URL Shortener System Design

Which Tech - Stack Choices

Cloud Provider: Google Cloud Platform (GCP)

- Seamless App Engine deployment
- Free-tier friendly
- Easy integration with managed services

Database: MongoDB Atlas (Cloud-hosted NoSQL DB)

- Schema flexibility, scalable, geo-friendly

Backend Framework: Node.js with Express

- Lightweight, fast, perfect for APIs

Frontend Framework: React.js

- Component-based, async-friendly, modern

What Flow - System Flow (Step-by-Step)

1. User enters a long URL in the React app.
2. React sends POST to /api/shorten.
3. Backend checks existing, else generates short code.
4. Saves in MongoDB, returns short URL.
5. When clicked, backend redirects and logs analytics.
6. Dashboard hits /api/analytics/:shortCode for metrics.

Which APIs - Design of API Endpoints

1. POST /api/shorten - returns a short URL.
2. GET /api/redirect/:shortCode - redirects to original URL and logs data.
3. GET /api/analytics/:shortCode - returns analytics like click counts, locations, timeline.

How You Will Optimize the Flow

URL Shortener System Design

Performance

- MongoDB indexes, async logging, App Engine Standard

Scalability

- Stateless backend, horizontally scalable MongoDB

Error Handling

- Proper status codes, try-catch blocks, frontend 404s

Demo Code - Critical Snippets

1. Shorten URL Logic

- Checks DB, uses nanoid, saves and returns.

2. Redirection with Analytics

- Looks up URL, logs IP/country, redirects.

3. Analytics Retrieval

- Aggregates by country and last 30 days.

Optional Enhancements

- Custom aliases, QR codes, authentication, expiry, rate limiting.