Park Station Manufacturing

# 

# Software Design Template

Version 1.0

## Table of Contents

[Software Design Template 1](#_Toc21086455)

[Table of Contents 2](#_Toc21086456)

[Document Revision History 2](#_Toc21086457)

[Executive Summary 3](#_Toc21086458)

[Requirements 3](#_Toc21086459)

[Design Constraints 3](#_Toc21086459)

[Rationale 3](#_Toc21086460)

## Document Revision History

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 09/11/2024 | Park Station Manufacturing | In this revision, the Executive Summary section now includes client names and condenses the problem description into 2-3 sentences, outlining the client’s needs and goals. Business and technical requirements are clearly listed for each client. The Design Constraints section identifies three specific constraints and provides rationale for each, explaining their impact on the project. These changes ensure a clear and focused summary of the client’s needs and constraints for the software design project. |

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

Client’s Name: Park Station Manufacturing

Problem Summary: Park Station Manufacturing needs to create training videos using high-quality video editing software available only on Macs. However, their current setup uses Windows exclusively.

Business Requirements: Produce high-quality training and educational videos.

Technical Requirements: Use Mac-compatible video editing software despite the existing Windows environment.

## Requirements

Business Requirements: Produce high-quality training and educational videos.

Technical Requirements: Use Mac-compatible video editing software despite the existing Windows environment.

## Design Constraints

Constraint 1: Mac-Only Software

Explanation: The requirement for Mac-only software necessitates finding a way to use Mac-based tools within a Windows infrastructure, possibly involving virtual machines or remote access solutions.

Constraint 2: Infrastructure Compatibility

Explanation: The current Windows-only infrastructure presents challenges for integrating Mac software. This constraint may require additional solutions or modifications to facilitate compatibility.

Constraint 3: Cost of Acquisition

Explanation: The expense of acquiring Macs and software licenses could impact the overall budget. This constraint requires evaluating the financial implications and exploring cost-effective alternatives.

## Rationale

* Mac-Only Software: This constraint affects the choice of tools and may involve additional infrastructure changes to accommodate Mac software in a Windows environment.
* Infrastructure Compatibility: The need to integrate Mac software with Windows systems can complicate development and may require additional resources or solutions to ensure seamless operation.
* Cost of Acquisition: Budget constraints related to acquiring Macs and software affect the overall feasibility of the project and necessitate careful financial planning.

**Recommendations Section**

Recommendation:

For The Gaming Room’s Draw It or Lose It game, I recommend using a cloud-based platform like Amazon Web Services (AWS) or Microsoft Azure. Both of these platforms offer cross-platform support and allow the game to expand into multiple computing environments (e.g., Windows, macOS, iOS, Android). Cloud platforms also offer flexibility, scalability, and reliability necessary for managing traffic spikes and distributed gameplay.

* + Rationale: A cloud platform provides infrastructure as a service (IaaS) and platform as a service (PaaS) solutions, enabling seamless deployment across multiple operating systems. It also supports high availability, auto-scaling, and global reach, which is crucial for multiplayer online games.

2. Operating Systems Architectures

Description of Operating Platform Architectures:  
The chosen cloud platform, AWS or Azure, leverages virtualized environments to host services on various operating systems. These platforms utilize a multi-tenant architecture where multiple applications share underlying hardware while operating independently within their respective virtual machines or containers. Here's how it aligns with *The Gaming Room*’s needs:

* Virtual Machines (VMs): Allow hosting different instances of the game on different operating systems. For example, a VM can run on Linux for backend services, while another can support Windows for administrative interfaces.
* Containers: Tools like Docker and Kubernetes allow packaging the game’s components into containers. This improves portability between different environments (e.g., testing, production) and operating systems without worrying about compatibility issues.

3. Storage Management

Recommended Storage System:  
I recommend using cloud storage solutions, such as Amazon S3 (Simple Storage Service) or Azure Blob Storage. These provide scalable object storage suitable for storing player data, game states, and assets such as images, sounds, and other media files associated with *Draw It or Lose It*.

* Rationale:
  + Durability and Scalability: Cloud storage systems offer virtually unlimited storage with high availability and durability. The data can be replicated across regions to ensure availability and prevent data loss.
  + Access Control: Access permissions and roles can be defined at the storage layer, ensuring security for sensitive user data and game resources.

4. Memory Management

Explanation of Memory Management Techniques:  
The recommended platform (AWS or Azure) provides robust memory management techniques through its use of virtual memory and dynamic memory allocation. Here’s how these techniques work for *Draw It or Lose It*:

* Memory Allocation: Both cloud platforms automatically allocate memory resources based on the needs of the application. For instance, during peak times when more players are accessing the game, memory resources can be scaled up, ensuring that the game doesn’t experience slowdowns.
* Caching: Memory caching solutions like Amazon ElastiCache or Azure Redis Cache can be employed to reduce latency by caching frequently accessed data (e.g., player profiles, game states). This helps optimize memory usage and improves performance for distributed systems.

5. Distributed Systems and Networks

Explanation of Distributed Software and Networking:  
To enable communication between various platforms, *Draw It or Lose It* will need to implement a distributed architecture with an underlying network infrastructure that supports cross-platform play. Here’s how this can be accomplished:

* Microservices Architecture: The game can be broken down into smaller, independently deployable services (e.g., matchmaking, player management). These microservices can be distributed across different regions or data centers for redundancy.
* Networking: The game will leverage cloud-based load balancers (like AWS Elastic Load Balancer or Azure Load Balancer) to distribute traffic evenly between servers. These networks ensure optimal connectivity between clients and servers, reducing latency and preventing outages.
* Resilience to Failures: Both platforms offer mechanisms to automatically detect and recover from outages. Services like AWS Route 53 or Azure Traffic Manager can reroute traffic if one region becomes unavailable, ensuring continuous gameplay.

Network Dependencies:

* Latency: Since the game will be deployed in multiple regions, low-latency connections must be established between data centers and players. Cloud platforms offer edge locations and content delivery networks (CDNs) like CloudFront or Azure CDN to minimize latency for global users.
* Outages: Cloud platforms offer high redundancy through multi-region deployments, meaning if one server or region goes down, others can take over, minimizing downtime.

6. Security

Protecting User Information Across Platforms:  
Security is paramount for *Draw It or Lose It*. The following methods will ensure data is secure both in transit and at rest:

* Encryption:
  + Data at Rest: All user data, including personal information and game progress, should be encrypted using services like AWS Key Management Service (KMS) or Azure Key Vault.
  + Data in Transit: Use Transport Layer Security (TLS) to encrypt communications between clients and servers, preventing unauthorized access during data transmission.
* Authentication and Authorization:
  + Implement strong authentication mechanisms, such as OAuth 2.0 and Multi-Factor Authentication (MFA), for users logging into the game.
  + Use Identity and Access Management (IAM) roles to enforce least privilege access to game resources, ensuring that only authorized systems and individuals can access sensitive data.
* Regular Security Audits and Compliance: Utilize services like AWS Shield or Azure Security Center to continuously monitor security threats and ensure compliance with standards such as GDPR or CCPA, protecting users’ privacy rights.
* DDoS Protection: Implement cloud-based DDoS protection services like AWS Shield or Azure DDoS Protection to mitigate the risk of attacks that could disrupt gameplay.