**✅ V1 Feature List for *Timekin***

**📓 Core Journal Entry Features**

1. **Create Journal Entry**
   * Text-based entry (title + rich text or plain body)
   * Optional photo upload
   * Optional location tagging (manual or auto via device location)
   * Optional mood tag (emoji or color-based)
2. **View Journal Entries**
   * List of past entries with preview (date, title, thumbnail)
   * Detail page for each entry
3. **Edit/Delete Entry**
   * Full CRUD on journal entries
   * Soft delete (for future-proofing)

**🗄️ Data & Media Storage**

1. **Image Uploads**
   * Store uploaded media in cloud storage (e.g., AWS S3 or equivalent)
   * Return signed URL to client
2. **Location Tagging**
   * Integrate with location API or allow manual pin drop
   * Store coordinates and human-readable location

**👤 Basic User Management (Minimal)**

1. **Auth (Optional for V1)**
   * Assume single-user mode to avoid scope creep
   * Add auth later (Firebase Auth, Auth0, or custom OIDC) as bonus V2

**🧱 Infrastructure & DevOps**

1. **CI/CD Setup**
   * GitHub Actions deploy on push to main
   * Auto-deploy to a service (Render, Railway, or ECS)
2. **Infrastructure as Code**
   * Terraform provisioning: storage bucket, DB, and deployment infra

**🧪 Testing & Validation**

1. **Basic Unit Tests**
   * Cover API logic (entry creation, media upload)
   * Optional: frontend input validation tests

**🧾 Resume Bullet Outcome**

1. **Bullet-ready Deliverables**

* API built with Node.js + Express (or Fastify)
* DB integration (Postgres or MongoDB)
* Image storage via S3-compatible layer
* CI/CD to cloud
* Frontend in React/TypeScript, mobile-first

**🔚 Definition of Done (V1)**

* A user can create, view, edit, and delete journal entries with photo + location.
* Entries are stored in DB; photos in cloud storage.
* App is deployed, publicly accessible, with clean README and working CI/CD.
* All code is in GitHub. Project is buildable with one command.
* One-pager or video demo exists for portfolio.
* Resume bullet drafted based on completed work.

**🧭 Optional (Nice-to-Have) Guiding Docs**

These aren't essential but could help with structure, clarity, or collaboration later.

1. **Architecture Diagram**
   * Visual map of how frontend, backend, DB, storage, and infra interact
   * Helps with onboarding, storytelling, and debugging
2. **Entity/Data Model (ERD or JSON structure)**
   * Defines JournalEntry shape (text, image, mood, location, timestamp)
   * Helpful early for schema decisions and front-end planning
3. **API Contract (OpenAPI or Markdown spec)**
   * Route-by-route breakdown of expected request/response format
   * Optional but makes frontend/backend coordination cleaner
4. **.env Example File**
   * Early documentation of required secrets and environment config
5. **Personal Dev Journal (Logbook)**
   * Even 1–2 bullet points per week: What you did, learned, got stuck on
   * Becomes useful when interviewing or writing project posts
6. **Future V2 Ideas / Icebox**
   * Parking lot for features that are “nice-to-have later” like:
     + Audio recordings
     + Entry sharing / export
     + Search & filters
     + User auth
     + Tags or nested categories