**System Security Overview and Data Protection Measures**

Our system is designed with a multi-layered approach to security, ensuring both the safety and privacy of user data throughout the entire process. Below is a detailed explanation of how encryption, privacy, and data management work within our MERN stack application:

**1. Authentication and Token Management**

* **JWT (JSON Web Token):** We use JWT for user authentication. When a user logs in, a JWT is generated, which includes an encrypted payload containing the user's ID. This token is signed using a secret key, making it tamper-proof.
* **Token Hashing:** To add an extra layer of security, the generated JWT token is hashed using bcrypt before being stored in the database. This ensures that even if the database were compromised, the actual tokens cannot be easily used or reverse-engineered​.
* **Token Verification:** During subsequent requests, the token is verified against its hashed counterpart in the database. Only valid, non-expired tokens that match the stored hash are accepted, ensuring that only authenticated users can access the system​.

**2. Data Encryption**

* **Data at Rest:** All sensitive data stored in our MongoDB database, such as user information and transcription history, is encrypted using CryptoJS.AES with a strong secret key. This encryption ensures that even if the database is accessed by unauthorized users, the data remains unintelligible​.
* **Data in Transit:** We enforce HTTPS across all communications between the client and server, ensuring that data in transit is encrypted and secure from interception.

**3. Data Lifecycle Management**

* **Ephemeral Data Handling:** On the client side, data is encrypted using a hash key before being stored. This data is held only as long as necessary and is promptly removed from the client once it is no longer needed, minimizing the risk of data exposure​.
* **Data Retention Policies:** User-related data, specifically transcription history, is automatically purged from the system after 30 days. We employ a scheduled cron job that runs daily to identify and delete any records older than 30 days, ensuring that no outdated or unnecessary data is retained in our system​.

**4. Server Security**

* **Environment Configuration:** We use environment variables to manage sensitive configurations like database connection strings and JWT secrets. This approach isolates sensitive information from the codebase, reducing the risk of accidental exposure.
* **Database Security:** Our MongoDB instance is configured with secure access controls, ensuring that only authorized services can connect. Additionally, we utilize Mongoose to interface with MongoDB, which provides schema validation to enforce data integrity​.

**Summary**

Our system's architecture prioritizes the confidentiality, integrity, and availability of user data through rigorous encryption, secure authentication practices, and a proactive approach to data lifecycle management. These measures ensure that user privacy is maintained and that our application remains resilient against potential security threats.