MoodBot API Documentation

MoodBot API Documentation

Chapter 1: MoodBot API Documentation

Overview:

MoodBot API is a FastAPI backend offering three core NLP endpoints-analyze mood, detect crisis, and summarize text-leveraging Hugging Face models. The backend securely manages API keys using environment variables and supports CORS for frontend integration. The frontend is a React app communicating with this backend via JSON POST requests.

Chapter 2: Step-by-step Explanation:

1. Project Structure

- backend/main.py: The FastAPI application defining the API endpoints. It contains:
  - Environment variable loading for the Hugging Face API token (using python-dotenv or platform secrets).
  - CORS middleware configuration to allow requests from frontend domains.
  - Endpoint definitions /analyze_mood, /detect_crisis, and /summarize that accept JSON payloads with text and return processed responses using Hugging Face inference API.
- frontend/src/App.js: React component managing user input, mode selection (analyze, crisis, summarize), and displaying responses fetched from the backend.
- .env (local or secret on cloud platforms): Stores sensitive environment variables like HF_TOKEN to avoid exposing secrets in code or version control.
- .gitignore: Ensures node_modules, .env, and other unnecessary or sensitive files are not committed to the repository.

2. Backend Setup

- Load environment variables securely using .env locally or secret management on platforms like Render or Vercel.
- Implement CORS middleware in FastAPI to allow frontend access, specifying allowed origins or enabling all during development.
- Define API routes with input validation via Pydantic models to ensure correct request formats.
- Use the Hugging Face API for inference, sending requests with proper authorization headers.

3. Frontend Setup

- Build the React UI with input box, buttons for mode selection, and output display.

- Replace the backend URL (e.g., http://localhost:8000 during local dev or deployed backend URL on Render) in fetch calls.
- Handle API responses and errors gracefully, showing messages when the backend is unreachable.

4. Deployment
- Backend: Deploy on platforms like Render or Heroku. Use their environment variable management to set your HF_TOKEN securely and enable CORS.
- Frontend: Deploy on Vercel, Netlify, or similar platforms. Ensure the frontend fetch URLs point to the live backend URL.
- Optionally, during local testing, use Ngrok or similar tunneling tools to expose your backend publicly for frontend integration.

5. Security and Best Practices
- Never commit .env or tokens to public repositories.
- Use platform secrets or environment variable settings for production.
- Implement proper CORS policies; restrict origins in production.
- Keep dependencies updated and monitor vulnerabilities.

6. Usage
- Send JSON POST requests to /analyze_mood, /detect_crisis, or /summarize with the payload: { "text": "your input text" }.
- Receive structured JSON responses with mood labels, crisis detection boolean, or summarized text.
- Integrate easily with any frontend or API client.

Chapter 3: This documentation should help developers understand the project architecture, set u