This document describes the security measures taken in the Lost-and-Found mobile application, “LoboLocate” to protect user data. The app handles personal data such as name, email, birthdate, and item details. Although it does not handle sensitive financial information, robust measures are taken in place to protect user privacy, prevent unauthorized access, and ensure data integrity.

1. **User Data Protection**
   1. **Data Storage and Access**
      1. **Data Storage:** User data, including name, email, birth date, and items details are securely stored in a Mongo database. Only required information such as userIDs, itemIDs is stored to an authorized administrator. MongoDB itself has many security features such as encryption for data in transit, and rest, authentication, role-based access control, etc.
      2. **Access Control:** The app restricts access to user data by implementing token-based authentication using JWT (JSON Web Tokens) during login and session handling. This ensures that only authenticated users can access or modify their data.
   2. **Password Security**
      1. **Hashing and Salting:** User passwords are hashed using bcrypt and salted before storage in the database, ensuring that plain text-passwords are never stored. In the worst-case scenario, an unauthorized person viewing these encrypted passwords would not be able to figure out the actual password.
      2. **Authentication Flow:** Passwords are verified upon login through secure token-based authentication (JWT). Session tokens are only valid for a limited time to minimize risk of unauthorized access.
   3. **Sensitive Data Exposure**
      1. **Data Encryption:** All sensitive data transmitted between the client and the server is encrypted using HTTPS (TLS/SSL).
      2. **Sensitive Fields:** Private user data, such as user IDs, names, and email addresses, are not exposed to unauthorized users. Only necessary fields are exposed during data retrieval and only for authenticated sessions.
      3. **User Identifiers:** Unique user identifiers (\_id) are used to access user-specific data. The identifiers are not exposed directly in the app interface to avoid accidental data leakage.
2. **Network Security**
   1. **Secure API Communication**
      1. **HTTPS Protocol:** All API requests from the client to the server are made over HTTPS, ensuring data integrity and encryption during transmission.
      2. **API Authentication:** Each API endpoint is protected by requiring a valid JWT token, ensuring that only authenticated requests are processed by the server.
      3. **Socket.IO security:** Real-time data transfer for chat functionality uses secure socket communication and is limited to authenticated users only.
   2. **Prevention of Unauthorized Access**
      1. **Token Verification:** All API requests require a JWT token which is verified for authentication. If the token is invalid or expired, the request is rejected, and the user is redirected to the login screen.
      2. **Rate Limiting and Throttling:** The server implements rate limiting on API endpoints to prevent potential abuse through brute-force attacks or data scraping.
3. **Data Privacy Compliance**
   1. **User Consent**
      1. **User Agreement:** User must accept the app’s terms and conditions upon signing up, ensuring informed consent for data collection and usage as outline in the app’s privacy policy.
   2. **User Data Rights**
      1. **Data Access and Deletion:** Users can request their data history and delete reports they have created. Deleting an item history or account data permanently removes it from the MongoDB database.
      2. **Data Retention Policy:** The app follows a data retention policy to delete unclaimed items after 90 days and permanently removes user data upon account deletion to comply with privacy best practices.
4. **Secure Coding Practices**
   1. **Input Validation and Sanitization**
      1. **Sanitization:** User inputs are sanitized to prevent SQL injection and Cross-Site Scripting (XSS) attacks.
      2. **Backend Validation:** The server validates all data received through API endpoints to ensure it adheres to expected formats, preventing malicious data injections from APIs.
   2. **Error Handling and Loggin**
      1. **Error Messages:** User-facing error messages do not disclose sensitive information. Detailed error logs are stored on the server for internal use and troubleshooting without exposing sensitive information.
      2. **Logging Sensitivity:** Logs avoid capturing sensitive data directly. Any sensitive data logged temporarily for debugging is handled securely and deleted after use.
5. **Third-Party Integrations**
   1. **Secure API Key Management**
      1. **Environmental Variables:** All sensitive API keys, including those for Google Vision API, are stored securely in environment variables, not hard coded in the app codebase.
      2. **Access Control:** API access is restricted to authorized users only.
   2. **Third-Party Compliance:** This app ensures that all third-party services used comply with data privacy regulations. For instance, image process through Google Vision API adheres to Google’s data privacy and security standards, ensuring user data is managed responsibly.
6. **Periodic Security Audits:** Period testing allows use to check for penetration testing and scan any vulnerabilities within the application if it was to be implemented at the University of New Mexico. Security audits are logged for auditing and tracking potential unauthorized activities.