BYU Survival Tool

A personal, privacy-first website dashboard for deadlines, schedule, checklists, and quick actions — written as a build-ready spec you can paste into Claude.

What this document is: A detailed Product + UX + Technical spec for a small web app. Hand it to Claude and ask it to generate the project (code + UI) from these requirements.

What this document is not: A vague idea list. Every section includes concrete behaviors, screens, data structures, and acceptance criteria.

1. Product Overview

One-liner: A single place you open every day to see what matters at BYU: today's classes, upcoming deadlines, key links, and a small task/checklist system — without relying on invasive third-party trackers.

Primary user: Ike (a detail-oriented BYU student) who wants a fast, calm, reliable dashboard that reduces cognitive load.

Core promise: Open the site and immediately know: (1) what's next, (2) what's due soon, (3) where to click, (4) what to do today.

2. Guiding Principles

• Fast and quiet: opens in under ~1 second on a typical campus Wi■Fi connection; minimal animations.

• Privacy-first: default is local-only data; no tracking; no ads; no third-party analytics.

• Low-friction input: manual entry should be quick; importing should be optional and simple.

• Accessible by design: works for red-blind/protanopia; never encode meaning using only color.

• Offline-friendly: the dashboard should still show cached info when offline.

• Delight via clarity: UI should feel like a cockpit, not a social network.

3. Target Outcomes

• Reduce 'what am I forgetting?' anxiety by showing deadlines and next actions.

• Make BYU admin (links, dates, processes) one-click.

• Provide a lightweight task/checklist system (not a full todo app).

• Optionally track credits/GPA scenarios with a simple calculator.

4. Information Architecture

The site should have a tiny set of pages. Most usage is on the Dashboard.

Route	Name	Purpose	Shown in Nav?
/	Dashboard	Today view: schedule, due soon, quick links, “next action”.	Yes (default)
/tasks	Tasks	Simple tasks + checklists; quick capture; due dates.	Yes
/courses	Courses	Course list; per-course links; optional grade weights; notes.	Yes
/deadlines	Deadlines	Calendar-like list view; filters; add/edit.	Yes
/tools	Tools	GPA / credits calculators; utilities.	Yes
/settings	Settings	Data import/export, theme, privacy, backup.	Yes

Navigation: left sidebar on desktop; bottom tab bar on mobile. Keep nav labels short.

5. MVP Scope vs. Nice-to-Have

Area	MVP (must ship)	Nice-to-have (later)
Schedule	Manual class schedule + “today” view + next class highlight.	iCal import; auto semester switching.
Deadlines	Manual deadlines; due-soon list; notifications optional.	Canvas integration (if feasible), recurring deadlines.
Tasks	Quick capture; due date; simple checklists; per-course tagging.	Kanban, priority scoring, streaks.
Links	Per-course + global quick links.	Context-aware links (e.g., show testing center near exams).
Data	Local storage + export/import JSON.	Cloud sync (optional), end-to-end encryption.
UI	Clean, minimal, accessible.	Themes, widgets, drag-and-drop layout.

6. Core User Flows

These are the exact workflows the website must support. Each flow ends with an observable outcome.

Flow A - Morning check (10 seconds)

• Open Dashboard.

• See Next Class (time + location) and Due Soon list.

• Click one quick link (Canvas or class page).

• Optional: add a quick task like “read ch 3” in under 5 seconds.

Flow B - Add a deadline (30 seconds)

• Click “Add Deadline”.

• Select course (or None).

• Enter title, due date/time, optional notes, optional link.

• Deadline appears instantly in Due Soon and Deadlines page.

Flow C - Week planning (2-5 minutes)

• Open Deadlines page.

• Filter to next 7 days.

• Add 1-3 tasks per deadline.

• Mark one ‘Most Important’ task for the week (single pin).

Flow D - Course setup at semester start (10 minutes)

• Add courses (name, code, meeting times, location, instructor).

• Add per-course links (Canvas, syllabus PDF, textbook, office hours).

• Optional: add grading weights (Exam 40%, HW 30%, etc.).

Flow E - Export / backup (30 seconds)

• Open Settings.

• Click Export JSON.

• File downloads. (Optionally auto-include timestamp.)

7. Dashboard Requirements (the main page)

The Dashboard is a grid of cards. Default layout should work on laptop and phone without customization.

Card	What it shows	Rules / behavior

Next Class	Next scheduled class today; time range; location.	If no classes today, show “No classes today” and highlight next class tomorrow.
Due Soon	Deadlines due in next 7 days (or user setting).	Sorted by due date/time; show course tag; overdue at top.
Today Tasks	Tasks due today + pinned task.	Max 8 shown; show ‘View all’ if more.
Quick Links	Top 6 links (Canvas, email, library, degree audit, etc.).	User-editable; open in new tab; allow custom icons.
Capture	Single input box: “Add task or deadline…”	Natural language optional (e.g., “Bio lab report Fri 5pm”). If parsing fails, open modal.
Status	Tiny summary: “2 classes left, 3 due soon, 1 overdue”.	Should be readable at a glance; no color-only meaning.

Dashboard micro-interactions

• Keyboard shortcut: ‘/’ focuses Capture input; Enter saves.

• Hover states are subtle; avoid flashy transitions.

• Each item has a 3-dot menu for Edit / Duplicate / Delete.

• Use confirmation for Delete, but keep it quick (undo toast).

8. UI Wireframes (text-only)

These are rough layouts to guide Claude’s implementation. Keep them clean and consistent.

```
[Sidebar] BYU Survival Tool [Search/ /]
Dashboard
Tasks
Courses
Deadlines
Tools
Settings

---------------------------------------------------------------------------
| Next Class | Due Soon |
| 10:00-10:50 CHEM 101 | Fri 5:00p - Bio Lab Report (BIO 100) |
| BNSN 120 | Sat 11:59p - Reading Quiz (HIST 201) |
| [Directions] [Canvas] | Overdue: Calc HW 4 (MATH 112) |
---------------------------------------------------------------------------
| Today Tasks | Quick Links |
| [ ] Read ch 3 (BIO) | [Canvas] [Email] [Library] [MyMAP] [BYU] |
| [ ] Start outline (HIST) | [Testing Center] [Registrar] |
| [Pinned] Finish lab data | |
---------------------------------------------------------------------------
| Capture: [ Add task or deadline...__________________________ ] [Add] |
---------------------------------------------------------------------------

Tasks [ + New Task ] [Filter: All | Today | Overdue | Course ] [Search]
---------------------------------------------------------------------------
| [ ] Finish lab outline BIO 100 due Fri 5:00p (...) |
| [ ] Study set 3 CHEM 101 due Sat (...) |
| [x] Email advisor Admin done (...) |
---------------------------------------------------------------------------
Right panel (optional):
- Details, notes, checklist items, links, created date, history
```

9. Visual Design + Accessibility

Design must work for protanopia (red-blind). Use icons, labels, and shape; do not rely on red/green differences.

• Use a neutral palette (grays + one accent). Avoid meaning encoded only by red/green.

• Overdue items: use an icon (■) + label “Overdue” and optionally bold text; do not rely on color.

• Ensure minimum 4.5:1 contrast for body text.

• Focus states: clearly visible keyboard focus outline.

• Mobile: tap targets at least 44px height.

• Typography: system font; readable sizes; generous spacing; avoid dense walls of text.

Performance: avoid heavy UI libraries; keep bundle small; lazy-load non-dashboard pages.

10. Data Model (local-first)

Default storage is local (browser). Users can export/import a single JSON file.

```
Type Course:
- id: string (uuid)
- code: string (e.g., "CHEM 101")
- name: string (e.g., "General Chemistry")
- term: string (e.g., "Winter 2026")
- meetingTimes: [{ day: "Mon", start: "10:00", end: "10:50", location: "BNSN 120" }]
- links: [{ label: "Canvas", url: "..." }, { label: "Syllabus", url: "..." }]
- colorTag: string (optional, decorative only)

Type Deadline:
- id: uuid
- title: string
- courseId: uuid | null
- dueAt: ISO datetime string
- notes: string (markdown ok)
- link: string | null
- status: "open" | "done"
- createdAt: ISO datetime

Type Task:
- id: uuid
- title: string
- courseId: uuid | null
- dueAt: ISO datetime | null
- pinned: boolean
- checklist: [{ id, text, done }]
- notes: string
- status: "open" | "done"
- createdAt: ISO datetime

Type Settings:
- dueSoonWindowDays: number (default 7)
- weekStartsOn: "Sun" | "Mon"
- theme: "light" | "dark" | "system"
- enableNotifications: boolean (default false)
```

Import / Export requirements

• Export: download a single JSON containing courses, tasks, deadlines, and settings.

• Import: accept that JSON; validate schema; show summary of what will be imported.

• Conflict strategy (simple): either Replace All or Merge by id (MVP can do Replace All).

• Autosave: every change writes to local storage.

11. Key Components + Behaviors

Component	Purpose	Behavior
Item row	Used for tasks and deadlines. Shows title, tag, due date, and quick actions.	Click row opens details. Checkbox toggles done. 3-dot menu: Edit, Duplicate, Delete.
Details modal	Edit item without navigating away.	Fields: title, course, due date/time, notes, link. Save/Cancel.
Course picker	Select course for tasks/deadlines.	Searchable; shows course code + name.
Date/time input	Reliable date-time entry.	Use native HTML input or a lightweight picker. Must handle timezone correctly.
Toast / undo	Quick feedback.	On delete: show “Deleted - Undo” for ~6 seconds.

12. Technology Recommendations (Claude can implement)

You can choose any stack, but here are safe defaults for a small, fast site.

Layer	Recommended	Notes
Frontend	Next.js (App Router) or Vite + React	Choose the one you’re most comfortable deploying.
UI	Tailwind CSS	Fast to build clean layouts; good responsive control.
Storage	localStorage / IndexedDB	local-first; consider IndexedDB if data grows.
State	Zustand or React Context	Keep it simple.
Parsing	Optional: chrono-node (natural language dates)	If too heavy, skip for MVP.
Deployment	Vercel / Netlify (static) or Railway (Node)	Static is easiest if no server is needed.
Testing	Playwright (smoke tests)	Optional; at least do manual test checklist.

Important: Avoid third-party analytics by default. If you need telemetry, use a self-hosted, privacy-respecting option and keep it opt-in.

13. Security + Privacy

• No trackers. No Google Analytics. No ad networks.

• No third-party fonts by default (use system fonts).

• Only call third-party domains when the user clicks a link they added.

• Do not store secrets (tokens) unless the user explicitly chooses and understands the risk.

• Prefer static hosting; avoid a database unless needed.

• Provide a “Delete all data” button in Settings (clears local storage).

Optional advanced mode (later)

• Encrypted export: user sets a passphrase; export JSON encrypted locally.

• Cloud sync with end-to-end encryption (requires careful threat modeling).

14. Acceptance Criteria (MVP)

• Dashboard shows Next Class, Due Soon, Today Tasks, Quick Links, Capture.

• User can add/edit/delete courses, tasks, and deadlines.

• Due Soon list updates immediately and sorts correctly.

• Overdue items are clearly marked without relying only on color.

• Data persists across refresh; export/import works.

• Mobile layout usable; keyboard navigation works on desktop.

• App works offline with last saved data.

15. Build Plan (Suggested Milestones)

Milestone 1 - Skeleton (1 session)

• Set up project, routing, layout, navigation.

• Build Dashboard grid with placeholder cards.

Milestone 2 - Data + CRUD (1-2 sessions)

• Implement local storage layer + data types.

• Add Courses CRUD.

• Add Deadlines CRUD.

• Add Tasks CRUD.

Milestone 3 - Dashboard logic (1 session)

• Compute next class and due soon.

• Pin task behavior.

• Capture input -> creates Task (basic).

Milestone 4 - Polish + export/import (1 session)

• Export/import JSON.

• Empty states, undo toast, keyboard shortcut.

• Accessibility pass.

Milestone 5 - Nice-to-have (optional)

• Natural language parsing in Capture.

• Calendar view.

• Notifications.

16. Copy-Paste Prompt for Claude

Paste the following into Claude (and attach this PDF).

```
You are an expert full-stack engineer and UX designer. Build a privacy-first BYU
Survival Tool website as described below.

Requirements:
- Implement the routes: / (Dashboard), /tasks, /courses, /deadlines, /tools,
/settings.
- Local-first storage (localStorage or IndexedDB). Provide JSON export/import and a
“Delete all data” button.
- Dashboard cards: Next Class, Due Soon (next 7 days), Today Tasks (incl. pinned),
Quick Links (editable), and a Capture input with / shortcut.
- Data types: Course, Deadline, Task, Settings as specified in the document. Use UUIDs
and ISO datetimes.
- Accessibility: do not rely on red/green; overdue uses icon + label; strong focus
states; mobile tap targets.
- Performance: minimal dependencies; no third-party analytics; system fonts.
- Provide the complete project code, with clear instructions to run locally and deploy
(static hosting preferred if possible).
- Include a short manual test checklist to verify acceptance criteria.

UI direction:
- Clean, calm dashboard. Grid cards on desktop, stacked cards on mobile. Sidebar on
desktop, bottom nav on mobile.
- Minimal animations, clear typography, generous spacing.

Now generate:
1) project file tree
2) key files with full code
3) setup and deployment instructions
4) test checklist
```