# Seline UX Overhaul 2026

2026-02-19

# Seline UX Overhaul 2026 — Navigation & Implementation Checklist

*Meeting date: 2026-02-19 | All plans verified against codebase | Gap analysis incorporated*

## Quick Links

| # | Document | Topic | Priority | Owner |
|---|---|---|---|---|
| 01 | Create Agent Modal | Replace full-page wizard with inline modal | P1 | Umut |
| 02 | Wizard Simplification | Remove 5 steps, merge Preview+Success | P2 | Umut |
| 03 | Default Tools | Pre-select all utility tools for new agents | P3 | Umut |
| 04 | Agent Card Cleanup | Replace 7-button row with `•••` dropdown | P4 | Umut |
| 05 | Agent Duplicate | New API + UI to copy agents | P5 | Umut |
| 06 | Workflow Sections | Add Workflows / Agents section headers | P6 | Umut |
| 07 | Slash Skill Picker | `/` in chat input opens skill browser | P7 | Umut |
| 08 | Deferred Items | Node graph, marketplace, agent onboarding | Future | Both |
| 09 | Launch Plan | Mar 4 Product Hunt launch strategy | Now | Both |

**Implementation order: 03 → 01 → 02 → 04 → 05 → 06 → 07** (doc 03 unlocks doc 01; doc 04 unlocks doc 05)

## One-Line Task Reference

### Doc 03 — Default Tools *(do this first)*

- [ ] `lib/characters/templates/resolve-tools.ts` — Add `export` to `ALWAYS_ENABLED_TOOLS` and `UTILITY_TOOLS`
- [ ] `lib/characters/templates/resolve-tools.ts` — Add `delegateToSubagent` to `UTILITY_TOOLS`
- [ ] `lib/characters/templates/resolve-tools.ts` — Add and export `DEFAULT_ENABLED_TOOLS = [...ALWAYS_ENABLED_TOOLS, ...UTILITY_TOOLS, "webSearch"]`

- [ ] `lib/characters/templates/seline-default.ts` — Import `DEFAULT_ENABLED_TOOLS` and use as base for `SELINE_STATIC_TOOLS`
- [ ] `components/character-creation/terminal-wizard.tsx` — Change `enabledTools: ["docsSearch"]` → `enabledTools: DEFAULT_ENABLED_TOOLS`
- [ ] `components/character-creation/terminal-pages/capabilities-page.tsx` — Add `delegateToSubagent` to `BASE_TOOLS` array
- [ ] `components/character-creation/terminal-pages/capabilities-page.tsx` — Change prop default `initialEnabledTools = ["docsSearch"]` → `= DEFAULT_ENABLED_TOOLS`
- [ ] `components/character-creation/terminal-pages/capabilities-page.tsx` — Remove `tavilyKey` from `webSearch` dependencies
- [ ] `components/character-picker.tsx` — Remove `tavilyKey` from `webSearch` dependencies (inline tool editor)

### Doc 01 — Create Agent Modal *(depends on doc 03)*

- [ ] `components/character-creation/create-agent-modal.tsx` — **Create new file** with Quick Create + From Template tabs
- [ ] `components/character-picker.tsx` line ~1628 — Replace `<Link href="/create-character">` with `<button onClick={() => setCreateModalOpen(true)}>`
- [ ] `components/character-picker.tsx` line ~1843 — Replace empty-state `<Link>` with button opening the same modal
- [ ] `components/character-picker.tsx` — Add `const [createModalOpen, setCreateModalOpen] = useState(false);`
- [ ] `components/character-picker.tsx` — Add `<CreateAgentModal open={createModalOpen} onOpenChange={setCreateModalOpen} onCreated={() => loadCharacters()} />`
- [ ] `locales/en.json` — Add `picker.createModal.*` namespace (12 keys)
- [ ] `locales/tr.json` — Add same `picker.createModal.*` namespace in Turkish

### Doc 02 — Wizard Simplification

- [ ] `components/character-creation/terminal-wizard.tsx` — Update `PROGRESS_PAGES` to `["identity", "capabilities", "mcpTools"]`
- [ ] `components/character-creation/terminal-wizard.tsx` — Change post-identity navigation from `navigateTo("knowledge")` → `navigateTo("capabilities")`
- [ ] `components/character-creation/terminal-wizard.tsx` — Remove render blocks for `knowledge`, `embeddingSetup`, `vectorSearch`, `preview`
- [ ] `components/character-creation/terminal-wizard.tsx` — Remove imports: `KnowledgeBasePage`, `EmbeddingSetupPage`, `VectorSearchPage`, `PreviewPage`, `UploadedDocument`
- [ ] `components/character-creation/terminal-wizard.tsx` — Remove `documents` from `WizardState` and `initialState`
- [ ] `components/character-creation/terminal-wizard.tsx` — Remove handler functions: `handleKnowledgeSubmit`, `handleVectorSearchSubmit`, `handleEmbeddingSetupSubmit`,

`handleEmbeddingSetupSkip`

- [ ] `components/character-creation/terminal-wizard.tsx` — Add `handleFinalizeAgentWithTools(tools: string[])` function (accepts tools directly to avoid setState async race)
- [ ] `components/character-creation/terminal-wizard.tsx` — Update `handleCapabilitiesSubmit` to call `handleFinalizeAgentWithTools` when no MCP servers
- [ ] `components/character-creation/terminal-wizard.tsx` — Update `MCPToolsPage.onComplete` to call `handleFinalizeAgentWithTools(state.enabledTools)` (was: `navigateTo("preview")`)
- [ ] `components/character-creation/terminal-wizard.tsx` — Update `CapabilitiesPage.onBack` to `navigateTo("identity", -1)` (was: routing through vectorSearch/embeddingSetup)
- [ ] `components/character-creation/terminal-wizard.tsx` — Narrow i18n cast on line ~122 to remove old page ids
- [ ] `components/ui/wizard-progress.tsx` — Remove 4 entries: `knowledge`, `embeddingSetup`, `vectorSearch`, `preview`
- [ ] `components/ui/wizard-progress.tsx` — Remove now-unused icon imports (`BookOpen`, `Database`, `Eye`)
- [ ] `components/character-creation/terminal-pages/capabilities-page.tsx` — Update `onSubmit` signature to accept optional `vectorConfig`
- [ ] `components/character-creation/terminal-pages/capabilities-page.tsx` — Add "Advanced Options" collapsible section with vector search toggle
- [ ] `components/character-creation/terminal-pages/success-page.tsx` — Add `tagline` and `enabledTools` props; display agent summary
- [ ] `components/character-creation/terminal-pages/success-page.tsx` — Remove "Configure Another Agent" button
- [ ] `locales/en.json` + `locales/tr.json` — Remove orphaned keys (knowledgeBase, embeddingSetup, vectorSearchPage, preview sections)
- [ ] `locales/en.json` + `locales/tr.json` — Add new keys for SuccessPage summary + CapabilitiesPage advanced section

### Doc 04 — Agent Card Cleanup *(do before doc 05)*

- [ ] `components/character-picker.tsx` — Add `MoreHorizontal, Copy, Puzzle` to lucide-react import (NOT `MoreHorizontalIcon`, `CopyIcon`, `PuzzleIcon`)
- [ ] `components/character-picker.tsx` — Add `DropdownMenu, DropdownMenuContent, DropdownMenuItem, DropdownMenuSeparator, DropdownMenuTrigger` import
- [ ] `components/character-picker.tsx` line ~1657 — Add `relative group` to `AnimatedCard` className (both are required)
- [ ] `components/character-picker.tsx` — Add `•••` `DropdownMenu` with 8 items inside card header
- [ ] `components/character-picker.tsx` — Use `onSelect` not `onClick` on all `DropdownMenuItem`s
- [ ] `components/character-picker.tsx` — Add `onClick={(e) => e.stopPropagation()}` to `DropdownMenuContent` (not just trigger)

- [ ] `components/character-picker.tsx` — Use `openDeleteDialog(character)` not `handleDelete(char.id)` for Delete item
- [ ] `components/character-picker.tsx` — Use `openMcpToolEditor(character)` not `openMcpEditor(char)` for MCP item
- [ ] `components/character-picker.tsx` — Add Dashboard item: `router.push("/dashboard")`
- [ ] `components/character-picker.tsx` — Add stub `handleDuplicate`: `toast("Duplicate coming soon")`
- [ ] `components/character-picker.tsx` — Remove entire old bottom button row (lines ~1729-1793)
- [ ] `components/character-picker.tsx` `AgentCardInWorkflow` component — Apply same `relative group` + `DropdownMenu` treatment
- [ ] `components/character-picker.tsx` — `•••` trigger opacity: `opacity-40 group-hover:opacity-100` (not `opacity-0`) for mobile visibility

### Doc 05 — Agent Duplicate

- [ ] `app/api/characters/[id]/duplicate/route.ts` — **Create new file** with `POST` handler
- [ ] Use `requireAuth + getOrCreateLocalUser` pattern (NOT `getLocalUser()`)
- [ ] Insert into `characters` table copying `metadata` JSON blob (NOT individual columns — they don't exist)
- [ ] Clear `workflowId`, `workflowRole`, `inheritedResources` from duplicated metadata
- [ ] Copy `agentSyncFolders` with correct drizzle field names: `recursive` (NOT `is_recursive`), `includeExtensions` (NOT `file_extensions`)
- [ ] Set `inheritedFromWorkflowId: null` and `inheritedFromAgentId: null` on copied folders
- [ ] Copy `agent_plugins` rows from join table
- [ ] Copy `character_images` rows (metadata only — same file path on disk)
- [ ] Add ownership check: `if (source.userId !== dbUser.id) → 403`
- [ ] Use `await params` (Next.js 15 async params)
- [ ] Strip existing ` (copy)` suffix before appending: `source.name.replace(/ \(copy\)$/, "") + " (copy)"`
- [ ] `components/character-picker.tsx` — Replace stub `handleDuplicate` with real: `resilientPost(…, {retries: 0})` → `loadCharacters()` → `toast.success("Agent duplicated")`

### Doc 06 — Workflow Sections

- [ ] `components/character-picker.tsx` line ~1369 — Upgrade existing `<h3>` to `<h2>` flex-row layout with divider
- [ ] `components/character-picker.tsx` — Inline `+ New Workflow` button calls `setWorkflowCreatorOpen(true)` (NOT `setShowCreateWorkflow`)
- [ ] `components/character-picker.tsx` — Disable `+ New Workflow` button when `allStandaloneCharacters.length === 0`
- [ ] `components/character-picker.tsx` — Remove old standalone "Create Workflow" button block (lines ~1352-1364)

- [ ] `components/character-picker.tsx` — Add "Agents" section header `<h2>` before the agent grid
- [ ] `components/character-picker.tsx` — Show workflows section header even when `filteredWorkflowGroups.length === 0` (with empty-state message)
- [ ] `components/character-picker.tsx` — Keep search bar in current position (or rename label to "Search agents & workflows...")
- [ ] `components/character-picker.tsx` — Replace `confirm()` for `removeSubagentFromWorkflow` with `<AlertDialog>`
- [ ] `components/character-picker.tsx` — Replace `confirm()` for `deleteWorkflowGroup` with `<AlertDialog>`
- [ ] `components/character-picker.tsx` — Add deleted-initiator placeholder: `{!initiator && <Badge variant="destructive">Agent deleted</Badge>}`
- [ ] `locales/en.json` + `locales/tr.json` — Add `agents.sectionTitle: "Agents"`

### Doc 07 — Slash Skill Picker

- [ ] `components/assistant-ui/thread.tsx` — Add state: `showSkillPicker`, `skillPickerQuery`, `selectedSkillIndex`
- [ ] `components/assistant-ui/thread.tsx` — Detect `/` using `inputValue.slice(0, cursorPosition)` (NOT full string)
- [ ] `components/assistant-ui/thread.tsx` — Load skills: `GET /api/skills?characterId={character.id}&status=active` (MUST include `status=active`)
- [ ] `components/assistant-ui/thread.tsx` — Get `character.id` from `useCharacter()` context (NOT a prop)
- [ ] `components/assistant-ui/thread.tsx` — Gate fetch: `if (!character?.id || character.id === "default") return`
- [ ] `components/assistant-ui/thread.tsx` — Integrate keyboard handler into existing delegation chain (after FileMentionAutocomplete handler)
- [ ] `components/assistant-ui/thread.tsx` — Handle both `Enter` AND `Tab` as selection keys
- [ ] `components/assistant-ui/thread.tsx` — On select: insert `"Run the {skill.name} skill "` (NOT `/run skillName`)
- [ ] `components/assistant-ui/thread.tsx` — Picker UI: place OUTSIDE `ComposerPrimitive.Root` (same placement as FileMentionAutocomplete)
- [ ] `components/assistant-ui/thread.tsx` — Use `onMouseDown + e.preventDefault()` on picker buttons (NOT `onClick` — avoids textarea blur race)
- [ ] `components/assistant-ui/thread.tsx` — Show "needs input" badge for skills with `inputParameters`
- [ ] `components/assistant-ui/thread.tsx` — Two empty states: "no skills at all" vs "no skills match query"

---

## Pre-Launch Checklist (Umut)

- [ ] Apple Developer enrollment + Mac signing + notarize DMG
- [ ] CONTRIBUTING.md added to repo
- [ ] README refreshed with screenshots, topics, GIF demo
- [ ] All 7 features implemented and tested
- [ ] Demo video recorded (60-90 sec)
- [ ] 5 short-form wow-moment clips recorded
- [ ] HN "Show HN" post written

## Pre-Launch Checklist (Duhan)

- [ ] Discord server created and set up
- [ ] Landing page with email capture live
- [ ] 200+ pre-launch emails collected
- [ ] Product Hunt page draft ready (12:01 AM PST Mar 4)
- [ ] Comparison table and description corrected (vs Open WebUI/AnythingLLM/Jan.ai)
- [ ] 5 newsletter press kit emails sent
- [ ] 5 influencer DMs sent
- [ ] Reddit accounts established (r/LocalLLaMA activity)
- [ ] 10-20 beta testers found and briefed
- [ ] Real estate analyzer demo GIF recorded
- [ ] 50-plugin marketplace list in marketplace.json format

## Architecture Quick Reference

| Where things live | Path |
| --- | --- |
| Agent templates | `lib/characters/templates/` |
| Tool definitions | `lib/ai/tool-registry/tool-definitions.ts` |
| Character API routes | `app/api/characters/` |
| Skills API | `app/api/skills/route.ts` |
| Workflow API | `app/api/workflows/` |
| Settings API | `app/api/settings/route.ts` |
| Chat input | `components/assistant-ui/thread.tsx` |
| Home page (character picker) | `components/character-picker.tsx` |
| Agent creation wizard | `components/character-creation/terminal-wizard.tsx` |
| DB schema | `lib/db/sqlite-character-schema.ts` |
| i18n English | `locales/en.json` |
| i18n Turkish | `locales/tr.json` |

## Key Correctness Notes

| Mistake to avoid | Correct version |
|---|---|
| `getLocalUser()` in API routes | `requireAuth(req)` + `getOrCreateLocalUser(userId, ...)` |
| `params.id` in Next.js 15 routes | `const { id } = await params` |
| `refetchCharacters()` | `loadCharacters()` |
| `toast({ description: "..." })` | `toast("...")` or `toast.success("...")` |
| `handleDelete(char.id)` | `openDeleteDialog(character)` |
| `openMcpEditor(char)` | `openMcpToolEditor(character)` |
| `setShowCreateWorkflow(true)` | `setWorkflowCreatorOpen(true)` |
| `MoreHorizontalIcon` | `MoreHorizontal` (no Icon suffix) |
| `PuzzleIcon` / `CopyIcon` | `Puzzle` / `Copy` |
| `is_recursive` in agentSyncFolders | `recursive` (drizzle field name) |
| `file_extensions` in agentSyncFolders | `includeExtensions` |
| `data.id` from template create | `data.characterId` |
| `data.name` from quick-create | `data.agent.name` |
| POST body: `{ name, purpose, ... }` (flat) | `{ character: { name }, metadata: { purpose, ... } }` (nested) |
| `resilientFetch({ method: "POST" })` | `resilientPost(url, body, options)` |
| `onClick` on DropdownMenuItem | `onSelect` |
| `/run skillName` in chat | `"Run the skillName skill"` |
| Regex on full inputValue | `inputValue.slice(0, cursorPosition)` |

# 01 — Create Agent Modal (Replace Full-Page Wizard Entry)

## Problem

Clicking "Create New Agent" on the home page navigates away to `/create-character` — a 9-step full-page terminal wizard. This is a massive commitment for a new user who just wants to try the app.

**Meeting quote:**

> *"When you click on create an agent, it should open it in a pop-up modal where you choose the pre-made models and see them very fast without going to creating agent screen." "Mostly you should be ready to chatting after this."*

---

## Current Behavior

```
Home page (/)
    |
    ▼
[Create New Agent card]  ← is a <Link href="/create-character">
    |
    ▼ (full page navigation)
/create-character ← 9-step wizard
    Identity → Knowledge → Embedding → VectorSearch → Capabilities → MCP → Preview → Success
```

---

## Target Behavior


## Dependency: Implement Doc 03 First

`DEFAULT_ENABLED_TOOLS` used in the Quick Create flow does not yet exist in the codebase. Doc 03 must be implemented before this modal to export that constant from `lib/characters/templates/resolve-tools.ts`.

## Files to Create / Modify

### New File: `components/character-creation/create-agent-modal.tsx`

```
"use client";

import { useState, useEffect } from "react";
import { useRouter } from "next/navigation";
import Link from "next/link";
import { Loader2 } from "lucide-react";
import { Dialog, DialogContent, DialogHeader, DialogTitle } from "@/components/ui/dialog";
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";
import { Button } from "@/components/ui/button";
import { toast } from "sonner";
import { resilientPost } from "@/lib/utils/resilient-fetch";
import { DEFAULT_ENABLED_TOOLS } from "@/lib/characters/templates/resolve-tools";

interface CreateAgentModalProps {
  open: boolean;
  onOpenChange: (open: boolean) => void;
  onCreated: () => void;
}

interface Template {
  id: string;
  name: string;
  tagline: string;
  category?: string;
}

export function CreateAgentModal({ open, onOpenChange, onCreated }: CreateAgentModalProps) {
  const router = useRouter();
  const [activeTab, setActiveTab] = useState<"quick" | "template">("quick");
  const [inputValue, setInputValue] = useState("");
  const [isLoading, setIsLoading] = useState(false);
  const [templates, setTemplates] = useState<Template[]>([]);
  const [error, setError] = useState<string | null>(null);

  // Load templates on mount
  useEffect(() => {
    if (!open) return;
    fetch("/api/characters/templates")
```

```
      .then((r) => r.json())
      .then((data) => {
        if (data?.templates) setTemplates(data.templates);
      })
      .catch(() => {/* fail silently */});
  }, [open]);

  // Quick Create flow — 2-step:
  // Step 1: POST /api/characters/quick-create { concept } → { success, agent: { name, tagline, purpose } }
  // Step 2: POST /api/characters { character: {name, tagline}, metadata: {purpose, enabledTools} }
  //         → { character: { id } }  (status: "active" by default)
  // Step 3: router.push(`/chat/${id}`)
  const handleQuickCreate = async () => {
    if (inputValue.trim().length < 10) return;
    setIsLoading(true);
    setError(null);

    try {
      // Step 1: Expand concept
      const { data: expandData, error: expandErr } = await resilientPost<{
        success: boolean;
        agent: { name: string; tagline: string; purpose: string };
      }>("/api/characters/quick-create", { concept: inputValue.trim() }, { retries: 0, timeout: 30000 });

      if (expandErr || !expandData?.agent) {
        throw new Error(expandErr || "Failed to generate agent profile");
      }

      const { name, tagline, purpose } = expandData.agent;

      // Step 2: Create active agent — POST /api/characters sets status "active" directly
      const { data: createData, error: createErr } = await resilientPost<{
        character: { id: string };
      }>(
        "/api/characters",
        {
          character: { name, tagline },
          metadata: { purpose, enabledTools: DEFAULT_ENABLED_TOOLS },
        },
```

```
        { retries: 0 }
      );

      if (createErr || !createData?.character?.id) {
        throw new Error(createErr || "Failed to create agent");
      }

      onCreated();
      onOpenChange(false);
      router.push(`/chat/${createData.character.id}`);
    } catch (err) {
      const msg = err instanceof Error ? err.message : "Something went wrong";
      // Check for "no model configured" scenario
      if (msg.includes("model") || msg.includes("provider") || msg.includes("API key")) {
        setError("No AI provider configured. Please add an API key in Settings first.");
      } else {
        setError(msg);
      }
      toast.error("Failed to create agent");
    } finally {
      setIsLoading(false);
    }
  };

  // Template flow:
  // POST /api/characters/templates/{id}/create → { success, characterId, templateId }
  const handleUseTemplate = async (templateId: string) => {
    setIsLoading(true);
    setError(null);

    try {
      const { data, error: err } = await resilientPost<{
        success: boolean;
        characterId: string;  // key is "characterId" not "id"
        templateId: string;
      }>(`/api/characters/templates/${templateId}/create`, {}, { retries: 0 });

      if (err || !data?.characterId) {
        throw new Error(err || "Failed to create agent from template");
      }

      onCreated();
```

```
      onOpenChange(false);
      router.push(`/chat/${data.characterId}`);
    } catch (err) {
      setError(err instanceof Error ? err.message : "Failed to create from template");
      toast.error("Failed to create agent from template");
    } finally {
      setIsLoading(false);
    }
  };

  const isQuickCreateDisabled = isLoading || inputValue.trim().length < 10;

  return (
    <Dialog open={open} onOpenChange={onOpenChange}>
      {/* DialogContent already provides a built-in × close button — do NOT add another */}
      <DialogContent className="sm:max-w-[440px] bg-terminal-cream border-terminal-border font-
mono">
        <DialogHeader>
          <DialogTitle className="font-mono text-terminal-dark">
            Create an Agent
          </DialogTitle>
        </DialogHeader>

        {error && (
          <p className="text-red-600 text-xs font-mono px-1">{error}</p>
        )}

        <Tabs value={activeTab} onValueChange={(v) => setActiveTab(v as "quick" | "template")}>
          {/* Tab styling needs terminal overrides — default shadcn uses bg-muted */}
          <TabsList className="w-full bg-terminal-dark/10 rounded-md p-1">
            <TabsTrigger
              value="quick"
              className="flex-1 font-mono text-sm data-[state=active]:bg-terminal-cream data-[s
tate=active]:text-terminal-green data-[state=active]:shadow-sm"
            >
              Quick Create
            </TabsTrigger>
            <TabsTrigger
              value="template"
              className="flex-1 font-mono text-sm data-[state=active]:bg-terminal-cream data-[s
tate=active]:text-terminal-green data-[state=active]:shadow-sm"
            >
```

```
              From Template
            </TabsTrigger>
          </TabsList>

          <TabsContent value="quick" className="mt-4 space-y-4">
            <textarea
              value={inputValue}
              onChange={(e) => setInputValue(e.target.value)}
              onKeyDown={(e) => {
                if (e.key === "Enter" && (e.metaKey || e.ctrlKey) && !isQuickCreateDisabled) {
                  handleQuickCreate();
                }
              }}
              placeholder="Describe your agent in one sentence..."
              className="w-full h-24 rounded border border-terminal-border bg-white px-3 py-2 f
ont-mono text-xs text-terminal-dark resize-none focus:outline-none focus:ring-1 focus:ring-term
inal-green"
              disabled={isLoading}
            />
            <p className="text-xs text-terminal-muted font-mono">
              {inputValue.length}/10 min chars
            </p>
            <Button
              onClick={handleQuickCreate}
              disabled={isQuickCreateDisabled}
              className="w-full bg-terminal-green hover:bg-terminal-green/90 text-white font-mo
no"
            >
              {isLoading ? (
                <>
                  <Loader2 className="h-4 w-4 animate-spin mr-2" />
                  Creating...
                </>
              ) : (
                "Create & Chat →"
              )}
            </Button>
          </TabsContent>

          <TabsContent value="template" className="mt-4">
            {/* 8 templates total — grid-cols-3 on md+, grid-cols-2 on mobile */}
            <div className="grid grid-cols-2 sm:grid-cols-3 gap-2 max-h-64 overflow-y-auto">
```

```
              {templates.length === 0 ? (
                <p className="col-span-3 text-xs text-terminal-muted font-mono text-center py-
4">
                  Loading templates...
                </p>
              ) : (
                templates.map((tmpl) => (
                  <button
                    key={tmpl.id}
                    onClick={() => handleUseTemplate(tmpl.id)}
                    disabled={isLoading}
                    className="rounded-lg border border-terminal-border p-3 text-left cursor-po
inter hover:border-terminal-green transition-colors disabled:opacity-50 disabled:cursor-not-all
owed bg-white"
                  >
                    <p className="font-mono text-xs font-medium text-terminal-dark line-clamp-
1">
                      {tmpl.name}
                    </p>
                    {tmpl.tagline && (
                      <p className="font-mono text-[10px] text-terminal-muted line-clamp-2 mt-
0.5">
                        {tmpl.tagline}
                      </p>
                    )}
                    {isLoading ? (
                      <Loader2 className="h-3 w-3 animate-spin mt-2 text-terminal-green" />
                    ) : (
                      <span className="text-[10px] text-terminal-green font-mono mt-1 block">
                        Use →
                      </span>
                    )}
                  </button>
                ))
              )}
            </div>
          </TabsContent>
        </Tabs>

        <div className="border-t border-terminal-border/40 pt-3 mt-2">
          <Link
            href="/create-character"
```

```
            className="text-xs text-terminal-muted hover:text-terminal-dark font-mono transitio
n-colors"
            onClick={() => onOpenChange(false)}
          >
            Advanced setup →
          </Link>
        </div>
      </DialogContent>
    </Dialog>
  );
}
```

---

### Modify: `components/character-picker.tsx`

**Two locations** must both be changed (not just one):

### Location 1 — Main Create Card (line ~1628)

```
// BEFORE:
<AnimatedCard data-animate-card hoverLift className="bg-terminal-cream/50 hover:bg-terminal-cre
am cursor-pointer">
  <Link href="/create-character" className="block h-full">
    ...
  </Link>
</AnimatedCard>

// AFTER — preserve AnimatedCard with data-animate-card for entrance animation:
<AnimatedCard
  data-animate-card
  hoverLift
  className="bg-terminal-cream/50 hover:bg-terminal-cream cursor-pointer"
  onClick={() => setCreateModalOpen(true)}
>
  <div className="flex flex-col items-center justify-center h-full min-h-[200px] gap-4 p-6">
    <div className="w-16 h-16 rounded-full bg-terminal-green/10 flex items-center justify-cente
r shadow-sm">
      <Plus className="w-8 h-8 text-terminal-green" />
    </div>
    <div className="text-center">
      <p className="font-medium font-mono text-terminal-dark">{t("create")}</p>
      <p className="text-sm text-terminal-muted font-mono">{t("createDescription")}</p>
    </div>
  </div>
</AnimatedCard>
```

### Location 2 — Empty State Button (line ~1843)

```
// BEFORE:
<Link href="/create-character">
  <AnimatedButton className="gap-2 bg-terminal-green hover:bg-terminal-green/90 text-white font
-mono">
    <Plus className="w-4 h-4" />
    {t("create")}
  </AnimatedButton>
</Link>

// AFTER:
<AnimatedButton
  onClick={() => setCreateModalOpen(true)}
  className="gap-2 bg-terminal-green hover:bg-terminal-green/90 text-white font-mono"
>
  <Plus className="w-4 h-4" />
  {t("create")}
</AnimatedButton>
```

### Add state + modal render

Add at the top of the component with other state:

```
const [createModalOpen, setCreateModalOpen] = useState(false);
```

Add near the end of the component JSX, alongside other dialogs:

```
<CreateAgentModal
  open={createModalOpen}
  onOpenChange={setCreateModalOpen}
  onCreated={() => loadCharacters()}  // loadCharacters() — NOT refetchCharacters()
/>
```

Add import:

```
import { CreateAgentModal } from "@/components/character-creation/create-agent-modal";
```

---

## API Endpoints Used

| Endpoint | Method | Purpose | Returns |
|---|---|---|---|
| `/api/characters/quick-create` | POST | Expand concept into name/tagline/purpose | `{ success, agent: { name, tagline, purpose } }` |
| `/api/characters` | POST | Create active agent with tools | `{ character: { id, ... } }` |
| `/api/characters/templates` | GET | Load template list | `{ templates: [...] }` |
| `/api/characters/templates/{id}/create` | POST | Create from template | `{ success, characterId, templateId }` |

**Why `POST /api/characters` instead of `POST /api/characters/draft`:**

- `/api/characters` sets `status: "active"` automatically — no extra PATCH needed
- `/api/characters/draft` hardcodes `status: "draft"` regardless of what you pass; requires a second PATCH call
- Both endpoints accept the nested body `{ character: {...}, metadata: {...} }`

---

## Quick Create — Corrected Two-Step Flow

```
Step 1: POST /api/characters/quick-create
  Body:    { concept: "A researcher that finds papers..." }
  Returns: { success: true, agent: { name: "ResearchBot", tagline: "...", purpose: "..." } }
           ↑ IMPORTANT: data is nested under "agent" key

Step 2: POST /api/characters
  Body:    {
             character: { name: "ResearchBot", tagline: "..." },
             metadata:  { purpose: "...", enabledTools: DEFAULT_ENABLED_TOOLS }
           }
           ↑ IMPORTANT: nested body shape — "purpose" goes in metadata, not in character
  Returns: { character: { id: "abc123", ... } }
           ↑ IMPORTANT: character id is at data.character.id

Step 3: router.push(`/chat/abc123`)
```

---

## Template Flow — Corrected

```
Step 1: POST /api/characters/templates/{templateId}/create
  Body:    {}
  Returns: { success: true, characterId: "abc123", templateId: "..." }
           ↑ IMPORTANT: key is "characterId" not "id"

Step 2: router.push(`/chat/abc123`)
  ↑ Use data.characterId — NOT data.id
```

---

## i18n Keys Required

Add to **both** `locales/en.json` and `locales/tr.json` under `picker.createModal`:

```
"createModal": {
  "title": "Create an Agent",
  "quickCreateTab": "Quick Create",
  "fromTemplateTab": "From Template",
  "descriptionPlaceholder": "Describe your agent in one sentence...",
  "createAndChat": "Create & Chat",
  "creating": "Creating...",
  "advancedSetup": "Advanced setup →",
  "errorExpansion": "Failed to generate agent profile. Check your AI provider settings.",
  "errorCreation": "Failed to create agent. Please try again.",
  "useTemplate": "Use →",
  "loadingTemplates": "Loading templates...",
  "noTemplates": "No templates found",
  "minCharsHint": "Minimum 10 characters"
}
```

---

## UX Details & Edge Cases

- **Empty input**: "Create & Chat" button disabled when textarea has fewer than 10 chars
- **⌘+Enter**: Submits Quick Create (keyboard shortcut)
- **Escape key**: Close modal (handled by Dialog automatically)
- **Mobile**: Modal goes full-screen on mobile ( `sm:max-w-[440px]` on desktop)

- **After creation**: Close modal, call `loadCharacters()`, navigate to chat
- **No model configured**: Show specific error "Please configure an AI provider in Settings first" (detect by error message containing "model", "provider", or "API key")
- **Auth expiry**: 401 errors from mutation calls should show "Session expired, please reload"
- `{retries: 0}`: All creation calls must pass `retries: 0` to prevent duplicate agent creation on 5xx errors
- **Templates**: 8 templates exist (including Seline default); all show in the grid
- **DialogContent close button**: Already built into the component — do NOT add a manual ✕ button

---

## Verification Steps

1. Open home page → click "Create Agent" → modal opens (no full-page navigation)
2. Quick Create tab: type "A cooking assistant" (10+ chars) → click Create & Chat → spinner → chat opens
3. Templates tab: click "Use →" on Data Analyst → spinner → chat opens
4. "Advanced setup →" link → navigates to /create-character wizard
5. Press Escape or click backdrop → modal closes without creating
6. After creation → home page agent list shows new agent (loadCharacters refreshes)
7. Empty state button (no agents page) → also opens modal
8. If no AI provider configured → shows specific error message, not generic one

---

## Gap Analysis & Missing Considerations

> *The following were identified by codebase research on 2026-02-19 and have been incorporated into the implementation plan above. Kept here for historical reference.*

| # | Issue | Severity | Resolution in Plan |
|---|---|---|---|
| 1 | `draft/route.ts` ignores `status: "active"` | **Critical** | Use `POST /api/characters` instead |
| 2 | `quick-create` returns `{ agent: {...} }` not flat object | **Critical** | Read `data.agent.name` etc. |
| 3 | Template create returns `characterId` not `id` | **Critical** | Use `data.characterId` |
| 5 | `DEFAULT_ENABLED_TOOLS` does not exist yet | **Blocker** | Implement doc 03 first |
| 6 | Second create link in empty state not mentioned | **High** | Updated Location 2 section |
| 14 | No i18n keys defined | **High** | Full key list added |
| 17 | `resilientPost` retries non-idempotent creation | **High** | `{ retries: 0 }` on all creation calls |
| 18 | PATCH body is nested, not flat | **High** | Using `POST /api/characters` avoids this |
| 7 | `AnimatedCard data-animate-card` must be preserved | **Medium** | Preserved in corrected snippet |
| 8 | `refetchCharacters()` doesn't exist — use `loadCharacters()` | **Medium** | Fixed |
| 9 | Toast uses `sonner` not shadcn useToast | **Medium** | `toast.error()` / `toast.success()` |
| 10 | Spinner: `Loader2 animate-spin` | **Medium** | Corrected in component code |
| 12 | DialogContent already has ✕ — don't add another | **Medium** | Note added |
| 13 | TabsList needs terminal aesthetic override | **Medium** | className overrides added |
| 20 | No model configured → show specific error | **Medium** | Error detection added |
| 23 | `quick-create` LLM call may take 15s — increase timeout | **Medium** | `{ timeout: 30000 }` added |

# 02 — Wizard Simplification (Remove Knowledge, Optional Embeddings, Merge Preview+Success)

## Problem

The full `/create-character` wizard has 9 steps. Three of them are either deprecated, unnecessary for most users, or redundant:

- **Knowledge** — document uploads; deprecated, folder syncing replaces it
- **Embedding Setup + Vector Search** — required today but should be optional/advanced
- **Preview** — redundant with Success; meeting says combine them

**Meeting quotes:**

> *"Knowledge should be completely deleted from all app because it's unnecessary. We have folder syncing." "Embeddings were necessary at one point but now it is not since we have options to go without syncing — this can be advanced." "Capabilities should be again fully configured by us — user should not need to see them at first." "We can combine [Preview and Success] but we should keep the celebration confetti."*

---

## Current Wizard Flow (9 steps)

```
Intro → Identity → Loading → Knowledge → EmbeddingSetup → VectorSearch → Capabilities → [MCP] → Preview → Success
```

## Target Wizard Flow (4 steps)

```
Intro → Identity → Loading → Capabilities (+optional vector toggle inline) → [MCP if configured] → Success (with summary)
```

MCP step stays conditional (only shown if user has MCP servers configured). Progress bar will show: Start (completed) → Identity (active) → Capabilities → MCP Tools (if configured) — **4 dots maximum, 3 without MCP**, not "2" as originally stated.

---

## Change 1: Update `WizardPage` Type and `PROGRESS_PAGES`

### File: `components/character-creation/terminal-wizard.tsx`

**Note:** The `WizardPage` type at line 27 already does NOT include `"knowledge"`, `"embeddingSetup"`, `"vectorSearch"`, or `"preview"` — it was partially cleaned up. However, `PROGRESS_PAGES` at line 40 still references all of them.

**Find `PROGRESS_PAGES` array (line ~40) and replace:**

```
// BEFORE
const PROGRESS_PAGES: WizardPage[] = ["identity", "knowledge", "embeddingSetup", "vectorSearch", "capabilities", "mcpTools", "preview"];


// AFTER
const PROGRESS_PAGES: WizardPage[] = ["identity", "capabilities", "mcpTools"];
//                                                                  ↑ keep mcpTools — removing it hides the progress bar on that page
```

**Find the navigation call after identity submit (line ~163) and replace:**

```
// BEFORE
navigateTo("knowledge");


// AFTER
navigateTo("capabilities");
```

**Find and remove render blocks for old pages:**

```
// REMOVE the KnowledgeBasePage block:
{currentPage === "knowledge" && (
  <KnowledgeBasePage
    onSubmit={handleKnowledgeSubmit}
    onBack={() => navigateTo("identity")}
  />
)}

// REMOVE the EmbeddingSetupPage block:
{currentPage === "embeddingSetup" && (
  <EmbeddingSetupPage ... />
)}

// REMOVE the VectorSearchPage block:
{currentPage === "vectorSearch" && (
  <VectorSearchPage ... />
)}

// REMOVE the PreviewPage block:
{currentPage === "preview" && (
  <PreviewPage ... />
)}
```

**Remove imports for removed pages:**

```
// REMOVE these 4 imports from the import block at the top:
// KnowledgeBasePage,
// EmbeddingSetupPage,
// VectorSearchPage,
// PreviewPage,

// REMOVE this type import:
// import type { UploadedDocument } from "./terminal-pages/knowledge-base-page";
```

**Remove state variables:**

```
// REMOVE from WizardState interface:
// documents: UploadedDocument[];

// REMOVE from initialState:
// documents: [],

// REMOVE unused handler functions:
// handleKnowledgeSubmit()
// handleVectorSearchSubmit()
// handleEmbeddingSetupSubmit()
// handleEmbeddingSetupSkip()
```

**Update the i18n type cast (line ~122) to remove removed page IDs:**

```
// BEFORE
label: t(step.id as "intro" | "identity" | "capabilities" | "mcpTools" | "knowledge" | "embeddi
        ngSetup" | "vectorSearch" | "preview"),

// AFTER
label: t(step.id as "intro" | "identity" | "capabilities" | "mcpTools"),
```

---

## Change 2: Update WIZARD_STEPS in wizard-progress.tsx

**File:** `components/ui/wizard-progress.tsx`

The `WIZARD_STEPS` array currently has 8 entries. Remove 4 of them:

```
// REMOVE these 4 entries (and their now-unused icon imports):
{ id: "knowledge", label: "Knowledge", icon: <BookOpen /> },       // REMOVE
{ id: "embeddingSetup", label: "Embeddings", icon: <Database /> }, // REMOVE
{ id: "vectorSearch", label: "Folders", icon: <Database /> },      // REMOVE
{ id: "preview", label: "Preview", icon: <Eye /> },                // REMOVE

// KEEP these 4 entries:
{ id: "intro", label: "Start", icon: <Sparkles /> },
{ id: "identity", label: "Identity", icon: <User /> },
{ id: "capabilities", label: "Capabilities", icon: <Wrench /> },
{ id: "mcpTools", label: "MCP Tools", icon: <Plug /> },
```

Also remove the now-unused lucide icon imports: `BookOpen`, `Database` (if only used for those entries), `Eye`.

---

## Change 3: Merge Preview → Directly Submit on Capabilities

### File: `components/character-creation/terminal-wizard.tsx`

The current `handleCapabilitiesSubmit` (line ~171) only navigates to preview or MCP — it does no API call. The finalization PATCH (`handleFinalizeAgent` at line ~289) is currently called from `PreviewPage.onConfirm`.

**Corrected approach: move finalization directly into the capabilities/MCP flow.**

Update `handleCapabilitiesSubmit`:

```
// BEFORE:
const handleCapabilitiesSubmit = (enabledTools: string[]) => {
  setState((prev) => ({ ...prev, enabledTools }));
  if (hasMcpServers === false) {
    navigateTo("preview");   // ← remove this
  } else {
    navigateTo("mcpTools");
  }
};

// AFTER:
const handleCapabilitiesSubmit = (enabledTools: string[]) => {
  setState((prev) => ({ ...prev, enabledTools }));
  if (hasMcpServers === false) {
    // No MCP step — finalize directly
    // Call handleFinalizeAgent but we need tools from the new state
    // Since setState is async, pass tools directly:
    handleFinalizeAgentWithTools(enabledTools);
  } else {
    navigateTo("mcpTools");
  }
};
```

Create a new `handleFinalizeAgentWithTools` that accepts tools directly (to avoid async setState race):

```
const handleFinalizeAgentWithTools = async (tools: string[]) => {
  if (!draftAgentId) return;
  setIsSubmitting(true);
  setError(null);
  navigateTo("loading");

  try {
    const { data, error: patchError } = await resilientPatch<{ error?: string }>(
      `/api/characters/${draftAgentId}`,
      {
        character: {
          name: state.identity.name,
          tagline: state.identity.tagline || undefined,
          status: "active",
        },
        metadata: {
          purpose: state.identity.purpose,
          enabledTools: tools,  // use passed tools, not state.enabledTools (async)
          enabledMcpServers: state.enabledMcpServers,
          enabledMcpTools: state.enabledMcpTools,
          mcpToolPreferences: state.mcpToolPreferences,
        },
      }
    );

    if (patchError) {
      throw new Error(data?.error || patchError || "Failed to create agent");
    }

    setState((prev) => ({ ...prev, createdCharacterId: draftAgentId, enabledTools: tools }));
    navigateTo("success");
  } catch (err) {
    setError(err instanceof Error ? err.message : "Creation failed");
    navigateTo("capabilities", -1);
  } finally {
    setIsSubmitting(false);
  }
};
```

Also update `MCPToolsPage.onComplete` (currently navigates to "preview" — line ~401):

```
// BEFORE:
onComplete={() => navigateTo("preview")}

// AFTER (MCP tools are already saved in handleMCPToolsSubmit — finalize here):
onComplete={() => handleFinalizeAgentWithTools(state.enabledTools)}
```

---

## Change 4: Update CapabilitiesPage Back Navigation

**File:** `components/character-creation/terminal-wizard.tsx`

The back navigation for Capabilities currently routes back through the old embedding/vector pages:

```
// BEFORE:
onBack={() => navigateTo(vectorDBEnabled ? "vectorSearch" : "embeddingSetup", -1)}

// AFTER:
onBack={() => navigateTo("identity", -1)}
```

The `vectorDBEnabled` state is still fetched from `/api/settings` on mount. Repurpose it to pre-populate the new inline Advanced Options toggle initial state (checked if already enabled, unchecked if not).

---

## Change 5: Add Optional Vector Search Toggle to Capabilities Page

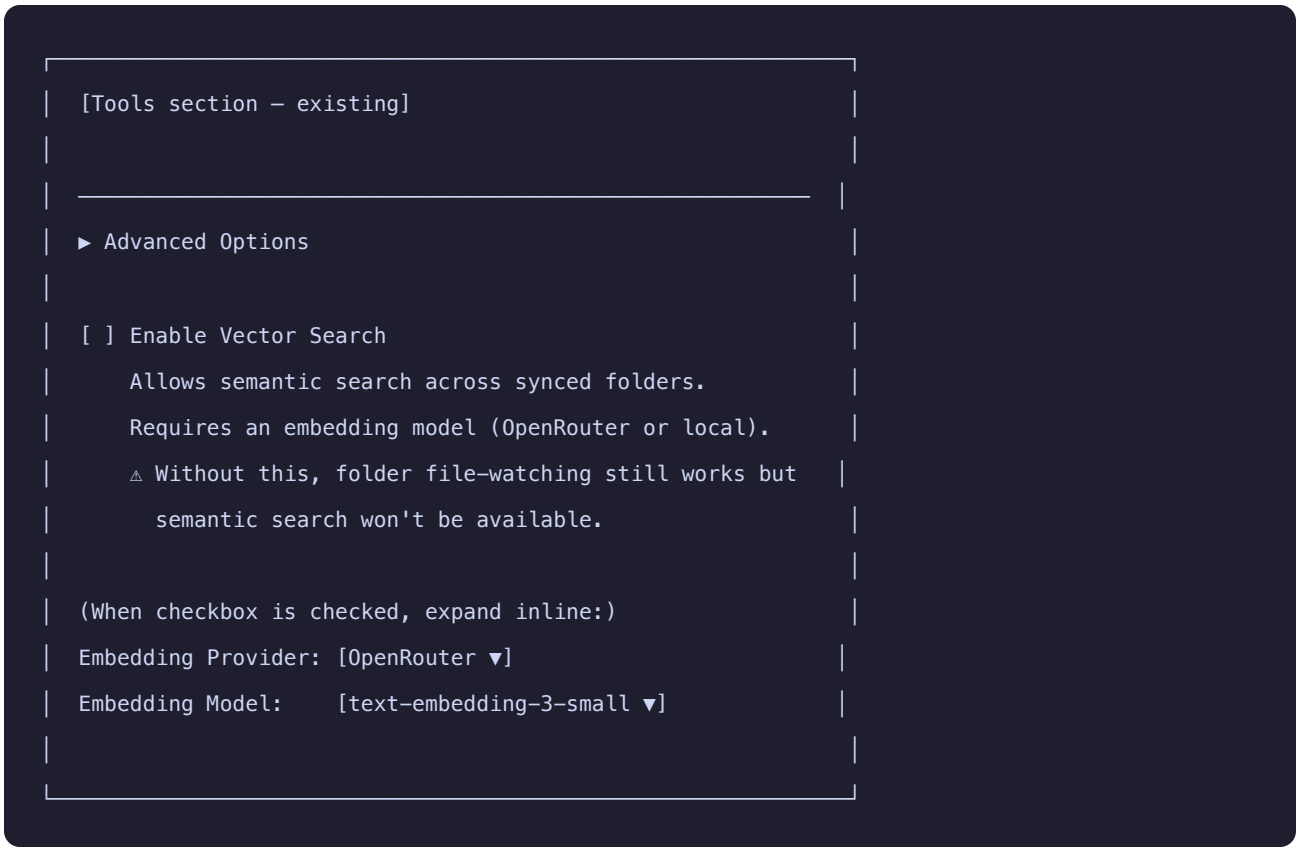
**File:** `components/character-creation/terminal-pages/capabilities-page.tsx`

Update `onSubmit` signature to pass vector config:

```
// BEFORE
onSubmit: (enabledTools: string[]) => void;

// AFTER
onSubmit: (enabledTools: string[], vectorConfig?: { provider: string; model: string; apiKey?: s
        tring }) => void;
```

Add a collapsible "Advanced Options" section **below** the tool categories, before the Continue button:




- Initial state: `vectorSearchEnabled = vectorDBEnabled` (from settings fetch in wizard)
- When checked: show provider/model dropdowns (reuse logic from `EmbeddingSetupPage`)
- When wizard finalizes with `vectorSearchEnabled === true`: call `resilientPut("/api/settings", { embeddingProvider, embeddingModel, vectorDBEnabled: true })` **before** the character PATCH, because `vectorSearch` tool's `isVectorDBEnabled()` reads settings at request time

---

## Change 6: Extend SuccessPage with Agent Summary

### File: `components/character-creation/terminal-pages/success-page.tsx`

Extend props to accept agent config for display:

```
// New props (data comes from wizard state, not API response):
interface SuccessPageProps {
  characterId: string;
  characterName: string;
  tagline?: string;          // ← new
  enabledTools: string[];    // ← new
  onConfigureAnother?: () => void;  // ← remove this (meeting: not necessary)
  onGoHome: () => void;
}
```

Add agent summary block above action buttons:


Pass from wizard:

```
{currentPage === "success" && state.createdCharacterId && (
  <SuccessPage
    characterId={state.createdCharacterId}
    characterName={state.identity.name}
    tagline={state.identity.tagline}
    enabledTools={state.enabledTools}
  />
)}
```

---

## i18n Cleanup Required

Remove these now-orphaned keys from **both** `locales/en.json` and `locales/tr.json`:

- `characterCreation.knowledgeBase.*` (all keys)
- `characterCreation.embeddingSetup.*` (all keys)
- `characterCreation.vectorSearchPage.*` (all keys)
- `characterCreation.preview.*` (all keys)
- `characterCreation.progress.knowledge`
- `characterCreation.progress.embeddingSetup`

- `characterCreation.progress.vectorSearch`
- `characterCreation.progress.preview`
- `characterCreation.success.configureAnother` (removed button)

New keys to add for the SuccessPage summary and the Advanced Options toggle (at minimum):

- `characterCreation.success.toolsEnabled` — "Tools enabled ({count}):"
- `characterCreation.capabilities.advancedOptions` — "Advanced Options"
- `characterCreation.capabilities.enableVectorSearch` — "Enable Vector Search"
- `characterCreation.capabilities.vectorSearchHint` — "Allows semantic search across synced folders."

---

## Resulting Wizard Flow (Visual)


## Files Summary

| File | Action |
| --- | --- |
| `terminal-wizard.tsx` | Remove knowledge/embedding/vectorSearch/preview pages; update PROGRESS_PAGES; fix nav flow; add `handleFinalizeAgentWithTools`; update back nav |
| `capabilities-page.tsx` | Update `onSubmit` signature; add collapsible Advanced Options with vector search toggle |
| `wizard-progress.tsx` | Remove 4 entries (knowledge, embeddingSetup, vectorSearch, preview); remove unused icon imports |
| `success-page.tsx` | Accept + display agent config summary ( `tagline` , `enabledTools` ); remove "Configure Another" |
| `terminal-pages/knowledge-base-page.tsx` | No longer imported by wizard — can be deleted |
| `terminal-pages/embedding-setup-page.tsx` | No longer imported by wizard — can be deleted |
| `terminal-pages/vector-search-page.tsx` | No longer imported by wizard — can be deleted |
| `terminal-pages/preview-page.tsx` | No longer imported by wizard — can be deleted |

## Verification Steps

1. Navigate to `/create-character` → progress bar shows Start + Identity + Capabilities (+ MCP if configured)
2. No "Knowledge" step appears anywhere
3. Capabilities page loads with all tools pre-checked (after doc 03 lands)
4. Advanced Options section exists, collapsed by default
5. Toggle "Enable Vector Search" → provider/model pickers appear inline
6. Click "Confirm & Create" → goes directly to Loading then Success (no Preview page)
7. Success page shows agent name, tagline, and key tools with count
8. "Start Chatting" navigates to chat
9. MCPToolsPage "Complete" → finalizes and navigates to Success (not Preview)

---

## Gap Analysis & Missing Considerations

> *The following were identified by codebase research on 2026-02-19 and have been incorporated into the plan sections above. Kept here for historical reference.*

| # | Issue | Resolution |
|---|---|---|
| 1 | WizardPage already partially cleaned; PROGRESS_PAGES still has old entries | Plan updated — only PROGRESS_PAGES needs fixing |
| 2 | `finalizeAgentCreation` doesn't exist — only `handleFinalizeAgent` | Replaced with `handleFinalizeAgentWithTools` accepting tools directly |
| 3 | MCPToolsPage `onComplete` still navigates to "preview" | Updated to call `handleFinalizeAgentWithTools(state.enabledTools)` |
| 4 | `handleStepClick` casts any string to WizardPage — stale WIZARD_STEPS entries cause broken nav | WIZARD_STEPS reduced to 4 entries in Change 2 |
| 5 | WIZARD_STEPS has 8 entries; plan only removed 1 | Changed to remove 4 entries |
| 6 | `t(step.id as ...)` cast must be narrowed | Updated cast in Change 1 |
| 7 | `UploadedDocument` type import from knowledge-base-page will break | Remove import + remove `documents` from WizardState |
| 10 | `CapabilitiesPage.onSubmit` signature must change | Change 5 updates signature |
| 12 | PROGRESS_PAGES must include mcpTools | `"mcpTools"` added to PROGRESS_PAGES |
| 13 | SuccessPage data comes from wizard state, not API response | Confirmed — pass from state |
| 14 | Many i18n keys become orphaned | Full cleanup list added |
| 16 | Back navigation currently routes via old pages | Changed to simple `navigateTo("identity", -1)` |
| 17 | Embedding settings PUT needed before PATCH | Addressed in Change 5 |
| 18 | "2 progress dots" statement is wrong — actually 3-4 | Corrected in target flow description |

# 03 — Pre-select All Utility Tools by Default

## Problem

New agents created via the wizard start with only `["docsSearch"]` enabled. This means a freshly created agent cannot execute commands, run skills, schedule tasks, memorize things, or do web searches until the user manually enables them.

**Meeting quote:**

> *"Capabilities should be again fully configured by us. All the necessary tools should come selected. User should not need to see them at first and default agent should have all the capabilities that should maintain the full working integration of all the application."*

---

## Current State

**`components/character-creation/terminal-wizard.tsx`** — line ~54:

```
enabledTools: ["docsSearch"],  // only one tool — severely limited
```

**`lib/characters/templates/resolve-tools.ts`** — both constants are module-private (NO `export`):

```
const ALWAYS_ENABLED_TOOLS = [
  "docsSearch", "localGrep", "readFile", "editFile", "writeFile", "executeCommand"
];

const UTILITY_TOOLS = [
  "calculator", "memorize", "runSkill", "scheduleTask",
  "sendMessageToChannel", "showProductImages", "updatePlan", "updateSkill"
  // "delegateToSubagent" is MISSING
];
// NOTE: webSearch and webBrowse are NOT in either array — they're added conditionally
```

---

## Target State

Every new agent (wizard and quick-create modal) should start with **all always-enabled + all utility tools + webSearch** pre-selected. Users can deselect in Capabilities if needed, but the default is fully functional.

`webBrowse` is intentionally excluded from the static default because it requires Firecrawl or a local scraper — enabling it when the user has neither configured causes a confusing "checked but broken" state in the UI.

---

## Changes

### File 1: `lib/characters/templates/resolve-tools.ts`

**Three changes:**

1. Add `export` keyword to both existing private constants
2. Add `delegateToSubagent` to `UTILITY_TOOLS`
3. Add `webSearch` to default set (DuckDuckGo works without any API key)
4. Export new `DEFAULT_ENABLED_TOOLS` constant

```
// CHANGE: add "export" + add "delegateToSubagent":
export const ALWAYS_ENABLED_TOOLS = [
  "docsSearch",
  "localGrep",
  "readFile",
  "editFile",
  "writeFile",
  "executeCommand",
] as const;

// CHANGE: add "export" + add "delegateToSubagent":
export const UTILITY_TOOLS = [
  "calculator",
  "memorize",
  "runSkill",
  "scheduleTask",
  "sendMessageToChannel",
  "showProductImages",
  "updatePlan",
  "updateSkill",
  "delegateToSubagent",   // ← add this
] as const;

// ADD new export — the static default for all new agents:
// webSearch is included because DuckDuckGo fallback needs no API key
// webBrowse is NOT included — it silently disappears at runtime without Firecrawl/local scrape
        r
export const DEFAULT_ENABLED_TOOLS: string[] = [
  ...ALWAYS_ENABLED_TOOLS,
  ...UTILITY_TOOLS,
  "webSearch",
];
```

**Why `webBrowse` is excluded from `DEFAULT_ENABLED_TOOLS`:** The tool is registered with `enableEnvVar: "FIRECRAWL_API_KEY"` in the registry. Without Firecrawl or a local scraper configured, `isToolEnabled()` returns `false` and the tool is stripped at runtime. But in the Capabilities page UI, it would show as *checked but locked/disabled* (the `webScraper` dependency badge). Showing users a checked tool they cannot use is confusing. `webBrowse` remains in `resolveSelineTemplateTools` where it is added conditionally only when the dependency is met.

**Why `webSearch` IS included:** The `tavilyKey` dependency shown in the Capabilities UI is stale — `webSearch` now uses DuckDuckGo as a free fallback when no Tavily key is set (see `resolveSelineTemplateTools` line 103). The UI dependency badge needs a separate fix (see Change 5 below).

---

### File 2: `lib/characters/templates/seline-default.ts`

Import and use `DEFAULT_ENABLED_TOOLS` as the base for the display preview list. The dynamic resolver (`resolveSelineTemplateTools`) still runs at creation time and adds `webBrowse`, `vectorSearch` conditionally — the static list is only used for template preview display.

```
import { DEFAULT_ENABLED_TOOLS } from "./resolve-tools";


// Replace SELINE_STATIC_TOOLS with a reference to DEFAULT_ENABLED_TOOLS
// plus the conditionally-available tools (for display purposes only):
const SELINE_STATIC_TOOLS: string[] = [
  ...DEFAULT_ENABLED_TOOLS,
  "vectorSearch",  // shown in preview even though conditional — resolveSelineTemplateTools han
        dles runtime
  "webBrowse",     // shown in preview even though conditional
];
```

This keeps the Seline template preview showing the full expected capability set (including conditional tools) while using `DEFAULT_ENABLED_TOOLS` as the base.

---

### File 3: `components/character-creation/terminal-wizard.tsx`

```
// ADD import:
import { DEFAULT_ENABLED_TOOLS } from "@/lib/characters/templates/resolve-tools";


// CHANGE initial state (line ~54):
// BEFORE:
enabledTools: ["docsSearch"],


// AFTER:
enabledTools: DEFAULT_ENABLED_TOOLS,
```

---

### File 4: `components/character-creation/terminal-pages/capabilities-page.tsx`

Two changes:

#### 4a. Add `delegateToSubagent` to `BASE_TOOLS`:

```
// Find the BASE_TOOLS array and add:
{ id: "delegateToSubagent", name: "Delegate to Sub-Agent", category: "utility", ... },
// This prevents a flash-of-unchecked on first render since DEFAULT_ENABLED_TOOLS includes it
// but the async API fetch may not have returned yet
```

#### 4b. Update the prop default:

```
// BEFORE (line ~279):
initialEnabledTools = ["docsSearch"],

// AFTER:
initialEnabledTools = DEFAULT_ENABLED_TOOLS,
// Add import: import { DEFAULT_ENABLED_TOOLS } from "@/lib/characters/templates/resolve-tool
        s";
```

---

### File 5: `components/character-creation/terminal-pages/capabilities-page.tsx` — Fix `webSearch` Dependency

The `webSearch` tool currently declares `dependencies: ["tavilyKey"]` in the capabilities page BASE_TOOLS, causing the checkbox to appear locked when no Tavily key is configured. This is stale since DuckDuckGo makes the tool functional without any key.

```
// Find the webSearch entry in BASE_TOOLS and remove the dependency:
// BEFORE:
{ id: "webSearch", ..., dependencies: ["tavilyKey"] },

// AFTER:
{ id: "webSearch", ..., dependencies: [] },
// or simply omit the dependencies field
```

Also fix the same stale dependency in `components/character-picker.tsx` (inline tool editor BASE_TOOLS around line 118):

```
// Find webSearch in the character-picker BASE_TOOLS and remove tavilyKey dependency
```

---

## Tool Reference Table

| Tool ID | Category | Included in DEFAULT_ENABLED_TOOLS | Notes |
|---|---|---|---|
| `docsSearch` | knowledge | ✓ | Vector similarity search in indexed docs |
| `localGrep` | knowledge | ✓ | Regex search through synced files |
| `readFile` | knowledge | ✓ | Read files from disk |
| `editFile` | knowledge | ✓ | Edit files on disk |
| `writeFile` | knowledge | ✓ | Create/overwrite files |
| `executeCommand` | knowledge | ✓ | Run shell commands |
| `webSearch` | search | ✓ | Web search (DuckDuckGo free; Tavily if key set) |
| `webBrowse` | search | ✗ | Requires Firecrawl API key or local scraper |
| `calculator` | utility | ✓ | Math calculations |
| `memorize` | utility | ✓ | Save facts to agent memory |
| `runSkill` | utility | ✓ | Execute a skill/plugin |
| `updateSkill` | utility | ✓ | Modify a skill |
| `scheduleTask` | utility | ✓ | Create cron/interval tasks |
| `sendMessageToChannel` | utility | ✓ | Send to chat channels |
| `showProductImages` | utility | ✓ | Display product images |
| `updatePlan` | utility | ✓ | Update agent's task plan |
| `delegateToSubagent` | utility | ✓ | Delegate to a sub-agent (was missing; now added) |

**Not in default set (optional/gated):**

- `vectorSearch` — requires embeddings setup (conditional in `resolveSelineTemplateTools`)
- `webBrowse` — requires Firecrawl or local scraper (conditional)
- `workspace` — requires `devWorkspaceEnabled` setting (double-gated: DB flag + runtime check)
- All image generation tools — require API keys
- `webQuery`, `firecrawlCrawl` — require specific API keys
- `describeImage`, `patchFile` — in EXCLUDED_TOOLS (intentionally off)

**Note on checked-but-locked tools:** Several tools in `DEFAULT_ENABLED_TOOLS` have UI dependency badges in the Capabilities page:

- `executeCommand`, `readFile`, `editFile`, `writeFile`, `localGrep` — depend on `syncedFolders`
- For new agents with no sync folders configured yet, these 5 tools will appear checked but locked/disabled in the Capabilities wizard

This is acceptable UX because: (a) the agent can still use `webSearch`, `memorize`, `runSkill`, etc. without sync folders, and (b) the lock icon communicates that the feature requires setup rather than being broken.

---

## Scope of This Change

This change applies to:

1. **Wizard new-agent flow** — via `initialState.enabledTools = DEFAULT_ENABLED_TOOLS`
2. **Seline default template creation** — via updated `SELINE_STATIC_TOOLS` base
3. **Quick-create modal** (doc 01) — modal passes `DEFAULT_ENABLED_TOOLS` in the `POST /api/characters` body

**Out of scope:**

- Other templates (social-media-manager, data-analyst, etc.) — each has its own tool set tailored to its purpose
- Plugin import sub-agents — use `InheritedAgentConfig.enabledTools` from plugin manifest
- `POST /api/characters` (direct creation via API) — no server-side injection; client must pass the right tools

---

## Verification Steps

1. Create a new agent via Quick Create modal → agent's tool count shows 17 tools
2. Create a new agent via wizard → Capabilities page loads with all tools pre-checked
3. `webSearch` checkbox is NOT locked/disabled (no `tavilyKey` dependency badge)
4. `webBrowse` is NOT pre-checked (dependency not met for most users)
5. `delegateToSubagent` appears in the tool list for workflow agents
6. Default Seline template preview shows its tool set (including vectorSearch, webBrowse for display)

---

## Gap Analysis & Missing Considerations

> *The following were identified by codebase research on 2026-02-19 and have been incorporated into the plan above. Kept here for historical reference.*

| # | Issue | Resolution |
|---|---|---|
| 1 | `ALWAYS_ENABLED_TOOLS` and `UTILITY_TOOLS` not exported | `export` added to both in Change 1 |
| 2 | `DEFAULT_ENABLED_TOOLS` didn't exist | Created and exported in Change 1 |
| 3 | `webSearch` / `webBrowse` NOT in `ALWAYS_ENABLED_TOOLS` | `webSearch` added explicitly; `webBrowse` excluded with explanation |
| 4 | `webBrowse` has Firecrawl gate — silently disappears | Excluded from DEFAULT_ENABLED_TOOLS; remains in dynamic resolver |
| 5 | `webSearch` UI shows stale `tavilyKey` dependency | Change 5 removes the stale dependency from capabilities-page and character-picker |
| 6 | `delegateToSubagent` missing from wizard BASE_TOOLS | Change 4a adds it |
| 9 | `CapabilitiesPage` prop default was `["docsSearch"]` | Change 4b updates it to `DEFAULT_ENABLED_TOOLS` |
| 10 | Plan showed wrong draft POST schema | Using `POST /api/characters` in modal (doc 01) instead |
| 11 | Checked-and-locked tools (syncedFolders-dependent) | Acknowledged in tool reference table; acceptable UX |
| 13 | Other creation paths don't use DEFAULT_ENABLED_TOOLS | Scope explicitly limited in plan |

# 04 — Agent Card Cleanup (Replace Icon Button Row with Overflow Menu)

## Problem

Each agent card on the home page has 7 text-and-icon buttons squished into the bottom row: Edit | Tools count | Folders | MCP Tools | Plugins | Dashboard | Delete

This is visually noisy, hard to understand at a glance, and overwhelming for new users.

**Meeting quote:**

> *"Here's the agents list — we can talk about how we're going to do here — there is so much clutter: edits, tool enabling, folders, MCP tools, configure plugin, dashboard, delete — however all these are kind of necessary to have the advanced customization." "It should be put a bit further behind in a more unionized compact form."*

---

## Current Card Layout

```
┌──────────────────────────────────────────────────────┐
│  🟢  ResearchBot                                     │ ← active dot (on avatar)
│      "Your paper discovery engine"                   │
│                                                      │
│  [webSearch] [readFile] [memorize] +9                │ ← tool badges
│                                                      │
│  [Continue Chat ▶]  [New Chat +]                     │ ← primary buttons
│                                                      │
│  ✏ Edit  🔧 3 tools  🗄 Folders  🔌 MCP  🔌 Plugins  📊 Dashboard  🗑 Delete  │ ← 7 cluttered buttons
└──────────────────────────────────────────────────────┘
```

Note: The current bottom row uses **text + icon buttons** (not icon-only), from a mix of `@phosphor-icons/react` (`Wrench`, `Database`, `Pencil`, `Trash`, `Plug`, `ChartBar`) and `lucide-react` (`User`, `MessageCircle`, `PlusCircle`).

---

## Target Card Layout


The `•••` menu opens a dropdown with all 8 management actions:

## Implementation

### File: `components/character-picker.tsx`

#### Step 1: Add required imports

```
// Add to lucide-react import line:
import { Plus, Loader2, ..., MoreHorizontal, Copy, Puzzle } from "lucide-react";
//                                          ↑ NOT MoreHorizontalIcon  ↑ NOT PuzzleIcon
//                                                                      Puzzle is the corr
ect lucide name

// DropdownMenu already imported or add:
import {
  DropdownMenu,
  DropdownMenuContent,
  DropdownMenuItem,
  DropdownMenuSeparator,
  DropdownMenuTrigger,
} from "@/components/ui/dropdown-menu";
```

#### Step 2: Add `relative group` to the card wrapper

`AnimatedCard` does NOT have `relative` or `group` by default. Both are required:

```
// BEFORE (line ~1657):
<AnimatedCard key={character.id} data-animate-card hoverLift className="bg-terminal-cream">

// AFTER:
<AnimatedCard key={character.id} data-animate-card hoverLift className="bg-terminal-cream relat
ive group">
//                                                                                  ↑↑
both required
```

#### Step 3: Add the `•••` trigger button to the card header area

Place inside the card's `<div className="p-4 pb-2">` wrapper, using `absolute top-2 right-2`:

```
{/* ••• overflow menu trigger — fades in on hover (touch: always visible at 40% opacity) */}
<DropdownMenu>
  <DropdownMenuTrigger asChild>
    <button
      className="absolute top-2 right-2 p-1 rounded opacity-40 group-hover:opacity-100 transiti
on-opacity hover:bg-terminal-dark/10 focus:opacity-100 focus:outline-none"
      onClick={(e) => e.stopPropagation()}
      aria-label="Agent options"
    >
      <MoreHorizontal className="w-4 h-4 text-terminal-muted" />
    </button>
  </DropdownMenuTrigger>
  <DropdownMenuContent
    align="end"
    className="font-mono text-sm"
    onClick={(e) => e.stopPropagation()}
    // ↑ IMPORTANT: stopPropagation on content too, not just trigger
  >
    <DropdownMenuItem onSelect={() => openIdentityEditor(character)}>
      {/* Use Pencil from @phosphor-icons/react — already imported */}
      <Pencil className="w-3.5 h-3.5 mr-2" />
      Edit Info
    </DropdownMenuItem>
    <DropdownMenuItem onSelect={() => openToolEditor(character)}>
      <Wrench className="w-3.5 h-3.5 mr-2" />
      Manage Tools
    </DropdownMenuItem>
    <DropdownMenuItem onSelect={() => openFolderManager(character)}>
      <DatabaseIcon className="w-3.5 h-3.5 mr-2" />
      Sync Folders
    </DropdownMenuItem>
    <DropdownMenuItem onSelect={() => openMcpToolEditor(character)}>
      {/* Function name is openMcpToolEditor — NOT openMcpEditor */}
      <Plug className="w-3.5 h-3.5 mr-2" />
      MCP Tools
    </DropdownMenuItem>
    <DropdownMenuItem onSelect={() => openPluginEditor(character)}>
      {/* Both MCP and Plugins used Plug icon; use Puzzle from lucide for Plugins */}
      <Puzzle className="w-3.5 h-3.5 mr-2" />
      Plugins
    </DropdownMenuItem>
```

```
      <DropdownMenuItem onSelect={() => router.push("/dashboard")}>
        {/* Dashboard button was in the original row but MISSING from original plan */}
        <BarChart2 className="w-3.5 h-3.5 mr-2" />
        Dashboard
      </DropdownMenuItem>
      <DropdownMenuSeparator />
      <DropdownMenuItem onSelect={() => handleDuplicate(character.id)}>
        {/* Requires doc 05 implementation — stub until then */}
        <Copy className="w-3.5 h-3.5 mr-2" />
        {/* Copy from lucide-react — NOT CopyIcon */}
        Duplicate
      </DropdownMenuItem>
      <DropdownMenuSeparator />
      <DropdownMenuItem
        onSelect={() => openDeleteDialog(character)}
        // Function is openDeleteDialog(character: CharacterSummary) — NOT handleDelete(char.id)
        className="text-red-600 focus:text-red-600"
      >
        <Trash2 className="w-3.5 h-3.5 mr-2" />
        Delete
      </DropdownMenuItem>
    </DropdownMenuContent>
  </DropdownMenu>
```

**Key correctness notes:**

- `onSelect` not `onClick` on `DropdownMenuItem` (Radix-idiomatic; auto-closes menu; better keyboard/pointer compat)
- `stopPropagation` on **both** trigger button AND `DropdownMenuContent` to prevent card-level click handlers
- `MoreHorizontal` not `MoreHorizontalIcon` (lucide has no `Icon` suffix)
- `Puzzle` not `PuzzleIcon` (lucide, no `Icon` suffix)
- `Copy` not `CopyIcon` (lucide, no `Icon` suffix)
- `openMcpToolEditor(character)` not `openMcpEditor(char)` — wrong name was never defined
- `openDeleteDialog(character)` not `handleDelete(char.id)` — takes full object, opens AlertDialog
- `handleDuplicate` does not exist yet — stub with `toast("Coming soon")` until doc 05 lands

### Step 4: Stub `handleDuplicate` until doc 05 is implemented

```
const handleDuplicate = (characterId: string) => {
  toast("Duplicate feature coming soon");
  // TODO: implement after doc 05 API endpoint is built
};
```

### Step 5: Remove the old bottom button row

Find the block containing all the management buttons (lines ~1729-1793 in character-picker.tsx) and remove the entire `<div className="flex items-center gap-1">` wrapping the 7 buttons.

---

## Active Session Indicator

The green pulsing dot is `absolute -top-1 -right-1` relative to the Avatar's `relative` wrapper **inside the card**, not at the card's corner. There is **zero overlap** with the `•••` button which is `absolute top-2 right-2` relative to the card root. No repositioning needed.

---

## Mobile / Touch Device Note

On touch screens, `:hover` states are unreliable. Instead of fully hiding ( `opacity-0` ) the `•••` button, use a reduced-but-visible default:

```
opacity-40 group-hover:opacity-100 focus:opacity-100
```

This ensures the button is always visible on touch devices (at 40% opacity) and becomes fully visible on desktop hover/focus. The existing session-item dropdown uses `opacity-0 group-hover:opacity-100` which is invisible on mobile — this approach is intentionally different.

---

## `AgentCardInWorkflow` Also Needs Updating

The `AgentCardInWorkflow` component (lines 212–380 in `character-picker.tsx` ) has its own copy of the action row (lines ~322-358) with the same buttons. The plan must also update this component to use the `•••` menu pattern. Otherwise workflow-member agent cards will still show the old text-button row — an inconsistent UI.

Add `relative group` to its `AnimatedCard` wrapper and apply the same `DropdownMenu` structure.

## Tool Count Display

After removing the wrench button row, the total enabled tool count disappears (it was shown as e.g. "3 tools" on the wrench button). The tool badge row above still shows the top 3 tool names + "+N more" overflow count. This is considered acceptable — the decision is explicit.

---

## Verification Steps

1. Home page → agent cards show NO bottom button row
2. `•••` button visible at reduced opacity (always, on all devices); fully opaque on hover
3. Click `•••` → dropdown with 8 items (Edit, Tools, Folders, MCP, Plugins, Dashboard, Duplicate, Delete)
4. "Edit Info" → opens identity editor dialog ✓
5. "Manage Tools" → opens tool editor dialog ✓
6. "Sync Folders" → opens folder sync manager ✓
7. "MCP Tools" → opens MCP tool editor dialog ✓
8. "Plugins" → opens plugin editor dialog ✓
9. "Dashboard" → navigates to /dashboard ✓
10. "Duplicate" → shows "Coming soon" toast (stub until doc 05)
11. "Delete" shows in red → opens AlertDialog confirmation ✓
12. Active session green dot still visible, no overlap with `•••`
13. `AgentCardInWorkflow` also has the `•••` menu

---

## Gap Analysis & Missing Considerations

> *The following were identified by codebase research on 2026-02-19 and have been incorporated into the plan above. Kept here for historical reference.*

| # | Issue | Resolution |
|---|---|---|
| 1 | Current row has 7 text+icon buttons, not 5 icon-only | Plan updated to reflect reality |
| 2 | `MoreHorizontalIcon` → use `MoreHorizontal` | Corrected throughout |
| 3 | `handleDelete(char.id)` → use `openDeleteDialog(character)` | Corrected |
| 4 | `openMcpEditor` → use `openMcpToolEditor` | Corrected |
| 5 | `handleDuplicate` doesn't exist | Stub added; doc 05 tracks the real implementation |
| 6 | `PuzzleIcon` doesn't exist → use `Puzzle` | Corrected |
| 7 | `CopyIcon` doesn't exist → use `Copy` | Corrected |
| 8 | Dashboard button omitted from plan | Added as menu item |
| 9 | `AnimatedCard` needs both `relative` and `group` | Both added |
| 10 | `onSelect` not `onClick` on DropdownMenuItems | Corrected |
| 11 | `stopPropagation` needed on content too | Added to DropdownMenuContent |
| 12 | AgentCardInWorkflow not mentioned | Added as explicit task |
| 14 | Mobile: `opacity-0 group-hover` invisible on touch | Changed to `opacity-40` always-visible approach |

# 05 — Agent Duplicate / Copy Feature

## Problem

There is no way to copy an agent. Users want to assign the same agent to multiple workflow trees, or create a variation of an existing agent without rebuilding from scratch.

**Meeting quote:**

> *"You want to create copies of the agents you already have and quickly assign them to new work trees." "Of course the agents should be copyable and can be present in different work trees. Currently I think there is no way of doing that."*

---

## Current State

- No duplicate endpoint exists
- No duplicate UI action exists
- The only creation paths are: new wizard, quick-create, template

---

## What Gets Copied

| Field | Copied? | Notes |
|---|---|---|
| `name` | ✓ | Append " (copy)" |
| `tagline` | ✓ | Exact copy |
| `metadata` (all) | ✓ | Copies purpose, enabledTools, systemPrompt, mcpServers/Tools, etc. — all in one JSON blob |
| `enabledPlugins` (metadata cache) | ✓ | Copied via metadata; authoritative copy via agent_plugins table |
| `syncFolders` | ✓ (paths only) | Copy folder paths with `status: "pending"` — will auto-index |
| `character_images` | ✓ | Copy image row metadata (points to same file on disk) |
| `agent_plugins` rows | ✓ | Copy plugin assignments from join table |
| `memories` | ✗ | Fresh start for memory |
| `is_default` | ✗ | Copy is never the default |
| `sessions / chat history` | ✗ | Not copied |
| workflow linkage | ✗ | `workflowId`, `workflowRole`, `inheritedResources` cleared — copy is standalone |
| `inheritedFromWorkflowId` on folders | ✗ | Set to null — copy's folders are standalone |

---

## API Implementation

### New File: `app/api/characters/[id]/duplicate/route.ts`

**Critical corrections from gap analysis — read carefully:**

- Use `sqlite-character-schema.ts` imports (not shim files)
- Use `requireAuth` + `getOrCreateLocalUser` pattern (NOT `getLocalUser()`)
- All agent config is in the `metadata` JSON column — no individual columns for `purpose`, `enabledTools`, etc.
- `agentSyncFolders` uses camelCase drizzle field names: `recursive` (not `is_recursive`), `includeExtensions` (not `file_extensions`)
- Next.js 15 requires `await params` (params is a Promise)
- Must check ownership before duplicating (security)

```ts
import { NextRequest, NextResponse } from "next/server";
import { requireAuth } from "@/lib/auth/local-auth";
import { getOrCreateLocalUser, loadSettings } from "@/lib/settings/settings-manager";
import { db } from "@/lib/db/sqlite-client";
import {
  characters,
  agentSyncFolders,
  characterImages,
} from "@/lib/db/sqlite-character-schema";
import { agentPlugins } from "@/lib/db/sqlite-plugins-schema";
import { eq, and } from "drizzle-orm";

type RouteParams = { params: Promise<{ id: string }> };

export async function POST(req: NextRequest, { params }: RouteParams) {
  try {
    // Auth — same pattern as every other route in this codebase:
    const userId = await requireAuth(req);
    const settings = loadSettings();
    const dbUser = await getOrCreateLocalUser(userId, settings.localUserEmail);

    const { id } = await params;  // Next.js 15: params is a Promise

    // Fetch the source character
    const [source] = await db
      .select()
      .from(characters)
      .where(eq(characters.id, id))
      .limit(1);

    if (!source) {
      return NextResponse.json({ error: "Agent not found" }, { status: 404 });
    }

    // Ownership check — same as GET/PATCH/DELETE in app/api/characters/[id]/route.ts
    if (source.userId !== dbUser.id) {
      return NextResponse.json({ error: "Forbidden" }, { status: 403 });
    }

    // Create the duplicate character
    // All config is in the "metadata" JSON column — copy it wholesale,
```

```
    // then clear workflow-linkage fields from the copy's metadata
    const sourceMetadata = (source.metadata as Record<string, unknown>) || {};
    const duplicateMetadata = {
      ...sourceMetadata,
      // Clear workflow linkage so the copy is a standalone agent
      workflowId: undefined,
      workflowRole: undefined,
      inheritedResources: undefined,
    };

    const [newChar] = await db
      .insert(characters)
      .values({
        // drizzle uses camelCase field names (mapped to snake_case SQL columns internally):
        userId: dbUser.id,
        name: `${source.name} (copy)`,
        displayName: source.displayName ? `${source.displayName} (copy)` : null,
        tagline: source.tagline,
        status: "active",
        isDefault: false,
        metadata: duplicateMetadata,
      })
      .returning();

    const newId = newChar.id;

    // Copy sync folder paths (not the indexed data)
    const sourceFolders = await db
      .select()
      .from(agentSyncFolders)
      .where(eq(agentSyncFolders.characterId, id));

    if (sourceFolders.length > 0) {
      await db.insert(agentSyncFolders).values(
        sourceFolders.map((f) => ({
          // drizzle camelCase field names (NOT snake_case SQL column names):
          characterId: newId,
          userId: dbUser.id,
          folderPath: f.folderPath,
          displayName: f.displayName,
          recursive: f.recursive,           // ← NOT "is_recursive"
          includeExtensions: f.includeExtensions,  // ← NOT "file_extensions"
```

```
        excludePatterns: f.excludePatterns,
        isPrimary: f.isPrimary,
        syncMode: f.syncMode,
        chunkPreset: f.chunkPreset,
        indexingMode: f.indexingMode,
        syncCadenceMinutes: f.syncCadenceMinutes,
        fileTypeFilters: f.fileTypeFilters,
        maxFileSizeBytes: f.maxFileSizeBytes,
        chunkSizeOverride: f.chunkSizeOverride,
        chunkOverlapOverride: f.chunkOverlapOverride,
        reindexPolicy: f.reindexPolicy,
        status: "pending",  // reset — will auto-index on next background sync cycle
        // Explicitly null out workflow provenance for the copy:
        inheritedFromWorkflowId: null,
        inheritedFromAgentId: null,
      }))
    );
  }

  // Copy plugin assignments from the agent_plugins join table
  // (the metadata.enabledPlugins cache is already copied above, but the
  //  authoritative data lives in agent_plugins rows)
  const sourcePlugins = await db
    .select()
    .from(agentPlugins)
    .where(and(eq(agentPlugins.agentId, id), eq(agentPlugins.enabled, true)));

  if (sourcePlugins.length > 0) {
    await db.insert(agentPlugins).values(
      sourcePlugins.map((p) => ({
        agentId: newId,
        pluginId: p.pluginId,
        workflowId: null,  // copy is standalone, not workflow-inherited
        enabled: true,
      }))
    );
  }

  // Copy avatar/character images (metadata row only — image files stay on disk)
  const sourceImages = await db
    .select()
    .from(characterImages)
```

```
      .where(eq(characterImages.characterId, id));

    if (sourceImages.length > 0) {
      await db.insert(characterImages).values(
        sourceImages.map((img) => ({
          characterId: newId,
          localPath: img.localPath,
          url: img.url,
          isPrimary: img.isPrimary,
          imageType: img.imageType,
          imagePrompt: img.imagePrompt,
          imageSeed: img.imageSeed,
        }))
      );
    }

    return NextResponse.json({ character: newChar }, { status: 201 });
  } catch (error) {
    console.error("[Duplicate Agent] Error:", error);
    return NextResponse.json({ error: "Failed to duplicate agent" }, { status: 500 });
  }
}
```

---

## UI Implementation

### File: `components/character-picker.tsx`

Replace the stub `handleDuplicate` from doc 04 with the real implementation:

```
const handleDuplicate = async (characterId: string) => {
  try {
    // Use resilientPost (not resilientFetch with method:"POST")
    // retries: 0 to prevent duplicate creation on retry
    const { data, error } = await resilientPost<{ character: { id: string } }>(
      `/api/characters/${characterId}/duplicate`,
      {},
      { retries: 0 }
    );
    if (error || !data?.character) throw new Error(error || "Unknown error");

    // loadCharacters() — NOT refetchCharacters() (doesn't exist)
    await loadCharacters();

    // sonner toast API: toast("msg") not toast({ description: "msg" })
    toast.success("Agent duplicated");
  } catch (err) {
    toast.error("Failed to duplicate agent");
  }
};
```

The "Duplicate" menu item in the `•••` dropdown (from doc 04) calls this handler:

```
<DropdownMenuItem onSelect={() => handleDuplicate(character.id)}>
  <Copy className="w-3.5 h-3.5 mr-2" />
  Duplicate
</DropdownMenuItem>
```

---

## Important: Auto-Reindexing on Folder Copy

Setting folder `status: "pending"` means the next call to `syncStaleFolders()` (which runs on startup and on a timer) will pick up all copied folders and begin indexing. If the source agent had many large sync folders, all will index simultaneously — potentially hitting EMFILE limits on macOS (see project MEMORY.md).

**Decision point:** Accept auto-reindexing (consistent with any newly-added folder) OR set `status: "paused"` and require the user to manually trigger sync. The implementation above uses `"pending"` (auto-reindex). Change to `"paused"` if EMFILE concerns outweigh convenience.

---

## Duplicate Name Deduplication

There is no `UNIQUE` constraint on `(userId, name)` in the characters table. Repeated duplications produce "Agent (copy)", "Agent (copy) (copy)", etc. To avoid accumulation, strip an existing " (copy)" suffix before re-appending:

```
const baseName = source.name.replace(/ \(copy\)$/, "");
const dupName = `${baseName} (copy)`;
```

---

## "Add to Workflow" Quick Action

Beyond simple duplication, the `•••` menu for standalone agents should also offer "Add to Workflow →":

```
[•••] menu for standalone agent:
  ✏ Edit Info
  ...
  ──────────────
  ⧉  Duplicate
     Add to Workflow...  ← opens select dialog
  ──────────────
  🗑  Delete
```

**API for "Add to Workflow":**

The workflow PATCH endpoint accepts a discriminated union. The action to add an agent is:

```
// PATCH /api/workflows/{workflowId}
{
  action: "addSubagent",   // ← NOT adding to an "agents array"
  agentId: string,
  syncFolders: boolean (optional)
}
```

**Important constraints:**

- `assertAgentNotInActiveWorkflow` is enforced server-side — an agent in an active/paused workflow cannot be added to another. API returns HTTP 400: "Agent already belongs to an active workflow"
- New members added via `addSubagent` are always added as `"subagent"` role. To make an agent the initiator, use a separate `action: "setInitiator"` call

- The "Add to Workflow" dialog must show only workflows where the agent is NOT already a member
- The dialog must handle the 400 error gracefully and show it to the user

**Dialog wireframe:**


---

## Verification Steps

1. Home page → hover agent card → click `•••` → see "Duplicate" option
2. Click Duplicate → toast "Agent duplicated" → new card appears with "(copy)" suffix
3. Duplicate inherits all tools (via metadata copy), plugins, sync folder paths
4. Duplicate has `status: active` and appears immediately in the list
5. Duplicate has empty chat history (fresh start)
6. If source agent had sync folders → copy shows same paths with `status: pending`
7. If source agent had an avatar image → copy shows same avatar
8. Duplicating "Agent (copy)" → produces "Agent (copy)" (not "Agent (copy) (copy)")
9. Cannot duplicate another user's agent → 403 response

---

## Gap Analysis & Missing Considerations

> *The following were identified by codebase research on 2026-02-19 and have been incorporated into the plan above. Kept here for historical reference.*

| # | Issue | Resolution |
|---|---|---|
| 1 | Wrong schema imports (shim files) | Changed to `sqlite-character-schema.ts` |
| 2 | `getLocalUser()` wrong auth pattern | Changed to `requireAuth` + `getOrCreateLocalUser` |
| 3 | All agent config is in `metadata` JSON blob | Insert uses `metadata: {...spread...}` not individual columns |
| 4 | Wrong `agentSyncFolders` column names | `recursive` (not `is_recursive`), `includeExtensions` (not `file_extensions`) |
| 5 | Plugin assignments in `agent_plugins` join table | Added plugin copy block |
| 6 | Next.js 15 async params | `type RouteParams = { params: Promise<{id: string}> }` + `await params` |
| 7 | Wrong `resilientFetch({method:"POST"})` | Changed to `resilientPost` |
| 7b | `refetchCharacters()` doesn't exist | Changed to `loadCharacters()` |
| 7c | `toast({description:"..."})` wrong API | Changed to `toast.success()` / `toast.error()` |
| 8 | Copying folders triggers auto-reindex | Acknowledged with decision point |
| 9 | "Add to Workflow" API is `addSubagent` action, not array update | Corrected |
| 10 | `character_images` not copied | Added image copy block |
| 11 | No ownership check | `if (source.userId !== dbUser.id) → 403` added |
| 12 | "(copy) (copy)" accumulation | Name deduplication logic added |
| 14 | Missing required sync folder fields | All columns copied including `userId`, indexing settings |

# 06 — Workflow UI: Section Headers & Hierarchy Clarity

## Problem

The home page mixes standalone agents and workflow-grouped agents in the same area with no clear visual separation. The relationship between them is confusing, and there is no easy way to add a standalone agent to an existing workflow.

**Meeting quotes:**

> *"We need to create a separation between this but still protect the ease of accessibility and visibility." "User when they land on the character picker should see both of these modes nicely but be able to switch them nicely." "What we need to make it better is the cluttering and the layout and unique visual touch that will separate us from being basic."*

---

## Important Distinction: Workflows vs Workspaces

These are two completely separate systems — do not conflate them:

| | Workflows | Workspaces (WorkspaceDashboard) |
|---|---|---|
| State variable | `workflowGroups` | Inside `workspace-dashboard.tsx` |
| API | `GET /api/workflows?status=all` | `GET /api/workspaces` |
| What it is | Multi-agent orchestration groups | Active git worktrees (Developer Workspace feature) |
| Visibility | Always | Only when `devWorkspaceEnabled === true` |

`WorkspaceDashboard` must remain above the "Workflows" section, exactly where it is (lines 1341-1350). It is a separate feature gated by `devWorkspaceEnabled` and should not be reorganized under "Workflows".

---

## Current Layout

```
[WorkspaceDashboard]                    ← only when devWorkspaceEnabled
[h3 "Workflows" — no divider, buried]   ← partially exists already
[WorkflowCard: MainWorkflow]
  └── AgentA, AgentB
[WorkflowCard: ResearchWorkflow]
  └── AgentC
[Create Agent card] [Standalone agents...]  ← no "Agents" header
```

---

## Target Layout

```
[WorkspaceDashboard]                    ← unchanged

── Workflows ──────────────────────── [+ New Workflow]

[WorkflowCard: MainWorkflow] (collapsed by default)
[WorkflowCard: ResearchWorkflow] (collapsed by default)

── Agents ──────────── [🔍 Search agents & workflows...]

[Create Agent card] [Standalone AgentD] [Standalone AgentE]
```

---

## Implementation

### File: `components/character-picker.tsx`

#### Step 1: Upgrade existing Workflows section header (line ~1369)

A `<h3>` heading already exists at line 1369 inside `{filteredWorkflowGroups.length > 0 && (...)}`. Convert it to the full flex-row layout with divider and inline "New Workflow" button:

```
// BEFORE (line ~1369 — inside the existing workflows guard):
<h3 className="font-mono text-sm font-medium text-terminal-muted uppercase tracking-wider">
  {t("workflows.sectionTitle")}
</h3>

// AFTER — upgrade to full flex-row with divider and inline button:
<div className="flex items-center gap-3 mb-4">
  <h2 className="font-mono text-xs font-semibold tracking-widest text-terminal-muted uppercase
whitespace-nowrap">
    {t("workflows.sectionTitle")}
  </h2>
  <div className="flex-1 h-px bg-terminal-border/40" />
  <button
    className="text-xs font-mono text-terminal-muted hover:text-terminal-dark transition-colors
disabled:opacity-40 disabled:cursor-not-allowed"
    onClick={() => setWorkflowCreatorOpen(true)}
    // ← "setWorkflowCreatorOpen" NOT "setShowCreateWorkflow" (which doesn't exist)
    disabled={allStandaloneCharacters.length === 0}
    // ↑ disable when no standalone agents to add — opening the dialog would show empty dropdow
n
    title={allStandaloneCharacters.length === 0 ? "Create a standalone agent first" : undefine
d}
  >
    + New Workflow
  </button>
</div>
```

### Step 2: Show Workflows section even when 0 workflows exist

Currently the entire section is hidden when `filteredWorkflowGroups.length === 0`. Add an empty state so the section is discoverable:

```
{/* Always show the Workflows section once there's at least one agent */}
{standaloneCharacters.length > 0 || workflowGroups.length > 0 ? (
  <>
    {/* Section header — always visible */}
    <div className="flex items-center gap-3 mb-4">
      ...same header as above...
    </div>

    {/* Workflow cards — only when they exist */}
    {filteredWorkflowGroups.length > 0 ? (
      filteredWorkflowGroups.map((wf) => (
        ...existing workflow card render...
      ))
    ) : (
      <p className="font-mono text-xs text-terminal-muted mb-6">
        No workflows yet. Create your first multi-agent workflow above.
      </p>
    )}
  </>
) : null}
```

### Step 3: Remove the old standalone "Create Workflow" button block

The old button at lines 1352-1364 (guarded by `allStandaloneCharacters.length > 0`) is now replaced by the inline button in the section header. Remove the standalone button block.

### Step 4: Add "Agents" section header

Before the agent card grid (the `<div ref={gridRef} ...>` at line ~1621):

```
{/* Agents section header */}
<div className="flex items-center gap-3 mb-4 mt-6">
  <h2 className="font-mono text-xs font-semibold tracking-widest text-terminal-muted uppercase
whitespace-nowrap">
    {/* Add i18n key: t("agents.sectionTitle") → "Agents" */}
    {t("agents.sectionTitle")}
  </h2>
  <div className="flex-1 h-px bg-terminal-border/40" />
</div>
```

## Search Bar Positioning

**Do NOT move the search bar** into the Agents section header. The current `searchQuery` state filters BOTH standalone agents AND workflows. Moving it visually under "Agents" would confuse users trying to search for a workflow by name.

Options (pick one):

1. **Keep in current position** above both sections — simplest, no risk
2. **Rename the label** to `"Search agents & workflows..."` to accurately reflect its scope
3. **Add two separate search inputs** (one per section) with separate state — most complex

Recommended: Option 2 (rename label only, keep position).

---

## Replace `confirm()` Dialogs with AlertDialog

Two destructive workflow actions currently use browser `confirm()` — inconsistent with the rest of the UI which uses Shadcn `<AlertDialog>`:

- `removeSubagentFromWorkflow` (line ~974): `if (!confirm("Remove this sub-agent...")) return`
- `deleteWorkflowGroup` (line ~985): `if (!confirm("Delete this workflow group?...")) return`

Replace both with `<AlertDialog>` components consistent with how agent deletion is handled. This is part of the "unique visual touch" polish.

---

## Workflow Card Visual Notes

Status color coding already fully implemented at lines 1377-1382:

```
const statusColor =
  wf.status === "active" ? "bg-green-100 text-green-700 border-green-200" :
  wf.status === "paused" ? "bg-yellow-100 text-yellow-700 border-yellow-200" :
                           "bg-gray-100 text-gray-500 border-gray-200";
```

This renders as a `<Badge>` — no work needed. Only missing piece: dynamic border color on the `<Card>` element itself (currently always `border-terminal-border`).

Collapse by default: already implemented — `expandedWorkflows` initializes as `new Set()`. The one auto-expand heuristic (single workflow auto-expands) at lines 873-876 should be preserved.

Avatar overlapping circles: Use `<Avatar>` / `<AvatarFallback>` components, since many agents have no image and rely on initials fallback. Overlap via negative margin: `className="-ml-2 first:ml-0"`.

## Deleted Initiator Agent Handling

Schema: `ON DELETE CASCADE` on `agentWorkflows.initiatorId` — if the initiator is deleted, the entire workflow row is deleted. No orphaned-workflow state in DB.

UI gap: if a workflow is fetched and the initiator is then deleted before re-fetch, `initiator` computed at line 1374 will be `undefined`. The "Run" and "Share Folder" buttons both check `if (initiator)` and silently do nothing. Add a placeholder:

```
{!initiator && (
  <Badge variant="destructive">Agent deleted</Badge>
)}
```

## i18n Keys Needed

Add to both `locales/en.json` and `locales/tr.json`:

```
"agents": {
  "sectionTitle": "Agents"
}
```

The `workflows.sectionTitle` key already exists. The "+ New Workflow" button text can be a hardcoded string or a new key `workflows.newWorkflow`.

## Full Home Page Layout (ASCII Wireframe)


---

## Verification Steps

1. Home page → WorkspaceDashboard (if enabled) appears above Workflows section
2. Workflows render under "—— Workflows ——" section header with divider
3. "+ New Workflow" button appears inline in the Workflows header
4. "+ New Workflow" disabled when all agents are in workflows (no standalone agents left)
5. Zero workflows → section header visible with empty-state message
6. Standalone agents render under "—— Agents ——" section header with divider
7. Search bar visible above both sections (not moved to Agents only)
8. `confirm()` dialogs replaced with `<AlertDialog>` for workflow destructive actions

---

## Gap Analysis & Missing Considerations

> *The following were identified by codebase research on 2026-02-19 and have been incorporated into the plan above. Kept here for historical reference.*

| # | Issue | Resolution |
|---|---|---|
| 1 | `setShowCreateWorkflow` doesn't exist | Changed to `setWorkflowCreatorOpen` |
| 2 | Workflows section header already partially exists | Plan updated to upgrade it, not create from scratch |
| 3 | "New Workflow" button guard broken when section moves | Button disabled when `allStandaloneCharacters.length === 0` |
| 4 | Workflows vs Workspaces conflated | Section added distinguishing the two systems |
| 5 | Search bar filters both agents AND workflows | Keep in current position; rename label to reflect scope |
| 6 | Collapse-by-default already implemented | Noted; preserve auto-expand heuristic for single workflow |
| 8 | `workflowCreatorOpen` dialog needs standalone agents | Button disabled when `allStandaloneCharacters.length === 0` |
| 9 | One-workflow-per-agent constraint | Agent-already-in-workflow 400 errors must be handled in "Add to Workflow" UI |
| 10 | Deleted initiator leaves silent UI bug | Add `{!initiator && <Badge variant="destructive">Agent deleted</Badge>}` |
| 11 | Status color coding already implemented | No CSS work needed; only card border color is missing |
| 14 | `confirm()` dialogs for destructive actions | Replace with `<AlertDialog>` |

# 07 — Slash Command Skill Picker in Chat Input

## Problem

There is no way to discover or invoke skills directly from the chat. Users either go to Settings → Plugins or drop a file into the chat. The meeting requested a marketplace-like experience accessible from the chat input with a `/` keystroke.

**Meeting quote:**

> *"I know that one agent does it in its chat session where the prompt input field is — you just write slash and you see all available skills in your database." "Maybe we can do a Mac app search — Spotlight search — where you press command and on the chat sessions again like a marketplace component modal you would see all the skills listed."*

---

## Critical: What `/` Picker Inserts

**The `/run skillName` command does NOT work as a chat message.** The agent invokes skills exclusively through the `runSkill` tool (called internally by the LLM when it infers intent). There is no parser that converts `/run skillName` text into a tool call.

On skill selection, insert **natural language** instead:

```
Run the realEstateScraper skill
```

The agent's system prompt (from `formatSkillsForPromptFromSummary()`) already tells the LLM to call `runSkill` when user intent matches a trigger example. Natural language reliably triggers this.

Alternative for a more structured approach: call `threadRuntime.append({ role: "user", content: "Run the realEstateScraper skill" })` directly, bypassing the textarea entirely.

---

## Target Behavior

```
User types "/" at the start of a message (or after a space):


┌──────────────────────────────────────────────────────────┐
│  Available Skills                    [🔍 search...]      │
│  ──────────────────────────────────────────────────────  │
│  ▶ realEstateScraper    real estate image → renovation ideas  │
│    researchAgent        deep web research + summarize         │
│    socialMediaBot       post to social media on schedule      │
│    dataAnalyzer         analyze CSV/Excel files               │
│    meetingNotes         transcribe and summarize meetings     │
│  ──────────────────────────────────────────────────────  │
│  ↑↓ navigate   Tab/Enter to select   Esc to close        │
└──────────────────────────────────────────────────────────┘


   [/ realEstate..._____________________________________]  [Send]
```

When selected, inserts:

```
Run the realEstateScraper skill
```

(not `/run realEstateScraper`)

---

## Implementation

### Architecture Notes Before Starting

1. **Chat input is a `<textarea>` in `components/assistant-ui/thread.tsx`** — confirmed at line 1663. Not contentEditable.
2. **`characterId` comes from context** (`useCharacter()` at line 1026) — not a prop on `Composer`
3. **Popover must be placed OUTSIDE `ComposerPrimitive.Root`** — same pattern as `FileMentionAutocomplete` at line 1618 — otherwise it's clipped by the composer box dimensions
4. **No `command.tsx` or `cmdk` library exists** — follow `FileMentionAutocomplete` pattern (custom div + buttons)

### Step 1: Add Slash Detection State to Composer

In `components/assistant-ui/thread.tsx` — find the `Composer` function:

```
// Add state:
const [showSkillPicker, setShowSkillPicker] = useState(false);
const [skillPickerQuery, setSkillPickerQuery] = useState("");
const [selectedSkillIndex, setSelectedSkillIndex] = useState(0);
```

### Step 2: Detect `/` Using Cursor Position (Not Full String)

**Critical:** Run the regex against `inputValue.slice(0, cursorPosition)` — not the full string. The `cursorPosition` state already exists in the Composer (it's maintained in `onChange`). This prevents false triggers when `/` appears mid-string with cursor elsewhere.

```
const handleInputChange = (value: string, cursor: number) => {
  setInputValue(value);
  setCursorPosition(cursor);

  // Detect "/" at start or after whitespace — slice to cursor position only
  const textToCursor = value.slice(0, cursor);
  const slashMatch = textToCursor.match(/(^|\s)\/(\w*)$/);
  if (slashMatch) {
    setShowSkillPicker(true);
    setSkillPickerQuery(slashMatch[2] || "");
    setSelectedSkillIndex(0);
  } else {
    setShowSkillPicker(false);
    setSkillPickerQuery("");
  }
};
```

### Step 3: Load Skills

```
const { character } = useCharacter();  // already called in Composer

const [skills, setSkills] = useState<SkillRecord[]>([]);

useEffect(() => {
  // Gate on character id existing and not being the "default" placeholder
  if (!character?.id || character.id === "default") return;

  // status=active to exclude draft and archived skills
  resilientFetch<{ skills: SkillRecord[] }>(
    `/api/skills?characterId=${character.id}&status=active`
  ).then(({ data }) => {
    if (data?.skills) setSkills(data.skills);
  });
}, [character?.id]);

// Filter by query
const filteredSkills = skills
  .filter((s) =>
    s.name.toLowerCase().includes(skillPickerQuery.toLowerCase()) ||
    s.description?.toLowerCase().includes(skillPickerQuery.toLowerCase())
  )
  .slice(0, 8);
```

### Step 4: Integrate into `handleKeyDown` Delegation Chain

The existing `handleKeyDown` at lines 1148-1158 already delegates to `FileMentionAutocomplete` via a forwarded-ref `handleKeyDown` method. Add the slash picker to the **same chain**:

```
const handleKeyDown = (e: React.KeyboardEvent) => {
  // 1. @mention handler first (existing)
  const mentionHandler = (mentionRef.current as unknown as { handleKeyDown?: ... }).handleKeyDo
        wn;
  if (mentionHandler && mentionHandler(e)) return;

  // 2. Slash skill picker handler
  if (showSkillPicker) {
    if (e.key === "ArrowDown") {
      e.preventDefault();
      setSelectedSkillIndex((i) => Math.min(i + 1, filteredSkills.length - 1));
      return;
    }
    if (e.key === "ArrowUp") {
      e.preventDefault();
      setSelectedSkillIndex((i) => Math.max(i - 1, 0));
      return;
    }
    if (e.key === "Enter" || e.key === "Tab") {
      // Both Enter and Tab select — consistent with FileMentionAutocomplete
      e.preventDefault();
      if (filteredSkills[selectedSkillIndex]) {
        selectSkill(filteredSkills[selectedSkillIndex]);
      }
      return;
    }
    if (e.key === "Escape") {
      setShowSkillPicker(false);
      return;
    }
  }

  // 3. Default behavior (existing submit on Enter, etc.)
  ...existing handlers...
};
```

### Step 5: On Skill Selected

```
const selectSkill = (skill: SkillRecord) => {
  // Insert natural language — NOT "/run skillName"
  const textToCursor = inputValue.slice(0, cursorPosition);
  const textAfterCursor = inputValue.slice(cursorPosition);

  // Replace the trailing "/ query" in the text-to-cursor portion
  const newTextToCursor = textToCursor.replace(/(^|\s)\/\w*$/, (match) => {
    const prefix = match.startsWith(" ") ? " " : "";
    return `${prefix}Run the ${skill.name} skill `;
  });

  setInputValue(newTextToCursor + textAfterCursor);
  setShowSkillPicker(false);

  // Focus back on input
  setTimeout(() => inputRef.current?.focus(), 0);
};
```

### Step 6: Skill Picker Popover UI

**Placement:** Outside `ComposerPrimitive.Root`, inside the outer `<div className="relative w-full">` — same position as `FileMentionAutocomplete`:

```
{/* Placed BEFORE ComposerPrimitive.Root, not inside it */}
{showSkillPicker && (
  <div
    className="absolute bottom-full left-0 right-0 mb-2 z-50 rounded-lg border border-terminal-
border bg-terminal-cream shadow-lg font-mono text-sm overflow-hidden"
  >
    {/* Header */}
    <div className="flex items-center justify-between px-3 py-2 border-b border-terminal-borde
r/40">
      <span className="text-xs text-terminal-muted font-semibold tracking-wider uppercase">
        Skills
      </span>
      {skillPickerQuery && (
        <span className="text-xs text-terminal-muted">"{skillPickerQuery}"</span>
      )}
    </div>

    {/* Skill list or empty states */}
    <div className="max-h-52 overflow-y-auto">
      {skills.length === 0 ? (
        // No skills at all for this agent
        <div className="px-3 py-4 text-xs text-terminal-muted text-center">
          No skills available yet — drop a .md skill file into the chat,
          or visit Settings → Plugins.
        </div>
      ) : filteredSkills.length === 0 ? (
        // Skills exist but none match the query
        <div className="px-3 py-4 text-xs text-terminal-muted text-center">
          No skills match "{skillPickerQuery}"
        </div>
      ) : (
        filteredSkills.map((skill, i) => (
          <button
            key={skill.id}
            className={cn(
              "w-full flex items-start gap-3 px-3 py-2 text-left transition-colors",
              i === selectedSkillIndex
                ? "bg-terminal-green/10 text-terminal-dark"
                : "hover:bg-terminal-dark/5 text-terminal-dark/80"
            )}
            // IMPORTANT: onMouseDown + preventDefault to avoid blur race
```

```
            // DO NOT use onClick — textarea loses focus before click fires
            onMouseDown={(e) => {
              e.preventDefault();  // prevents textarea blur
              selectSkill(skill);
            }}
            onMouseEnter={() => setSelectedSkillIndex(i)}
          >
            <span className="text-terminal-green font-bold text-xs mt-0.5 shrink-0">/</span>
            <div className="flex-1 min-w-0">
              <span className="font-medium">{skill.name}</span>
              {skill.description && (
                <span className="ml-2 text-xs text-terminal-muted truncate">
                  {skill.description}
                </span>
              )}
            </div>
            <div className="flex items-center gap-2 shrink-0">
              {/* Show "requires input" indicator for skills with required parameters */}
              {skill.inputParameters && Object.keys(skill.inputParameters).length > 0 && (
                <span className="text-[10px] text-amber-500 border border-amber-200 rounded px-
1">
                  needs input
                </span>
              )}
              {skill.category && (
                <span className="text-[10px] text-terminal-muted">{skill.category}</span>
              )}
            </div>
          </button>
        ))
      )}
    </div>

    {/* Footer hint */}
    <div className="px-3 py-1.5 border-t border-terminal-border/40 text-[10px] text-terminal-mu
ted/60 flex gap-3">
      <span>↑↓ navigate</span>
      <span>Tab/Enter to select</span>
      <span>Esc to close</span>
    </div>
  </div>
)}
```

```
<ComposerPrimitive.Root ...>
  {/* existing textarea here */}
</ComposerPrimitive.Root>
```

---

## Skills API Endpoint

Use existing: `GET /api/skills?characterId={id}&status=active`

**Critical:** The `status=active` parameter is required. Without it, `listSkillsForUser` returns skills of ALL statuses (draft, active, archived). Draft skills the user is still editing would appear in the picker and fail when "run".

Response shape (non-`all` path):

```
{ "skills": [{ "id", "name", "description", "category", "icon", "inputParameters", "triggerExam
        ples", "runCount" }] }
```

**Note on plugin skills:** `GET /api/skills` returns only DB skills (created in the skill editor or from conversations). Plugin skills (installed from `.zip` packages) are resolved at runtime by `listRuntimeSkills()` and have no public REST endpoint. The picker will therefore show DB skills only. Plugin skills are discoverable via Settings → Plugins. Accept this limitation or add a new `GET /api/skills/runtime` endpoint that calls `listRuntimeSkills()`.

---

## Skills With Required Parameters

Skills with `inputParameters` show a "needs input" badge. When the user selects such a skill, the inserted text includes a hint:

```
Run the realEstateScraper skill (I'll need: listingUrl)
```

The agent will ask for the missing parameter in its response if the user sends without providing it.

---

## Popover Position

Position the popover **above** the input field ( `bottom-full` ) to avoid covering chat messages. The chat input is at the bottom of the viewport so upward expansion is always correct. The sticky container does not have `overflow: hidden`, so the absolute positioning works without a Portal.

## Verification Steps

1. Open a chat session
2. Type `/` → skill picker appears above input
3. Continue typing `/res` → list filters to matching skills
4. Arrow key up/down → selection moves
5. Press Tab or Enter → input updates to "Run the skill"
6. Press Escape → picker closes, input unchanged
7. Click on a skill (mousedown) → skill selected, focus returns to textarea (no blur race)
8. If agent has no skills → "No skills available yet" message shown
9. If query matches nothing → "No skills match '...'" message shown
10. Skills with required params show amber "needs input" badge
11. Click outside the picker → picker closes (blur on textarea)
12. Draft/archived skills do NOT appear in picker

---

## Gap Analysis & Missing Considerations

> *The following were identified by codebase research on 2026-02-19 and have been incorporated into the plan above. Kept here for historical reference.*

| # | Issue | Resolution |
|---|---|---|
| 1 | `/run skillName` not a parsed command | Changed to insert natural language |
| 2 | Regex on full string causes false triggers | Changed to slice to cursor position |
| 3 | `characterId` from context not prop | Use `useCharacter()` hook |
| 5 | Popover must be outside `ComposerPrimitive.Root` | Placement note added |
| 6 | No `command.tsx` — follow `FileMentionAutocomplete` pattern | Custom div pattern used |
| 7 | API returns all statuses without filter | `?status=active` added |
| 9 | Required params need visual indicator | "needs input" badge added |
| 10 | `handleKeyDown` must join existing delegation chain | Chain integration shown |
| 10b | `Tab` key not handled | Added Tab as selection key |
| 11 | Plugin skills not in `/api/skills` | Documented as known limitation |
| 12 | "No skills" vs "no match" distinction | Two separate empty states added |
| 13 | `onClick` causes blur race — use `onMouseDown` + `preventDefault` | Corrected in picker button |

# 08 — Deferred Items (Future Sprint)

These topics came up in the meeting and are important, but are explicitly NOT part of the current implementation sprint. They are documented here so nothing is lost.

---

## D1 — Node Graph Workflow Editor

**Meeting quote:**

> *"So what we need to make it better — we can think of how maybe ComfyUI handles this where they connect the nodes that will communicate with each other."*

**What it is:** Replace the current dropdown-based workflow creation (select main agent, select sub-agents one by one) with a visual canvas editor where agents are nodes and connections are drag-drawn edges.

**Why deferred:** Major UI component requiring a canvas library (reactflow, d3, or custom SVG). Does not block any current feature.

**Prior art to reference:**

- React Flow — most established, good docs
- ComfyUI's node editor
- n8n's workflow builder

**Engineering blockers to resolve before starting:**

1. **Schema migration required.** The current schema has no position data. `agent_workflows` has only `id`, `userId`, `name`, `initiatorId`, `status`, `metadata`, timestamps. `agent_workflow_members` has only `workflowId`, `agentId`, `role`, `sourcePath`, `metadataSeed`. A migration adding a `layout` JSON column to `agent_workflows` (for canvas zoom/pan state) and a `position` JSON column to `agent_workflow_members` (for per-node x/y) is required.
2. **One-workflow-per-agent constraint.** `assertAgentNotInActiveWorkflow` (`lib/agents/workflows.ts:387`) throws if an agent is already in an active workflow. A node graph that allows the same agent node to appear in multiple canvases simultaneously requires removing or relaxing this constraint at the backend level.
3. **Existing CSS tree visualization.** Lines 1568-1611 in `character-picker.tsx` render an initiator-to-subagents tree with CSS borders. A canvas editor would either replace this in a new route (`/workflows/[id]`) or coexist — either way the current visualization needs a plan.

4. **Folder sync latency.** Connecting agents in the graph triggers `createManualWorkflow` / `addSubagentToWorkflow` which runs folder sync operations. The UI must account for this latency (loading state, or defer sync to an explicit "Apply" step).

**When to pick up:** After doc 01-06 home page UX stabilizes. Estimate: next major sprint.

---

## D2 — Plugin Marketplace Content

**Meeting quote:**

> *"I think there is an opportunity to provide them with easy to install plugins — this might not be a hard task to just gather 300-500 folders of plugins from the internet." "Who coming to an agentic framework would expect plugins to work."*

**What it is:** Populate the existing (currently empty) marketplace with real, curated plugins.

**Current state of the marketplace:**

- The marketplace UI and infrastructure exist
- The one-click "Install" button ( `components/plugins/marketplace-browser.tsx` lines 175-178) is **stubbed** — it shows a toast instructing the user to manually download and upload the zip. The download-from-source pipeline (fetching from GitHub, URL, npm, pip) is not implemented. Engineering work is needed beyond "1-2 days to wire up an endpoint."

**Plugin format requirement for Duhan's content work:** Plugins must conform to the Anthropic Claude Code plugin standard. Any content gathered from GitHub/AutoGPT/LangChain that lacks a `.claude-plugin/plugin.json` manifest or `SKILL.md` file will be **rejected** by the import parser. A conversion/wrapping step is needed for most "found" plugins.

**Catalog format for Duhan:** The marketplace catalog must be a valid `MarketplaceManifest` JSON object (defined in `lib/plugins/types.ts` ). Deliver as a `marketplace.json` file, not a spreadsheet.

**Duhan's task (non-engineering):** Research and compile a database of 50-100 initial plugins. For each entry in `marketplace.json` :

- `name` (string, required)
- `description` (1-2 sentences, optional but important for UI)
- `version` (optional)
- `category` (research / productivity / social / finance / code / image)
- `tags` (array, optional)
- `source` (GitHub URL or direct .zip URL)

**Engineering tasks before marketplace content can ship:**

1. Build the download-from-source pipeline (fetch zip from GitHub/URL → pipe to import parser)

2. Add auto-fetch mechanism (there is currently no cron/background job to update catalog)
3. Reserve a name for the Seline default marketplace (unique constraint on name per user)

**Sources for Duhan to research:**

- Claude Code's skill/slash command ecosystem
- GitHub public repositories tagged with relevant topics
- Agent frameworks: AutoGPT, CrewAI, LangChain tools
- Custom Seline skills created internally (Umut's use cases: researcher, worktree, social media bot, AI girlfriend agent, real estate analyzer)

**When to pick up:** Duhan can start content curation in parallel. Engineering pipeline: next sprint after current UX sprint.

---

## D3 — Agent-Driven Onboarding

**Meeting quote:**

> *"What the talk is: we have a primary agent that is all preconfigured and user comes, starts chatting — and you will create the agent, you will set the tools, the agent will be persisted and ready to use in the future." "Is it going to be an agent-driven process or a human-driven process?"*

**What it is:** A conversational onboarding where the user chats with a meta-agent ("Seline Setup") that asks questions and automatically creates a configured agent for them. No wizard UI at all.

**Prior art:** Claude's own onboarding, ChatGPT's "Create a GPT" flow.

**Engineering gaps larger than originally noted:**

1. **No agent-callable tools for creating agents.** The full tool registry ( `lib/ai/tools.ts` ) has no `createAgent` , `setAgentTools` , `setAgentSyncFolder` , or `setAgentSystemPrompt` tools. These must be authored from scratch before any meta-agent can build agents.
2. **The existing `quick-create` endpoint is not reusable.** It's a one-shot HTTP call using `generateObject` . An agent-driven flow requires multi-turn streaming chat via `POST /api/chat` .
3. **Security escalation risk.** An agent that can create agents and assign tools is a privilege escalation vector. Required mitigations before building:
    - Tool creation calls must require human-in-the-loop approval (similar to memory approval flow)
    - The meta-agent's own `enabledTools` must be restricted to agent-management tools only (no shell execution, no file write)
    - Default to a safe tool set for newly created agents; require explicit user confirmation before enabling file/network tools

**When to pick up:** After D2's download pipeline is built (pre-populated agents from marketplace make D3 compelling). Needs a detailed security spec before implementation begins.

---

## D4 — Unified Per-Agent Settings Hub

**Meeting quote:**

> *"We should definitely combine the plugins, skills into a consistent page — both for customizing per-agents and both for seeing the marketplaces — manage them all in one place." "Currently if you go to edit one of them there is basic info, custom prompt — again all these are necessary but it should be put a bit further behind in a more unionized compact form."*

**What it is:** A single `/agents/[id]/settings` page with tabs:

- Identity (name, tagline, purpose, avatar, system prompt)
- Tools (capabilities, MCP)
- Knowledge (sync folders, vector search)
- Skills & Plugins
- Memory
- Schedules

**Why deferred:** The backend data is already accessible via individual routes. The work is UI reorganization — complex but not urgent for launch.

**What actually exists today:**

- `app/agents/[id]/memory/page.tsx` ✓
- `app/agents/[id]/schedules/page.tsx` ✓
- `app/agents/[id]/skills/page.tsx` ✓
- `app/agents/[id]/page.tsx` ✗ (missing — no index page)
- `app/agents/[id]/settings/page.tsx` ✗ (missing)
- `app/agents/[id]/layout.tsx` ✗ (missing — no shared tab bar)

**Migration complexity:**

- Six separate dialogs in `character-picker.tsx` (tool editor, identity editor, folder manager, MCP, plugin, delete) would migrate into tabs
- Internal links from `chat-sidebar`, `thread.tsx`, `schedule-list.tsx` point to old routes — need redirects or updates
- `Knowledge` tab has no existing route — must be built from scratch
- `Skills & Plugins` tab has no existing route — must be built from scratch

**When to pick up:** After launch stabilizes. Estimate: 2-3 weeks post-launch.

## D5 — Sharing Use Cases & Demo Agents

**Meeting quote:**

> *"We need screenshots, we need use case showcases, we need results, we need tests and we need comparisons with other agents."*

**Known use cases to document:**

1. **Researcher agent** (Umut) — Codebase research + web research
2. **Git worktree parallel agent** (Umut) — Multiple tasks in parallel GitHub work trees
3. **AI persona agent** (Umut) — Personal assistant on phone, sends videos, has memory, assigns tasks
4. **Social media automation** (Umut) — Monitors social platforms, converses with users overnight
5. **Real estate analyzer** (Duhan) — Scrapes listings, analyzes images, renovation recommendations
6. **Meeting notes** — Transcribe + summarize meetings (template agent)

**Duhan's task:** Record screen demos of items 5 and 6. Write 2-3 sentence descriptions for each use case for website/social content.

**Engineering gap:** There is no shareable agent export format. An agent's full config (tools, sync folders, system prompt, memory, schedules) cannot be exported to a file without bundling it as a plugin. If the use-case page is meant to let users "clone" a demo agent, this feature must be built separately. The existing plugin import system (`app/api/plugins/import/route.ts`) supports `PluginAgentEntry` definitions, which could be used as a workaround.

There is also no `/examples` or `/use-cases` page in the app. Assets in `seline-web/` consist only of `README.md`, `index.html`, `demo.gif`. Building a use-cases gallery or landing page page is a separate project.

**When to pick up:** In parallel with launch preparation.

---

## D6 — Launch Plan (Non-Engineering)

**Meeting decisions:**

- Launch target: March 4, 2026 (Product Hunt day — see doc 09)
- Need: budget allocation, platform list, content assets

See `09-launch-marketing-plan.md` for the full launch plan with corrected details.

---

## Dependencies Between Deferred Items

| | D1 | D2 | D3 | D4 | D5 |
|---|---|---|---|---|---|
| **D1** | — | — | — | — | — |
| **D2** download pipeline | — | — | Enables D3 | — | — |
| **D3** | — | Benefits from D2 content | — | — | — |
| **D4** | — | — | — | — | — |
| **D5** | — | — | — | — | — |

D2's download pipeline is the highest-leverage deferred engineering task — it unblocks D3's compelling demo and makes the marketplace actually usable.

# 09 — Launch & Marketing Plan

## Overview

Seline launch target: **March 4, 2026 (Tuesday)** — Tuesday is the best Product Hunt day Team: Umut (engineering + demos), Duhan (content + distribution + user research)

---

## Critical Pre-Launch Blockers (Must Fix Before Launch)

| # | Blocker | Owner | Status |
|---|---|---|---|
| 1 | **Mac code signing** — app shows "damaged/unidentified" error without it | Umut | README says "will sign in two days" — confirm done |
| 2 | **CONTRIBUTING.md** — open source projects need this for community building | Umut | Missing |
| 3 | **Landing page** — no place to capture waitlist emails before launch | Umut + Duhan | Missing |
| 4 | **Beta testers** — 10-20 testers before Mar 4 launch | Duhan recruits | Not planned |

---

## Messaging & Positioning

### Primary Target User (Pick ONE for launch)

**Developers who want AI agents that do real work on their machine.** Business users and power users are secondary audiences — launch with a focused developer message.

### "Runs locally" → Fix the Messaging

The current "Runs locally" claim is inaccurate. Seline stores data locally but requires cloud API keys (Anthropic, OpenRouter, etc.) for LLM calls. Fix to:

- **Correct**: "Local-first — your data, your API keys, your machine"
- **Correct**: "Open source desktop app. Bring your own API keys, pay only for what you use."
- **Avoid**: "Runs locally" / "Your data stays yours" (implies no cloud dependencies)

### Corrected Tagline

Current: "Build AI agents that actually work. Chat, code, research, schedule & automate. Open source. Runs locally."

**Recommended alternatives:**

- "Seline — AI agents that research, code, and message you the results. Open source desktop app." ← workflow angle
- "Seline — Your AI agents, running on your machine. Bring your own API keys." ← ownership angle
- "Seline — The open-source AI desktop app where agents actually ship work." ← developer angle

### Seline's Real Differentiators (Lead With These)

Based on codebase analysis vs. competitors, Seline's unique combination:

1. **Multi-channel bot deployment** — WhatsApp + Telegram + Slack + Discord from a desktop app (no competitor does this)
2. **Deep Research with cited sources** — 6-phase research workflow, Perplexity-quality, free/self-hosted
3. **Task scheduling with channel delivery** — set agent to research overnight, get results in your Telegram at 8 AM
4. **Video assembly** — no other AI agent platform generates videos with Remotion
5. **8 one-click agent templates** — most competitors have zero

### Cost Transparency (Address This Proactively)

Users will immediately ask "what does it cost?" — have an answer ready:

- **Seline itself**: Free (MIT license, open source)
- **API costs**: Typically 5 – 20/*monthformoderateusage*(*ClaudeAPI* 3-15/1M tokens)
- **Ollama**: Free (local models, requires GPU/CPU)

### Updated Comparison Table

| Feature | Seline | Open WebUI | AnythingLLM | Jan.ai | ChatGPT |
|---|---|---|---|---|---|
| Persistent agents | ✓ | ✗ | partial | ✗ | ✗ |
| File system access | ✓ | ✗ | ✓ (RAG only) | ✗ | sandbox only |
| Scheduling | ✓ | ✗ | ✗ | ✗ | ✗ |
| Multi-agent workflows | ✓ | ✗ | ✗ | ✗ | ✗ |
| Channel bots (WhatsApp/TG/Slack) | ✓ | ✗ | ✗ | ✗ | ✗ |
| Deep research | ✓ | ✗ | ✗ | ✗ | paid |
| Skills/plugins | ✓ | ✓ | partial | ✗ | ✓ (GPTs) |
| Local LLM (Ollama) | ✓ | ✓ | ✓ | ✓ | ✗ |
| Open source | ✓ | ✓ | ✓ | ✓ | ✗ |
| Desktop app | ✓ | ✗ | ✓ | ✓ | ✓ |
| Video generation | ✓ | ✗ | ✗ | ✗ | ✗ |

Note: Compare against Open WebUI, AnythingLLM, Jan.ai — not ChatGPT/Claude.ai (unfair comparison) or AutoGPT (lost mindshare since 2023).

---

## Phase 1: Pre-Launch Preparation (Feb 19 → Mar 3)

### 1.1 — App Assets (Umut + Duhan together)

**Screenshots needed (6 total):** | # | Screen | What to Show | |---|--------|-------------| | 1 | Home page | Clean agent grid, 3-4 agents, workflow section visible | | 2 | Chat in action | Agent mid-task, tool calls visible, clean UI | | 3 | Create Agent Modal | Quick create popup, minimal and fast | | 4 | Workflow tree | Multi-agent workflow, connection visible | | 5 | Real estate use case | Duhan's scraper result card output | | 6 | Settings / tools | Shows extensibility (many tools, clean layout) |

**Demo video (60-90 sec, Umut records):**

```
Script outline:
0–5s   : "Meet Seline — your AI agent platform"
5–20s  : Home screen tour → click Create Agent → type sentence → chatting in 10s
20–35s : Agent doing real work (web search + file edit in chat)
35–50s : Workflow: two agents running in parallel on different tasks
50–65s : Scheduling: agent runs overnight, you check results in the morning
65–75s : "Open source. Free to use. Your agents, your data."
```

**GIFs (Duhan records, 10-15 sec each):**

1. Create agent in 10 seconds (quick create modal)
2. Real estate analyzer output
3. Agent writing code + committing to git
4. Scheduling an agent task
5. *(New)* Agent delivering research results to Telegram/WhatsApp

**Short-form videos (15-30 sec "wow moment" clips — Umut records, Duhan distributes):**

- Agent creating a PR
- Scheduling a task in 10 seconds
- Agent researching and delivering to WhatsApp at 8 AM
- Post to YouTube Shorts, TikTok, Instagram Reels, X simultaneously

### 1.2 — App Description Copy (Duhan drafts, Umut reviews)

**Short description (300 words):**

```
Seline is an open-source AI agent platform that lets you build, customize, and run AI agents
for any task — on your own machine, with your own API keys.

Unlike cloud-based tools, Seline agents persist. They remember conversations, execute real
tasks on your computer, browse the web, manage files, run schedules, and work in multi-agent
teams. And they can message you results on WhatsApp, Telegram, or Slack.

Key capabilities:
• Chat with any LLM (Claude, GPT-4, local models via Ollama)
• Agents that read/write files, run code, and search the web
• Multi-channel bots: deploy an agent to WhatsApp, Telegram, or Slack
• Deep research: 6-phase research with cited sources (Perplexity-style, free)
• Schedule agents to run overnight and deliver results in the morning
• Multi-agent workflows where agents delegate to each other
• Build and share skills (custom mini-programs) to extend agent abilities
• Developer Workspace: agents work on git branches, submit PRs
• 8 ready-made templates: Social Media Manager, Data Analyst, Meeting Notes, and more

Built for developers, researchers, and power users who want to automate real work with AI.

Open source (MIT). Desktop app for Mac and Windows. Pay only for API usage — no subscription.
```

### 1.3 — Create Discord Server (Duhan, 2 weeks before launch)

Set up Discord with channels: `#announcements`, `#general`, `#show-your-agent`, `#bug-reports`, `#feature-requests`, `#agent-templates`. This is needed before launch since success metrics include "Discord joins."

### 1.4 — Landing Page with Email Capture (Umut + Duhan, 1 week before launch)

Simple one-page site with email signup to capture 200-500 pre-launch subscribers. These become Day 1 upvoters and stargazers. Even a Carrd.co or Notion page with an email form works.

### 1.5 — Beta Testing (Duhan recruits, Feb 28 → Mar 3)

Recruit 10-20 beta testers. Sources: personal network, X/Twitter, r/LocalLLaMA, AI Discord servers. Focus on: install issues on Mac/Windows, API key config confusion, first-run UX.

### 1.6 — Product Hunt Preparation (Duhan, 2+ weeks before launch)

- Start following/engaging on Product Hunt NOW (30+ days before launch)
- Build maker profiles with follower base
- Pre-launch page with follower capture

## Phase 2: Launch Day (March 4, 12:01 AM PST)

**Launch time: 12:01 AM PST** — NOT 6 AM. Product Hunt resets at midnight PST; launching at 12:01 AM gives the full 24-hour window.

### Priority Order

#### Tier 1 — High impact, same day

| Platform | Who | Action |
|---|---|---|
| **Product Hunt** | Duhan runs, Umut reviews copy | Full listing, 12:01 AM PST, maker "first comment" ready |
| **Hacker News (Show HN)** | Umut writes, posts | Technical angle: multi-agent + open source + channel bots |
| **X / Twitter** | Duhan | Thread with demo video + wow-moment clips |
| **GitHub** | Umut | README refresh with screenshots, badges, topics, use cases |

#### Tier 2 — Same week

| Platform | Who | Action |
|---|---|---|
| **Reddit** | Duhan | r/LocalLLaMA, r/SelfHosted, r/MachineLearning, r/SideProject |
| **LinkedIn** | Duhan | Company + personal post with story angle |
| **Discord servers** | Duhan | AI Tinkerers, LocalLLaMA Discord, etc. |

*Note: r/LocalLLaMA requires established account (≥2 weeks activity, <10% self-promotion). Start contributing now.*

#### Tier 3 — Week 2-3

| Platform | Who | Action |
|---|---|---|
| **YouTube** | Duhan (or Umut records) | 5-min walkthrough video |
| **Dev.to / Hashnode** | Umut | Technical writeup: "How I built a multi-agent platform" |
| **Indie Hackers** | Duhan | Share story, use case outcomes |
| **AppSumo** | — | Deprioritize — audience mismatch for developer tool with BYOK model |

### Newsletter Outreach (Duhan, 2 weeks before launch)

Reach out to at least 5 of these with a press kit (screenshots, description, founder story):

- **TLDR** (1.2M+ subscribers) — "cool tools" section
- **Ben's Bites** — AI-specific newsletter
- **The Rundown AI** — daily AI newsletter
- **Console.dev** — curates interesting open-source tools
- **Changelog** — podcast + newsletter, covers open source
- **JavaScript Weekly / Node Weekly** — Seline is Electron/Next.js/Node

### Developer Influencer Outreach (Duhan, 2 weeks before launch)

Specific targets to DM (cold outreach is free):

- **Fireship** (3M+ subs) — covers dev tools in "100 seconds" format
- **Matt Wolfe** — AI tools reviewer
- **All About AI** — covers open-source AI specifically
- **NetworkChuck** — covers self-hosted and local AI
- **Theo (t3.gg)** — covers Next.js/TypeScript ecosystem

### GitHub Trending Strategy (launch day)

- Add repo topics: `ai`, `agent`, `llm`, `desktop-app`, `electron`, `open-source`, `local-ai`, `mcp`, `whatsapp-bot`, `multi-agent`
- Push stars on launch day simultaneously with Product Hunt
- Ask early supporters to star the repo at 12:01 AM

---

## Phase 3: User Acquisition (Ongoing post-launch)

### Daily Commitment

- **Umut**: 10 min/day — respond to comments, DMs, GitHub issues
- **Duhan**: 30 min/day — engage with replies, share use cases, talk to 1 user

### Content Calendar (Duhan owns, Umut reviews)

**Week of launch:**

- Day 1 (launch day): Demo video + Product Hunt + HN + GitHub push
- Day 2: Real estate use case GIF + description
- Day 3: Git worktree parallel agent demo

- Day 4: Scheduling overnight agent use case
- Day 5: "How to build your first agent in 10 seconds"

**Ongoing weekly:**

- 1x use case showcase (video or GIF + description)
- 1x feature highlight (a tool, a capability — especially underrated features)
- 1x short-form wow-moment clip
- 1x response to community question/feedback

### User Research (Duhan)

- Talk to 1 user per day for the first 2 weeks post-launch (10-min call)
- Track: what they tried to do, what was confusing, what delighted them
- Share summary with Umut weekly
- Key question: "What would make you recommend Seline to a friend?"

---

## Phase 4: Budget Allocation

| Category | Suggested Budget | Owner |
|---|---|---|
| Apple Developer Program (Mac signing) | $99/year | Umut |
| Product Hunt promotion | $50-100 | Duhan |
| X/Reddit paid ads (only after seeing organic traction) | $100-200 | Duhan |
| Influencer/creator outreach | $0 (cold DMs first) | Duhan |
| Design assets (if freelancer needed) | TBD | Duhan |

**Rule:** No paid ads until organic launch shows traction signal (>100 GitHub stars, >200 PH upvotes). Measure first week, then decide.

---

## Work Distribution Summary

### Umut's Engineering Tasks (In Priority Order)

See docs 01-07 for full implementation specs.

1. **Doc 03** — Export `DEFAULT_ENABLED_TOOLS` from resolve-tools.ts (unblocks doc 01)
2. **Doc 01** — Create Agent Modal ( `components/character-creation/create-agent-modal.tsx` )

3. **Doc 02** — Wizard simplification (remove Knowledge step, merge Preview+Success)
4. **Doc 04** — Agent card `•••` overflow menu
5. **Doc 05** — Agent duplicate API + UI
6. **Doc 06** — Home page section headers (Workflows / Agents)
7. **Doc 07** — Slash skill picker in chat input
8. **Mac code signing** — Apple Developer enrollment, sign + notarize

### Umut's Marketing Tasks

- Record 60-90 sec demo video
- Record 5 short-form wow-moment clips (15-30 sec each)
- Record GIFs: worktree agent, scheduling, code agent
- Write HN "Show HN" post copy
- Refresh README with screenshots, topics, CONTRIBUTING.md
- Review all Duhan copy before publishing

### Duhan's Tasks (Parallel with dev)

**Week 1 (Feb 19-24):**

- [ ] Set up Discord server with channels
- [ ] Draft corrected app tagline + description (use updated messaging above)
- [ ] Draft comparison table (vs. Open WebUI, AnythingLLM, Jan.ai — not ChatGPT)
- [ ] Build Product Hunt page draft (screenshots from Umut when ready)
- [ ] Record real estate analyzer demo GIF
- [ ] Compile initial list of 50 plugins for marketplace (in `marketplace.json` format — see doc 08 D2)
- [ ] Create simple landing page with email capture

**Week 2 (Feb 24-Mar 3):**

- [ ] Recruit 10-20 beta testers, run tests
- [ ] Schedule Product Hunt for Mar 4, 12:01 AM PST (not 6 AM)
- [ ] Prepare Reddit posts (different angle per community, establish account first)
- [ ] Write LinkedIn post (personal story angle)
- [ ] Newsletter outreach to 5+ newsletters with press kit
- [ ] DM influencer/YouTuber targets
- [ ] X/Twitter thread draft with embedded demo video
- [ ] Prep "maker first comment" for Product Hunt page

**Launch Day (Mar 4):**

- [ ] Post all content simultaneously at 12:01 AM PST
- [ ] Respond to all comments within 15 minutes

- [ ] Ask supporters to upvote and star on GitHub

**Ongoing:**

- [ ] Talk to 1 potential user per day (10 minutes)
- [ ] Document feedback in shared notes doc
- [ ] Maintain content calendar (3 posts/week)

---

## Timeline

```
Feb 19 (today) → Dev sprint starts + Duhan begins content prep
Feb 24 (Mon)   → Core dev features complete (docs 01–04 done)
Feb 28 (Fri)   → All features + assets ready; beta testers start
Mar 3 (Mon)    → Beta feedback addressed; Product Hunt page finalized
Mar 4 (Tue)    → LAUNCH DAY 12:01 AM PST (NOT 6 AM)
Mar 11         → Week 1 metric review; paid ads decision
```

---

## Success Metrics (First Week)

| Metric | Goal |
|---|---|
| Product Hunt upvotes | 200+ |
| GitHub stars (new) | 100+ |
| App downloads | 500+ |
| Discord joins | 50+ |
| HN upvotes | 50+ |
| Email signups (pre-launch) | 200+ |
| User interviews | 10+ |
| Waitlist/email conversion to install | >20% |

**Go/No-Go criteria for paid ads:** If GitHub stars < 100 after week 1, revisit messaging before spending on ads.

---

## Gap Analysis & Missing Considerations

> *The following were identified by codebase audit and platform research on 2026-02-19 and have been incorporated into the plan above. Kept here for historical reference.*

| # | Issue | Resolution |
|---|---|---|
| 1 | "Runs locally" is inaccurate — requires cloud API keys | Messaging corrected throughout |
| 2 | Comparison table has inaccuracies (ChatGPT has desktop app) | Table rebuilt vs. correct competitors |
| 3 | AutoGPT lost mindshare — wrong competitor | Replaced with Open WebUI, AnythingLLM, Jan.ai |
| 4 | Product Hunt 6 AM PST is wrong — should be 12:01 AM PST | Corrected |
| 5 | Mac code signing is launch blocker | Added as Critical Pre-Launch Blocker #1 |
| 6 | No Discord server planned | Added as Phase 1 task |
| 7 | No short-form video strategy | Added wow-moment clips to content plan |
| 8 | No newsletter outreach | Added newsletter list + Duhan task |
| 9 | No influencer outreach list with names | Added specific targets |
| 10 | No GitHub Trending strategy | Added repo topics + star coordination |
| 11 | No beta testing phase | Added beta testing (Feb 28 - Mar 3) |
| 12 | No landing page / email capture | Added as Critical Pre-Launch Blocker #3 |
| 13 | No cost transparency messaging | Added "Seline itself is free" narrative |
| 14 | AppSumo not suitable for BYOK developer tool | Deprioritized with explanation |
| 15 | Seline's unique features undersold | Updated description highlights channels, deep research, video |
| 16 | CONTRIBUTING.md missing for open source | Added as Critical Pre-Launch Blocker #2 |
| 17 | No "Why not just use ChatGPT?" narrative | Added as messaging recommendation |
| 18 | No go/no-go criteria for paid ads | Added metrics gate |