**The Gaming Room**

# **Software Design Template**

Version 1.0

## **Table of Contents**

Software Design Template……………………………………………………………………………..1

Table of Contents………………………………………………………………………………………. 2

Document Revision History…………………………………………………………………………… 2

Executive Summary……………………………………………………………………………………. 3

Requirements…………………………………………………………………………………………… 3

Design Constraints……………………………………………………………………………………... 3

Rationale…………………………………………………………………………………………………. 3

## **Document Revision History**

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 05/15/2022 | Kenneth Paulsen | Added Summary, Requirements and Constraints |

Instructions: Fill in all bracketed information on page one (the cover page), in the Document Revision History table, in the footer, and below each header. Under each header, remove the bracketed prompt and write your own paragraph response covering the indicated information.

## **Executive Summary**

*Our new client, The Gaming Room, wants to develop a web based game for multiple platforms. This game is based on their current game, Draw It or Lose It which is currently only available as an Android app.*

## **Requirements**

***Business Requirements:***

*Web based game must be based on their current game: Draw It or Lose It.*

***Technical Requirements:***

*App must be compatible on multiple operating devices*

*App will rely on Client/Server architecture*

*App must be compatible to work on multiple web browsers*

## **Design Constraints**

*1.* The *app will need to be developed with web based framework/programming languages. Given the scope of the project, this will require multiple frameworks/languages.*

*2. Programming language versions and app features would need to be balanced to account for different browser versions (i.e. some functionality on newer browsers isn’t compatible with older browser versions).*

*3.* *For full functionality, the app will require backend and front end development. This requires either front end/back end developers and/or full stack developers on the project.*

4. UX/UI designs and functionality will need to be responsive and customized to accommodate different devices. A UI on a mobile device will have different factors to consider compared to a desktop utilizing app.

5.Testing and QA will be larger in scope and more complex due to the breadth of desired compatible operating devices.

## 

## **Rationale**

*1.*  **The *app will need to be developed with web based framework/programming languages******-*** *The clients require the app to be compatible with multiple operating devices (i.e. desktop, mobile, etc.). For the best results, this requires programming languages that are optimized for each field (web, mobile). Frameworks can also be used to help speed up the development process. Depending on the complexity and scope, this could require one or several frameworks*

*2.* ***Programming language versions and app features would need to be balanced to account for different browser versions -*** As new features are added to browsers configurations change and some of these features aren’t backwards version compatible. Implementing similar functionality between browser versions would add more work and time however this would also ensure the app is accessible for an optimal number of people. While the app will never be accessible to 100% of Internet users, finding an optimal balance is important for both technical and business reasons.

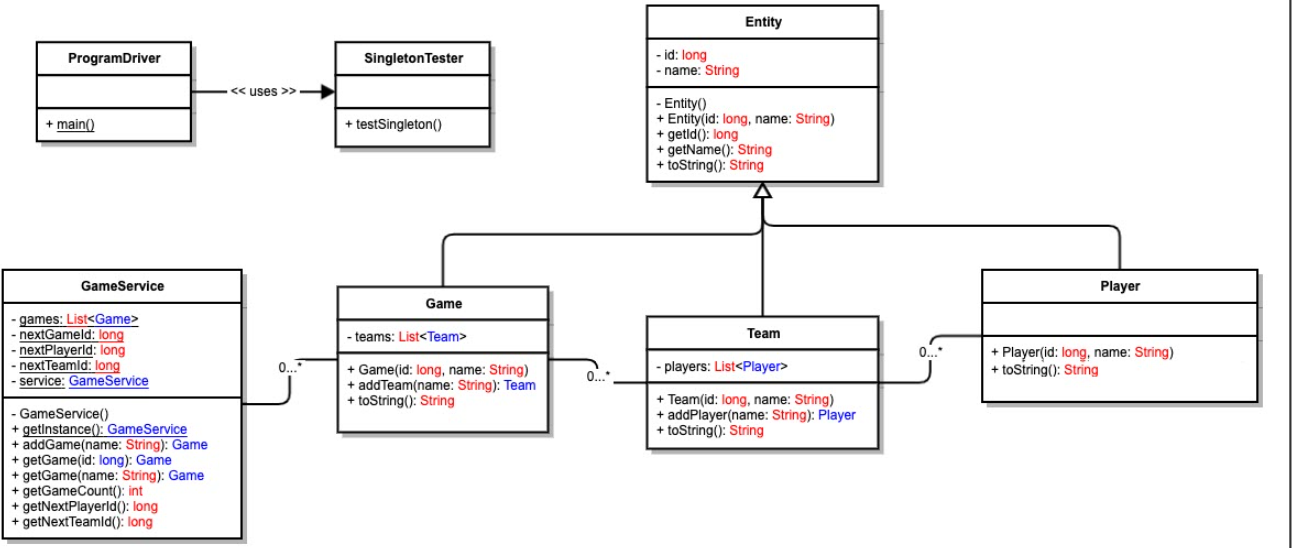
*3.* ***For full functionality, the app will require backend and front end development –***  The majority of web apps utilize both front end and back end functionality. Front end primarily focuses on user experience and input. Back end primarily focuses on Database optimizations and API calls between different parts of the application. Based on the desired features, the app will require both front and back end development. Each side of development has its own special skill sets and knowledge needed to develop effectively. Depending on the capabilities of the development team this could be individuals that specialize in one of these areas or a generalist that can do both.

4. **UX/UI designs and functionality will need to be responsive and customized to accommodate different devices-** Depending on what device is being used will determine the proper layout needed when using the app. For example, mobile phones have constraints desktops do not have such as smaller viewing dimensions, different orientations and  *fewer input options. Some UI features supported on a desktop, such as navigation bars, have limited mobile support. These differences will require the development team to customize the UI/UX experiences for each type of device.*

5. **Testing and QA will be larger in scope and more complex due to the breadth of desired compatible operating devices -** The scope and complexity of this project means more resources need to be allocated for testing and QA purposes. Supported features will be different across operating devices as well as different browsers (and their versions). For the app to work across these different scenarios, this requires extensive and rigorous QA, testing and debugging. Failure to do so will result in additional resources being spent over the long term fixing bugs and features.

**Domain UML explanation:**

The classes Game, Team and Player all share an inheritance with the Entity Class. There are also many multiplicity relationships in this diagram. GameService class has a 0 to many relationship with the Game class, meaning each GameService can have 0 or many instances of the Game class. This exact relationship also occurs between the Game class and the Team class (0 to many teams per Game class) and between Team and Player classes (0 to many players per team). In terms of OOP principles expressed in the diagram we have Inheritance where Game, Team and Player classes are inheriting from Entity class. We also have examples of Encapsulation in the Entity class. The variables id and name are private. They are given initial values when the Entity class is created. However they can only be accessed using the getter methods in the class and cannot be changed after initialization. There also exists polymorphism through method overloading. For example, the GameService class has a getGame method that can look up a game by either Id or by its name. Since it has several ways of accomplishing the same task, this is an example of polymorphism.

**