.

**🔐 1. How does your system handle data? Does it consider data privacy issues?**

**Answer:** Yes, data privacy and security were a key priority in the development of this system. The backend is built using **MongoDB**, which is a NoSQL database that is resistant to **SQL injection attacks**, thus offering an added layer of data protection. Additionally:

* **Passwords are hashed** before being stored in the database using secure hashing algorithms, ensuring that even if data were to be breached, passwords would be unreadable.
* The system uses **protected routes** — unauthorized users cannot access the dashboard or sensitive user data unless they are logged in.
* **Authentication tokens** are used to validate user sessions securely.
* **Password strength validation** is enforced during user registration, encouraging the use of strong, secure passwords.

**🌐 2. How does the backend manage user data?**

**Answer:** User data is efficiently and securely managed through a modular backend architecture using **Express.js**. Here’s how it's structured:

* **Routes Folder:** Manages API endpoints. For example, /admin is a protected route only accessible by admin users.
* **Controllers Folder:** Handles business logic like creating, reading, updating, and deleting user data.
* **Middleware Folder:** Implements access control. For instance, only admins can access certain routes or functionalities, and users without proper authentication are blocked.
* User roles and privileges are checked through middleware before any sensitive actions can be performed.

**⚖️ 3. What ethical considerations have you implemented?**

**Answer:** Ethical usage is fundamental, especially for a mental health platform. Some key ethical considerations include:

* **Authentication-first access:** Users must be authenticated to view or update personal dashboards or journals.
* **Consent-based onboarding:** Users must agree to the platform’s **terms and conditions** before they can use the service.
* **No exposure of sensitive data:** Sensitive health data is not shared with third parties.
* **Role-based access control:** Only authorized personnel (like admins) can access platform-wide analytics or moderate user content.

**🏥 4. Will the system comply with healthcare guidelines and regulations?**

**Answer:** Yes, the system is designed with **healthcare best practices** in mind. While it is not a diagnostic tool, it supports emotional well-being and self-reflection through journaling. Key features that support compliance include:

* **Admin-level system management:** Admins oversee the platform's activities and ensure ethical use.
* The system does **not collect clinical data**, so it avoids direct violation of sensitive health regulations.
* In future iterations, we plan to align more closely with standards such as **HIPAA (US)** or **Kenya’s Data Protection Act** by implementing further encryption and anonymization techniques.

**🧠 Other Potential Questions and Model Answers**

**🛠️ 5. What tech stack did you use and why?**

**Answer:** The tech stack includes:

* **Frontend**: HTML, CSS, JavaScript (or React if applicable)
* **Backend**: Node.js with Express.js
* **Database**: MongoDB (using MongoDB Atlas for cloud-based access)

This stack was chosen for its scalability, flexibility, and rich ecosystem. MongoDB’s document-based structure is ideal for storing dynamic journal entries and user data. Express.js simplifies API creation and request handling.

**🔒 6. How do you handle user authentication and authorization?**

**Answer:** The system uses:

* **JWT (JSON Web Tokens)** for secure authentication and maintaining user sessions.
* **Role-based access control** ensures that only authorized users (e.g., admins) can access or modify sensitive parts of the system.
* Middleware checks the token and user role before granting access to protected routes.

**📈 7. How does the system improve user mental well-being?**

**Answer:** The system supports emotional well-being by:

* Allowing users to **journal their thoughts** and feelings in a private, secure environment.
* Providing **visual insights** like mood trends, emotional graphs, and reflection prompts.
* Offering a **safe space** for self-expression, which encourages mindfulness and emotional resilience.

**🌍 8. How is this system localized for Kenyan users?**

**Answer:** The system takes into account the Kenyan context by:

* Using **simple, user-friendly interfaces** for all literacy levels.
* Focusing on **offline/low-data design principles** for accessibility.
* Including **culturally relevant prompts** in the journaling tool to encourage emotional self-awareness.

**🧪 9. How did you test the system for performance and security?**

**Answer:** Testing involved:

* **Functional testing** of all user flows (registration, journaling, login/logout, etc.).
* **Security testing**, especially on password strength, route protection, and injection resistance.
* **Role-based tests** to ensure non-admin users can't access restricted areas.
* **Stress testing** was done with multiple dummy users to ensure performance under load.

**📦 10. What are future plans for this system?**

**Answer:** Planned future enhancements include:

* Integration with **licensed counselors or chatbots** for users needing extra support.
* Advanced **AI-driven sentiment analysis** on journal entries.
* **Mobile app version** for easier access.
* Implementation of **data encryption at rest**, and **multi-factor authentication**.

**💻 11. Where is user data stored and how is it accessed?**

**Answer:** User data is stored securely in **MongoDB Atlas**, a cloud-based version of MongoDB. The backend server (Express.js) connects to MongoDB using the **MongoDB URI**. Data can be visualized in tools like:

* **MongoDB Compass** (desktop)
* **MongoDB Atlas Dashboard** (cloud)

Access to these platforms is secured and restricted to developers/admins only.

**🛑 12. What happens when a user tries to access restricted content?**

**Answer:** If an unauthenticated or unauthorized user tries to access protected routes (e.g., /admin), the **middleware** intercepts the request and:

* Returns a **403 Forbidden** or **401 Unauthorized** response.
* Redirects to the login or error page, depending on the frontend logic.