Feedback From The Instructor

No details provided regarding the implementation. For final submission please do submit the design sketch and readme file mentioning how to run your code also, no testing (Unit testing, tests for edge cases, integration tests etc.) seems to be done at this stage so please implement that as well. - Please add better comments in your code to get better points for code quality - Also remove the commented codes from code files if not using.

Modifications Made for the Final Submission

1. Implementation Details

Instructor's Feedback: No details provided regarding the implementation.

Actions Taken:

- Added a comprehensive **overview** in the README, explaining each part of the system.
- **System Components**:
  - **ContentServer**: Reads weather data, converts it to JSON, and sends it to the AggregationServer.
  - **AggregationServer**: Processes GET/PUT requests, stores data, and uses a Lamport clock for synchronization.
  - **GETClient**: Sends GET requests and retrieves weather data.
- **Lamport Clock**: Ensures event synchronization across all components.

2. Design Sketch and README

Instructor's Feedback: For final submission please do submit the design sketch and readme file mentioning how to run your code.

Actions Taken:

- **Design Sketch**: A detailed diagram of how the ContentServer, AggregationServer, and GETClient interact, including the flow of data and the use of Lamport clocks.
- **README**:
  - Project Overview: Clear summary of the project's purpose.
  - Running Instructions: Step-by-step guide for running the servers.

    - Dependencies: List of required libraries.
    - Testing Instructions: How to run unit and integration tests.

3. Testing (Unit, Edge Case, and Integration Tests)

Instructor's Feedback: , no testing (Unit testing, tests for edge cases, integration tests etc.) seems to be done at this stage so please implement that as well.

Actions Taken:

- **Unit Tests**: Implemented to verify key functionalities of each component (ContentServer, AggregationServer, GETClient).
- **Edge Cases**: Tested cases such as empty files, malformed JSON, missing data, and empty PUT requests.
- **Integration Tests**: Simulated multiple clients interacting with the server, ensuring data consistency even with concurrent requests and large data transfers.
- **Test Automation**: Automated testing setup using JUnit to streamline the testing process.

4. Code Quality Improvements

Instructor's Feedback: Please add better comments in your code to get better points for code quality - Also remove the commented codes from code files if not using.

Actions Taken:

- **Improved Comments**: Added detailed explanations for each class and method, as well as inline comments to clarify complex logic.
- **Removed Unused Code**: Cleaned up unnecessary and commented-out code to enhance readability and maintainability.