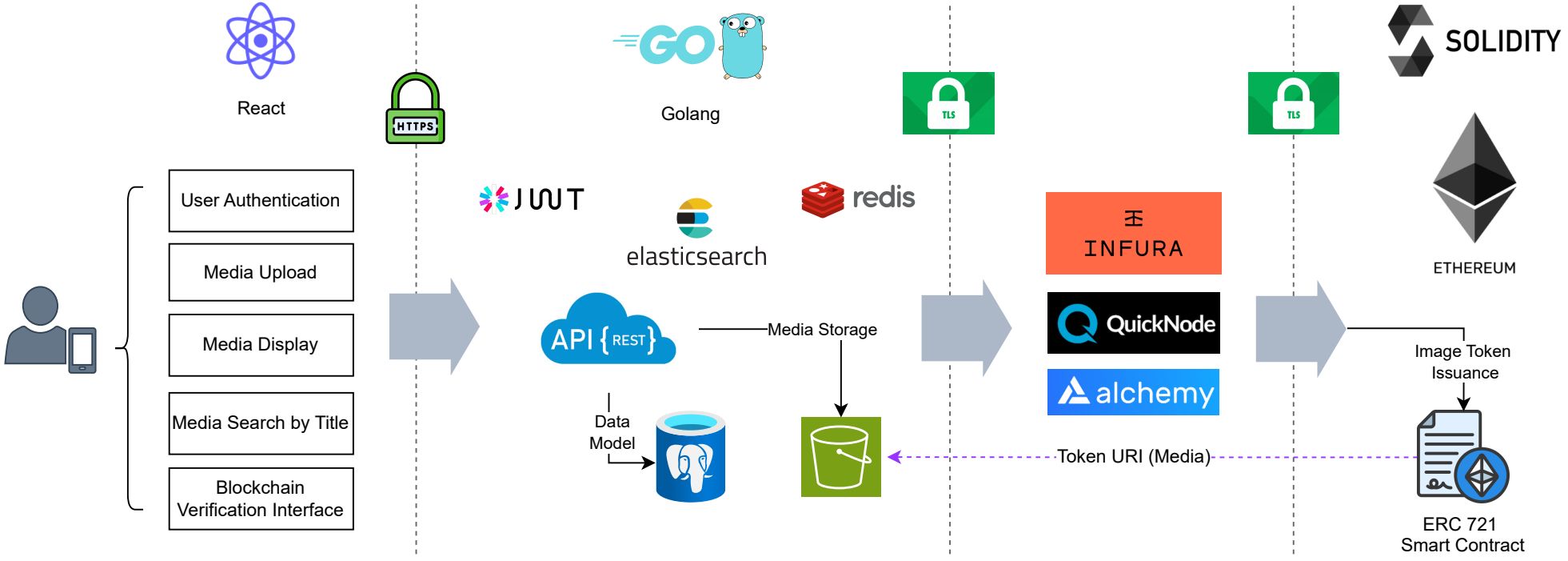

Front End

- Adopt responsive design for accessibility across devices.
- Secure user authentication flow using JWT tokens.
- Upload interface with preview and metadata (title, description) input.
- Interface to initiate blockchain verification and display results.

Back End

- **User Management**
  - RESTful API endpoints for user registration, login, and profile management.
  - Use JWT for secure authentication and session management.
- **Media Management**
  - API endpoints for uploading, fetching, and deleting media files.
  - Integrate with cloud storage SDK for direct media file management.
  - Store media metadata in the PostgreSQL database.
- **Search Engine**
  - Integrate Elasticsearch with the application for efficient title-based searching.
  - Index media titles and associated metadata for quick retrieval.
- **Blockchain Interface**
  - Deploy a smart contract on Ethereum for proof of existence.
  - Utilize Web3 APIs for interacting with the Ethereum blockchain.
  - Functions for submitting media hash to the blockchain and verifying existence.

Storage

- Amazon S3 buckets for media storage with proper access controls.
- PostgreSQL database schema designed for fast reads and writes, with tables for users, media files, and blockchain proofs.

Scalability & Performance

- Use a microservices architecture for the backend to scale components independently.
- Implement caching strategies for frequently accessed data.
- Optimize media file storage and delivery using CDN.
- Monitor and adjust Ethereum gas usage for cost-effective blockchain interactions.

Security

- Implement SSL/TLS for secure data transmission.
- Hash user passwords before storing them in the database.
- Use access control mechanisms in both the application and storage layers.
- Validate and sanitize all user inputs to prevent SQL injection and XSS attacks.
- Use OpenZeppelin Smart Contracts secure design