Design Documentation: Journal Companion

1. Introduction

The **Journaling Companion Application** is a full-stack web system that supports reflective journaling, daily check-ins, and AI-generated prompts. The app is designed for users who want a private, guided journaling practice augmented by intelligent suggestions.

The system integrates:

- **React** on the frontend for interactive UI.
- **FastAPI** on the backend for a modern, async web service.
- **Postgres** for persistent data storage.
- **Groq AI (Llama 3.1)** for prompt generation.
- **Docker** for environment consistency and easy deployment.

This documentation outlines the design decisions, architecture, data model, and future considerations.

---

2. System Architecture

2.1 High-Level Overview

The app is structured as a three-tier system:

1. **Frontend (React + Vite)**
    - Provides an interactive interface for journaling, check-ins, and insights.
    - Manages authentication, renders prompts, and displays insights.

2. **Backend (FastAPI)**
    - Exposes RESTful endpoints for authentication, journal entry management, check-in storage, and AI prompt generation.
    - Implements JWT-based authentication for secure access.
    - Interacts with the database and calls Groq AI for prompt generation.
3. **Database (Postgres)**
    - Stores structured data: users, journal entries, and check-ins.
    - Exposed through connection pooling (`psycopg2.pool`).
    - Enforced with schemas to maintain relational integrity.
4. **AI Service (Groq + Llama)**
    - Receives journal history as input.
    - Generates one empathetic journaling question under 20 words.
    - Augmented with safety filters to avoid harmful prompts.

2.2 Deployment with Docker

- **Postgres**: containerized with persistent volumes.
- **Backend**: containerized FastAPI + Uvicorn server.
- **Frontend**: React app built with Vite, served via Nginx in a container.
- **docker-compose.yml** orchestrates the multi-service stack.

---

3. Backend Design (FastAPI)

3.1 Core Features

- **Authentication**
    - JWT-based, using `python-jose` and `passlib[bcrypt]`.
    - Routes: `/auth/register`, `/auth/login`, `/auth/me`.
- **Journal Entries API**
    - `POST /api/entries`: create entry.
    - `GET /api/entries`: list entries with pagination.
- **Check-ins API**
    - `POST /api/checkins`: record mood, stress, energy, struggles.
    - `GET /api/checkins`: retrieve past check-ins.
- **Prompt Generation API**
    - `POST /api/generate-prompt`: calls Groq AI Llama model.
    - Short-circuits to safe crisis messages if concerning terms detected.

3.2 Safety Layer

- Regex scanning detects keywords like *suicide, harm myself, overdose*.
- Crisis fallback prompts replace unsafe outputs.
- A hotline note is attached to all AI prompts for extra safety.

---

4. Database Design (Postgres)

4.1 Schema

- **Users Table**

    - id UUID PK
    - username TEXT UNIQUE
    - hashed_password TEXT
    - created_at TIMESTAMPTZ
- **Journal Entries Table**
    - id UUID PK
    - user_id UUID FK → users.id
    - title TEXT
    - content TEXT
    - mood TEXT
    - tags TEXT[]
    - is_private BOOLEAN
    - created_at TIMESTAMPTZ
- **Check-ins Table**
    - id UUID PK
    - user_id UUID FK → users.id
    - mood_score INT
    - stress INT
    - energy INT

    - `struggles TEXT[]`
    - `note TEXT`
    - `created_at TIMESTAMPTZ`

4.2 Access Layer

- **Connection pooling** via `psycopg2.pool`.
- Query functions for CRUD operations (e.g., `insert_entry`, `get_entries`).
- Queries parameterized to avoid SQL injection.

---

5. Frontend Design (React + MUI)

5.1 Pages

- **HomePage**
    - Displays the latest AI-generated prompt.
    - Allows creating new journal entries.
- **CheckInPage**
    - Collects mood, stress, energy, struggles, and reflection.
    - Stores results via `/api/checkins`.
- **InsightsPage**
    - Aggregates data from entries and check-ins.
    - Displays metrics:
        - Days journaled

        - Average mood, stress, energy
        - Most common struggles
        - Top emotions/topics
- **HistoryPage**
    - Lists past journal entries with timestamps.
- **LoginPage**
    - Provides user authentication with JWT.
    - Stores token in `localStorage`.

5.2 UI Components

- **Layout.tsx** → shared header, logout button, notification reminders.
- **PromptCard** → renders daily prompt.
- **StatCard** → reusable card for insights metrics.

5.3 Notifications (In-App)

- Implemented as a **Snackbar** that reminds the user to check in if they haven't logged one today.

---

6. AI Prompt Generation (Groq Llama)

6.1 Template

- Empathetic, short, user-centric journaling questions.
- Constraints:
    - ≤ 20 words.

    - Always ends with a question mark.
    - No lists, no prefaces, no quotes.

6.2 Examples

- Input: *“I felt calm walking by the lake.”*
- Output: *“What moments recently made you feel peaceful and why?”*

6.3 Crisis Handling

- If concerning text is present, skip the model.
- Respond with safe, supportive fallback questions (e.g., *“What support would feel most helpful to you right now?”*).
- Always append hotline note.

---

7. Security Considerations

- **JWT Authentication** protects all API routes.
- **Password Hashing** with bcrypt prevents plaintext storage.
- **CORS** restricted to frontend origin.
- **Parameterized Queries** prevent SQL injection.
- **Docker Isolation** keeps services sandboxed.

---

8. Future Work

- **Push Notifications** → real reminders outside the site.

- **User Preferences** → mood color themes, journaling goals.
- **Advanced Analytics** → streak tracking, mood trend charts, AI insights.
- **Full Sign Up Functionality** → Allow new users to make accounts.

---

9. Conclusion

The Journaling Companion Application integrates modern full-stack technologies to provide a safe, empathetic journaling experience. With Postgres for structured storage, FastAPI for secure API design, React for interactive UI, and Groq AI's Llama model for prompt generation, the system is both extensible and user-centered.

Dockerization ensures the entire stack runs reproducibly, making it easy to deploy locally or to the cloud. Safety layers and crisis handling protect users while maintaining the supportive role of the AI coach.

This design establishes a solid foundation for continued enhancements like notifications, deeper insights, and larger scale.