URL Shortener Microservice – Design Document

1. Overview The URL Shortener Microservice provides:

2. Shortening of long URLs with optional custom shortcodes

3. Redirection to the original URL via shortcodes
4. Tracking of click analytics (timestamp, referrer, and coarse location)

5. Retrieval of URL statistics including click history

6. Architecture Type: Single Microservice (Backend-only, REST API) Components:

7. Express.js Server: Handles HTTP requests and routes.

8. In-Memory Database (Map): Temporary storage for URL metadata and clicks.

9. Logging Middleware: Custom logging used for error and info tracking, avoiding console.log.

10. Data Modeling Short URL Object: { "originalUrl": "string", "createdAt": "ISODate", "expiry": "ISODate", "clicks": [{ "timestamp": "ISODate", "referrer": "string", "location": "string" }] } Keys: shortcode -> unique identifier, serves as key in Map/DB. Clicks: array storing each click event for analytics.

11. Technology Selections

12. Backend Framework: Express.js

13. Language: Node.js (ES Modules)
14. Logging: Custom middleware
15. Data Storage: In-memory Map
16. UUID / Shortcode Generation: Random alphanumeric strings

17. Time Handling: JavaScript Date

18. Key Design Decisions

19. Accept optional user-provided shortcode or generate randomly.

20. Routes: POST /shorturls, GET /\:shortcode, GET /shorturls/\:shortcode.
21. Click Analytics: timestamp, referrer, location.
22. Expiry Handling: default 30 minutes, 410 Gone if expired.

23. Logging: Structured JSON for all key events.

24. Assumptions

25. Users are pre-authorized; no authentication.

26. Input validation ensures valid URLs.
27. Click location is a placeholder.
28. Single instance; horizontal scaling requires external storage.

29. In-memory DB is temporary.

30. Scalability & Maintainability

31. Stateless API design.

32. Map-based DB can be replaced with Redis/MongoDB.
33. Logging and structured error handling.

34. Modular code design.

35. +--------------------+

| Client |

| (Browser / API) |

+---------+----------+

|

| HTTP Requests (POST / GET)

v

+--------------------+

| Express.js API |

| (Server Layer) |

+---------+----------+

|

+------------+------------+

| |

v v

```
+--------------------+ +--------------------+

| POST /shorturls | | GET /shorturls/\:id |

| - Validate input | | - Retrieve stats |

| - Generate shortcode| | - Return JSON |

| - Set expiry | +--------------------+

| - Store in DB |

+--------------------+

|

v

+--------------------+

| In-memory DB |

| (Map / key-value) |

| shortcode -> { |

| originalUrl, |

| createdAt, |

| expiry, |

| clicks[] |

| } |

+--------------------+

|

v

+--------------------+

| Logging Middleware |
| - Structured JSON |

| - Logs errors, |

| creation, clicks |

+--------------------+
```

This design ensures a robust, maintainable, and scalable microservice for URL shortening with analytics. It balances simplicity for evaluation with extensibility for production deployment.