**Daily Journal Sentiment Tracker - Development Roadmap**

**Project Overview**

Building a daily journal sentiment tracker with Python Flask backend, HTML/CSS/JS frontend, TextBlob for sentiment analysis, and Chart.js for visualization.

**Tech Stack**

* **Backend**: Python + Flask + TextBlob
* **Frontend**: HTML/CSS/JavaScript + Chart.js
* **Data Storage**: SQLite (beginner-friendly)
* **Deployment**: Railway/Render (backend) + Vercel/GitHub Pages (frontend)

**Development Steps**

**Phase 1: Backend Setup (Flask API)**

* [ ] **Step 1.1**: Set up Python virtual environment
* [ ] **Step 1.2**: Install dependencies (Flask, TextBlob, Flask-CORS, sqlite3)
* [ ] **Step 1.3**: Create basic Flask app structure
* [ ] **Step 1.4**: Set up SQLite database with entries table
* [ ] **Step 1.5**: Create database initialization script
* [ ] **Step 1.6**: Build POST /api/entries endpoint (save journal entry + calculate sentiment)
* [ ] **Step 1.7**: Build GET /api/entries endpoint (retrieve all entries with sentiment)
* [ ] **Step 1.8**: Add CORS configuration for frontend communication
* [ ] **Step 1.9**: Test API endpoints locally

**Phase 2: Frontend Development**

* [ ] **Step 2.1**: Create basic HTML structure (index.html)
* [ ] **Step 2.2**: Add CSS styling for journal form and chart container
* [ ] **Step 2.3**: Include Chart.js library via CDN
* [ ] **Step 2.4**: Create JavaScript functions for:
  + Submitting new journal entries
  + Fetching entries from API
  + Processing data for Chart.js
* [ ] **Step 2.5**: Implement Chart.js line chart for mood visualization
* [ ] **Step 2.6**: Add form validation and user feedback
* [ ] **Step 2.7**: Test frontend locally with backend

**Phase 3: Integration & Testing**

* [ ] **Step 3.1**: Test complete flow (write entry → see sentiment → view chart)
* [ ] **Step 3.2**: Handle edge cases (empty entries, API errors)
* [ ] **Step 3.3**: Add loading states and error messages
* [ ] **Step 3.4**: Optimize chart display (date formatting, colors, etc.)

**Phase 4: Deployment**

* [ ] **Step 4.1**: Prepare backend for deployment (requirements.txt, environment variables)
* [ ] **Step 4.2**: Deploy Flask backend to Railway/Render
* [ ] **Step 4.3**: Update frontend API URLs to production backend
* [ ] **Step 4.4**: Deploy frontend to Vercel/GitHub Pages
* [ ] **Step 4.5**: Test production deployment end-to-end

**Phase 5: Enhancements (Optional)**

* [ ] **Step 5.1**: Add date filtering for chart
* [ ] **Step 5.2**: Implement basic authentication
* [ ] **Step 5.3**: Add entry editing/deletion
* [ ] **Step 5.4**: Improve UI/UX design
* [ ] **Step 5.5**: Add more sentiment metrics (subjectivity, emotion categories)

**Key Files Structure**

sentiment-tracker/

├── backend/

│ ├── app.py # Main Flask application

│ ├── database.py # Database setup and operations

│ ├── requirements.txt # Python dependencies

│ └── .env # Environment variables

├── frontend/

│ ├── index.html # Main HTML page

│ ├── style.css # Styling

│ ├── script.js # JavaScript logic

│ └── README.md # Frontend deployment instructions

└── README.md # Project documentation

**Quick Resume Points**

* **Current Phase**:2.7
* **Last Completed Step**: 1.9
* **Next Action**: degbugging
* **Issues/Notes**:   
  :5000/favicon.ico:1   
  Failed to load resource: the server responded with a status of 404 (NOT FOUND)

**Deployment URLs (Fill when ready)**

* **Backend API**: [Your Railway/Render URL]
* **Frontend**: [Your Vercel/GitHub Pages URL]

**Dependencies to Install**

# Backend

pip install flask textblob flask-cors

# Download TextBlob corpora

python -m textblob.download\_corpora

**Testing Commands**

# Test backend locally

python app.py

# Test API endpoints

curl -X POST http://localhost:5000/api/entries -H "Content-Type: application/json" -d '{"text":"I feel great today!", "date":"2024-01-01"}'

curl http://localhost:5000/api/entries