📘 Timekin – Product Brief

1. Overview

Timekin is a personal journaling application focused on capturing life stories and preserving personal legacy through multimedia entries. Inspired by the absence of ancestral records in the founder’s own family, Timekin aims to give individuals a tool to document their lives in a meaningful and lasting way.

The MVP (V1) will serve as a functional and well-architected full-stack showcase project, with a focus on clarity, modularity, and future extensibility.

2. Value Proposition

A journal is more than just a daily log — it’s a living artifact. Timekin makes it easy for people to leave a rich legacy through:

Text-based entries

Photo/video attachments

Geolocation tagging

Clean, minimal user interface

Secure and durable cloud storage

This app should feel like an elegant tool for memory preservation — not just note-taking.

3. Target User

Individuals who want to privately record and preserve memories for future generations

Users who value simplicity and emotional significance over social features

Anyone who wants a journaling app without noise, ads, or dopamine-chasing UX

4. Product Goals (V1)

Build a complete full-stack app with proper infrastructure, storage, authentication, and API design

Create a user-friendly journaling interface with support for:

Writing entries

Uploading media (photos/videos)

Storing metadata (timestamp, optional location)

Viewing entries by date

Deploy on a public domain with CI/CD automation

Maintain clean code and documentation for portfolio/demo purposes

5. Scope of V1 (Definition of Done)

✅ Create, read, and delete journal entries

✅ Upload media assets and associate with entries

✅ Tag entries with timestamp and location (manual or browser-based)

✅ Secure API for journal data and media

✅ Store all data reliably (MongoDB for entries, S3 for media)

✅ CI/CD for backend and frontend

✅ README documentation, tech stack doc, and setup instructions

6. Constraints

Project must be completable within ~40 hours over 20 weeks

Focus is on clarity, modularity, and dev onboarding — not maximizing features

All features should be MVP-level: no mobile responsiveness, search, edit/update entries, or user sharing for V1

7. Success Metrics

✅ A new job or interview opportunity resulting from the project

✅ Strong resume bullet based on end-to-end technical execution

✅ Clear, compelling walkthrough of the project from tech and product perspective

✅ Optional: deployment metrics (e.g., uptime, performance, page load)

8. Guiding Principle

“Leave behind something worth remembering.”

Every decision should serve the goal of preserving moments with care, clarity, and future-friendly architecture.

🎓 20-Week Course Schedule (2 Days/Week, 2 Hours/Day)

This is now aligned with the full V1 feature list and “definition of done.”

PHASE 1 – FOUNDATION (Weeks 1–4)

Goal: Establish tooling, boilerplate, and workflows.

Week 1 – Setup & Planning

• Day 1: Repo setup, GitHub basics, README scaffold, local dev env

• Day 2: Pick deployment stack (Render/Railway), CI/CD hello world (GH Actions)

Week 2 – Frontend Skeleton

• Day 3: Create-react-app or Vite bootstrapping, page routing

• Day 4: Build static pages (Home, Journal List, Entry View, New Entry Form)

Week 3 – Backend API Skeleton

• Day 5: Express/Fastify API scaffold, test endpoint

• Day 6: Hook up frontend to fetch sample API data

Week 4 – DevOps & Infra as Code Intro

• Day 7: Create IaC plan (Terraform: DB, bucket, service), begin implementation

• Day 8: Apply Terraform to create S3 bucket or equivalent, connect storage

PHASE 2 – CORE FEATURES (Weeks 5–12)

Goal: Build out CRUD, storage, and media handling.

Week 5 – DB Setup

• Day 9: Choose schema (Postgres vs Mongo), define journal model

• Day 10: Migrate DB and test inserts from local API

Week 6 – Create Entry API

• Day 11: POST /entries — Accept text, image, metadata

• Day 12: Frontend journal form wired to backend POST

Week 7 – View Entries (List & Detail)

• Day 13: GET /entries — List view

• Day 14: GET /entries/:id — Detail view

Week 8 – Edit/Delete Entries

• Day 15: PATCH & DELETE endpoints

• Day 16: Frontend edit/delete wiring

Week 9 – Media Uploads

• Day 17: Setup signed S3 upload URLs

• Day 18: Allow image uploads in frontend, store S3 link in DB

Week 10 – Location Tagging

• Day 19: Frontend location picker / browser geolocation

• Day 20: Store geolocation + address in DB

Week 11 – Mood Tagging + Misc UX

• Day 21: Mood tags (emoji or palette), optional field

• Day 22: UX polish for create/edit forms

Week 12 – Test Coverage

• Day 23: Add unit tests to backend routes

• Day 24: Frontend form validation & snapshot tests

PHASE 3 – POLISH, DEMO, DEPLOY (Weeks 13–20)

Goal: Launch app, prepare portfolio/storytelling.

Week 13 – CI/CD Polish

• Day 25: Build-on-commit config (GitHub Actions)

• Day 26: Setup production vs staging environments

Week 14 – Production Deployment

• Day 27: Final deploy of frontend + backend

• Day 28: Smoke test full app in prod

Week 15 – UX Polish

• Day 29: 404 page, loading states, empty states

• Day 30: Accessibility and performance pass

Week 16 – README & Docs

• Day 31: Document API, setup, .env.example, architecture notes

• Day 32: Record Loom walkthrough or screenshots

Week 17 – Resume & Story Crafting

• Day 33: Draft 2–3 resume bullets

• Day 34: Create short LinkedIn post + project summary

Week 18 – Mock Interviews & Pitching

• Day 35: Practice explaining architecture in 1:00

• Day 36: Practice whiteboard: “How did you build this?”

Week 19 – Targeted Job Applications

• Day 37: Apply to 5 companies with project link

• Day 38: Customize resume to highlight project tech stack

Week 20 – Final Audit & Sprint Retrospective

• Day 39: Identify missing features or bugs, wrap up

• Day 40: Reflect, log lessons learned, plan V2 roadmap