Group Discussion and Consolidation: Priorities and Roadmap for a Static shadcn/ui Frontend

Project Context, Goals, and Consolidation Approach

Context This project ships a shadcn/ui-based React frontend that serves static assets, reads playlist.json as the primary data source, and operates without a live backend; Supabase is disabled for now and treated as a future option only [1]. Under a frontend-only constraint, we will implement robust client-side logic for “Top 10” ordering from createdAt and client-side search and indexing to meet MVP and Sprint 2 goals [1]. To avoid ambiguous date behavior across browsers and locales, all createdAt values will use ISO 8601 with an explicit “Z” or offset, and we will parse only ISO strings to ensure deterministic ordering [2]. We will use date-fns for comparisons to keep sorting and formatting pure and reliable [3]. For user-facing date presentation, we will render with Intl.DateTimeFormat to ensure locale- and timezone-appropriate formatting [4]. For search, we will prefer an indexed, client-side library that balances footprint and responsiveness for medium datasets and wire the UI with accessible combobox and listbox patterns aligned to WAI-ARIA Authoring Practices [1].

Goals We will deliver a consolidated priorities list spanning features and content and a staged schedule that distinguishes MVP from Sprint 2, grounded in practices appropriate to a frontend-only application. The outcome is a single, traceable backlog and a release plan that can be executed without backend dependencies while maintaining quality and accessibility across the UI.

Stakeholder synthesis and traceability We will consolidate input from Emma, Bob, Alex, David, and Iris through a structured process that preserves traceability and aligns on outcomes. First, we will group observations and requests with thematic clustering and affinity mapping to surface common needs and pains. Next, we will build an Opportunity Solution Tree (OST) that maps one clearly defined product outcome to customer opportunities, candidate solutions, and assumption tests, making implicit assumptions explicit and enabling compare-and-contrast decisions [5]. We will follow the OST process by defining the outcome at the top, interviewing and mapping opportunities, selecting a target opportunity, brainstorming multiple solutions, and breaking those into assumptions and tests to iteratively refine the tree [6]. We will create a User Story Map linked to OST opportunities so features are contextualized along the customer journey and we avoid a “flat backlog” of context-free items [7]. Traceability will be maintained by linking each backlog item to its OST node and tagging the originating stakeholder report to communicate rationale and enable meaningful feedback throughout the project [5].

Prioritization framework and communication We will use RICE as the primary scoring method because it quantifies “impact per time worked” in a transparent, repeatable way that fits a static frontend MVP under time and resource constraints [8]. RICE factors and formula will be applied consistently: Reach, Impact, Confidence, and Effort, with the RICE Score computed as (Reach × Impact × Confidence) / Effort [9]. The Impact scale and Confidence tiers will follow Intercom’s guidance, and Effort will be estimated in whole person-months or 0.5 for well under a month [9].

RICE rubric (used for scoring and ranking) We will apply the following rubric with one primary outcome per scoring cycle, then sort by RICE and adjust for dependencies and table-stakes transparently [9].

Criterion	Scale/Units	Guidance
Reach	Number affected over a defined period	Use analytics or historical usage; set a consistent period
Impact	3, 2, 1, 0.5, 0.25	Tie to one primary outcome (e.g., adoption or conversion)
Confidence	100%, 80%, 50%	Reflect strength of data backing Reach/Impact/Effort
Effort	Person-months (whole numbers or 0.5)	Include design, frontend, QA, PM; round up for risk
RICE Score	(Reach × Impact × Confidence) / Effort	Sort, then review dependencies/table-stakes

To communicate inclusion/exclusion in timeboxed releases, we will add MoSCoW tags (Must/Should/Could/Won't) to items, noting MoSCoW improves clarity but does not rank items within buckets [8].

Estimation approach and quality gates We will estimate with Planning Poker using a Fibonacci-based story point scale (1–2–3–5–8) to engage all roles, reduce anchoring via simultaneous card reveals, and converge through discussion for each backlog item [10]. Story points are a relative measure of size and uncertainty, and we will not equate points to hours to avoid the common pitfall that undermines estimation's purpose [11]. We will apply Atlassian's guidance to keep estimates high-level, cap task size and split large items, and recalibrate using completed reference stories and team velocity for sprint forecasting [7]. Items will meet a Definition of Ready before sprint inclusion so goals and acceptance criteria are clear and work can start without major unknowns [12]. We will apply a shared Definition of Done to every increment and define item-specific Acceptance Criteria that are testable and concrete to ensure shippable quality and reduce rework [13]. We will visualize and standardize DoD and AC on the board to enforce explicit policies, improve consistency, and provide a reliable path from "in progress" to "done" [14].

Pitfalls to avoid

- Anchoring during estimation is mitigated through simultaneous card reveals and consensus-building in Planning Poker [10].
- Do not equate story points to hours; treat points as relative size and use velocity to forecast realistically [11].
- Avoid over-precision and detailing deep backlog items likely to change; refine regularly and involve all roles to keep the backlog healthy and ready for sprint planning [7].

Transition to priorities, tasks, dependencies, and schedule In the next sections, we will translate this foundation into a consolidated priorities list, measurable tasks with owners and estimates, explicit dependencies, and a clear MVP and Sprint 2 delivery plan aligned to the frontend-only constraints [1]. Dependencies will be documented, such as "Top 10" ordering relying on createdAt in playlist.json with strict ISO 8601 parsing for deterministic sorting [2]. Search will depend on index construction and tuning to balance footprint and latency for a static React app using an indexed library suited to medium datasets [15]. All plans will note that the backend is disabled and Supabase remains a future option to avoid hidden dependencies in MVP and Sprint 2 [1].

Consolidated Priorities and Release Plan (MVP and Sprint 2)

This section consolidates the five stakeholder inputs into an outcome-oriented, traceable plan for a static React app using shadcn/ui, with no backend at launch and playlist.json as the sole data source [1].

Stakeholder themes and opportunity map

We synthesized the inputs using an Opportunity Solution Tree to align on one outcome and make assumptions explicit before selecting target opportunities [5]. We then sketched a lightweight user story map to avoid a flat backlog and to

sequence work along the discovery and browse-to-play journey [7].

- Outcome: Enable users to quickly discover and browse the most relevant sessions/resources in a static environment, then find specific items via search, with high accessibility and predictable performance.
- Opportunities (O) derived from themes:
- O1. Deterministic discovery: “Top 10” must be correct and stable, based on createdAt in playlist.json.
- O2. Findability at scale: Client-side search with responsive typeahead and relevant results.
- O3. Accessibility by default: Keyboard and screen reader support for search/discovery UI.
- O4. Static robustness: Unambiguous dates, predictable formatting across locales, and resilient handling of malformed data.
- O5. Delivery quality: Clear AC/DoR/DoD to reduce rework and ensure shippable increments.

Stakeholder theme	Mapped opportunity	Why it matters
Correct “Top 10” ordering	O1	Trust and credibility of the homepage/landing experience
Fast, relevant search	O2	Users can find content beyond the Top 10
A11y and keyboard UX	O3	Inclusive access, compliance, and usability
Static data hygiene (ISO, formatting)	O4	Deterministic behavior in a backend-free app
Predictable delivery, fewer regressions	O5	Faster iteration, clear quality gates

Prioritization approach (RICE + MoSCoW)

We use RICE to score “impact per time worked,” with Reach (users per scoring period), Impact (3/2/1/0.5/0.25), Confidence (100/80/50%), and Effort (person-months), then sort by RICE and adjust only for dependencies and table-stakes [9]. We add MoSCoW tags to communicate release inclusion for a fixed MVP/Sprint 2 timeframe, acknowledging MoSCoW does not rank within buckets [8]. The plan is constrained to a frontend-only static app, so items requiring a backend are excluded or explicitly deferred [1].

Scoring cycle notes:

- Period: initial scoring assumes 1,000 monthly active users; we will recalibrate with real analytics post-MVP per the rubric guidance to use a consistent period [9].

Constraints and architectural guardrails

- Supabase is disabled; backend is future-only. All functionality must run entirely client-side against static assets.
- playlist.json is the canonical data source; createdAt must be ISO 8601 with explicit “Z” or offset for unambiguous parsing [2].
- We will parse only ISO date strings and guard against invalid inputs, since Date.parse is reliable for ISO but not arbitrary formats [16].

RICE-prioritized backlog with MoSCoW tags

The following initial scores guide release slicing; they will be re-baselined after MVP telemetry.

ID	Feature	MoSCoW	Reach	Impact	Confidence	Effort (PM)	RICE
F1	Deterministic “Top 10” from createdAt	Must	800	2	0.8	0.5	2560
F2	ISO enforcement + Intl date formatting	Must	1000	1	1	0.5	2000
F3	Search (MiniSearch) with exact-prefix typeahead	Must	600	2	0.8	1	960
F4	A11y combobox/listbox for search results	Must	600	1	0.8	0.5	960
F5	Quality gates: AC/DoR/DoD baseline	Must	1000	1	0.8	0.5	1600
F6	Fuzzy search + auto-suggest + field boosting	Should	600	1	0.8	1	480
F7	Filters/sorting controls (date, tag)	Should	500	0.5	0.8	0.5	400
F8	Prebuilt JSON index experiment + lazy load	Could	1000	0.5	0.5	1	250
F9	Resilience: invalid createdAt handling, DST tests, skeleton/error UI	Should	1000	1	0.8	0.5	1600

MVP scope and acceptance criteria

MVP focuses on the table-stakes discovery and search path with correctness, a11y, and baseline quality.

- F1 Deterministic “Top 10” from createdAt (Must)
- Acceptance criteria
  - Given playlist.json with valid ISO 8601 createdAt values ending in “Z” or explicit offsets, the Top 10 list orders items strictly by descending timestamp with stable ties.
  - Items with missing or invalid createdAt are excluded from the Top 10 and logged for data remediation.
  - Unit tests cover ISO date-only vs date-time, explicit offsets, and DST boundary cases to ensure deterministic ordering.
- Success metrics
  - Functional: 100% ordering tests pass across browsers.
  - Performance: median Top 10 render under 100 ms on a 1,000-item dataset on a mid-tier device.
- F2 ISO enforcement + Intl date formatting (Must)
- Acceptance criteria
  - Data loader rejects or quarantines non-ISO createdAt values; only ISO with “Z” or explicit offsets enter the UI pipeline.
  - All date displays use Intl.DateTimeFormat with an explicit timeZone configuration.
  - A visible data-quality banner surfaces the count of rejected items when any are found.
- Success metrics
  - Data hygiene: 0 non-ISO dates admitted to UI; 100% of date renders use Intl.
- F3 Search (MiniSearch) with exact-prefix typeahead (Must)
- Acceptance criteria

    - MiniSearch indexes title and tags at startup and supports prefix-based typeahead; no fuzzy expansion is enabled in MVP.
    - Query latency p95 ≤ 50 ms on a 5,000-item corpus on a mid-tier device.
    - Top five results visibly update as the user types and link correctly to details.
- Success metrics
    - Performance: p95 query latency ≤ 50 ms on the test corpus.
    - Adoption: ≥ 40% of sessions trigger at least one search within two weeks of launch.
- F4 A11y combobox/listbox for search results (Must)
- Acceptance criteria
    - The search input implements role=combobox with aria-expanded and aria-controls mapped to a popup listbox; Down Arrow moves focus into the popup per APG [17].
    - The results popup implements role=listbox with option children, supports Up/Down/Home/End, and tracks focus via aria-activedescendant [18].
    - Screen reader announcements include active option changes and selection; automated axe scan shows 0 high-severity violations on the search flow.
- Success metrics
    - Usability: 100% of keyboard interactions validated in manual a11y test cases.
    - Compliance: 0 high-severity a11y violations on the search and results UI.
- F5 Quality gates: AC/DoR/DoD baseline (Must)
- Acceptance criteria
    - Every story contains specific, testable acceptance criteria before entering a sprint [13].
    - A Definition of Ready checklist is applied during refinement before pull-in [12].
    - A shared Definition of Done checklist (code review, tests pass, PO acceptance) is applied before merge/release [14].
- Success metrics
    - Delivery: ≤ 5% of merged stories re-opened post-release in the MVP cycle.
    - Process: 100% of MVP stories show AC/DoR/DoD artifacts in the board.

Sprint 2 scope and acceptance criteria

Sprint 2 extends findability, resilience, and performance under the same static architecture.

- F6 Fuzzy search + auto-suggest + field boosting (Should)
- Acceptance criteria
    - MiniSearch enables fuzzy matching with a minimum exact prefix to cap expansions and sustain responsiveness [19].
    - Auto-suggestions appear after 2 characters with debounced queries; title matches are boosted over tags.
    - Relevance validation: at least 4 of top 5 results match intent in qualitative tests across 10 sample queries.
- Success metrics
    - Performance: p95 fuzzy query latency ≤ 100 ms on 5,000 items.

    - Quality: ≥ 80% top-5 relevance in moderated tests.
- F7 Filters/sorting controls (date, tag) (Should)
- Acceptance criteria
    - Users can sort by createdAt asc/desc and filter by tag; controls are toggleable and persist within session.
    - Controls are keyboard accessible and announced to screen readers with proper state.
    - Sorting/filtering interactions compose with search without errors.
- Success metrics
    - Engagement: ≥ 20% of search sessions use at least one filter or sort control.
- F8 Prebuilt JSON index experiment + lazy load (Could)
- Acceptance criteria
    - Build pipeline can output a compressed JSON index artifact; the app lazy-loads it on the first search interaction.
    - Memory ceiling: in-memory index ≤ 5 MB on load for the target corpus.
    - A/B toggle allows runtime comparison between runtime indexing and prebuilt index.
- Success metrics
    - Startup: first contentful paint unaffected (±5%) vs. MVP baseline.
    - Query speed: no regression vs. runtime indexing on p95 latency.
- F9 Resilience: invalid createdAt handling, DST tests, skeleton/error UI (Should)
- Acceptance criteria
    - Invalid or missing createdAt entries render in non-Top-10 contexts with a clear “Date unavailable” label and do not break sorting.
    - Expanded automated tests cover DST transitions and explicit offset parsing for deterministic outcomes.
    - Loading skeletons and error banners render within 100 ms on slow networks to maintain perceived responsiveness.
- Success metrics
    - Reliability: 0 unhandled exceptions in date parsing/sorting across supported browsers.
    - UX: ≤ 1% of sessions show an unhandled error banner.

Technical notes that anchor the plan

- Dates: createdAt must be ISO 8601 with “Z” or explicit offsets; this avoids local-time ambiguities and ensures deterministic sorting in the browser [2].
- Parsing: Only parse ISO; non-standard strings are unreliable with Date.parse, so invalid data must be detected and handled explicitly [16].
- Sorting: Convert to timestamps and sort numerically or use date-fns compareAsc/compareDesc for deterministic ordering [3].
- Display: Use Intl.DateTimeFormat with explicit options (e.g., timeZone) for locale-correct date rendering in a client-only app [4].
- Search engine: Prefer MiniSearch for a compact, client-side inverted index with prefix/fuzzy matching and auto-suggest suitable for medium datasets [15].

- Fuzzy performance: Require an exact prefix before enabling fuzzy to reduce candidate expansion and maintain latency budgets [19].
- A11y patterns: Implement combobox semantics on the search input and listbox semantics for results, with expected keyboard interactions and aria- attributes per APG [17].
- Listbox behavior: Track active option with aria-activedescendant and support Up/Down/Home/End navigation in the results list [18].

Dependencies and feasibility summary

- F1 depends on playlist.json providing valid ISO createdAt; F2 enforces this at load-time to protect F1 behavior [2].
- F3 depends on the search index being initialized; F4's a11y behaviors layer atop F3's UI scaffolding [15].
- F6 builds on F3 by enabling fuzzy and suggestions with prefix guards to control performance [19].
- F8 explores swapping runtime indexing for a prebuilt index to further reduce CPU and payload when the corpus grows, while keeping the static-only constraint [15].

All items are feasible within a static React/shadcn/ui app and avoid server calls, acknowledging that any backend integration is explicitly out of scope until a future iteration [1].

Next: task breakdowns, owners, estimates, and dependencies

The next section will decompose each feature into measurable tasks with owners, apply Planning Poker with Fibonacci story points, and enforce a size cap with splits for oversized items to improve predictability [7]. We will run Planning Poker collaboratively to converge on estimates and calibrate using reference stories [10].

Task Breakdown, Owners, Estimates, Dependencies, and Schedule

1) Estimation scale and capacity assumptions

- We use Planning Poker with a Fibonacci scale (0.5, 1, 2, 3, 5, 8, 13) to estimate relatively by complexity, uncertainty, and effort, engaging all roles to converge on a shared understanding [10].
- Do not equate story points (SP) to fixed hours; the indicative hour ranges below are for planning communication only, not a conversion rule, and we will manage scope via velocity and empirical delivery rather than time-per-point math [11].
- We cap individual tasks at 8 SP; if a task feels larger, we split it and re-estimate to improve predictability and flow before scheduling into a sprint [7].
- Initial team capacity assumption: 30–35 SP per two-week sprint across the team, subject to calibration after Sprint 1; forecasts are not guarantees and will be updated as new information emerges [20].

Indicative hour ranges per SP (guidance, not a contract) [7]:

- 1 SP ≈ 2–6 hours
- 2 SP ≈ 4–10 hours
- 3 SP ≈ 6–15 hours
- 5 SP ≈ 12–25 hours
- 8 SP ≈ 20–40 hours

2) Per-feature task breakdowns (F1–F9) Feature assumptions, aligned to the prior plan:

- F1: Top 10 computation based on createdAt in playlist.json using ISO 8601 and deterministic client-side sorting [2].

- F2: Client-side search using MiniSearch with accessible combobox/listbox UI [15].
- F3: Static JSON data schema/loader and validation pipeline (no backend) [2].
- F4: Date/time presentation via Intl.DateTimeFormat with explicit timeZone policy [4].
- F5: Accessibility patterns (combobox/listbox keyboarding and ARIA) [17].
- F6: Performance and indexing optimizations (runtime vs prebuilt index, fuzzy tuning) [15].
- F7: QA automation and regression (DST/date parsing, search, a11y) [2].
- F8: UI integration and pages (Home with Top 10, Search, Detail, Empty/Error states) [1].
- F9: Optional future-backend abstraction (non-blocking; Supabase disabled) [1].

F1 — Top 10 by createdAt (client-side, deterministic) | Task ID | Description | Owner | Estimate (SP,hours) | Acceptance checks | Dependencies | | --- | --- | --- | --- | --- | --- | | F1-T1 | Define createdAt spec (ISO 8601 with Z/offset) and update sample playlist.json | David | 2 SP, 4–10h | All sample items use unambiguous ISO 8601 with Z/offset; schema doc updated [2] | — | | F1-T2 | Implement safe parser: createdAt→Date, guard NaN, define fallback/exclusion policy | Bob | 3 SP, 6–15h | Invalid strings produce NaN and are excluded/logged per policy; unit tests cover null/empty/invalid [16] | F1-T1 | | F1-T3 | Implement Top 10 sort descending by timestamp with stable tie-break | Alex | 2 SP, 4–10h | Sorting uses getTime/compareDesc; ties break by title to ensure stable order; unit tests green [3] | F1-T2 | | F1-T4 | Edge-case tests: date-only vs date-time, explicit offsets, DST boundaries | Iris | 3 SP, 6–15h | Tests assert deterministic ordering across locales/timezones; DST cases included [2] | F1-T2 | | F1-T5 | Top 10 UI component (shadcn/ui list/cards), empty/error states | Emma | 3 SP, 6–15h | Renders 10 items max; graceful empty/error UI; responsive layouts | F1-T3 | | F1-T6 | Data ingestion doc: publisher checklist for createdAt | David | 1 SP, 2–6h | Markdown doc committed; linked from repo; examples include Z/offset [2] | F1-T1 |

F2 — Search with MiniSearch and accessible UI | Task ID | Description | Owner | Estimate (SP,hours) | Acceptance checks | Dependencies | | --- | --- | --- | --- | --- | --- | | F2-T1 | Choose MiniSearch config (fields, boosting, id) and initialize index | Alex | 3 SP, 6–15h | Index builds at runtime; fields/boosts documented; memory within budget [15] | F3-T2 | | F2-T2 | AddAll documents from loader; handle adds/removals | Bob | 2 SP, 4–10h | Dataset indexes without errors; reindex path documented [15] | F2-T1,F3-T3 | | F2-T3 | Combobox input with aria attributes and keyboard focus mgmt | Emma | 5 SP, 12–25h | role=combobox, aria-expanded/controls wired; Down Arrow moves into popup per APG [17] | F5-T1 | | F2-T4 | Results popup listbox with option items and navigation | Emma | 5 SP, 12–25h | role=listbox with option children; Up/Down/Home/End, aria-activedescendant tracked [18] | F2-T3,F5-T2 | | F2-T5 | Auto-suggest and prefix/fuzzy tuning for typeahead | Alex | 3 SP, 6–15h | Suggestions appear <100ms at 5k docs; minimal prefix enforced before fuzzy [19] | F2-T2 | | F2-T6 | Search result selection → navigation | Bob | 2 SP, 4–10h | Enter/click navigates; focus returns predictably; tests pass | F2-T4 | | F2-T7 | Search unit tests (query variants, fuzzy/prefix) | Iris | 2 SP, 4–10h | Tests for exact/prefix/fuzzy; latency assertions in CI [15] | F2-T5 |

F3 — Static data schema, loader, validation (no backend) | Task ID | Description | Owner | Estimate (SP,hours) | Acceptance checks | Dependencies | | --- | --- | --- | --- | --- | --- | | F3-T1 | Define JSON schema/types; required fields incl. createdAt | David | 2 SP, 4–10h | Schema published; type checks in build; createdAt required [2] | — | | F3-T2 | Implement loader to fetch static JSON and cache in memory | Bob | 3 SP, 6–15h | Single network fetch; memoized; retries with backoff | F3-T1 | | F3-T3 | Validation: reject malformed createdAt; sanitize missing fields | Alex | 3 SP, 6–15h | Invalid items dropped with reason; metrics logged; tests cover NaN parse [16] | F3-T2 | | F3-T4 | Data pipeline hook: emit normalized docs to search index | Bob | 2 SP, 4–10h | Emits normalized docs with stable ids; order-agnostic | F3-T3,F2-T1 | | F3-T5 | CI step: schema validation on PR | David | 1 SP, 2–6h | PR fails on invalid JSON; report lists offending items | F3-T1 |

F4 — Date/time presentation and locale policy | Task ID | Description | Owner | Estimate (SP,hours) | Acceptance checks | Dependencies | | --- | --- | --- | --- | --- | --- | | F4-T1 | Implement Intl.DateTimeFormat wrapper with explicit timeZone | Alex | 2 SP, 4–10h | Dates render consistently given locale/timeZone; snapshot tests [4] | F1-T2 | | F4-T2 |

Apply formatter in Top 10 and Detail views | Bob | 2 SP, 4–10h | All visible dates use wrapper; no direct Date.toString calls [4] | F4-T1,F8-T2 | | F4-T3 | Documentation: display policy (UTC vs local) | David | 1 SP, 2–6h | Policy recorded; examples for “medium/short” styles [4] | F4-T1 |

F5 — Accessibility baseline (combobox/listbox patterns) | Task ID | Description | Owner | Estimate (SP,hours) | Acceptance checks | Dependencies | | --- | --- | --- | --- | --- | --- | | F5-T1 | A11y design review: combobox interaction model | Emma | 2 SP, 4–10h | Interaction spec aligns with APG; flows documented [1] | — | | F5-T2 | aria-activedescendant and selection state handling | Bob | 3 SP, 6–15h | Active option tracked correctly; aria-selected reflects state [18] | F5-T1 | | F5-T3 | Keyboard support: type-ahead, Esc, Tab/Shift+Tab | Alex | 3 SP, 6–15h | Behavior matches combobox/listbox expectations [17] | F5-T2 | | F5-T4 | Automated a11y checks (axe) + manual screen reader smoke | Iris | 2 SP, 4–10h | No critical violations; SR announces roles/states properly [1] | F2-T4 |

F6 — Performance and indexing | Task ID | Description | Owner | Estimate (SP,hours) | Acceptance checks | Dependencies | | --- | --- | --- | --- | --- | --- | | F6-T1 | Runtime indexing budget and metrics (TTI/memory) | David | 2 SP, 4–10h | Budget set; metrics dashboard in CI for index size/time [15] | F2-T1 | | F6-T2 | Lazy-load index and code-split search route | Alex | 3 SP, 6–15h | Search bundle loads on demand; TTI unaffected on Home | F6-T1 | | F6-T3 | Prefix-length tuning and fuzzy distance guardrails | Bob | 2 SP, 4–10h | prefix_length enforced; 95th percentile query <150ms [19] | F2-T5 | | F6-T4 | Optional: prebuilt JSON index script (build-time) | Alex | 5 SP, 12–25h | Prebuilt index generated from playlist.json; behind flag; not used by default [15] | F3-T1 | | F6-T5 | Bundle size budget and CI check | David | 1 SP, 2–6h | CI fails if bundle exceeds budget; action items auto-created | F6-T2 |

F7 — QA automation and regression | Task ID | Description | Owner | Estimate (SP,hours) | Acceptance checks | Dependencies | | --- | --- | --- | --- | --- | --- | | F7-T1 | Unit tests for Date.parse vs ISO and NaN handling | Iris | 2 SP, 4–10h | Invalid formats → NaN; ISO parses as expected; tests green [16] | F1-T2 | | F7-T2 | DST boundary regression pack | Iris | 2 SP, 4–10h | Cases around spring/fall transitions; deterministic results [2] | F7-T1 | | F7-T3 | E2E: Top 10 ordering and search flows | Bob | 3 SP, 6–15h | Cypress/Playwright scenarios pass reliably | F1-T5,F2-T6 | | F7-T4 | A11y automated checks in CI | Alex | 2 SP, 4–10h | axe checks run per PR; no critical violations [1] | F5-T4 |

F8 — UI integration and pages (static React/shadcn/ui) | Task ID | Description | Owner | Estimate (SP,hours) | Acceptance checks | Dependencies | | --- | --- | --- | --- | --- | --- | | F8-T1 | Home page with Top 10 section and entry points | Emma | 3 SP, 6–15h | Home renders Top 10; link to Search; responsive | F1-T5 | | F8-T2 | Detail page: metadata and formatted createdAt | Bob | 2 SP, 4–10h | Visible date via Intl wrapper; deep linkable [4] | F4-T1 | | F8-T3 | Search page layout using combobox/listbox | Emma | 3 SP, 6–15h | Layout integrates F2 widgets; empty states | F2-T4 | | F8-T4 | Empty/error states for data fetch | Alex | 2 SP, 4–10h | Network errors surfaced; retry; accessible alerts | F3-T2 |

F9 — Optional future-backend abstraction (non-blocking; Supabase disabled) | Task ID | Description | Owner | Estimate (SP,hours) | Acceptance checks | Dependencies | | --- | --- | --- | --- | --- | --- | | F9-T1 | Define DataSource interface (static JSON vs future API) | David | 2 SP, 4–10h | Interface documented; default impl = static JSON; feature-flagged | F3-T2 | | F9-T2 | Adapter skeleton for Supabase (disabled path) | Alex | 3 SP, 6–15h | Stub compiles; not bundled by default; behind env flag | F9-T1 | | F9-T3 | Risk check: ensure no runtime backend calls in MVP | Iris | 1 SP, 2–6h | CI check verifies adapters disabled in production build | F9-T2 |

3) Cross-cutting tasks (supporting multiple features) | Task ID | Description | Owner | Estimate (SP,hours) | Acceptance checks | Dependencies | | --- | --- | --- | --- | --- | --- | | X-T1 | Definition of Ready/Done adoption and board policies | David | 1 SP, 2–6h | DoR/DoD visible; used in PR templates; reduces rework [7] | — | | X-T2 | RICE rubric snapshot and trace links to backlog | David | 1 SP, 2–6h | Each feature links to RICE decision; exceptions documented [9] | — | | X-T3 | Performance smoke in CI (Lighthouse run) | Alex | 2 SP, 4–10h | Failing thresholds block merges; report attached to PR | F6-T1 | | X-T4 | Documentation hub for data/date/search policies | Emma | 2 SP, 4–10h | Central README links to F1/F2/F4 policies; kept updated [2] | F1-T6,F4-T3 | | X-T5 | Accessibility checklist for UI stories (AC templates) | Iris | 1 SP, 2–6h | AC templates include roles, keyboard, name/role/value [13] | F5-T1 |

4) Dependency map summary and risk hotspots

- Critical chains
- F1-T1→F1-T2→F1-T3→F1-T5→F8-T1: Data spec to Top 10 UI to Home integration is on MVP's critical path [2].
- F3-T2→F3-T3→F2-T1→F2-T2→F2-T3→F2-T4→F2-T6: Loader/validation to indexed, accessible search flow is the second MVP critical path [15].
- F4-T1→F4-T2: Date formatting wrapper before applying across views [4].
- F5-T1→F5-T2→F5-T3→F5-T4: A11y pattern spec through to checks to keep search usable for keyboard/AT users [1].
- F6-T2→F6-T3: Lazy-load then tune prefix/fuzzy for responsive search UX at scale [19].
- Risk hotspots
- Ambiguous dates in incoming JSON causing non-deterministic ordering; mitigated by strict ISO requirement and NaN guards in F1/F3 [16].
- Search latency under fuzzy queries on low-end devices; mitigated by prefix_length and lazy-loading search bundle [19].
- Accessibility regressions with custom combobox/listbox; mitigated by APG alignment and automated checks [1].
- Scope creep into backend work; mitigated by keeping F9 optional and feature-flagged; MVP uses static JSON only [1].

5) Schedule: MVP (Sprint 1) and Sprint 2 allocation Two-week sprints; sequencing honors critical dependencies; all tasks are frontend/static-only. Any future-backend abstraction work in F9 is optional and must not block MVP; Supabase remains disabled in all production builds for MVP and Sprint 2 [1].

Sprint 1 (MVP focus: Top 10, Search baseline, Data pipeline, A11y baseline, Core pages) | Area | Tasks | | --- | --- | | F1 Top 10 | F1-T1, F1-T2, F1-T3, F1-T5, F1-T6, F1-T4 | | F2 Search (baseline) | F2-T1, F2-T2, F2-T3, F2-T4 | | F3 Data pipeline | F3-T1, F3-T2, F3-T3, F3-T4 | | F4 Date/locale | F4-T1 | | F5 Accessibility | F5-T1, F5-T2 | | F7 QA | F7-T1, F7-T3 | | F8 UI integration | F8-T1, F8-T3 | | Cross-cutting | X-T1, X-T4 |

Notes:

- Critical path items are front-loaded: F1-T1→T3 and F3-T2→F2-T4 to ensure Home and Search are shippable; keyboarding and ARIA are included to avoid rework [1].
- Estimates sum within 30–35 SP capacity; any overflow moves to Sprint 2 after Planning Poker calibration [7].

Sprint 2 (Enhancements: formatting, performance, advanced a11y/tests, detail page, optional prebuilt index) | Area | Tasks | | --- | --- | | F2 Search (enhancements) | F2-T5, F2-T6, F2-T7 | | F4 Date/locale | F4-T2, F4-T3 | | F5 Accessibility | F5-T3, F5-T4 | | F6 Performance | F6-T1, F6-T2, F6-T3, F6-T5, F6-T4 (optional) | | F7 QA | F7-T2, F7-T4 | | F8 UI integration | F8-T2, F8-T4 | | F9 Backend abstraction | F9-T1, F9-T2, F9-T3 (all optional, behind flags) | | Cross-cutting | X-T2, X-T3, X-T5 |

Sequencing highlights:

- Apply the date formatter across views after wrapper stability (F4-T2 after F4-T1) [4].
- Performance tuning follows baseline search completion to measure real impact (F6 chain after F2 baseline) [15].
- A11y verification (F5-T4) and DST regressions (F7-T2) finalize quality gates before Sprint 2 release [1].

References

[1] ARIA Authoring Practices Guide | APG | WAI - W3C
[2] Date - JavaScript - MDN
[3] date-fns documentation - DevDocs
[4] Intl.DateTimeFormat() constructor - JavaScript - M...
[5] Opportunity Solution Trees: Visualize Your Discove...
[6] Opportunity Solution Trees: Visualize Your Discove...
[7] What are story points in Agile and how do you esti...
[8] Product Prioritization Frameworks - Productboard
[9] RICE: Simple prioritization for product managers -...
[10] How to Play Planning Poker and Involve the Whole T...
[11] Don't Equate Story Points to Hours - Mountain Goat...
[12] Definition of Done vs. Acceptance Criteria: A comp...
[13] Definition of Done vs Acceptance Criteria - Visual...
[14] Definition of Done vs Acceptance Criteria (and Why...
[15] MiniSearch, a client-side full-text search engine ...
[16] Date.parse() - JavaScript - MDN - Mozilla
[17] WAI-ARIA: Role=Combobox - DigitalA11Y
[18] ARIA: listbox role - MDN - Mozilla
[19] How to Use Fuzzy Searches in Elasticsearch | Elast...
[20] The age old question - Story Points vs Hours | Scr...