To run:  
  
Okay, let's break down how to run your project with the updated files and dependencies. This assumes you have Node.js, npm (or yarn/pnpm), and Python 3.x installed on your system.

1. Backend Setup (Node.js + Express):

* Navigate to Backend Directory: Open your terminal and navigate to the backend directory:

Bash

cd ai-chatbot/backend

* Install Dependencies:

Bash

npm install # or yarn install or pnpm install

* Run the Server:

Bash

npm start # or yarn start or pnpm start (if you added the scripts as suggested)

# OR for development (with nodemon):

npm run dev # or yarn dev or pnpm dev

This will start your Express server, likely on port 5000. You should see a message like "Server running on port 5000" in your terminal.

2. AI Service Setup (Python + Flask):

* Navigate to AI Service Directory:

Bash

cd ai-chatbot/ai\_service

* Create a Virtual Environment (Recommended): It's best practice to use a virtual environment for Python projects to isolate dependencies.

Bash

python3 -m venv .venv # Create a virtual environment (venv is a common name)

source .venv/bin/activate # Activate the virtual environment (Linux/macOS)

.venv\Scripts\activate # Activate the virtual environment (Windows)

* Install Dependencies:

Bash

pip install -r requirements.txt

* Set Environment Variable: You *must* set your Gemini API key as an environment variable. The code reads it from os.getenv("GEMINI\_API\_KEY").
  + Linux/macOS:

Bash

export GEMINI\_API\_KEY="your\_actual\_api\_key"

It is better to set the environment variable in your .bashrc or .zshrc file so it's available every time you open a terminal.

* + Windows:

PowerShell

$env:GEMINI\_API\_KEY = "your\_actual\_api\_key"

or set the environment variable in the System Properties.

* Run the Flask App:

Bash

python app.py

This will start your Flask app, likely on port 5001.

3. Frontend Setup (React + TypeScript):

* Navigate to Frontend Directory:

Bash

cd ai-chatbot/frontend

* Install Dependencies:

Bash

npm install # or yarn install or pnpm install

* Run the Development Server:

Bash

npm start # or yarn start or pnpm start

This will start the React development server. It usually opens your browser automatically to http://localhost:3000.

4. Putting It All Together:

1. Start Backend: In one terminal, run the backend server (npm start or npm run dev in the backend directory).
2. Start AI Service: In another terminal, activate the Python virtual environment (if you created one) and run the Flask app (python app.py in the ai\_service directory). *Make sure you've set the environment variable with your API key.*
3. Start Frontend: In a third terminal, run the React development server (npm start in the frontend directory).

Once all three are running, you should be able to access the chatbot in your browser at http://localhost:3000 (or whatever port your React app is running on).

Important Considerations:

* API Key: Double-check that you have correctly set the GEMINI\_API\_KEY environment variable. The AI service won't work without it.
* Ports: Make sure the ports are consistent. Your backend is using 5000, the AI service is using 5001, and the frontend will likely be on 3000. These should match the URLs in your code.
* CORS: For development, the cors() middleware in your backend is sufficient. But for production, you'll need to configure CORS more specifically to prevent security vulnerabilities.
* Testing: Test your chatbot thoroughly after getting it running. Try various inputs to make sure the communication between the frontend, backend, and AI service is working as expected.
* Error Handling: As mentioned before, the error handling is basic. Implement more robust error handling, especially in your backend and AI service, for a production-ready application.

--------  
  
Code:  
  
ai\_service/app.py:  
  
from flask import Flask, request, jsonify

import os

from dotenv import load\_dotenv

import google.generativeai # Correct import

load\_dotenv()

app = Flask(\_\_name\_\_)

# No need to instantiate anything

# gen\_ai = GoogleGenerativeAI() # Remove this line

@app.route("/ask", methods=["POST"])

def ask():

data = request.get\_json()

user\_message = data.get("message")

model = google.generativeai.getGenerativeModel(model="gemini-pro") # Use google.generativeai directly

result = model.generateContent({"contents": [{"role": "user", "parts": [{"text": user\_message}]}]})

reply = result.response.candidates[0].content.parts[0].text

return jsonify({"reply": reply})

if \_\_name\_\_ == "\_\_main\_\_":

app.run(port=5050)  
  
requirements.txt:

google-generativeai

python-dotenv

flask  
  
  
backend/.env

GEMINI\_API\_KEY = AIzaSyBNoUm9yStRsgXuNk1ZEIfYWA0w5OYYfyA  
  
  
backend/routes/chat.js:  
  
import axios from "axios";

import express from "express";

const router = express.Router();

router.post("/", async (req, res) => {

try {

const { message } = req.body;

const response = await axios.post("http://127.0.0.1:5001/ask", { message });

res.json({ reply: response.data.reply });

} catch (error) {

res.status(500).json({ error: "Error processing request" });

}

});

export default router;

backend/package.json

{

"type": "module",

"dependencies": {

"express": "^4.18.2",

"cors": "^2.8.5",

"dotenv": "^16.0.3",

"axios": "^1.3.4"

},

"devDependencies": {

"nodemon": "^3.0.1"

},

"scripts": {

"start": "node server.js",

"dev": "nodemon server.js"

}

}  
  
backend/server.js:

import cors from "cors";

import dotenv from "dotenv";

import express from "express";

import chatRoutes from "./routes/chat.js";

dotenv.config();

const app = express();

app.use(cors());

app.use(express.json());

app.use("/chat", chatRoutes);

const PORT = process.env.PORT || 5050;

app.listen(PORT, () => {

console.log(`Server running on port ${PORT}`);

});

frontend/public/index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<meta name="theme-color" content="#000000" />

<meta

name="description"

content="AI Chatbot using Google Gemini AI"

/>

<link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

<link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

<title>AI Chatbot</title>

</head>

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

<div id="root"></div>

</body>

</html>

frontend/src/api.ts

export const sendMessage = async (message: string) => {

const response = await fetch("http://localhost:5050/chat", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify({ message }),

});

return response.json();

};

frontend/src/app.tsx

export const sendMessage = async (message: string) => {

const response = await fetch("http://localhost:5000/chat", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify({ message }),

});

return response.json();

};

frontend/src/Chatbot.tsx

import React, { useState } from "react";

import { sendMessage } from "./api";

const Chatbot: React.FC = () => {

const [messages, setMessages] = useState<{ user: string; bot: string }[]>([]);

const [input, setInput] = useState("");

const handleSend = async () => {

if (!input.trim()) return;

const userMessage = input;

setInput("");

const response = await sendMessage(userMessage);

setMessages([...messages, { user: userMessage, bot: response.reply }]);

};

return (

<div style={{ maxWidth: "400px", margin: "auto", textAlign: "left" }}>

<div style={{ border: "1px solid #ccc", padding: "10px", minHeight: "300px" }}>

{messages.map((msg, index) => (

<p key={index}><strong>You:</strong> {msg.user} <br/><strong>Bot:</strong> {msg.bot}</p>

))}

</div>

<input

type="text"

value={input}

onChange={(e) => setInput(e.target.value)}

placeholder="Type a message..."

/>

<button onClick={handleSend}>Send</button>

</div>

);

};

export default Chatbot;  
  
frontend/src/index.tsx:

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App.tsx'; // Import your main App component

const root = ReactDOM.createRoot(document.getElementById('root') as HTMLElement); // Type assertion for TypeScript

root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);  
  
frontend/package.json

{

"name": "ai-chatbot-frontend",

"version": "1.0.0",

"description": "AI Chatbot using Google Gemini AI",

"main": "src/App.tsx",

"scripts": {

"start": "react-scripts start",

"build": "react-scripts build",

"test": "react-scripts test",

"eject": "react-scripts eject"

},

"dependencies": {

"axios": "^1.3.4",

"react": "^18.2.0",

"react-dom": "^18.2.0",

"react-scripts": "^5.0.1",

"typescript": "^4.9.5"

},

"devDependencies": {

"@types/react": "^18.2.20",

"@types/react-dom": "^18.2.6",

"dotenv": "^16.0.3"

},

"browserslist": {

"production": [

">0.2%",

"not dead",

"not op\_mini all"

],

"development": [

"last 1 chrome version",

"last 1 firefox version",

"last 1 safari version"

]

}

}

I have some errors in the code i am getting errors like "Failed to compile.

Module not found: Error: Can't resolve './api' in '/Users/navyasaijosyula/Documents/Gemini chatbot/frontend/src'

ERROR in ./src/Chatbot.tsx 6:0-36

Module not found: Error: Can't resolve './api' in '/Users/navyasaijosyula/Documents/Gemini chatbot/frontend/src'

webpack compiled with 1 error"

and "python app.py

/Users/navyasaijosyula/Documents/Gemini chatbot/ai\_service/.venv/lib/python3.9/site-packages/urllib3/\_\_init\_\_.py:35: NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled with 'LibreSSL 2.8.3'. See: https://github.com/urllib3/urllib3/issues/3020

warnings.warn(

\* Serving Flask app 'app'

\* Debug mode: off

WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.

\* Running on http://127.0.0.1:5050

Press CTRL+C to quit

127.0.0.1 - - [02/Feb/2025 17:47:09] "GET / HTTP/1.1" 404 -

127.0.0.1 - - [02/Feb/2025 17:47:10] "GET /favicon.ico HTTP/1.1" 404 -

[2025-02-02 17:52:30,396] ERROR in app: Exception on /ask [POST]

Traceback (most recent call last):

File "/Users/navyasaijosyula/Documents/Gemini chatbot/ai\_service/.venv/lib/python3.9/site-packages/flask/app.py", line 1511, in wsgi\_app

response = self.full\_dispatch\_request()

File "/Users/navyasaijosyula/Documents/Gemini chatbot/ai\_service/.venv/lib/python3.9/site-packages/flask/app.py", line 919, in full\_dispatch\_request

rv = self.handle\_user\_exception(e)

File "/Users/navyasaijosyula/Documents/Gemini chatbot/ai\_service/.venv/lib/python3.9/site-packages/flask/app.py", line 917, in full\_dispatch\_request

rv = self.dispatch\_request()

File "/Users/navyasaijosyula/Documents/Gemini chatbot/ai\_service/.venv/lib/python3.9/site-packages/flask/app.py", line 902, in dispatch\_request

return self.ensure\_sync(self.view\_functions[rule.endpoint])(\*\*view\_args) # type: ignore[no-any-return]

File "/Users/navyasaijosyula/Documents/Gemini chatbot/ai\_service/app.py", line 17, in ask

model = google.generativeai.getGenerativeModel(model="gemini-pro") # Use google.generativeai directly

AttributeError: module 'google.generativeai' has no attribute 'getGenerativeModel'

127.0.0.1 - - [02/Feb/2025 17:52:30] "POST /ask HTTP/1.1" 500 -

127.0.0.1 - - [02/Feb/2025 17:52:37] "GET / HTTP/1.1" 404 -

127.0.0.1 - - [02/Feb/2025 18:07:40] "GET / HTTP/1.1" 404 -

"