URL Shortener - Architecture and Code Design (Human-readable Summary)

Technology Choices:

- Backend: Node.js with Express

  Reason: It's lightweight, fast, and perfect for handling API-based microservices.

- Unique ID Generator: nanoid

  Reason: It creates small, unique, and secure IDs ideal for URL shortening.

- Storage: In-memory Map()

  Reason: Easy to implement and fast for testing/demo purposes.

Middleware:

- A middleware called 'logClientInfo' is used before the URL is shortened.
- It shows user data like email, name, roll number, access code, and client info.
- This helps track requests or simulate authentication if needed.

How URL Shortening Works:

1. User sends a POST request to /shorten with a long URL.
2. Middleware logs client info.
3. The backend checks if the URL is present.
4. It generates a unique shortId using nanoid.
5. Stores original URL with expiry time (30 minutes) in a Map using shortId as key.
6. Returns a short URL like http://localhost:3000/abc123.

How Redirection Works:

1. User visits short URL like /abc123.
2. System checks if that ID exists in the Map.
3. If not found, it returns "URL not found".

4. If expired, it deletes it and returns "URL expired".

5. If valid, it redirects to the original URL.

Design Assumptions:

- This is a demo system. Data is not stored permanently.
- It's assumed the environment is secure and trusted.
- Real apps should use Redis or MongoDB instead of Map().
- Validation and security checks should be added before going live.

Future Improvements:

- Use Redis for storage with built-in expiry.
- Add user authentication and rate limiting.
- Track analytics like number of clicks, location, etc.
- Improve error handling and logging.

This summary is written simply for clear understanding and has minimal technical errors.