox, PC, and Nintendo as well as Phones. It is based on many popular games such as draw and lose it, ludo and Snake and ladders. The game application has multiple consisting with bring in variety of players from all over the world. The application uses images from library of images tills it fully renders. My team does not wait around games to fix itself, but my team is fully engaging into guessing opportunity with in 30 second time limits. Right now, the game concept is in file projects, and we are following with meeting every morning where our game going and what other platforms, we can bring inn.

The game named and usernames must be unique so the user can check their usernames as well as their progress in games. The game may exist in memory at ang given time, so most be using singleton pattern to it. The main point will be the games must handle more teams with multiple player’s function.

## Requirements:

The games must have the ability to more than one teams involvement.

Each team will assign more than 10 players.

The game names and must be unique to allow user to check weather ID is taken by other players.

The game can exist in memory at any given time and when player quit the game due to network issues, they must re started the game get the best experiences.

## [Design Constraints](#_2et92p0):

Fewer constraints will exist within this games and projects; the form of making a consistent experience within this we-based and app-based application. The goal is to reach wider audiences with successfully where it includes training out team to keep up with maintaining program completion after done building up the game. Maintaining the program weather games became successful lunch or not because not all the operating platforms we get are not equally made or preforms better.

The software IOS and Android and many other operating system programs will be designed as an application available in the Google Play, Apple App Store, and other many gaming systems, each respective system will require a landing page with Get on the X platforms or available in X Gaming Systems. The game should be played from anywhere to any platforms to reach vast and verity of peoples. A complete cross-platform program is requiring the team will function normally with react-native-web and incorporate all modern brewers to Operating systems across Mac, Window, Linux, Android, IOS, Nintendo, X-box, and Sony operating systems. The user data management will hold user data privately and securely in a platform in independent way. We should hold end of our premises to our user by creating user login and authentication, with a restful approach to it.

## [System Architecture View](#_ilbxbyevv6b6):

This point is not included here but I do want because with-out support planning of design of building you cannot plan what type of building your making it. It is the reminder of systems and sub-systems present in the game applications that includes physical components and their tiers. Logical communication and storages aspects is also necessary to understand the architecture and must be provided.

## [Domain Model](#_8h2ehzxfam4o)

We have created the class for entity teams, Game service, game players, programmer and tester where we inheritance relationship games, teams and players. The dependency of the program creates the structure needed to iterate through lists that exist within the team and players classes, which helps enact a requirement for the game to ensure that there is only one instance of each team and each player during game.

The program-driver class holds the main function which open ups a singleton games memory where other class singleton-tester to prove there is one instance. Game-service use public get instance method to obtain singleton. Both encapsulation which helps prevent data being modified and implementation details will be hidden with demonstrates in singleton which is used for to create the static of Game-service. Game-service use polymorphism methods for overloading of the get game function where it allows invocation of using correct methods of checking signatures. Both get-game and game-service demonstrates the iterator pattern. Games-service can have 0 to any amount of children classes. Since game-service has a List<game> typed static variables; game has a team relation, and team has a player. The Entity class holds common attributes, behaviors from each player, team, and games inherit the demonstrates abstraction and aids in the forms of codes.

"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 | Apple include built in web-service where its commands a robust such as python and commands to configure to access the server. The Mac is highly intuitive with client’s creation policies compared to Window.  The downside of Mac is user is that it has limited hardware configurations and runs.  Up to Date:  It will run in minimal cost by running an HTML web socket provides a client a API to connect with nods.js or backend server. If the Server crash, we needed the authentication methods of web sockets sessions. | Linux also has powerful command systems, with the benefit of being cost-effective over Mac and Windows. Linux is more reliable and can run for long periods of time with good hardening practices.  Up to Date:  Linux has the best hardening, including strong and unique passwords, multi-factor authentication, SSH keygen, and software updates regularly.  Firebase can be used as a cloud-based database solution to scale to a more significant number of users but expect higher costs. For example, we can use a multi-factor token login system using react web JSON objects for user authentication with a Flask API. | Compared to other OSs, Windows has a wide variety of software. Users claim familiarity, ease of use, web extensions, and SQL support.  **Update**:  A Windows server solution can start at minimal costs, rolling a personal Node.js with RESTful authentication through a React-native web-based front-end. Code portability across browsers and Android and IOS systems. This solution can start cheaply and scale up if more computing power is needed.  Amazon offers **cloud-based** AWS micro server solutions, as does Heroku, which will allow apps to scale up infrastructure with more users. A Heroku microservice could hold API functions to help **load balance** users amongst game rooms while ensuring minimal downtime with DDoS mitigation, spoofing protection, destructive port scanning, and host-based firewalls. Compared to Linux and Mac, Windows has many unique tools that can only be virtualized on other systems. | The option to code the backend and host the mobile app is less viable than a Linux / Mac / Windows-based solution. This option would not work well with the current specific skillset we are seeking, as it would require a ground-up approach to coding and cross-app development.  It provides clients with the flexibility of having the app anywhere, anytime. However, developing the app requires adjustments to screen real estate differences—all screen size possibilities for tablets, smartphones, and browsers. |
| Client Side | Mac is costlier compared to Window and ease as use windows; it is moderate times to learns an intuitive interface.  It should hire independent web developer for under $Xk. A app can create hybrid rather allowing a portability with faster markets and cheaper costs. | <Linux is ideal for software and web developers because of its cost-effectiveness. Open-source programs that work in unison with the system. Compared to Mac and Windows defaults, Linux requires more time to learn.  **Update**:  Since React-native web can be implemented to support iOS, Android, and the Web platform with a single codebase, this solution is optimal for front-end client interaction. Front-end JWT-based authentication can connect to our Flask or Firebase database over secured web sockets on our server. | React-native for web codebase can also be worked on in a Windows environment using Visual Studio Code.  Summary of development needs:  **Flask dev, React native web dev, Security dev for proper authentication, upscale with cloud-based solutions** | Utilizing React will allow responsive media adjustments to conform to IOS and Android phone form factors and scale up the web interface. |
| Development Tools | The package manager to install Unix, Mac utilities, XCode IDE, Iterm2, Dash api brewers. In additional JavaScript, HTMLS, CSS, platforms languages. | In addition, JavaScript, HTML, CSS, React, react-native, react-native-web, npm, and yarn. Chrome development tools, MySQL. | In addition, JavaScript, HTML, CSS, React, react-native, react-native-web, npm, and yarn. Chrome development tools, MySQL. | Firefox, Opera, Samsung browser, Chrome, Metro browser. The website should work across all mobile browsers, and mobile browsers are required to test both the website and the app itself.  App: JavaScript should be enabled on IOS and Android to allow app access. Google Play Store and Apple App Store. |

## 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: The goal of this game is to make it for multiplatform and cross-gen operating system. Mobile and web based coded with react-native-web framework for any operating systems. This will make an implementation of native components, Api on the web, react Dom in front; with this it will run in any operating platforms.

Since recommendation is mac, 64x-based server; the game is web, android, IOS, and any other OP system compiling, and code testing is requiring Apple SDK. Which is make the game lunching code compile software where it can run smooth to given platforms SDK.

1. Operating Systems Architectures: Recommendation is flask on a mac x64 server with Api server-side language, develop in virtual environment with consistent of dependencies. Should ensure scalability with cost effective, and Heroku function manage load and balancing images which will store in AWS bucket.
2. Storage Management: Any backup should be automated and backup mirroring of server code must be user data compression for dependency config, and virtualized. Since there will be many networks latency for mobile and webapps we con optimize performance through optimize image, code compression, and caching for database queries Etc.… The best example will be Amazon has all the above-mentioned storage features which allows them to work efficiently, fast, and relatively inexpensive data storage.
3. Memory Management: So, the recommendation of debugging the leaks using given platforms for Android, Mac, and any other platform the game is lunching with using there coding software under profile setting. We must utilize web development and mobile app application techniques, images in memory objects, serving images via Amazon AWS; which allows faster downloads and reduce throttling on web bowers and remove discounted games session and objects from memory to avid leaks.
4. Distributed Systems and Networks: To reach the possible users, distribute the software from various sources; a web page for the app will have a link to the Facebook page and Twitter page for development updates, Google Play Store download, Apple App Store download, as well as Mac, Linux, and Windows packages in both x64 and x86 bit versions. Use WebSocket’s for low-overhead communication between server and user for in-game chat and image drawing—client-server model with a RESTful approach for user authentication. With many players, we need several replicated servers that connect to a Heroku-based function; this helps load balance users across regions for matchmaking in-game. We must have a database server to store user credentials.

Security: To protect user information on and between various platforms, consider the recommended operating platform's user protection and security capabilities. Encrypt access to critical user data on the database server. Ensure proper exception management and handle any exceptions. Implement platform-agnostic authentication with secure password recovery. Ensure software libraries and dependencies are updated within the virtualized instance—close unnecessary ports on the webserver. Recommended technologies and approaches for user authentication must involve collecting user data in a RESTFUL approach, preferably through a React or Angular javascript-based front end that is portable across devices, and sending that data to our web server, which contains the game logic. For example, the server may include a WebSocket on Node.js to render images persistently since they must be in real-time. Messages for a chat application can also use a Websocket on a Flask server.

Works Cited

*Design patterns - builder pattern*. Tutorialspoint. (n.d.). https://www.tutorialspoint.com/design\_pattern/builder\_pattern.htm.

YouTube. (n.d.). YouTube. https://www.youtube.com/watch?v=L6SK\_R8o7bQ