**RTT Messenger Project**

**Phase 1: Project Setup**

1. **Define Project Scope**
   * Clearly outline project goals, functionalities, and technologies used.
   * Decide on initial architecture: RESTful API design.
   * Create a version control repository (Git).
2. **Basic Project Structure**
   * Create a project structure with directories for client, server, and common (for shared classes).

**Phase 2: Implement Client and Server**

1. **Basic Server Implementation**
   * Create a basic server that listens for incoming connections (UDP for now).
   * Implement a method to acknowledge received messages.
2. **Basic Client Implementation**
   * Create a basic client that sends messages to the server.
   * Implement methods to receive and handle acknowledgments from the server.
3. **Message Handling**
   * Define a message class that includes fields for message ID, content, and timestamps.
   * Implement serialization/deserialization for message objects.

**Phase 3: Implement Communication and RTT Measurement**

1. **RTT Measurement**
   * Implement functionality to measure round-trip time for messages.
   * Store timestamps for sending and receiving messages to calculate RTT.
2. **Message ID Tracking**
   * Implement logic to generate and track unique message IDs for each communication.
3. **Delta Time Measurement**
   * Create a method to track time differences (deltas) between sending and receiving messages.

**Phase 4: Introduce RDT 3.0 Mechanisms**

1. **Implement RDT 3.0 Logic**
   * Develop reliable data transfer logic including:
     + Acknowledgments (ACK) and negative acknowledgments (NAK).
     + Retransmission strategies for lost messages.
2. **Error Handling**
   * Implement error handling for lost or corrupted messages, utilizing RDT 3.0 concepts.

**Phase 5: RESTful API Integration**

1. **API Design**
   * Design RESTful endpoints for client-server communication (e.g., /send, /acknowledge).
2. **API Implementation**
   * Implement the server to handle REST API requests, mapping them to internal message handling.
3. **Client API Calls**
   * Update the client to interact with the server using the REST API.

**Phase 6: UDP over HTTP and TCP Preparation**

1. **UDP over HTTP**
   * Investigate and implement a mechanism to send UDP packets over HTTP.
   * Ensure it aligns with existing server and client communication.
2. **Prepare for TCP Implementation**
   * Structure the codebase to allow easy switching to TCP later.
   * Define a TCP communication module that can replace the UDP module.

**Phase 7: Testing and Optimization**

1. **Testing**
   * Implement unit tests for individual components (message handling, RTT measurement).
   * Perform integration testing for client-server interactions.
2. **Performance Optimization**
   * Optimize message handling and RTT calculations for efficiency.
   * Monitor and adjust OS-level synchronization if necessary.

**Phase 8: Documentation and Future Improvements**

1. **Documentation**
   * Document the project setup, API endpoints, and usage instructions.
   * Create a user manual for future developers.
2. **Future Enhancements**
   * Identify potential features (e.g., TCP integration, UI, improved error handling).
   * Plan for additional iterations based on user feedback or performance metrics.

**Additional Considerations**

* **Monitoring OS Performance**: Consider using tools like JMX (Java Management Extensions) for monitoring JVM performance and OS-level metrics.
* **User Interface**: Optionally, consider building a simple UI to visualize communication and RTT metrics.