Git Repo: https://github.com/harshpreet14/journalling

---

1. Scope

Build a minimal web or mobile app that lets the user:

- Sign in with email and password.
- Record voice journals.
- Store audio files and transcripts:
  - Locally in the app, grouped by calendar day.
  - In Google Drive (audio + transcript) if authorized.
- Use AI to merge and reason over multiple recordings from the same day.

No marketing site, no extra pages. Straight to "Journal" and sign in.

---

2. Authentication and Landing Flow

Goals

- Simple sign-in.
- Single screen with "Journal" label, then sign-in form.

Tasks

1. Implement email/password authentication.
   - Create sign up and sign in with:
     - Email
     - Password
   - Basic validation and error states.
   - Password reset flow if needed.
2. Landing page layout:
   - Header text: "Journal".
   - Subsection: sign-in / sign-up form directly on the same screen.
   - No extra navigation, no marketing copy, no hero section.
3. Post-login behavior:
   - On successful login, go directly to the "Journal" screen listing entries by day.
   - Persist session so user does not have to sign in every time.

---

3. Journal Screen and UX

Goals

- Let user easily record a new entry.
- Show past entries grouped by day.
- Keep it utilitarian; function over aesthetics.

Tasks

1. Journal main view:
    - Top section: "Record new entry" with:
        - Record button (start / stop).
        - Recording timer indicator.
    - Below: list of days.
        - Each day shows:
            - Date (YYYY-MM-DD or localized).
            - Summary snippet of that day (e.g. first line of transcript or AI generated short summary if available).
            - Expand/collapse to show individual recordings for that day if needed.
2. Per-day drill-down:
    - When a day is expanded:
        - Show list of individual recordings:
            - Timestamp of each recording.
            - Short preview of transcript.
            - Playback button for the audio.
        - Option to see the merged "day-level" transcript / summary produced by AI.
3. Basic controls:
    - Delete recording.
    - Optional: rename a day or add a manual note.

---

4. Audio Recording and Local Storage

Goals

- Record audio from device mic.
- Store audio locally and tag it with correct date and time.

Tasks

1. Implement microphone recording:
    - Use appropriate API for platform (MediaRecorder or native mobile equivalent).
    - Capture audio in a reasonably compressed format (e.g. WebM/Opus or AAC).
2. Associate metadata:
    - For each recording, store:
        - id
        - user_id
        - created_at (timestamp)
        - local_path or blob reference
        - duration
        - Link to transcript record (when available)
3. Local persistence:
    - Implement a lightweight local database or storage:
        - Example: IndexedDB / SQLite / local file system depending on platform.
    - Ensure recordings and transcripts are available offline.
4. Background edge cases:
    - Handle cases where recording is interrupted or user navigates away.

---

5. Transcription

Goals

- Convert recorded audio to text.
- Prefer on-device transcription if realistic.

Tasks

1. Choose transcription approach:
    - Primary preference: on-device model (e.g. small Whisper variant or OS level speech-to-text) running locally.
    - Fallback: server-side transcription via API if on-device is not feasible. Make this configurable.
2. Integrate transcription pipeline:
    - After recording stops:
        - Trigger transcription job.
        - Store transcript text in local storage associated with the audio.
    - Show loading state while transcription is in progress.
3. Data model:
    - Transcript fields:
        - id
        - recording_id
        - text
        - created_at
        - Optional: language, confidence.
4. Error handling:
    - If transcription fails:
        - Keep audio anyway.
        - Show error and offer "retry transcription".

---

6. Google Drive Integration

Goals

- Automatically upload each recording and its transcript to Google Drive if user connects Google.
- Keep a clean folder structure in Drive.

Tasks

1. Google OAuth setup:
    - Add "Connect Google Drive" action in settings or journal screen.
    - Request appropriate Drive scopes with minimal permissions required.
    - Store and refresh tokens securely.
2. Drive file structure:
    - Create root folder, e.g. "JournalApp" (configurable).
    - For each day:
        - Optional: subfolder per date, e.g. JournalApp/2025-12-10/.
    - For each recording:

        - Upload audio file with naming scheme like `YYYY-MM-DD_HH-mm-ss_audio.ext`.
        - Upload transcript as `.txt` or `.md` with similar naming.
3. Sync logic:
    - When a new recording + transcript is ready:
        - Queue an upload job.
        - Retry on transient failures.
        - Mark local entry as `synced` or `sync_failed`.
4. Settings:
    - Allow toggling Drive sync on/off.
    - Show sync status indicators per entry.

---

7. AI Grouping and Day-Level Aggregation

Goals

- Group multiple recordings from the same calendar day into a single logical "day entry".
- Provide AI features that operate over all recordings in that day and across days.

Tasks

1. Day grouping logic:
    - Define grouping key: local calendar date based on `created_at` and user timezone.
    - All recordings whose timestamps fall on the same date are grouped.
    - If user records twice in a day or multiple days in one sitting:
        - Use timestamp of when the recording is created to assign to the correct date.
    - Handle edge cases around midnight and timezone.
2. Day-level aggregation:
    - For each day:
        - Merge transcripts of all recordings for that day in chronological order.
    - Save this merged transcript as a separate entity:
        - `day_id`, `date`, `full_text`.
3. AI processing per day:
    - Run an AI pass on the merged day transcript to:
        - Generate a short summary.
        - Optional: extract key themes, tasks, moods.
4. AI across multiple days:
    - Provide an interface to query over multiple days:
        - Example use cases:
            - "Show me themes from last week."
            - "What was I worried about most this month."
    - Backend or on-device vector search:
        - Index day-level transcripts and summaries.
        - Answer queries by retrieving relevant days and constructing responses.
5. Local-first preference:
    - If using an AI model:
        - Prefer running small models on device if the platform allows.
        - If using a cloud model, keep PII and raw audio exposure minimal and document that.

---

8. Local-first and Privacy Constraints

Goals

- Maximize on-device processing and storage.
- Minimize server dependencies.

Tasks

1. Architecture choices:
    - Keep user data (audio + transcript) stored locally by default.
    - Only send data off-device for:
        - Explicit Drive sync.
        - AI or transcription cloud services if on-device is not available.
2. Configurable cloud usage:
    - Settings to toggle:
        - "Use cloud transcription" on/off.
        - "Use cloud AI" on/off.
    - Clear warnings when data is sent off-device.
3. Documentation for user:
    - Simple explanation somewhere in-app:
        - Where data is stored.
        - When it goes to Google Drive.
        - When, if ever, it goes to external AI/transcription APIs.

---

9. Remove Unnecessary Homepage

Goals

- No marketing or landing homepage.

Tasks

1. Remove existing homepage route and assets.
2. Set application root route to the "Journal" sign-in screen.
3. Ensure navigation after login goes directly to journal list.
