ChatGPT

Repository Analysis

- **Tech Stack & Style:** The app is built on TanStack Start (React + TypeScript) with Tailwind CSS and Shadcn/UI for styling [1]. All layouts use utility classes (e.g. `min-h-screen bg-background text-foreground`) and custom theme tokens (like `bg-card`, `ring-border`) for a cohesive look. This means our base design is consistent and lightweight.
- **Brand Tone & Palette:** The existing tone is *calm and scholarly*. It uses an off-white background (`#FDF7EF`) and near-black text (`#181220`), with deep blue (`#3155FF`) as the primary accent and warm amber (`#F6A623`) as a secondary highlight [2]. Headings use an editorial serif (the `font-display` class) and body text a neutral sans-serif [2]. This conveys a trustworthy, studious feel.
- **UI Patterns:** The code follows the UI spec in many cases: multi-step onboarding with a step indicator; filter sidebar + scholarship cards; split-pane essay view. Components like cards (`rounded-2xl bg-card p-4 shadow-sm`) and buttons (`rounded-full bg-primary`) are used consistently [3] [4]. Many Shadcn primitives (Card, Button, Tabs, Dialog, Badge, Skeleton, Toast) are prescribed [5] and some (e.g. cards, badges) are custom-implemented with the theme classes.
- **Strengths:** Key flows have clear layouts. The **onboarding wizard** is centered and step-driven [6]. The **matches page** has a two-column layout (filters + results) at larger breakpoints [7]. Scholarship cards cleanly display name, sponsor, amount, tags and deadlines [8]. The **homepage/dashboard** summarizes "Recommended/In-Progress/Applied" lists in well-separated columns. Altogether, the app already feels **calm**, **focused**, and **well-structured**.
- **Weaknesses:** However, many intended features are not yet built. The **"Why this fits you" dialog, Application Planner, Rubric self-grade, and enhanced essay editor** aren't implemented. Some UI elements lack polish: for example, there is no global `<AppShell>` header/nav as suggested, and focus styles are inconsistent (many inputs use generic focus without a clear ring) [5]. Minor inconsistencies in styling exist (e.g. some badges use `bg-secondary`, others use bordered chips). Route naming is also inconsistent (code uses `/scholarship/$id` whereas docs expect `/scholarships/:id`). These gaps suggest opportunities for refactoring and completing missing states.
- **MCP/Component Usage:** The current code uses Shadcn UI's themes (color classes like `bg-background`, `text-primary-foreground`) but often hand-rolls layouts instead of employing high-level components. For instance, card and filter panel layouts are done with basic `<div>`s using Tailwind classes rather than leveraging a provided `Card` or `Sheet` component. Overall, the foundation is solid (Tailwind + Shadcn) [1] [5], but some components (tabs, dialogs, advanced form controls) are only partially used, leaving visual patterns to be unified.

Competitive and Pattern Research

- **Scholarship Platforms:** Going Merry, Bold.org, and similar sites use **filterable card grids**. We should replicate a **left-side filter panel** (country, level, fields, toggles) and a main list of `ScholarshipCard` components [8]. Cards highlight scholarship name, sponsor, amount, deadline, and tags (level, field, demographics) – much like job listings on Indeed or LinkedIn (with badges for attributes) [8]. It's effective to surface eligibility info (e.g. "Undergraduate", "Women in

STEM") as colored badges or chips, and use hover lifts/accents to show interactivity [9]. A "Reset filters" action (as in the empty-state design) is also standard.

- **Wizards & Onboarding:** For multi-step forms, Duolingo's approachable wizard offers cues: a **progress indicator or stepper** at top with labels, and "Next/Back" buttons. We'll use a horizontal stepper (`<Tabs>` or custom step bar [6]) showing "Basics – Background – About You," with the current step highlighted. Buttons should be prominent (solid primary) and disabled until required fields are valid. This pattern reduces cognitive load by focusing on one section at a time [10]. (Shadcn has example Stepper/Progress components; MUI's Stepper is another model.)
- **Editor / Writing Assistance:** For the essay drafting workspace, models like **Grammarly or Notion** inspire a split-screen editor. We'll implement a left-side rich editor (or styled `<textarea>`) and a right-side panel with tabs for "Rubric" and "Themes" [11]. When providing AI suggestions (outline, rewrite), we might use modals or slide-over sheets (MUI's Dialog/Drawer or Shadcn's `<Dialog>`/`<Sheet>`). For revision flows, a side-by-side diff UI (current vs suggested text) is ideal, similar to an inline comment or merge diff; React Bits has animated panel components that could enhance this.
- **Component Library Patterns:** We'll heavily lean on the Shadcn primitives as recommended [5]. For example, use `<Card>` for scholarship listings and dashboard stats (with `rounded-2xl shadow-sm` styling [3]), `<Button>` variants for primary/secondary actions, `<Tabs>` for rubric/themes, `<Dialog>` or `<Sheet>` for "Why this fits" and revision panels. MUI's components provide similar patterns: e.g. MUI's `<Autocomplete>` could serve the "Fields of study" selector, and `<DataGrid>` for the dashboard table if needed. However, a custom table/list is simpler for v1. Overall, adopt proven UX patterns (filters + list, cards, wizard, editor with side info) to meet user goals efficiently.

Aesthetic Directions (Art Direction)

We recommend **three potential styles**, with the first as the primary candidate:

1. **"Scholar Studio" (Baseline – Calm & Trustworthy):** This is the current direction. Tone is *calm, professional, and encouraging*. Pair an editorial serif (e.g. Playfair Display) for headings with a neutral sans-serif (Inter/Source Sans) for body [2]. The semantic color palette is light and warm: **Page background** #FDF7EF, **surface** (cards) #FFFFFF, **text** #181220 (almost black) [2]. Use **Primary Accent** = deep blue #3155FF and **Secondary** = amber #F6A623 for buttons, links, and highlights [2]. Motion should be subtle: gentle fades and slides on dialogs/hover [12], no flashy animations, respecting reduced-motion preferences [12]. This direction feels *reliable and calm*, emphasizing clarity.

2. **"Bright Progress" (Warm & Energetic):** A livelier alternative. Tone is *upbeat and motivational*. Font pair could flip: a clean sans-serif headline (e.g. Poppins) with a friendly serif (or the same sans) for body, creating a modern vibe. Colors would be brighter: crisp white or very light gray background, **Primary** = vibrant teal or green (trust/building focus), **Secondary** = bright orange or coral. This injects energy suitable for young students. Motion would include small "pops" or accent color pulses on success (e.g. a green checkmark animation) and smooth expansions for sidebar filters. Use gradients/illustrative icons sparingly for a fresh look. However, it's riskier as it departs from the existing calm brand.

3. **"Modern Minimal" (Clean & Focused):** A sleek, monochrome style with a single strong accent. Tone is *efficient, no-nonsense*. Use a sans-serif for all text (e.g. Montserrat), with sparing use of serif for emphasis if needed. Primary = navy or charcoal blue, Secondary = a single highlight color (lime or

cyan). The UI would minimize shadows and use more whitespace. Motion is equally restrained: micro-transitions (100–150ms) on hovers [12], with all focus states clearly defined (thicker rings). This appeals to tech-savvy users, but may feel colder.

Chosen Direction – "Scholar Studio": We recommend sticking with the *existing calm theme*. It matches the mission ("coach"-like, trustworthy), and is already mostly implemented [2] [1]. It uses accessible, high-contrast colors and an academic aesthetic, which works well for an educational tool. The blue/amber palette suggests both *credibility* (blue) and *warmth/optimism* (amber). Its motion guidelines (subtle transitions and visible focus rings) are well-described in our docs [13] [12], and we should continue them to refine the feel. (The other directions can inform future theming or dark mode variations but the baseline meets our needs.)

Deliverable 1 – UX Principles

1. **Clarity Over Cleverness:** Always present *one clear next action*. The onboarding wizard and match list should emphasize the obvious step (e.g. a highlighted "Continue" or "View Details" button) [10] [6]. Avoid overwhelming screens – break flows into digestible parts (the multi-step profile form is a good example).
2. *Implications:* Wizard headers explicitly say "Step X of Y" [14]; matches page shows count of results; highlight primary CTA (e.g. "Start essay") on each screen.
3. **Transparent Guidance:** Show *why things happen*. Display eligibility criteria and match rationale. For instance, eligibility sections (country, GPA, demographics) should be visible on detail pages [15], and a "Why this fits you" dialog should explain fit scores. This demystifies the AI: users see *facts* (criteria, weights) not just a black-box result.
4. *Implications:* Include an "Eligibility" panel on the detail screen (as per docs [15]). Use clear labels (e.g. "Priority for: Women in STEM" vs "Only open to: …") so students understand filters. Provide contextual help text for toggles/checkboxes (e.g. "Optional – used only for targeted matches").
5. **User in Control:** AI suggestions must be optional and reviewable. The student's actions drive the flow (never auto-submit). For example, "Generate draft" should *present* an essay for review (not auto-apply it), and "Accept revision" merges changes only after user confirmation [16] [17].
6. *Implications:* Buttons like "Accept Revision" or "Generate Outline" open modals/dialogs where the user chooses, rather than auto-changing text. Show diffs side-by-side for rubric improvements [18]. Always allow "Keep original" on edits.
7. **Inclusivity and Transparency of Data:** Treat sensitive inputs (demographics) carefully. Demographic filters are optional and clearly explained (with "why" text). UX text should use respectful language and privacy-conscious defaults.
8. *Implications:* Label optional sections ("This is optional…") [19]. Do *not* hide or downplay eligibility info; instead show "Check official page for full details" to cover omissions [20].
9. **Accessibility:** Ensure keyboard navigation, focus visibility, and contrast. Every interactive element (buttons, links, tabs) must be reachable via Tab with a clear focus ring [21]. Text ratios should meet WCAG standards (our dark-on-light scheme already has high contrast [2]).
10. *Implications:* All buttons use Tailwind's focus-visible ring classes (as in code: `focus-visible:ring-2 ring-accent`) [21]. Ensure form fields have `aria-labels` if needed. Design components (Cards, Tabs) should have role/label where applicable.

(Additional principles might include "Supportive Tone" – use encouraging copy; "Resilience" – handle errors graciously – these guide copywriting and error states below.)

Deliverable 2 – Design System & Tokens

Color Tokens (Tailwind theme): We define semantic color names that map to Tailwind and Shadcn conventions:
- `page-bg`: `#FDF7EF` (off-white background) [2]
- `surface`: `#FFFFFF` (card/panel background) [22]
- `text-main`: `#181220` (primary text) [2]
- `text-muted`: `#6B6175` (secondary text) [22]
- `primary`: `#3155FF` (brand blue accent) [2]
- `primary-foreground`: `#FFFFFF` (text on primary buttons)
- `secondary`: `#F6A623` (amber accent) [2]
- `secondary-foreground`: `#181220`
- Semantic: `success: #2E7D32` (green), `warning: #FFB300` (amber), `error: #D32F2F` (red) [23].

Typography:
- **Fonts:** Define `fontDisplay`: Playfair Display or Fraunces (serif) for headings, and `fontSans`: Inter (sans-serif) for body [2] [24].
- **Sizes:** Use a modular scale. E.g. `h1 ~2.5rem (40px)`, `h2 ~2rem`, `h3 ~1.25rem` [25]. Body text at 1rem (16px) with `text-sm` on smaller screens. Ensure consistent `line-height` and `margin-bottom` on headings (e.g. `mb-3`).

Spacing & Layout: Use a spacing scale (Tailwind's default: 4px * 1,2,3…). For components:
- Default container padding: `px-4 sm:px-6 lg:px-8`.
- Gaps between sections: `gap-6` or `gap-8`.
- Consistent padding inside cards/forms (e.g. `p-4` to `p-6`).

Radii & Shadows:
- Border radius tokens: e.g. `rounded-md` for small buttons/inputs, `rounded-lg` for larger elements, `rounded-2xl` for cards/surfaces [3].
- Shadows: default shadow-sm (`shadow-sm`) on cards, `shadow-md` on hover [26]. Focus ring uses `ring-2 ring-offset-2`.

Motion: Define timing tokens (e.g. `motion-duration:150ms`, `ease-out`) for subtle animations. All interactive transitions should use these. For example, cards use `transition-all duration-150` on hover [27]. Dialogs/Sheets fade + slide in (e.g. `ease-in-out`, `200ms`). Honor `prefers-reduced-motion`.

Tailwind Config Example: In `tailwind.config.js`, extend theme colors and fonts:

```
// tailwind.config.js
module.exports = {
  theme: {
    extend: {
      colors: {
        'page': '#FDF7EF',
        'card': '#FFFFFF',
        'primary': '#3155FF',
        'accent': '#3155FF',
        'accent-foreground': '#FFFFFF',
        'secondary': '#F6A623',
        'secondary-foreground': '#181220',
        'muted': '#6B6175',
        'border': '#E0E0E0',
        'success': '#2E7D32',
        'warning': '#FFB300',
        'error': '#D32F2F',
      },
      fontFamily: {
        display: ['Playfair Display', 'serif'],
        sans: ['Inter', 'sans-serif'],
      },
      borderRadius: {
        lg: '1rem',
        xl: '1.5rem',
      },
      spacing: {
        '9': '2.25rem',
        '11': '2.75rem',
      },
      // (Extend shadows, transition timing as needed)
    },
  },
}
```

This aligns Tailwind's theme with the tokens above, enabling usage like `bg-page`, `text-primary`, `font-display`, etc. Shadcn's `useTheme()` can also reference these aliases.

Deliverable 3 – Component Inventory & MCP Mapping

Layout & Navigation

- `<AppShell>` **/ Header:** Custom component wrapping each page. It contains the logo (wordmark *GoGetScholarship*), desktop nav links (`Matches`, `Dashboard`), and a user menu. On mobile, hide links behind a hamburger triggering a `<Sheet>` (side drawer). *States:* Sticky header on scroll, mobile menu slide-in.

```
// Example: Header component
<header className="sticky top-0 z-50 bg-page p-4 shadow-sm">
  <div className="mx-auto flex items-center justify-between max-w-6xl">
    <Link to="/" className="font-display text-xl">GoGetScholarship</Link>
    <nav className="hidden md:flex space-x-4">
```

```
      <Link to="/matches" className="text-sm font-medium text-
foreground">Matches</Link>
      <Link to="/dashboard" className="text-sm font-medium text-
foreground">Dashboard</Link>
    </nav>
    <div className="md:hidden">
      {/* Hamburger icon triggers mobile menu sheet */}
    </div>
    <UserMenu />  {/* Avatar + dropdown */}
  </div>
</header>
```

- **Footer:** A simple text footer (`GoGetScholarship – prototype`) if needed (per UI guide) with muted styling.

Data Display

- **ScholarshipCard:** For lists (matches, dashboard rows) – use Shadcn `<Card>` or a div with `rounded-xl bg-card shadow-sm`. Contains: Title (`h3.font-display`), sponsor (small muted text), amount (badge style), deadline (with icon), tags row of badges (level, fields) and demographic chips (amber accent) [8] [28]. Example snippet:

```
<div className="group rounded-xl bg-card p-4 shadow-sm ring-1 ring-border
                transition hover:-translate-y-0.5 hover:ring-primary/50">
  <h2 className="font-display text-base">{scholarship.name}</h2>
  <p className="text-xs text-muted">{scholarship.provider}</p>
  <div className="mt-3 flex flex-wrap gap-1.5 text-[11px]">
    {scholarship.levelTags.map(tag => (
      <Badge key={tag} variant="outline">{tag}</Badge>
    ))}
    {scholarship.fieldTags.map(tag => (
      <Badge key={tag} variant="filled">{tag}</Badge>
    ))}
    {scholarship.demographicTags.map(tag => (
      <Badge key={tag} variant="secondary">{tag}</Badge>
    ))}
  </div>
  <div className="mt-2 flex justify-between items-center text-xs text-
muted">
    <span>Workload: {scholarship.workload}</span>
    <Button size="sm" variant="primary">View details</Button>
  </div>
</div>
```

Here `<Badge>` and `<Button>` are Shadcn components; the layout matches the UI spec [8].

- **Statistics / Dashboard Cards:** Small stat cards for totals (tracked, ready, in-progress) use `rounded-lg bg-card p-4 shadow` and numeric display. These can use Shadcn `<Card>` or `<Statistic>` pattern.

- **Table/List:** For the dashboard main list, a simple `<table>` or flex list with rows of applications. Could use MUI `<DataGrid>` for quick implementation, but a custom list (divs) is fine given complexity. Ensure header row is sticky if long.

Forms & Inputs

- **Inputs/Selects/Textareas:** Use Shadcn `<Input>`, `<Select>`, `<Textarea>`, `<Checkbox>`, `<RadioGroup>` [5]. Style: rounded-full or -lg, subtle border. Example:

```
<label className="text-xs font-medium text-muted">Field of Study</label>
<Select className="w-full" options={fieldOptions} placeholder="Search
fields..." />
```

- **Checkbox/Toggle:** For boolean filters. Use `<Checkbox>` or styled `<input type="checkbox">` with label; ensure label clickable.
- **Button Variants:**
- `primary`: solid accent (bg-primary, text-white, `rounded-full`) [29].
- `secondary`: outline (`border-primary text-primary`).
- `ghost/link`: minimal text style (blue underline for links).
  Show a spinner icon on loading states.

Feedback & Utility

- **Toasts:** Use Shadcn `<Toast>` for success/error messages. E.g. green header with check icon or red with alert icon, auto-dismiss after 3s [30]. Copy should be friendly (see States below).
- **Skeletons:** Shadcn `<Skeleton>` component for loading placeholders [31]. For example, skeleton cards on `/matches` and grey blocks for detail sections before data loads.
- **Dialog/Sheet:** Use `<Dialog>` for modal overlays ("Why this fits you"), and `<Sheet>` (side drawer) for mobile menus or the revision panel (rubric improvements) [16]. E.g.:

```
<Dialog open={openExplain} onOpenChange={setOpenExplain}>
  <DialogContent>
    <DialogTitle>Why this fits you</DialogTitle>
    <ul className="mt-2 list-disc list-inside text-sm">
      <li>Strength: Your GPA meets the scholarship’s requirement</li>
      <li>Note: Needs stronger leadership examples.</li>
    </ul>
  </DialogContent>
</Dialog>
```

- **Chips/Badges:** Shadcn `<Badge>` for the small tag-like info. Use `variant="outline"` or color props to match meaning (blue for general, amber for demographics, green for "Ready"). For example, an “In progress” status badge could be `<Badge variant="secondary">In Progress</Badge>` [32].

- **Progress & Loaders:** Shadcn `<Progress>` for readiness meter (dashboard) and spinners inside buttons or dialogs.

Editor & AI Assist

- **Essay Editor:** A full-width `<Textarea>` or rich text editor (like TipTap or Editor.js) for the essay content. Show a prompt header above it. Save status indicator (e.g. “Saved • 2 min ago”) can be a tiny badge or text element. Example:

```
<div className="flex flex-col">
  <label className="text-sm font-semibold">Prompt (500 words max)</label>
  <Textarea className="min-h-[260px]" placeholder="Start writing..."
value={essay} onChange={...} />
  <div className="mt-2 flex items-center justify-between text-[11px] text-
muted">
    <button type="button">Generate outline</button>
    <span>Saved · 1 min ago</span>
  </div>
</div>
```

- **AI Assist Actions:** Buttons like “Generate outline”, “Draft from bullets”, “Rewrite” trigger API calls. If a feature requires extra input (e.g. bullets), show a small inline form or modal before calling the API.

- **Rubric/Revision UI:** After grading, the “Rubric” tab is a table of criteria (e.g. “Leadership – 3/5”) [33]. Each row has an “Improve” button opening a right-side sheet: a diff view (two columns of text). We can create a `<RevisionSheet>` component with two `<pre>` blocks side-by-side. Example stub:

```
<div className="flex gap-2 text-[11px] leading-snug">
  <pre className="w-1/2 bg-muted/10 p-2 rounded">Original paragraph
text...</pre>
  <pre className="w-1/2 bg-muted/10 p-2 rounded bg-accent/10">Suggested
revised text...</pre>
</div>
<div className="mt-2 flex justify-end gap-2">
  <Button variant="primary">Accept revision</Button>
  <Button variant="ghost">Keep original</Button>
</div>
```

- **AI Loading/Error States:** Use inline spinners (button `<Spinner>`) and disallow interaction while waiting.

Deliverable 4 – Layout & Responsive Patterns

Onboarding Wizard (`/onboarding`)

- **Desktop:** A single centered card (`max-w-xl mx-auto`) with a horizontal stepper at the top showing "Basics – Background – About You" [6] . Each step's fields are inside the card with ample padding. The card uses `rounded-2xl bg-card shadow-sm` [34] . Primary CTA ("Continue"/"Save & See Matches") is at the bottom in a button row.
- **Mobile (<md):** Full-width stack. The stepper can wrap or convert to a vertical list. Buttons remain fixed at bottom if needed. (UI note: mobile layout is single-column by default.)

Matches List (`/matches`)

- **Desktop (`lg+`):** Two-column layout. Left sidebar (`w-[260px] bg-card p-4 shadow-sm`) holds filter controls [35] . Right main area (`flex-1`) has the title and a grid of scholarship cards [36] .
- **Mobile (<lg):** Collapse filters into a toggled `<Sheet>` panel (triggered by a "Filters" button) [37] . Scholarship cards span full width (`mx-auto max-w-md`).

Scholarship Detail + Sidebar (`/scholarship/:id`)

- **Desktop (`md+`):** Two-column layout [38] . Left column (≈60%) is the main content: scholarship title, provider, amount, deadline, eligibility (e.g. country, level) and full description [39] . Right column (≈40%) is a sticky sidebar with "What they care about" (themes & weights) and "Application components" checklists [40] . Action buttons ("Start essay", "Plan", "Why fits") float at bottom or fixed in the sidebar.
- **Mobile:** Stack all sections vertically. Title → provider → summary → eligibility → components → actions. "Why this fits" still appears in a dialog overlay. The draft generation section becomes full-width under the detail, with the student summary form and resulting draft below.

Essay Drafting Split View (`/scholarship/:id/essay`)

- **Desktop (`md+`):** Fixed top bar (scholarship name, prompt, word limit, word count indicator). Below, a **split view**: left panel (2/3 width) for the essay editor (prompt at top, then `<Textarea>` and action buttons), right panel (1/3 width) with two tabs [11] . The "Rubric" tab lists criteria with scores and "Improve" buttons [41] ; the "Themes" tab shows theme chips. The side panel is sticky so it remains visible during scroll.
- **Tablet:** Similar split or stacked if space is tight; ensure word count and edit buttons are always visible.
- **Mobile:** Use a tabbed approach [42] . Show either Editor or Guidance (Rubric/Themes) at one time. The rubric results (if graded) appear below the editor, collapsible. Buttons like "Grade rubric" and "Improve this" are full-width.

Dashboard Summary (`/dashboard` or homepage)

- **Desktop:** Page title "Your applications" at top. A row of stat cards (Total Tracked, Ready, In Progress) beneath it [43] . Below, a table or list of tracked scholarships with columns: Scholarship (name + sponsor), Deadline, Status badge, Readiness meter, and an "Open" button [44] .

- **Lower Section:** “Low extra work suggestions” as a horizontally scrollable card list [45]; on desktop show 3 cards in view with arrow navigation.
- **Mobile:** Collapse stat cards into a vertical stack. Show tracked scholarships as a vertical list of cards instead of a table [46]. Suggestion cards stack or allow swipe.

Breakpoints: We use Tailwind’s mobile-first breakpoints: up to `sm` = stacked layouts, `md` (≥768px) for two-column detail & essay split, `lg` (≥1024px) for filters sidebar. Headings/tabs scale up at `sm`/`md` as needed [47] [37]. Sticky elements: main header stays on scroll, the essay sidebar is `sticky top-20` on desktop, and the “Next best action” card on the home page is fixed width on large screens.

Deliverable 5 – States (Empty, Loading, Error)

- **Profile Input (Onboarding):**
- *Loading:* Disable inputs and show a spinner inside the “Save” button when submitting. Use subtle overlay or skeleton for the form card if profiling data is being fetched.
- *Error:* Inline error text near the field or at top (“Please fill in at least your country and level.” as in the code [48]). Use a red text color (`text-destructive`). A toast (“Failed to save profile. Please try again.”) can appear on network failures.

- *Empty (skipped):* On optional steps, show a note (“You can skip this step. Your profile is still usable without it.”) as in the UI guide [49].

- **Matches List:**

- *Loading:* Show 4–6 skeleton cards in place of results [31] (e.g. `<Skeleton>` for each card’s title, text, and tags). Disable filters while loading.
- *Empty (no matches):* Display a centered message with an icon: “No scholarships match your current filters.” and a “Reset filters” button [50]. This mirrors the UI spec. Tone is explanatory, not blaming: e.g. “Oops, no results for that combination. Try changing the filters.”

- *Error:* If the match API fails, show a full-width alert box above the list: “We couldn’t load your matches right now. Please refresh or try again.” Include a retry button that refetches (using a `<Toast>` or inline banner).

- **Scholarship Detail:**

- *Loading:* Show skeleton blocks for each section (title bar, eligibility list, “What they care” card) [31].
- *Empty:* If a scholarship has no personality/rubric data yet, show an info box: “Rubric not configured for this scholarship yet.” and disable “Grade essay” [51]. (Copy example: “You can still draft your essay, but no rubric is available yet.”)

- *Error:* Inline alert (`bg-error/10 text-error`) with message from server (e.g. “Failed to load scholarship details.”) and a “Back to matches” link. For AI operations: on draft or explain-fit failure, show a toast: “We couldn’t get help from the AI right now. Your work is safe – try again in a moment.” [52]. This reassures the user.

- **Essay Workspace:**

- *Loading:* While generating a draft or outline, disable the form and show a spinner in the "Generate draft" button. For autosave, show "Saving…" status.
- *Empty Draft:* Initially (no draft yet), show placeholder text in the draft panel: "Your draft will appear here once generated. You can copy it into your editor." (As in the current UI example [53].) Use a muted font.
- *Error:* If draft API fails, display a small red text under the button: e.g. "Sorry, the AI failed to write a draft. Please try again." Keep the user's input intact.
- *Global Errors:* For any persistent issue, use a toast or modal summarizing the problem ("Network error, try refreshing") rather than leaving blank areas.

All error/empty states should use the brand voice: helpful, not accusatory. Use icons (alert-triangle for errors, neutral folder/search icon for empty) and maintain the same card/grid layout.

Deliverable 6 – Accessibility & Motion Guidelines

- **Keyboard Navigation:** Ensure all interactive elements are keyboard-accessible. Use `tabindex="0"` on custom elements and proper HTML form controls. The tab order should follow the visual order (e.g. form inputs in sequence, filters then content). Test that dropdowns/sheets can open via keyboard (Enter/Space) and close with Esc.

- **Focus States:** Every focusable component must have a visible focus ring. Tailwind's `focus-visible:ring-2 focus-visible:ring-offset-2 focus-visible:outline-none` has been used (see code) [21]. Check that the ring contrast is clear against backgrounds. For example, primary buttons get an accent-blue ring; card links get an underline or outline.

- **Contrast:** Use WCAG AA contrast or better. Our text colors (#181220 on #FDF7EF ~ 16:1) are very high contrast. Ensure any accent/disabled text (e.g. `#6B6175` on white) is ≥4.5:1. Buttons and badges should meet contrast rules (e.g. white text on #3155FF is good). Use tools to verify all key text pairs.

- **Reduced Motion:** For motion guidelines: keep animations subtle (150–200ms) and provide no-critical animations only. As in our spec [27], cards and buttons use `transition-all duration-150`. Dialogs/Sheets fade/slide on open (avoid long/easing loops). Always respect `prefers-reduced-motion`: disable or simplify animations if the user has that setting on.

- **ARIA & Labels:** Use ARIA labels where needed (e.g. icon buttons like <CalendarDays> should have `aria-label="Deadline"`). All form fields have `<label>` elements. For dynamic content (like toasts), use `role="status"` or `alert` with appropriate `aria-live`.

- **Motion Severity:** No fast or large movements. Hover lifts (`hover:-translate-y-0.5`) are minimal. Avoid layout shifts (reserve space for spinners/loading to prevent jank). Test with reduced motion on to ensure usability.

Deliverable 7 – Implementation Roadmap

We propose a two-phase rollout following the **Core vs Stretch** classification [54] [55]. Below are tasks with rough difficulty (1=easy, 5=hard) and UX impact (1=low, 5=high):

- **Phase 1 (Core Builds):**
- **Onboarding Wizard (V1):** *Difficulty 2, Impact 5.* Complete `/onboarding` route implementation with all steps, validation, and API save. *(Files:* `src/routes/onboarding.tsx`*.)*
- **Matches Page & Filters:** *Difficulty 3, Impact 5.* Build filter sidebar (country, level, fields, toggles) and scholarship card grid. Wire up API calls or mock data. *(Files:* `src/routes/matches.tsx`*, plus a* `<ScholarshipCard>` *component.)*
- **Scholarship Detail View:** *Difficulty 4, Impact 5.* Implement `/scholarship/:id` details layout (two-column) with about and eligibility sections [38]. Include “Application components” checklist and action buttons. *(Files:* `src/routes/scholarship/$id.tsx`*.)*
- **Essay Workspace (Drafting):** *Difficulty 4, Impact 5.* Under detail page (or a sub-route/tab), add the essay editor split-view with prompt, textarea, and AI controls [56]. Implement draft generation form and display area (with explanation text). *(May split into* `EssayWorkspace.tsx` *or similar.)*
- **Basic Planner (Task List):** *Difficulty 3, Impact 4.* Inline in the detail page (or modal), list tasks derived from requirements. Use checkboxes and date pickers. *(Files: possibly a new* `ApplicationPlan.tsx` *and updates to* `$id.tsx`*.)*
- **Dashboard Page:** *Difficulty 3, Impact 4.* Route `/dashboard` showing stats cards and list of tracked apps. Implement the “Low extra work” suggestions row (horizontally scrollable). *(Files:* `src/routes/dashboard.tsx` *or reuse* `index.tsx`*.)*

- **Global Layout:** *Difficulty 2, Impact 5.* Create an `AppShell` component with header/nav that wraps all pages. Ensure mobile menu (Sheet) and user profile menu are functional. *(Files:* `src/AppShell.tsx`*, modify root route layout.)*

- **Phase 2 (Polish & Stretch):**

- **“Why This Fits” Dialog:** *Difficulty 3, Impact 4.* In detail page, add a Dialog (triggered by a button) showing AI-explained fit bullet points [57] [58]. Call `/api/scholarships/:id/explain-fit`. *(Update:* `ScholarshipPage` *component.)*
- **Rubric Grading UI:** *Difficulty 4, Impact 5.* Implement the “Grade against rubric” flow: call `/api/essays/:id/grade`, populate the rubric tab with per-criterion scores and feedback [59] [60]. Then add the revision sheet UI (2-column diff) for each criterion. *(Files: new components under EssayWorkspace.)*
- **Planner Enhancements:** *Difficulty 3, Impact 3.* Enable editing of due dates (popover date-picker) and optimistic checkbox updating. Possibly use a drag-and-drop library for reordering (stretch). *(Files:* `ApplicationPlan.tsx`*, integrating a* `<Calendar>` *if needed.)*
- **Accessibility & Focus Tuning:** *Difficulty 2, Impact 4.* Audit keyboard flows and color contrast. Add missing `aria-` attributes and adjust CSS as needed (focus rings, alt texts). *(Files: everywhere components are interactive.)*
- **Error/Empty State Polish:** *Difficulty 2, Impact 3.* Ensure all states have proper copy and visuals (as outlined above). Add Toast notifications for errors [52], and skeleton placeholders.

- **Dark Mode (Stretch):** *Difficulty 3, Impact 2.* (If time) Add dark theme tokens and a toggle. Would involve extending tailwind config and toggling class on `<body>`.

Each task should include testing and review for UX consistency. The difficulty ratings assume familiarity with the existing code (TypeScript, TanStack Router). The referenced files are likely to be touched or created as noted. This roadmap aligns with our core features list [54] and allows staging advanced features for later.

Sources: Our plan is grounded in the current codebase and UI/UX docs (e.g. layout specs in [12], [14], and [44]) and follows established design patterns. Citations above point to specific guidance from the project's UX/UI requirements [2] [6] [8] and architecture notes [1].

---

[1] GoGetScholarship – Technical Architecture Report.pdf
file://file-7E57bJfzWJxxsthdWqytz6

[2] [10] [15] [16] [18] [20] [33] [38] [39] [40] [57] ux-requirements.md
https://github.com/krgrss/GoGetScholarship/blob/7e48b06f8b5e259b54c02634012b49863c7e3c67/docs/ux-requirements.md

[3] [5] [6] [7] [8] [9] [11] [12] [13] [21] [22] [23] [24] [25] [26] [27] [29] [30] [31] [37] [42] [43] [44] [45] [46] [47] [50] [51] [52] [54] [55] [56] [58] [59] [60] ui-requirements.md
https://github.com/krgrss/GoGetScholarship/blob/7e48b06f8b5e259b54c02634012b49863c7e3c67/docs/ui-requirements.md

[4] [28] [32] [35] [36] matches.tsx
https://github.com/krgrss/GoGetScholarship/blob/7e48b06f8b5e259b54c02634012b49863c7e3c67/src/routes/matches.tsx

[14] [19] [34] [48] [49] onboarding.tsx
https://github.com/krgrss/GoGetScholarship/blob/7e48b06f8b5e259b54c02634012b49863c7e3c67/src/routes/onboarding.tsx

[17] [41] [53] $id.tsx
https://github.com/krgrss/GoGetScholarship/blob/7e48b06f8b5e259b54c02634012b49863c7e3c67/src/routes/scholarship/$id.tsx