System Design Document

1. Overview

A minimal URL Shortener system consisting of:

- Frontend: Vite + React app (port `3000`)

- Backend: Node.js + Express API (port `4000`)

- Storage: In-memory (no database)

- Logging: Lightweight request logging on the backend and optional API call logging on the frontend

Primary capabilities:

- Create short URLs with optional custom shortcode and validity period

- Redirect via short URLs while capturing click statistics

- Retrieve stats for a short URL

- Frontend UI for creating links and viewing a history list (from backend)

- Simple real-time UX for click counts (optimistic increment + short refresh)

2. Requirements

Functional

- Create short URL with:

  - `url` (required)

  - `validity` in minutes (optional; default 30; min 1)

  - `shortcode` (optional; autogenerated if missing)

- Redirect via short URL (`/r/:shortcode`) and record click details

- Get stats for a short URL (clicks, createdAt, expiry, click details)

- Frontend to submit new URLs and list existing ones

Non-Functional

- No database (all data in memory)

- Dev-friendly: simple to run locally on Windows

- Basic input validation (protocol and slug/shortcode format)

- Lightweight observability (console logs)

- CORS from frontend to backend

- Portable folder structure

3. Architecture

- Client: React SPA served by Vite dev server on `http://localhost:3000`

- API: Express server on `http://localhost:4000`

  - Routes organized as `routes` + `controllers`

  - In-memory store shared as a module-scoped Map

  - CORS enabled for frontend origin

  - Trust proxy for IP extraction

High-level diagram (textual):

- User → Frontend (React) → Backend API (Express) → In-memory store

- Redirect flow: Browser → Backend `/r/:slug` → increments stats → 302 → Original URL

4. Key Components

Frontend (`Frontend Test Submission/`)

- `src/api.js`: API client using `fetch` with optional logging
  - Env: `VITE_API_BASE` (defaults to `http://localhost:4000`)
  - Env: `VITE_API_LOG` or dev mode to enable API logs
- `src/components/UrlShortener.jsx`: Create form with optional custom slug
- `src/components/HistoryList.jsx`: List links with open/delete actions and click counts
- `src/App.jsx`: Loads links, wires create/delete, optimistic click increment, error handling
- `vite.config.js`: Dev server on `3000`

Backend (`Backend Test Submission/`)

- `server.js`: Entrypoint that calls `createApp()` and listens
- `src/app.js`: Express app factory, CORS, JSON body parser, trust proxy, request logger, routes
- `src/routes/links.routes.js`: Route table
- `src/controllers/links.controller.js`: Business logic
- `src/store/linksStore.js`: In-memory Map with helpers
- `src/middleware/logger.js`: Request logging middleware

5. Data Model

In-memory entity: Link

- `slug` or `shortcode`: string (letters/numbers/`-`/`_`)
- `url`: string (must start with `http` or `https`)
- `createdAt`: ISO string
- `expiresAt`: ISO string or null

- `clicks`: number
- `clickDetails`: array of:
  - `timestamp` (ISO string)
  - `referer` (string | null)
  - `userAgent` (string | null)
  - `ip` (string | null)
  - `geo` (null for now)

6. API Specification

Base URL: `http://localhost:4000`

Primary endpoints:

- POST `/shorturls`
  - Body:
    - `url`: string (required, http/https)
    - `validity`: number (minutes, optional, default 30; min 1)
    - `shortcode`: string (optional; auto-generated if omitted)
  - Responses:
    - 201:
      - `{ "shortLink": "http://localhost:4000/r/<shortcode>", "expiry": "<ISO8601>" }`
    - 400:
      - `{ "error": "Invalid URL... / Shortcode may..." }`
    - 409:
      - `{ "error": "Shortcode already exists" }`

- GET `/shorturls/:shortcode`

  - Response 200:

    - `{ "shortcode": "...", "url": "...", "createdAt": "...", "expirationDate": "...", "clicks": n, "clickDetails": [ ... ] }`

  - Response 404:

    - `{ "error": "Not found" }`

Redirect endpoint:

- GET `/r/:slug`

  - Behavior:

    - If expired → 410 "Link expired"

    - Otherwise increments `clicks`, appends click detail, redirects (302) to `url`

Additional frontend-facing endpoints (existing):

- POST `/api/shorten` (simple create)

- GET `/api/links` (list all)

- DELETE `/api/links/:slug` (delete)

7. Frontend Behavior

- Create flow:

  - React form → `api.shorten({ url, slug? })` (legacy) for the app's list UI

  - Compatible with `/shorturls` if extended for stats UI

- History:

  - Renders list from backend `GET /api/links`

  - Open uses backend short URL (increments clicks server-side)
  - Optimistic increment in UI then re-fetch list after ~600ms
- Config:
  - `VITE_API_BASE`: points to backend
  - `VITE_API_LOG`: enable console logging for API calls

8. Validation and Error Handling

- URL validation:
  - Must parse as `new URL(value)` and have protocol `http:` or `https:`
- Slug/shortcode:
  - Regex `^[A-Za-z0-9-_]+$`
- Backend errors:
  - 400 for invalid inputs
  - 409 if custom shortcode already exists
  - 404 for unknown shortcode
  - 410 for expired links (redirect endpoint)

9. Security Considerations

- Open redirect risk: Allowed but constrained by protocol whitelist (http/https). For production, consider:
  - Domain allow-lists or risk acceptance
- CORS: Only frontend origin is allowed by default
- XSS: No user-rendered HTML from URL field; ensure safe rendering on frontend
- IP capture:

  - `app.set('trust proxy', true)` used for `x-forwarded-for`

  - For public deployment, set trusted proxy configuration correctly

- Rate limiting: Not implemented (add for production)

- Secrets: None used

10. Observability

- Backend:

  - Request logger writes method, URL, status, duration, IP, truncated UA (`src/middleware/logger.js`)

- Frontend:

  - API logger logs method, URL, status, timing, response/error (`src/api.js`)

  - Toggle via `VITE_API_LOG`

11. Deployment and Operations

- Local dev ports:

  - Frontend: `3000`

  - Backend: `4000`

- Port conflicts:

  - Find PID: `netstat -ano | findstr :4000`

  - Kill: `taskkill /PID <PID> /F`

- Env:

  - Backend: `PORT`, `FRONTEND_ORIGIN`

  - Frontend: `VITE_API_BASE`, `VITE_API_LOG`

- Persistence:

- In-memory only; restarts wipe data

12. Runbooks (Local)

Backend:

```powershell
cd "Backend Test Submission"
npm install
npm start
# Server -> http://localhost:4000
```

Frontend:

```powershell
cd "Frontend Test Submission"
npm install
npm run dev
# App -> http://localhost:3000
```

Create (PowerShell):

```powershell
$body = @{ url = "https://example.com"; validity = 45; shortcode = "demo1" } | ConvertTo-Json
Invoke-RestMethod -Method Post -Uri http://localhost:4000/shorturls -ContentType 'application/json' -Body $body
```

Stats:

```powershell
Invoke-RestMethod -Uri http://localhost:4000/shorturls/demo1
```

13. Trade-offs and Future Work

- In-memory store:
  - Pros: Simple, fast, no ops
  - Cons: Not persistent; single-instance only
  - Next: Add Redis/Postgres and repository layer
- Click tracking:
  - Currently captures timestamp, ip, referrer, UA; no geo
  - Next: Integrate IP geolocation; batch/limit click details retained
- URL generation:
  - Current alphabet and length ok for small scale
  - Next: Collision handling and retry for autogenerated slugs
- Frontend UX:
  - Optional “Details” panel for `/shorturls/:shortcode` stats
  - Dedicated routing (`react-router-dom`) for detail views

14. Folder Structure

- Frontend: `Frontend Test Submission/`
  - `src/components/` (`UrlShortener.jsx`, `HistoryList.jsx`)

- `src/api.js`
- `vite.config.js`, `.env.development`
- Backend: `Backend Test Submission/`
- `server.js`
- `src/app.js`
- `src/routes/links.routes.js`
- `src/controllers/links.controller.js`
- `src/store/linksStore.js`
- `src/middleware/logger.js`

15. Acceptance Criteria

- POST `/shorturls` creates a short link and returns `shortLink`, `expiry` (201)
- GET `/shorturls/:shortcode` returns stats including `clicks` and `clickDetails`
- GET `/r/:shortcode` increments `clicks` and redirects (410 if expired)
- Frontend can create, list, delete and open links; click count updates in UI quickly
- Logging appears in console for both backend requests and frontend API calls when enabled