**🔧 Phase-Wise Development Plan for AI Chatbot MVP**

**⚙️ Phase 1: Project Skeleton & Basic Setup**

**Objective:** Establish base structure for frontend and backend and verify end-to-end connection.

**Tasks:**

* Create project directories:

bash

CopyEdit

/frontend

/backend

/shared (optional utilities, models)

* Initialize Git repo and README.md.
* Backend:
  + Choose Python framework: ✅ *Flask* (for simplicity) or FastAPI.
  + Create /chat route with dummy response: {"reply": "Hello, I'm a bot!"}
  + Enable CORS.
  + Load environment variables using python-dotenv.
* Frontend:
  + Use HTML/CSS/JS or React (recommended).
  + Build simple UI with:
    - Chatbox area
    - Input text field
    - Submit button
* Connect frontend to backend using **AJAX/Fetch**.

✅ **Checkpoint**: Typing a message in frontend displays bot's dummy reply without reloading the page.

**🤖 Phase 2: AI Integration**

**Objective:** Replace dummy logic with real AI model (OpenAI GPT or open-source).

**Tasks:**

* Setup OpenAI API (or other selected AI tool).
* Secure API key in .env file.
* Enhance /chat backend endpoint:
  + Accept user message
  + Forward to AI
  + Return response
* Keep in-memory session context (e.g., a session dictionary or store in global context).

✅ **Checkpoint**: Users can talk to real AI and receive contextual replies in the same session.

**💬 Phase 3: Real-Time Interaction + UI Feedback**

**Objective:** Make chat smooth and interactive.

**Tasks:**

* Add "Bot is typing..." animation before AI reply.
* Scroll chat view to latest message.
* Ensure responsive layout (mobile + desktop).
* Optional: WebSocket for real-time updates (advanced, can use simple polling/AJAX initially).

✅ **Checkpoint**: Smooth chat experience with dynamic UI feedback and responsiveness.

**🛠️ Phase 4: Error Handling & Robustness**

**Objective:** Handle edge cases and build reliability.

**Tasks:**

* Handle backend errors (e.g., OpenAI timeout, invalid input).
* Frontend:
  + Show error message on failure.
  + Disable input during response processing.
* Backend:
  + Sanitize inputs
  + Handle missing or malformed requests
* Add logging (e.g., using Python logging).

✅ **Checkpoint**: App gracefully handles failure scenarios without crashing.

**📚 Phase 5: Documentation & Testing**

**Objective:** Prepare deliverables and ensure code quality.

**Tasks:**

* Add requirements.txt and .env.example.
* Write README.md with:
  + Setup instructions
  + Technology choices
  + Assumptions and limitations
* Add basic API tests (optional, e.g., with pytest or unittest).
* Comment code and organize structure clearly.

✅ **Checkpoint**: MVP complete and ready for submission.

**🌟 Phase 6: Bonus Features (Optional, Time Permitting)**

**Objective:** Enhance with extra functionality.

**Optional Tasks:**

* **Chat History:**
  + Store in SQLite/PostgreSQL/Firestore.
* **GCP Integration:**
  + Host on App Engine or Cloud Run.
* **Advanced UI:**
  + Emoji support, voice-to-text input, file uploads.
* **Multilingual Support:**
  + Use language detection + translate queries.
* **Rate Limiting / Caching:**
  + Use Redis or rate limiter middleware.

✅ **Checkpoint**: Advanced version showing extra features.

**✅ Summary Table**

| **Phase** | **Focus** | **Key Deliverables** |
| --- | --- | --- |
| 1 | Setup Skeleton | Frontend+Backend, dummy connection |
| 2 | AI Integration | Real-time AI responses |
| 3 | UI Polishing | Typing animation, mobile-friendly |
| 4 | Error Handling | Robust experience |
| 5 | Documentation | README, .env, requirements.txt |
| 6 | (Bonus) Features | Chat history, GCP, UI/UX extras |

**🛠 Tools & Stack Suggestions**

| **Component** | **Tech** |
| --- | --- |
| Frontend | HTML/CSS/JS or React (recommended) |
| Backend | Flask or FastAPI (for async) |
| AI | OpenAI GPT-4 via openai package |
| Hosting | Localhost for MVP, GCP optional |
| Env Vars | python-dotenv |
| CORS | Flask-CORS |
| Optional DB | SQLite/PostgreSQL |