Perfect timing 🙌 — let’s go through everything step by step, so you clearly understand **how the frontend and backend talk to each other** and what each part of the code does.

**🖥️ 1. The Frontend (Your Form + JS)**

**HTML Form**

<form id="contactForm">

<label for="name">Full Name</label>

<input type="text" id="fname" placeholder="Your name" required>

<label for="no">Phone no</label>

<input type="tel" id="no" placeholder="0712345678" required>

<label for="email">Email</label>

<input type="email" id="email" placeholder="example@gmail.com" required>

<label for="subj">Subject</label>

<input type="text" id="subj" placeholder="Your subject" required>

<label for="message">Message</label>

<textarea id="message" rows="5" placeholder="Your message"></textarea>

<button type="submit">Send Message</button>

</form>

<p id="status"></p>

* This is just a normal form.
* Instead of sending data directly to an email, it **sends data to your backend** using JavaScript.

**JavaScript (frontend → backend)**

document.getElementById("contactForm").addEventListener("submit", async (e) => {

e.preventDefault(); // stop the form from refreshing the page

const formData = {

fname: document.getElementById("fname").value,

no: document.getElementById("no").value,

email: document.getElementById("email").value,

subj: document.getElementById("subj").value,

message: document.getElementById("message").value

};

try {

const res = await fetch("http://localhost:5000/api/contact", {

method: "POST",

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

body: JSON.stringify(formData) // convert form data to JSON

});

const data = await res.json();

document.getElementById("status").textContent = data.message;

document.getElementById("status").style.color = data.success ? "green" : "red";

} catch (err) {

document.getElementById("status").textContent = "Error sending message.";

document.getElementById("status").style.color = "red";

}

});

**What’s happening here?**

* When the user hits **Send**, JavaScript grabs all the input values.
* It makes a **POST request** with fetch() to your backend API at http://localhost:5000/api/contact.
* The backend responds with JSON ({ success: true, message: "Message sent successfully!" }).
* The frontend shows that result in the <p id="status"></p> element.

So the **frontend never sends email directly** — it just passes the data to the backend.

**⚙️ 2. The Backend (Node + Express)**

**server.js**

const express = require("express");

const bodyParser = require("body-parser");

const dotenv = require("dotenv");

const cors = require("cors");

dotenv.config();

const app = express();

// Middleware

app.use(bodyParser.urlencoded({ extended: false }));

app.use(bodyParser.json());

app.use(cors()); // lets frontend (different port) call backend

// Import routes

const contactRoute = require("./routes/contact");

// Use routes

app.use("/api/contact", contactRoute);

// Start server

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {

console.log(`Backend running at http://localhost:${PORT}`);

});

**Explanation:**

* express → creates the backend server.
* bodyParser → allows backend to read JSON data from frontend.
* cors → fixes the issue of frontend (e.g., http://localhost:3000) making requests to backend (http://localhost:5000).
* dotenv → loads your secret email credentials from .env.
* app.use("/api/contact", contactRoute); → links the /api/contact URL to the routes/contact.js file.

**routes/contact.js**

const express = require("express");

const nodemailer = require("nodemailer");

const router = express.Router();

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

const { fname, email, no, subj, message } = req.body;

// Setup mail transport

const transporter = nodemailer.createTransport({

service: "gmail",

auth: {

user: process.env.EMAIL\_USER,

pass: process.env.EMAIL\_PASS

}

});

const mailOptions = {

from: email,

to: process.env.EMAIL\_USER, // your inbox

subject: subj,

text: `

Name: ${fname}

Email: ${email}

Phone: ${no}

Message:

${message}

`

};

// Send mail

transporter.sendMail(mailOptions, (err) => {

if (err) {

console.error(err);

return res.status(500).json({ success: false, message: "Failed to send message" });

}

res.json({ success: true, message: "Message sent successfully!" });

});

});

module.exports = router;

**Explanation:**

* router.post("/") → listens for POST requests from frontend.
* req.body → contains the JSON the frontend sent (name, email, message).
* nodemailer.createTransport(...) → sets up Gmail (or another service) to send email.
* transporter.sendMail(mailOptions) → actually sends the message to your inbox.
* res.json(...) → sends a response back to the frontend so it knows if the email was sent.

**🔄 3. Communication Flow**

1. User fills out the **form** in frontend.
2. JavaScript makes a **POST request** with the form data → http://localhost:5000/api/contact.
3. The **backend route** /api/contact catches that request.
4. Backend uses **Nodemailer** to send an email to your Gmail.
5. Backend responds with { success: true, message: "Message sent successfully!" }.
6. Frontend shows this message to the user.

**🎯 Why This Approach Works**

* Frontend **never stores your email password** → only backend knows it (via .env).
* The separation of frontend and backend means you can deploy them independently.
* Using JSON (fetch) makes frontend and backend communicate like a mini API.

👉 Do you want me to also explain **how you would deploy this** (so your form works online instead of only on localhost)?