URL Shortener App Design

1. Core Architectural Decisions

Client-Side Execution: I implemented the application to operate entirely within the browser environment. This design choice eliminates server-side dependencies for core functionality.

Feature Focus: My design prioritizes the URL shortening and clipboard copy operations.

2. Data Management

In-Memory Storage: I utilize a JavaScript object (urlMemory in script.js) for storing key-value pairs of short codes to long URLs. This data is volatile and persists only for the duration of the browser session.

Persistent Storage (Conceptual): A production system would necessitate a backend database for durable storage of URL mappings.

3. Technology Stack

React: I selected React for its component-based paradigm, facilitating modular UI development.

Tailwind CSS: I chose Tailwind for its utility-first CSS framework, enabling rapid styling.

4. URL Handling

Simulated Redirection: The application's short URLs are illustrative. Clicking a generated short URL directly navigates to the original long URL via client-side href assignment.

Server-Side Redirection (Conceptual): A complete solution would involve a server-side endpoint to process short codes and issue HTTP redirects.

5. Assumptions

Non-Persistence: I assumed the data persistence requirement was limited to the current browser session.

Security Scope: I did not implement server-side security measures, as the application operates purely on the frontend.