Q8: Security

To ensure the security of user data and prevent unauthorized access to forms and responses, I would implement a comprehensive security strategy covering:

1. **Authentication & Identity Management**
    - **Secure Authentication System**:
        - Password hashing using strong algorithms (e.g: bcrypt).
        - JWT tokens for stateless authentication with appropriate expiration.
        - Multi-factor authentication option for sensitive accounts.

    - **Session Management**:
        - Short-lived JWT tokens with refresh token mechanism.
        - Token invalidation on logout or suspicious activity.
        - Session timeout for inactive users.

2. **Authorization & Access Control**
    - **Fine-grained Permission Model:**
        - Role Based Access Control (READ, WRITE, ADMIN permissions) on documents, ensuring they can only access resources they are authorized to use.
        - Relationship Based Access Control (Client A is an editor of Document A) on documents, ensuring they can only access resources they are relationships with.
        - Document-level permissions stored in PostgreSQL.
        - Permission verification for all document operations.

    - **API Gateway Security:**
        - Request validation and sanitization.
        - Rate limiting to prevent brute force attacks.
        - Input validation to prevent injection attacks.

3. **Data Protection**
    - **Data Encryption:**
        - Encryption of data in transit using HTTPS.
        - Encryption of sensitive data at rest in PostgreSQL and MongoDB.
        - Encryption of JWT tokens using strong keys.

    - **Privacy Controls for Form Responses:**
        - Anonymous response option for respondents.
        - Data minimization principles in collecting respondent information.
        - Access controls for viewing response data.

    - **Data Isolation:**
        - Logical separation of tenant data in MongoDB and PostgreSQL.
        - PostgreSQL, which enforces ACID properties, is used for sensitive data like user accounts, permissions, and responses.
        - MongoDB is used for less sensitive data like documents and CRDT operations.
        - Strict database access controls from services.

4. **Infrastructure Security**
    - **Network Security:**
        - Private subnets for databases and services.
        - Network ACLs and security groups.

        - Web Application Firewall (WAF) for API Gateway.

    - **Websocket Security:**
        - Ensuring that Client interacts with the same Server throughout a session, reducing the risk of session hijacking.

    - **Container Security:**
        - Regular scanning of container images for vulnerabilities.
        - Principle of least privilege for container permissions.
        - Runtime security monitoring.

    - **Secrets Management:**
        - Secure storage of database credentials and API keys.
        - Rotation of credentials and certificates.
        - No hardcoded secrets in code or configuration.

5. **Monitoring & Incident Response**
    - **Security Monitoring:**
        - Logging of all authentication attempts and sensitive operations.
        - Real-time alerting for suspicious activities.
        - Regular security audits and penetration testing.

    - **Vulnerability Management:**
        - Regular dependency updates to patch vulnerabilities.
        - Security scanning in CI/CD pipeline.
        - Responsible disclosure program.

6. **Compliance Considerations**
    - **Data Governance:**
        - Clear data retention and deletion policies.
        - Compliance with relevant regulations (GDPR, CCPA).
        - Data access audit trails.

    - **Form Distribution Security:**
        - Secure distribution links with optional expiration.
        - Ability to revoke access to distributed forms.
        - Verification of respondent identity when required.