DidYouThough? – Accountability Engine

Product name DidYouThough? – Accountability Engine

One-liner Turn “I’ll take care of it” into a visible, trackable graph of tasks, decisions, and risks from meetings.

Product vision Help teams close the gap between what people say in meetings and what actually gets done. The product should capture action items from messy inputs, group them by people and initiatives, and give a simple view of who owes what.

Target users

- Product managers and team leads who run recurring syncs
- Project / program managers who own delivery across squads
- Founders and early stage teams who live in meetings
- Chiefs of staff who track cross-functional commitments

Problem statement and context

Scattered Tasks

Tasks are fragmented across notes, Slack messages, and individual memory. This makes it challenging to get a complete overview of active work items and progress.

Lack of Ownership Clarity

No single, clear view of who is responsible for what. This often leads to confusion, duplicated efforts, and tasks falling through the cracks.

Manual Follow-ups & Lost Decisions

Follow-ups require manual effort, and critical decisions get overlooked. Teams spend excessive time chasing updates instead of focusing on execution.

Why this matters: Business Impact

Missed Deadlines

Without clear task tracking and accountability, projects frequently run behind schedule, impacting delivery timelines.

Accountability Gaps

Unassigned or forgotten tasks create significant gaps in responsibility, leading to unfinished work and blame shifting.

Resource Inefficiency

Teams waste valuable time on manual follow-ups and redundant work, detracting from strategic priorities and productivity.

Scope

In scope (current version)

Meeting Content Ingestion

Audio uploads & pasted text

AI-Powered Processing

LLM for transcription & extraction

Interactive Dashboards

Tasks, people view, & meeting log

Modern UI

Animated dark-mode interface

Out of scope (for now)

User Management

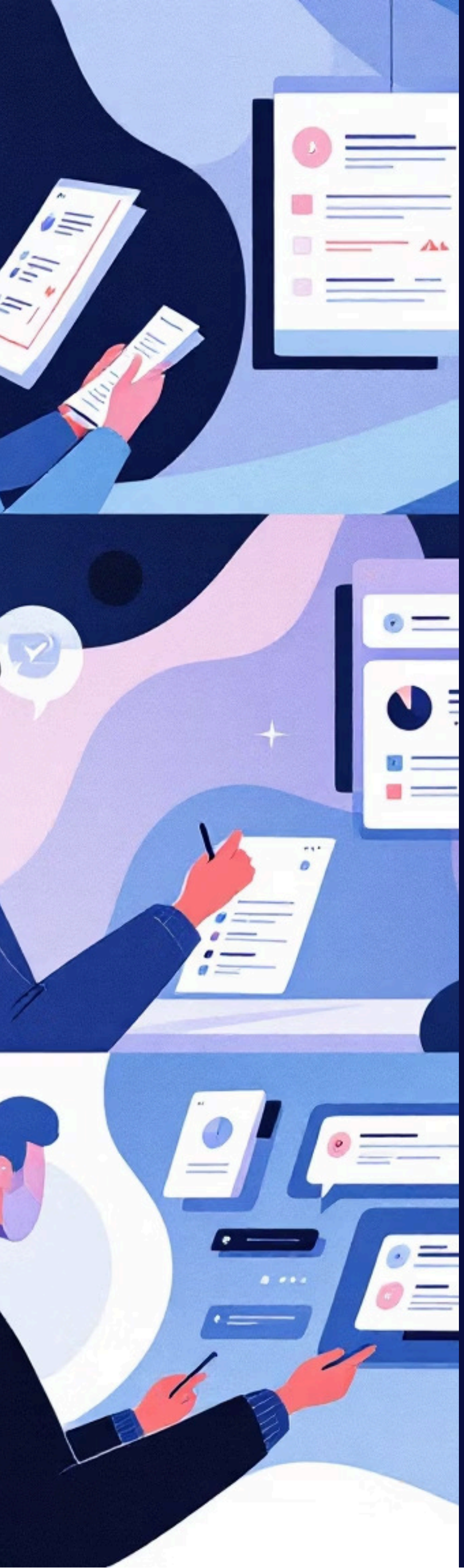
No persistence, accounts, or auth

External Integrations

No Slack, Jira, or similar APIs

Multi-Language & Mobile

No localization or dedicated app



Key user jobs

Capture commitments quickly from meetings

Users need to effortlessly turn meeting discussions into actionable items. This is crucial to prevent important decisions from being forgotten or miscommunicated, ensuring nothing falls through the cracks.

- Input: raw audio recording or rough notes
- Output: a clean list of tasks with owner, due date, priority, and initiative
- No extra admin work during or immediately after the meeting

Get visibility into who owns what

Clarity on task ownership is vital for team efficiency and accountability. Users need to quickly see who is responsible for each item to avoid duplication of effort and ensure timely completion.

- Clearly assign tasks to individuals
- Understand task distribution across the team
- Identify potential bottlenecks or overloaded team members

Track progress and send follow-ups

Managers and team leads require easy ways to monitor task progress and prompt updates. This streamlines workflows, reduces manual follow-ups, and keeps projects moving forward efficiently.

- Allow owners to mark tasks as complete
- Automate or simplify sending follow-up reminders via email
- Provide a clear overview of task statuses and impending deadlines

Report on completion and blockers

Regular reporting on task status and identifying blockers helps stakeholders stay informed and allows for proactive problem-solving. This ensures transparency and helps maintain project momentum.

- Summarize completed tasks and outstanding items
- Highlight critical blockers that require attention
- Maintain a historical record of progress for retrospectives

User flows

5.1 Capture a meeting from audio

01
Open Capture Tab

02
Upload Audio

03
Transcribe & Analyze

04
Extract Data

05
Store Tasks/Decisions

06
Success Message

5.2 Capture a meeting from text

01
Select Text/Notes

02
Paste Text

03
Analyze Text

04
Extract Data

05
Store Tasks/Decisions

5.3 Review and manage tasks (Dashboard)

01
Switch to Dashboard

02
View Metrics

03
View Charts

04
Filter Tasks

05
Mark Task Done

06
Metrics/Charts Update

5.4 View by person (People tab)

01
Go to People Tab

02
Select Person

03
View Person Card

04
View Tasks

05
Mark Task Done

06
Generate Follow-up

5.5 Review meeting history (Meeting Log)

01
Open Meeting Log

02
View Meetings

03
Expand Meeting

04
View Decisions

05
View Risks

06
Scan History

Functional requirements

Numbered so you can map to stories.

Capture

- User inputs meeting title and selects source (audio or text).
- Supports uploading audio files (MP3, M4A, WAV) for transcription.
- Accepts free-form text input for analysis.

Extraction

- Extracts tasks, decisions, and risks from audio or text.
- Outputs structured JSON data including task details like owner, due date, and priority.
- Gracefully handles errors during JSON parsing and data entry.

Dashboard

- Displays key metrics: open tasks, high priority tasks, initiatives, and meetings.
- Visualizes workload and priorities with donut and bar charts.
- Allows filtering of tasks by initiative, owner, and status.

People View

- Lists all unique task owners with pending and completed task counts.
- Shows a completion gauge for selected owners.
- Enables generating follow-up emails for an owner's open tasks.

Data Model

Task Object

Task ID: String
A unique identifier for each task.

Description: Text
A detailed explanation of the task to be completed.

Owner: String
The name of the individual responsible for the task.

Due Date: Date
The target date for the task's completion.

Priority: Enum (Low, Medium, High)
Indicates the urgency and importance of the task.

Initiative: String
The larger project or goal this task contributes to.

Status: Enum (To Do, In Progress, Done)
The current progress state of the task.

Source Meeting: Reference (Meeting ID)
Links the task to the specific meeting it was extracted from.

Created: Timestamp
The date and time when the task was first recorded.

Meeting Object

Meeting ID: String
A unique identifier for each meeting.

Date: Date
The calendar date when the meeting took place.

Name: String
The title or main topic of the meeting.

Type: Enum (Stand-up, Review, Brainstorm, etc.)
Categorizes the nature or purpose of the meeting.

Decisions: List of Text
A collection of key decisions made during the meeting.

Risks: List of Text
A collection of potential risks identified during the meeting.

Transcript: Text
The full transcribed text of the meeting discussion.

Relationship Overview

Each **Task** is directly linked to the **Meeting** from which it was extracted, providing clear context and traceability for all actions and discussions. This ensures that tasks are always associated with their origin point within the organizational workflow.

UX / UI requirements

Visual style

Key Design Principles:

- Dark theme with gradient background
- Glassmorphism cards with blur
- Rounded corners and soft shadows

High Priority

Medium Priority

Low Priority

Layout

Four main tabs for core navigation (Capture, Dashboard, People, Meeting Log). A dedicated left sidebar for branding, API key, global progress, and data management.

Micro-interactions

Subtle animations for live accountability (hero section pulse), task completion (confetti effect), and interactive hover states.

Accessibility

Ensuring text readability on dark backgrounds, avoiding tiny font sizes, and providing easy-to-click interactive elements.

Technical requirements

Stack

- Python
- Streamlit
- Pandas
- Groq Python client

External services

- Groq API key
- Whisper large v3 model
- Llama 3.x models

Performance

- Transcription latency driven by Groq; UI should show progress via st.status.
- Extraction call should run once per meeting input.
- UI should react fast when marking tasks complete (no extra API calls).

Security / privacy (MVP level)

- All data is in Streamlit session memory.
- No persistent storage.
- No direct PII masking yet.

Metrics and success

For v1, track (manually for now):

- Average number of tasks extracted per meeting
- Percentage of tasks marked complete within a week
- Share of high priority tasks completed vs created
- Number of meetings logged
- Number of unique owners engaged

Later, if telemetry is added:

- Time between meeting capture and first follow-up
- Repeat usage per user per week
- Drop-off points in the flow (capture vs dashboard vs people)

Roadmap (high level)

01

Now (v1, current code)

- Working capture, dashboard, people view, meeting log
- Session-scoped data
- Strong visual design and micro-interactions

02

Next (v1.5)

- Persistent storage (SQLite or Supabase)
- Simple editing of tasks (description, owner, due date, priority, initiative)
- Better date parsing and standard date picker for due dates
- Meeting tags and search in log

03

Later (v2+)

- User accounts and teams
- Slack / email integration (send follow-ups directly)
- Recurring meeting templates
- Role-based views (manager vs IC)
- Multi-language and time zone support

Gaps, Needs, and Solutions

This is the “honest gap analysis” for each area, presented as scannable stories.

Area 1 – Data persistence and reliability

Gap: All data lives in Streamlit session state, losing tasks/meetings on browser close or redeployment.

Need: Teams require tasks to persist across sessions and basic data durability.

Solution: Implement a lightweight database layer (e.g., SQLite/Supabase) to store and sync task/meeting data.

Area 2 – Multi-user and teams

Gap: The app assumes a single anonymous user with no concept of workspace or access control.

Need: Multiple teams need to use the app without seeing each other’s data, and team members should see shared tasks.

Solution: Add basic authentication and workspace IDs to filter tasks and meetings by active workspace.

Area 3 – Task editing and lifecycle

Gap: Task values are fixed after extraction; only status can be changed via checkboxes, with no editing of owner, priority, or due date.

Need: Teams need to clean up noisy AI extraction and adjust task details as work evolves.

Solution: Add inline editing for tasks (owner, priority, due date) and allow manual creation of new tasks.

Area 4 – Follow-ups and notifications

Gap: The app only generates email drafts, requiring manual sending, and lacks reminders or proactive nudges.

Need: Leads want automatic follow-ups to ensure tasks are completed without manual chasing.

Solution: Integrate with email/Slack for sending “nudge” messages and simple reminders for overdue tasks or weekly summaries.

Area 5 – Meeting capture fidelity

Gap: Extraction quality relies on one prompt and one LLM pass; there’s no way to review or correct raw JSON before saving.

Need: Users want to preview extracted tasks, fix errors, and accept them before committing to the database.

Solution: Show a “Review extracted tasks” screen with editable fields, and add a “Regenerate” button for poor outputs.

Area 6 – Search and navigation

Gap: There is no global search, and the meeting log is a simple list, preventing searches by keyword, initiative, or owner.

Need: Users need to quickly find information like past decisions or tasks related to specific initiatives.

Solution: Add a global search box with filters for meeting title, date range, initiative, and owner, and allow viewing related tasks from meetings.

Area 7 – Metrics and reporting

Gap: The dashboard lacks time trends and export options; the completion gauge is only a moment-in-time view.

Need: Leads require historical data to track improvement over time and share reports.

Solution: Use creation and completion timestamps to chart closure rates and task trends, and add CSV export for data.

Area 8 – Input flexibility and robustness

Gap: The current input expects "nice" audio/text, without explicit handling for very long transcripts, multi-language input, or Groq API errors.

Need: The tool should degrade gracefully and guide the user through failures.

Solution: Implement length checks with chunking for long text, handle Groq errors with clear messages, and detect multi-language input.

Area 9 – Governance and privacy

Gap: There is no clear story for data retention, redaction, or access logs; everything is local and in memory.

Need: Teams, especially in enterprises, require solutions for sensitive content and data management.

Solution: Add basic "soft delete" and retention settings with a database, allow users to redact/delete meetings, and document data flows.

Area 10 – Product positioning

Gap: The strong current experience lacks a clear "why this over X" narrative built into the product.

Need: Early users should clearly perceive the product's value compared to generic AI notetakers or heavy project tools.

Solution: Keep the UI focused on "commitments, not notes," reinforce the "Capture → Commitments → Follow-ups" story via copy, and add an onboarding walkthrough.