TD Chatbot - Student-Friendly Guide

Ye guide specially students ke liye banaya gaya hai jisse wo **TD Chatbot** ka backend aur logic samajh sake easily. Isme har technology, library, concept, flow, aur methods ka reason diya gaya hai.

---

1. Languages, Technologies & Libraries

Technology / Library	Purpose / Reason
Node.js	Backend JavaScript runtime environment. Server side logic run karne ke liye.
Express	Lightweight web framework. API routes create karne aur HTTP requests handle karne ke liye.
cors	Cross-Origin Resource Sharing. Frontend (browser) se backend requests allow karne ke liye.
body-parser	JSON requests parse karne ke liye. User ke messages backend me receive karne ke liye.
dotenv	Environment variables manage karne ke liye. API keys ko secure store karne ke liye.
openai	Connects backend to OpenAI GPT models for AI-based responses.
JavaScript (Vanilla)	Frontend logic (`script.js`) aur dynamic interactions handle karne ke liye.
JSON	Seed info (`seedInfo.js`) aur request/response format ke liye.

Reason for Use

- **Node.js + Express**: Fast and simple backend setup
- **cors + body-parser**: Securely handle frontend communication
- **dotenv**: Avoid hardcoding sensitive keys
- **openai**: AI responses for non-keyword queries
- **JSON**: Easy data structure for seed info and API responses

---

2. Workflow & Flowchart

Step-wise Flow

1. **User types a message** in frontend
2. `script.js` sends the message via POST `/api/chat`

3. **Backend (`backend.js`)** receives message
4. Checks **seedInfo.js** for keyword match
    - If found → send predefined instructions + optional link
5. Else → sends context-aware message to OpenAI GPT-4.1-mini
6. Backend responds with JSON: `{ reply: 'text', link: 'url' }`
7. Frontend displays message and clickable links

Flowchart

```
User
  |
v
Frontend (index.html + script.js)
  |
v
POST /api/chat
  |
v
Backend (backend.js)
  |---> Check seedInfo.js keywords
  |        |-> if match: return answer + link
  |---> Else: OpenAI GPT request
  |
v
JSON response { reply, link }
  |
v
Frontend displays message + clickable link
```

---

3. Key Concepts, Functions & Methods

Backend (`backend.js`)

- **express()** → Creates server instance
- **app.use(cors())** → Enables cross-origin requests
- **app.use(bodyParser.json())** → Parse incoming JSON
- **dotenv.config()** → Load `.env` variables
- **POST /api/chat** → Main API endpoint
- **sessions[userId]** → Stores chat history per user for context
- **knowledgeBase.find()** → Checks user message against keywords
- **openai.chat.completions.create()** → Sends message to AI GPT

Seed Info (`seedInfo.js`)

- **keyword** → Triggers predefined answer

- **answer** → Instructions text
- **link** → Optional direct redirect URL
- JSON array allows easy addition of new services

Frontend (`script.js`)

- **getElementById** → Access HTML elements (chatBox, userInput, sendBtn)
- **addEventListener('click' / 'keypress')** → Detect send button or Enter key
- **fetch()** → Send POST request to backend
- **addMessage(text, sender)** → Adds message to chatBox (user / bot)
- **scrollTop = scrollHeight** → Auto-scroll chat box

---

4. Student-Friendly Explanation of Concepts

- **In-memory sessions** → Remember conversation per user temporarily
- **Keyword-based responses** → Fast answer without AI call
- **AI fallback** → For non-keyword queries, GPT gives smart reply
- **Environment variables** → Keep API key secret, easy to change
- **JSON structure** → Organize seed info and API communication
- **Redirect links** → Provide convenience to user

---

5. Code Understanding (Backend + Seed Info + JS)

backend.js

1. **Setup & Imports** → Load libraries and env variables
2. **Express app setup** → Server listens on port 5000
3. **Session store** → Keep chat history in `sessions` object
4. **POST /api/chat** → Handles all incoming messages
5. **Keyword check** → Search seedInfo.js for direct answer
6. **AI request** → Call OpenAI if no keyword match
7. **Send response** → JSON `{ reply, link }` back to frontend

seedInfo.js

- Predefined responses stored as **JSON array**
- Each object: `keyword`, `answer`, `link`
- Makes adding new topics easy

script.js

- Capture user input
- Send message to backend using **fetch()**
- Display messages in **chatBox**

- Show bot typing indication
- Handle errors if backend not reachable

Ye guide ek student ko **full understanding** deta hai ki kaise backend, seed info, aur JS kaam karte hain, kaise AI + predefined info integrate hua hai, aur kaise messages frontend me display hote hain.