# Requirements Specification – coPlay

## 1. Introduction

This document describes what the coPlay system should do. It includes details for two types of use cases: 1.1 an internal system for 100 users in an organisation, and 1.2 a large-scale version that supports up to 1 billion users. These requirements will help guide how the system is designed, built, and tested.

## 2. Functional Requirements

These are the core features that must be supported by the system:

**FR-1**: Users can send chat messages using the POST /message endpoint.

**FR-2:** The system will decode messages from base64 and share them with all connected peers.

**FR-3:** Users can move disks in the Tower of Hanoi game by clicking towers using the GET /tower?tower=X endpoint.

**FR-4:** Tower movements are shared with all other web apps to keep the game state in sync.

**FR-5:** Clients will poll the /update endpoint regularly to get any new messages.

**FR-6:** Responses from /update will contain messages in JSON format with either a 'message' or 'tower' key.

**FR-7:** All connected browsers should show the same game and chat status.

**FR-8:** The system will support communication between peers using ZMQ first, and Zookeeper in later stages.

## 3. Non-Functional Requirements

These are quality-related goals and technical expectations for the system:

**R.1 Availability:** The system should keep running even if one or more peers go down (non-Byzantine faults).

**R.2 Scalability:** The system should handle more users and peers as needed, especially in the Zookeeper version.

**R.3 Latency:** The system must tolerate delayed messages and still behave normally.

**R.4 Consistency:** Every peer should see the same chat and game updates.

**R.5 Maintainability:** The codebase should be clean and modular so it’s easy to test or upgrade.

**R.6 Integrity:** If the system receives broken or unknown messages, it should skip them without crashing.

**R.7 Resilience**: If one peer fails or crashes, others should keep working.

**R.8 Extendibility:** The system design should make it easy to add Zookeeper support in future.

## 4. Scenario 1 – Organisation Deployment

This setup is for a group of around 100 people using the system inside a company. We'll use ZeroMQ for communication between webapps.

**4.1:** ZMQ will be used with PUSH/PULL sockets, so messages are shared between all webapps.

**4.2:** Each peer connects directly to other known ports.

**4.3:** Web browsers will fetch updates by polling /updating.

**4.4:** We'll add delay and reordering in messages to test if the system handles them correctly.

**4.5:** If a peer crashes, it won’t be restarted—this isn’t needed in this scenario.

## 5. Scenario 2 – Internet-Scale Deployment

This version is meant for massive use across the internet, with up to a billion users. It will use Zookeeper for coordination.

**5.1:** Zookeeper will store game and chat data using znodes.

**5.2:** Peers will use ephemeral znodes to register and detect who is online.

**5.3:** Watches will be used so peers get updates immediately without polling.

**5.4:** One peer will be elected as the leader to handle coordination.

**5.5:** If a peer disconnects, it can reconnect and sync state using the data stored in Zookeeper.

## 6. Conclusion

This document outlines the system’s required features and quality expectations. These will guide the development and testing of coPlay under both small and large-scale scenarios.