Scalability & Availability Considerations

Handling Increased Users or Data:

The system leverages Firebase Authentication and Realtime Database, both of which are designed to scale automatically as user demand increases. Firebase efficiently manages authentication requests, while the NoSQL structure of Realtime Database allows for fast, scalable data retrieval without complex queries. Additionally, only registered users can add data, ensuring that database writes are controlled and authenticated.

Load Balancing and Caching:

Firebase services inherently distribute traffic across multiple servers, reducing bottlenecks and improving response times. API requests made to Firebase Authentication and Realtime Database are handled by Google's infrastructure, which dynamically manages scaling without requiring manual load balancing.

Deployment Strategy and Uptime Considerations:

The system employs GitHub Actions for automated CI/CD, ensuring smooth and consistent deployments. Docker is used to containerize the application, providing a reliable and scalable runtime environment. Nginx serves as a reverse proxy, improving performance and handling incoming traffic efficiently. The backend leverages AWS Lambda for serverless execution, enabling automatic scaling based on demand. Since Firebase is used for authentication and database management, it integrates seamlessly with the deployed services. Rolling updates and blue-green deployments ensure minimal downtime, while monitoring tools help track system performance and reliability.