**Q7: Real-Time Collaboration**

The system employs several strategies to handle real-time collaboration on form creation and editing. It ensures consistency in what users see and experience:

1. **CRDT-Based Collaboration Model**
   To handle real-time collaboration on form creation and editing while ensuring consistency across users, the system implements a Conflict-free Replicated Data Type (CRDT) approach rather than Operational Transformation (OT). This choice provides several key benefits:
   - **Conflict-free merging**:
     - CRDTs mathematically guarantee that concurrent edits made by different Clients can be merged without conflicts, regardless of the order they're received.
   - **Eventual consistency**:
     - All connected Clients eventually see the same document state after all operations are applied.
   - **Decentralized coordination**: No central authority is required to resolve conflicts.

2. **Client-Side Implementation**
   The client maintains a local in-memory copy of the document and implements a hybrid approach for sending summarised CRDT operations:
   - **Optimized Event Parsing Hybrid Approach:**
     - When the user pauses typing (200ms delay).
     - At regular intervals as a safety net (every 2 seconds).
     - Upon reaching a character threshold (20 characters).
   - **Local-first editing:**
     - Changes are applied to the local document immediately.
     - Users see their own changes without waiting for server confirmation.
     - Provides a responsive experience even with network latency.
   - **Idempotency checks:**
     - Each operation includes a unique ID and version vector.
     - Client checks incoming operations against its history to prevent duplicate application.

3. **Server-Side Processing**
   The Real-time Collaboration Service handles distribution of operations:
   - **WebSocket connections:**
     - Maintained with sticky sessions through the load balancer.
     - Enables immediate broadcast of operations to all connected clients.
   - **Consistency mechanisms**
     - Distributed locks in MongoDB to prevent race conditions during operation storage.
     - Version vectors track causal relationships between operations
     - Version vectors combined with unique operation IDs to ensure that CRDT operations are applied exactly once (idempotency).
     - Operations are applied in causal order on all clients and servers.
   - **Resilience features**
     - Server caches the current document state in memory.
     - Operations are persisted to MongoDB before broadcasting.

        - Kafka provides a reliable asynchronous channel for operation distribution.

4. **Document Versioning**
   The system maintains document history and consistency through:
   - **Periodic snapshots:**
     - Kafka Consumer Service periodically creates document snapshots.
     - Snapshots are cached in Redis and stored in MongoDB.

   - **Version retrieval:**
     - Clients can request specific document versions.
     - Enables form distribution of stable versions regardless of ongoing edits.

5. **Ensuring Consistent User Experience**
   To ensure all users have a consistent experience:
   - **Convergence guarantees:**
     - CRDT properties ensure all clients eventually reach the same state.
     - Version vectors maintain causal relationships between edits.

   - **Failure recovery:**
     - If a server fails, a new server can reconstruct the document state.
     - If a client disconnects, it can receive missed operations upon reconnection.
     - Document snapshots provide efficient recovery points.

   - **Sticky Sessions:**
     - System uses sticky sessions for WebSocket connections, which ensures that a client remains connected to the same instance of the Real-time Collaboration Service.
     - This can help to maintain the order of messages and reduce the complexity of managing real-time updates.