# Test Plan – coPlay

## 1. Objective

To define and execute functional and fault tolerance test cases for the coPlay implemented using ZMQ and later ported to Zookeeper. This document covers testing strategies that validate system correctness and resilience under non-Byzantine faults.

## 2. Test Environment

* Python 3.9+
* Flask Web Framework
* ZMQ for messaging
* Kazoo/Zookeeper
* Browsers simulating clients polling via `/update` endpoint

## 3. Scope of Testing

* Validation of chat message broadcasting across all connected web apps.
* Synchronization of tower movements via GET requests.
* Polling-based update retrieval via `/update` endpoint.
* Simulation of faults: lag, crash, and reordering.

## 4. Functional Test Cases

All the test cases here are intended towards normal functionality of program. Here TC denotes test cases

|  |  |  |
| --- | --- | --- |
| TC-1 | Send base64 chat message via POST /message | All peers display the same message |
| TC-2 | Click tower and trigger GET (for single tower) | Disk movement reflects across all peers |
| TC-3 | Reset button clicked | Game resets |
| TC-4 | Open polling /update in 3 browsers | All remain in consistent game/chat state |

## 5. Fault Tolerance Test Cases

These tests simulate latency, message loss, and failure. Here FT denotes fault tolerance test

|  |  |  |
| --- | --- | --- |
| FT-1 | Inject delay before socket receive | Peers still update correctly after delay |
| FT-2 | Reorder tower messages in ZMQ | Correct disk logic without state corruption |
| FT-3 | Kill peer during active game | Remaining peers stay functional |
| FT-4 | Send the same message twice | No duplicate display in chat |
| FT-5 | Restart disconnected peer | Peer resumes correct state from z nodes |

## 6. Manual Testing

* Start 3 webapps on separate ports (like 5000, 5004, 5008)
* Open each in a separate browser.
* Send a message and perform tower moves.
* Observe cross-client consistency in updates.
* Screenshot to showcase changes

## 7. Code-Based Testing Strategy

First, enable ‘TESTING=True’ in ‘Wk0\_A2\_coPlay.py’ for automatic console testing. Then use `tests()` to simulate message send and validate correct state using assert statements or JSON inspection. And delays and message reorder can manually be added inside `/update`route or `recv\_json` loop.

For manual testing ‘TESTING=False’ in ‘Wk0\_A2\_coPlay.py’.