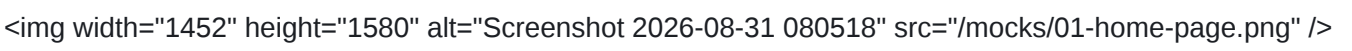
Build The Urlist

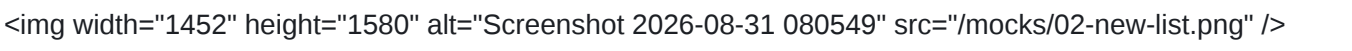
Build the complete application in this repository. Work autonomously from start to finish and stop only when the app is complete.

Mocks

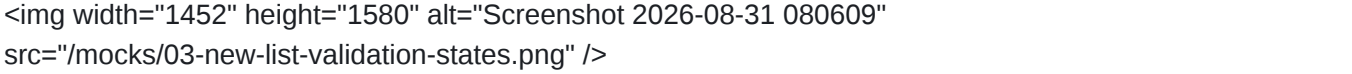
Home Page



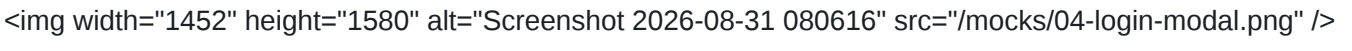
New List



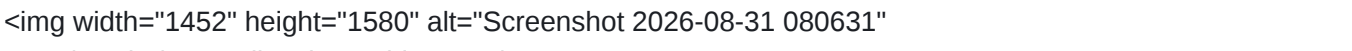
New List: Validation States



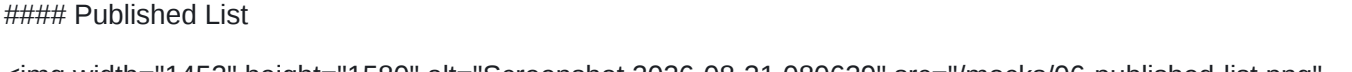
Login Modal



My Lists (Logged In)



Published List



Published List: QR Code

Technical specification and checklist

Before coding, create and commit `TECHNICAL_SPEC.md`. Keep it concise; do not restate this document.

It must contain:

- Architecture, data model, routes/API, security boundaries, and key assumptions
- An atomic Markdown checkbox for every requirement in this document, grouped by feature
- A verification method on every checkbox: unit test, browser test, command, or direct inspection

Use the format `- [ ] Requirement — Verify: method`. Maintain it throughout implementation. Check an item only after its implementation exists and its verification passes; reopen it if later work breaks it. Before finishing, resolve every unchecked item or leave it unchecked and report the exact blocker.

Stack and design

Use:

- Next.js App Router, React, strict TypeScript, Node.js, and npm
- SQLite with direct parameterized SQL through `better-sqlite3`; no ORM
- Vitest and Playwright
- Current stable package versions and a committed `package-lock.json`

Use the Node.js runtime, not Edge, for SQLite, sessions, file access, and metadata fetching.

Design system: use Bulma CSS for every screen (layout, navbar, cards, buttons, modals, forms, tags, dropdowns) and Font Awesome for all icons. Do not add a custom design system or a second CSS framework.

Product

The Urlist lets people create ordered lists of web links and publish each list at a public alias.

Publishing requires a signed-in user; there is no anonymous publishing. The Publish button reads **"Login to Publish"** and is disabled while signed out. List ownership belongs to the signed-in user who published it. The owner can edit and delete their list at any time while signed in. Deletion is **permanent**: the list's alias becomes immediately available to anyone. There is no soft delete, restore, tombstone, or anonymous list in this product.

Anyone can view an active list.

The server is authoritative: revalidate input, ownership, and alias availability on every write; ignore client-supplied owner IDs.

Routes

Route	Behavior
`/`	Home page
`/s/new`	Editor for the one local draft (new mode)
`/s/edit`	Editor for an owned, published list (edit mode) — **no alias in the URL**; the list being edited is the one currently loaded in the editor state
`/s/mylists`	Current user's lists (login required)
`/s/terms`	Static Terms of Service page (reasonable content is fine)
`/{vanity}`	Public list — **a single path segment only**

404 page (everything else, including multi-segment paths): show the text **"Sorry, there's nothing at this address."**

Static routes take priority over `/{vanity}`. Reserved first segments: `s`, `api`, `__test`.

Navigation bar

A navbar. Left: brand/logo (alt "urlist logo"); on mobile, a menu toggle that expands the menu. Menu items in this order, each an icon plus label:

1. `New` — a "create/add" icon
2. `My Lists` — a "user/account" icon — **only when signed in**
3. `About` — external link `https://aka.ms/theurlist`, a "help/question" icon
4. `Terms` — link to `/s/terms`, an "info" icon

Right: the theme dropdown (an icon-only button showing the active theme's icon), then either the Login item (a "sign in" icon, label "Login", opens the login modal) or the signed-in user's avatar and name followed by a dropdown chevron; that dropdown shows "Signed in with {provider}" and a "Log Out" item (a "sign out" icon).

Clicking `New` while on `/s/new` with a non-empty draft must pop a confirm modal: title **"Clear this list?"**, prompt **"This will reset the current list and you will lose all changes. Are you sure you want to do that?"** with OK/Cancel. On OK, reset the draft and go to `/s/new`. Otherwise, clicking `New` resets the draft and goes to `/s/new`.

Document title: "The Urlist - Share the internet".

Home page

Two stacked sections. The top one sits on the page's base background; the bottom one is a distinct, slightly darker full-bleed band.

Top section — two columns on desktop (the illustration is hidden on mobile), both columns **top-aligned** (the copy starts at the top of the section; it is not vertically centered against the illustration):

• Left column, in order, generously spaced:
1. H1 (large, **medium weight — not bold**): **Group links, Save & Share them with the world** — the single words "Group", "Save", and "Share" rendered in the primary teal color at the same weight as the rest of the sentence.
2. Paragraph (regular body text): **Add links to a list and share it with one simple URL.**
3. Paragraph (regular body text): **Create a list anonymously or login to save, manage, and edit your lists.**

• Right column: an original hero illustration (lazy loaded, roughly 500×500; in place of the reference app's `banner-logo-large.svg`). Its subject is **three cascading browser-window frames**, each offset down and to the right of the one behind it, alternating light / dark / light window chrome, each with a tab strip and an address bar reading `theurlist.com` and an otherwise empty page — it evokes a shared list opening in any browser. Draw it as an inline SVG using Bulma's color palette (background, border, text, and primary colors) with Bulma's default radii, keep it quiet, and do not copy the source SVG.

Bottom section ("Get Started") — a full-bleed band on a distinct background (the reference renders it as a very wide sheet with a gently rounded top edge rising from the bottom of the page; a full-width band with a subtle rounded or straight top edge is fine). It runs to the bottom of the page, filling the remaining viewport height when the page is shorter than the window. Inside, on the normal content width:

1. A centered, bold, primary-colored H2 **Get Started**
2. The first-link input widget, **left-aligned and spanning the full content width** (not centered or narrowed): the helper text directly above the input, then the input.

First-link input widget:

• Helper text above the input (left-aligned, regular-weight body text — not a small bold form label): **"Enter a link and press enter"**
• Extra-large text input spanning the full width of its container, with large placeholder/value text (visibly the biggest input on the page after the editor's vanity field), placeholder `http://example.com`
• Submit with Enter key. Validation: non-empty after trim; must parse as an absolute `http` or `https` URL (a bare domain-like value may be prefixed with `http://`); the host (after stripping a leading `www.`) must be a DNS-like host containing at least one dot. Invalid: apply the invalid input state and show **"That doesn't look like a valid URL"**
• On valid submit: add the link to the local draft, clear the input, refocus it, start metadata enrichment for that link, navigate to `/s/new`.

Draft and editor

Keep one draft per browser profile in browser-local storage. It holds the list's alias, description, and links with their order. It must survive reload, navigation, login, and logout. Clear it only after successful publication or a confirmed reset via the `New` guard above.

Both editor pages (`/s/new` and `/s/edit`) show, top to bottom:

1. the **publish bar** — a full-width band directly under the navbar;
2. the first-link input widget (the same "Enter a link and press enter" control as the home page) for adding more links — on **both** pages, new and edit mode;
3. when the list has links, the **"Links"** heading with its right-aligned hint, then the link rows;
4. on `/s/edit` only, the **"Delete This List"** button below the rows (see Delete).

Publish bar

Not a card. A full-width band directly under the navbar, on a background distinct from the page body below it (the reference separates it with a soft shadow). Three columns on desktop, stacked on mobile:

• **Vanity Url** (label above): the largest field in the app — a tall input whose value renders in large, medium-weight type. Tooltip *"Optional: Enter a vanity url for this list (i.e. my-list becomes theurlist.com/my-list). If you leave this box blank, we'll generate a random vanity for you."* While the user types: if it contains anything other than letters, numbers, or dashes, show **"Vanity URLs can only contain letters, numbers, and dashes."** Otherwise, after ≥300ms of the value being stable, check availability against the server; if it is in use, show **"This vanity URL is already in use. Please choose another."** On `/s/edit` the input is **disabled** (the alias never changes after publication) and visibly rendered in a disabled, muted/filled state.
• **Description** (label above): a normal bordered textarea, 2 rows. Tooltip *"Optional: The description will show up as the title on your public list page."*
• **Publish** button (large, primary, bold, rounded; top-aligned with the two fields): while signed out it is disabled with the label **"Login to Publish"** (clicking it opens the login modal). While signed in it is enabled only when the alias is valid (or blank) and the list has at least one link. On success the server response becomes the editor state and the browser navigates to `/{vanity}`. On failure keep all user input and show an error.

Link list editor

When the list has links, show a primary-colored "Links" heading with a right-aligned hint **"Drag links to

re-order"**, then one row per link in stored order.

Row anatomy — rows are cards of text, not forms:

- A drag grip icon at the far left, **outside** the card, vertically centered on it. It is the only drag handle.
- A delete icon (an "×") at the far right, **outside** the card, vertically centered on it. Clicking it removes the row immediately with **no confirmation**.
- Between them, a card (surface with a soft shadow) of uniform height (about 120px in the reference) containing, left to right:
- the 64px link image, vertically centered, or a Font Awesome "link" icon placeholder (centered in a 64px Bulma `image` box) when there is no image. On mobile the 64px image is replaced by a 24px thumbnail sitting inline to the left of the title;
- a text column: the **Title** (bold, one line, truncated with an ellipsis), the **Description** (regular, smaller, muted, clamped to about two lines with an ellipsis), and at the bottom the destination **URL** (small text in the link color, one line, truncated with an ellipsis).
- While the row's metadata fetch is in flight, a thin progress bar spans the top edge of the card; when the fetch settles it disappears.

In-place editing (this is the key interaction): the title, description, and URL are rendered as plain text — there are no visible input boxes, borders, field backgrounds, labels, or save buttons inside the row. All three are nevertheless editable in place: clicking, tapping, or tabbing into the text places a caret and lets the author type directly; the only visual affordance is a subtle hover/focus cue (for example a thin accent edge or focus ring around the piece of text being edited). The placeholders **"Enter a title"** and **"Enter a description"** appear only while that value is empty. Edits update the draft/editor state immediately as they are typed; there is no per-row save — the list is saved when Publish is pressed. Implement the three fields as real form controls (borderless single-line inputs for title and URL, a borderless textarea for the description, or an equivalent) with visually hidden labels so they remain keyboard- and screen-reader-operable. Fetched metadata fills only fields the author has not typed into; it never replaces typed text.

There are no per-link status badges or error messages: failed metadata simply leaves the text as it is (or empty) and clears the progress bar.

Reordering is **drag-and-drop only**, via the grip handle, with smooth animation (the whole card moves). No other reorder affordances.

Allow duplicate destination URLs as separate entries. No product-level link limit. Each link keeps a stable server ID, destination URL, optional title, optional description, optional image, and position. Persist order.

Live metadata

Fetch and parse the live destination page **server-side** after a link is added. Cap the fetch at 20 seconds. Failure keeps the link with whatever metadata was obtained (or none) and clears the row's progress bar; publication is never blocked by it. Preserve any manual title/description edits the author has made.

Extract metadata with this precedence:

- Title: `<title>` tag → `og:title` → `twitter:title` → first `<h1>` → `og:site_name`
- Description: `og:description` → `twitter:description` → `meta[name=description]`

• Image: `og:image` → `twitter:image` → `apple-touch-icon` → `mask-icon` → shortcut icon → `itemprop=image` → fetch the destination's `/favicon.ico`; resolve relative image URLs against the final post-redirect URL

The fetcher must be SSRF-safe: HTTP/HTTPS only, no credentials embedded, reject non-public and internal addresses, revalidate DNS on every redirect to prevent rebinding, follow at most five redirects, limit content to 2 MiB, accept only HTML/XHTML, and never forward app credentials. Empty metadata is a successful result.

Aliases and publication

An alias is **one segment** of letters, numbers, and hyphens (normalized to lowercase; 1–50 characters). It is globally unique among **active** lists (deleted aliases are immediately reusable, because deletion is permanent). Enforce the same live validation messages as the publish bar on the server.

If the alias is blank at publication, generate an available 7-character random alias from lowercase letters and digits.

Block publish/save when the list is empty or the alias is invalid or already in use. On failure, keep all user input. There is no additional "permanent" confirmation for publication.

Login and ownership

Show a login modal following the reference app's structure: a heading "Sign in to" with the logo, then three full-width Bulma buttons (`button is-fullwidth`), in this order, each with its provider's icon:

1. **"with Twitter/X"**
2. **"with GitHub"**
3. **"with Google"**

This app is local and must not call any real identity provider. Each button signs in as a stable fictional user stored in SQLite; map each provider button to a distinct mock identity (at least two distinct users total) and render that user's name and avatar in the navbar. Ownership is the (stable user ID, provider) pair, not the display name.

Use a server-verifiable HTTP-only session cookie with an appropriate SameSite policy; use Secure in production HTTPS while allowing local HTTP development. Login survives reload; logout ends the session. The navbar's signed-in state and all owner-only pages derive from the server session, never from client storage.

An owner can load, edit, and delete their list. Non-owner requests for an owner's list fail (401 on update/delete). My Lists requires login.

My Lists

Login required. H2 (large, medium weight, primary color): **"My Lists"**. Then a responsive grid of uniform-height tiles (about 195px in the reference): 1 column on mobile, 2 on tablet, 3 on desktop, 4 on widescreen.

• First tile: a dashed-border placeholder tile of the same size as the others, containing a large **+**

centered above the text **"Create new list"**; clicking it starts a fresh draft and goes to `/s/new`.
• Per-list tile: a card (surface with a soft shadow) with a primary-colored pill tag reading **"{N} Links"** (e.g. "4 Links") positioned over the tile's top-left edge, straddling the card's top border; a subtle CSS dot-pattern texture at the tile's top-right corner (in place of the reference app's `bg.png` dots); inside the card, left-aligned, the vanity URL as the title (medium weight, larger) and the description below it in regular text (omitted when empty). The whole tile is clickable (pointer cursor) and loads that list into the editor state, then goes to `/s/edit`.
• While loading: 3 skeleton tiles. If the request fails, show an empty grid without error text.

There is **no Deleted section and no Restore action**.

Delete

The edit page (`/s/edit`) shows a full-width danger (red) button labeled **"Delete This List"** below the link list. Clicking it opens a danger confirm modal with title **"Delete this list?"** and body **"The url {vanity} will be released for others to use."** (the alias rendered in the danger color). On confirm: permanently delete the list (content, links, and alias), reset the editor state, and navigate to `/s/new`. There is no restore.

Public list

`/{vanity}` renders one active list from the server. While loading show a large primary-colored H2 reading **"Loading {vanity}"** followed by 5 skeleton link rows.

On success:

• Heading: the list's **description** (large, primary color). No fallback needed.
• Share row (left, a connected group of icon buttons, all opening in a new tab, URL-encoded):
• X/Twitter: `https://twitter.com/intent/tweet?text={description} https://theurlist.com/{vanity}`
• Facebook: `https://www.facebook.com/sharer/sharer.php?u=https://theurlist.com/{vanity}`
• LinkedIn: `https://www.linkedin.com/shareArticle?mini=true&summary={description}&url=https://theurlist.com/{vanity}`
• View toggle (right, a connected group of two icon buttons with an accessible selected state): **"View as List"** and **"View as QR Code"**.
• QR view: a centered, scannable QR code SVG of the URL `https://theurlist.com/{vanity}` (4x module scale, colors `#121212` on `#F9FAFC`, error correction medium).
• List view: link cards in stored order, with the same card anatomy as the editor rows (64px image or placeholder on the left, bold title, description, small destination URL) but read-only — no grip, no delete, no editing. Each card is itself a link opening the destination in a new tab (`rel="noopener noreferrer"`); the title falls back to the destination URL; the description is omitted when empty.
• Below the cards, a link **"Report this list"** that opens a mailto to `support@theurlist.com` with subject "LinkBundle Flagged" and the list URL in the body.

When the alias does not resolve to an active list, show the not-found state instead: a large Font Awesome "sad face" or "search" icon (in place of the reference app's mascot image), H2 **"We couldn't find that Urlist"**, and H3 **"But don't be sad! That means {vanity} is still available."** where the alias is a link. Clicking that alias starts a fresh draft pre-filled with it and goes to `/s/new`. There is no tombstone or "deleted" state.

Theme, responsive UI, and accessibility

Provide a theme control in the navbar: a hoverable dropdown whose button shows the currently active theme's icon, with items (icon + label, checkmark on the active one): **Light**, **Dark**, **System**. Persist the choice (e.g. `localStorage["preferredTheme"]`, default `"system"`), apply it via a `data-theme` attribute on `<html>`, and set it with an inline script before first paint to avoid a wrong-theme flash. System follows the OS preference. Use Bulma's built-in dark mode (`data-theme="dark"` / `data-theme="light"`) rather than hand-rolled theme CSS, and provide light and dark logo variants of your own (not the reference app's `logo-dark.svg`).

Support current Chromium from desktop down to 320 CSS pixels with no page-level horizontal scrolling; the editor, modals, and public page must remain usable.

Meet WCAG 2.2 AA, including full keyboard operation, reduced-motion support, sensible focus management, and announced status/error changes. In-place editable text in link rows must be reachable with Tab, have an accessible name, and show a visible focus indicator.

Show truthful loading, empty, success, blocked, and error states. Errors must explain recovery. Never present failure as success.

Storage and security

Use versioned SQL migrations or an idempotent versioned initializer. Enable SQLite foreign keys. Use transactions for publish, save, delete, and reset.

Persist users (mock identities), lists, ownership/timestamps, links, and positions. Published data must survive a complete restart. Configure the database path by environment variable with a safe local default; do not commit database files.

Treat user text and fetched metadata as untrusted when rendering, prevent stored/reflected script execution, encode share parameters, apply CSRF protection, and keep internals out of errors.

Provide a development/test-only deterministic reset, preferably `POST /__test/reset` returning `204`. It clears lists and sessions and restores the mock identities. Disable or protect it in production.

Scripts, tests, and documentation

Provide npm scripts:

```text
dev
build
start
lint
typecheck
test
test:e2e
db:init
```

db:reset
```

Use Vitest for URL and alias validation, random alias generation, metadata parse precedence (title/description/image including the favicon fallback), ownership, SQLite transactions, and share URL construction.

Use Playwright for: the home page first-link flow (valid and invalid → "That doesn't look like a valid URL"); draft persistence across reload and logout; publishing while signed out (disabled "Login to Publish" → login modal with the three provider buttons → publish → landing on `/{vanity}`); live "already in use" and "letters, numbers, and dashes" validation; edit mode (disabled alias; adding a link from the editor's input widget; in-place editing of a row's title, description, and URL by clicking the text and typing, with no visible field chrome; drag-and-drop reorder); "Delete This List" confirmation ("The url … will be released for others to use.") and the alias being immediately claimable by another user; My Lists (create-new tile, "N Links" tags, skeleton loading, click-through to `/s/edit`); the public page (skeleton "Loading …", description heading, X/Facebook/LinkedIn share URLs, List/QR toggle with a scannable QR of `https://theurlist.com/{vanity}`, "Report this list" mailto); the not-found state ("We couldn't find that Urlist" + click-the-alias pre-filled draft flow); the terms page and the 404 page; the New-draft "Clear this list?" guard; theme switching, persistence, and no flash; and desktop plus 320px mobile layouts of the home, editor, and public pages.

Tests use a separate temporary database and must not depend on order.

Ship a README covering setup, environment variables, database, commands, mock login, reset, tests, and assumptions.

## Completion

Before you finish:

1. Run `npm run lint`, `npm run typecheck`, `npm test`, `npm run test:e2e`, and `npm run build`.
2. Start the production build and confirm the app responds.
3. In a real Chromium browser, complete every user journey: login/logout for at least two mock identities via the three provider buttons; signed-out → disabled "Login to Publish" → login → publish; owned-list create, drag-reorder, and in-place field editing; non-owner denial (401); hard delete and alias re-claim by a second user; My Lists create/edit flows; public links/QR/share; the not-found pre-filled-alias flow; the terms and 404 pages; the New-draft guard; themes; and desktop/mobile layouts.
4. Confirm active data survives a complete restart.
5. Reconcile `TECHNICAL_SPEC.md` against this instruction, fix every missing or incorrect requirement, and confirm each checked item has evidence.
6. Repeat affected validation and leave no unchecked item unless it has a reported external blocker.

Report what you built, assumptions, architecture, exact command results, and anything incomplete with its reason. Do not claim a check passed unless you ran it successfully.
```