Multi-Workflow System — Hiring Test

Goal Build a small web app with a Dashboard and two workflows:

1. Estimates — turn artifacts (scope, transcript, notes) into a line-item estimate through a structured process.

2. Contracts — generate & review agreements using an LLM, guided by our policies, track versions/notes, and show concise before/after change proposals.

A Copilot side panel accepts natural language, can read/write across workflows (e.g., “use estimate X to draft an MSA/SOW”), and executes actions.

We care about what you ship and how you vibe-code (prompts, patterns, speed).

Ground Rules

- **You pick the stack.**
- **Persistence required.** Any DB/files.
- **Demoable:** either a hosted URL **or** local run in ≤10 minutes.
- **LLM keys:** include `.env.example` with placeholders (e.g., `OPENAI_API_KEY`, `ANTHROPIC_API_KEY`, etc.). No offline fallback required.

---

1. Dashboard

- Two cards: **Estimates** and **Contracts**, each with a count + last updated.
- Click through to each workflow's list page.

---

2. Estimates Workflow

Purpose Structured estimation process through six stages: Artifacts → Business Case → Requirements → Solution/Architecture → Effort Estimate → Quote. Each stage has entry criteria and approval gates.

Estimation process

1. **Artifacts:** Create project and attach ≥2 artifacts (transcripts, documents, notes). Advance when ready.
2. **Business Case:** LLM generates business case from artifacts (scope, outcomes, constraints). Edit and approve to advance.
3. **Requirements:** LLM generates requirements summary from artifacts. Edit and validate to advance.
4. **Solution/Architecture:** Document approach and solution architecture, tech stack, risks. Approve to advance.
5. **Effort Estimate:** LLM generates WBS with tasks, assigned roles, hours per task, and assumptions from all prior artifacts. Approve to advance.
6. **Quote:** Apply rates to roles, add payment terms and timeline. Export CSV or copy-to-clipboard. Mark delivered.

Must-have screens

- **Projects list:** Show all projects with current stage indicator, filter by stage.
- **Project detail:** Stage stepper/progress indicator at top, current stage content panel, stage transition history timeline with timestamps/approvers.
- **Stage transitions:** Validation before advancing (e.g., artifacts uploaded, approval recorded), clear advance/approve buttons per stage.

Copilot commands context-aware per stage. All edits after LLM generation performed via agent.

---

3. Contracts Workflow

Purpose This workflow helps us create and review contracts the way we actually work. You add our policy rules and a couple of example agreements, then the system uses an LLM to draft new MSAs/SOWs that follow those rules. When a client sends their own draft, you can run a review that proposes only the material changes, showing exact before → after text with a one-line why. You can apply selected changes to produce a new version and keep a simple timeline of versions and notes.

Contract capabilities

- **Policy setup (do this first):** Add Policy Rules and Example Agreements (MSA, SOW, NDA, etc).

Then handle contracts in any order:

- **Create new agreement:** Generate LLM-powered agreement from policies:

    - Choose type (MSA or SOW) + counterparty
    - System feeds Policy Rules + Example Agreements to LLM
    - Save as Agreement v1

- **Review client draft:** Upload/paste client's agreement → save as incoming v1 → run policy-based review:

    - LLM returns material change proposals based on your policies
    - Each proposal shows exact `before → after` text + rationale
    - Apply selected changes to create new version

- **Validate against estimate:** Link SOW to an estimate to check alignment:

    - Agent validates against WBS (tasks, roles, hours) and quote (rates, totals, terms)
    - Returns discrepancies (e.g., missing WBS tasks, payment mismatch)

- **Version management:** Add new versions and notes; view timeline of all changes.

Must-have screens

- **Policy management:** List of policy rules and example agreements, add/edit/delete functionality.
- **Agreements list:** Show all agreements with type (MSA/SOW/NDA), counterparty, current version number.
- **Agreement detail:** Current agreement text, version selector/timeline on side, linked estimate indicator if applicable.
- **Review screen:** Side-by-side view showing proposed changes with before/after text, checkboxes to accept/reject each change.

Copilot commands context-aware. All edits after LLM generation performed via agent.

Nice-to-have (optional)

- Redline/diff preview (as enhancement to Review screen).
- DOCX/PDF generation (export feature from Agreement detail).
- "Send for signature" mock/adapter (action from Agreement detail).

---

4. Copilot (context-aware, cross-workflow)

Purpose
The Copilot is a right-side assistant you can talk to in plain English. It knows where you are (which agreement or project) and can read and update data—like adding notes, adjusting estimate lines, or applying specific change proposals. It also works across workflows (e.g., “Create a new MSA and SOW for Project Apollo; use the scope and estimate from the Estimates workflow”). Think of it as the command line for the whole app—fast, context-aware, and consistent.

Must-have

- **Contracts examples**
    - “Summarize pushbacks on this agreement.”
    - “Add a note: ‘Net 45 acceptable with 2% discount.’”
    - “Apply the payment-terms proposals and create a new version.”
- **Estimates examples**
    - "Increase Backend hours by 10%."
    - "Add a QA line: 40h at $90/hr, rationale 'regression pass'."
    - "What's the current total for Project Apollo?"
- **Cross-workflow examples**
    - “Create a new MSA and SOW for Project Apollo; use the scope and estimate from the Estimates workflow.”
        - (This should fetch the project’s artifacts/estimate and feed them into the Contracts LLM prompt to draft appropriate agreements.)
- Copilot must call your server functions/tools and **update the UI** accordingly (no manual refresh).

Deliverables

- **Repo** with:
    - `README.md`: hosted URL **or** ≤10-min local run steps; seed instructions.

- `AI_ARTIFACTS.md`: Document your AI-assisted development:
    - Include the actual prompts you used for each phase of your development process (planning, architecture, implementation, testing, etc.)
    - Show the progression from initial problem understanding to final implementation
    - Include at least one example where AI output required significant fixing and your follow-up prompts
    - Note which IDE/tool you used (Cursor, Claude Code, Copilot, etc.)
- `APPROACH.md` (1 page max):
    - Which AI tools you used and why
    - Your prompt strategy in 3-5 bullets
    - Biggest pivot/surprise during implementation
    - What you'd do differently with more time
- `TESTING.md`: how to run tests; include tests for at least 3 key server functions (e.g., proposal application, copilot action, estimate generation).
- `.env.example` with placeholder values for required keys (e.g., `OPENAI_API_KEY=your-key-here`). Do NOT include actual API keys.

- **Loom (≤15 min)**: Demo + quick process explanation:
    - Demo the working app
    - Show your IDE and walk through one complex prompt/refinement from AI_ARTIFACTS.md

---

Scoring (100 + bonus)

- AI assisted coding process (20): Structured approach (spec → architecture → implementation), effective prompt engineering, clear refinement strategy, documented pivots.
- Workflow completeness (25): Estimates (6-stage process with validations); Contracts (policy-guided draft, proposals, cross-reference with estimate).
- Copilot (25): Context awareness, 6+ real actions, cross-workflow creation (MSA/SOW from Estimates), solid error handling.
- Engineering quality (20): Clean model, validations, seeds, tests on ≥3 server funcs, clear structure.
- Product polish (10): Intuitive UI, sensible defaults, demonstrates AI leverage effectively.

- Bonus (+10): Redline/diff (+5), DOCX/PDF or signature mock (+5).