BACKEND API DESIGN:

Method	Route	Description
GET	`/admin/vouchers-summary`	Returns voucher usage stats
POST	`/issue-voucher`	Issues voucher to user email
POST	`/redeem-voucher`	Allows user to redeem voucher
GET	`/admin/vouchers/:course`	View all vouchers of a specific course
POST	`/api/admin/vouchers/upload`	Uploading voucher from admin

there will be 2 models:

Courses

```
{
  _id: "JS101",
  name: "JavaScript Basics",
  totalVouchers: 100
}
```

Vouchers

```
{

  _id: "JS101-XZ4B7Q-001",
  courseId: "JS101",
  used: true,
  issuedTo: "user@example.com",
  usedOn: "2025-08-05"
}
```

For emailing voucher, we will use services like **SendGrid**, **Nodemailer**, or **Mailgun.**

ON COURSE PURCHASE FLOW

1. User clicks "Buy" → Redirects to checkout.
2. After successful purchase (Stripe/PayPal webhook or response):
    - Call backend API: `POST /issue-voucher`
    - Send courseId and user email.

📨 Voucher Issuing API (on backend)

The backend:

- Finds an **unused voucher** from DB for the course.
- Marks it as used.
- Associates it with user email.
- Sends email to the user with voucher code.
- Returns status to frontend.
- Admin panel updates automatically via state or polling.

VOUCHER REDEMPTION LOGIC

When user uses the voucher:

1. Enter voucher code during checkout.
2. Frontend sends it to: `POST /redeem-voucher`
3. Backend:
    - Validates format.
    - Checks if exists, not used.
    - Marks as used.
    - Proceeds with free purchase.

ADMIN PANEL (React)

Courses list with:

- Total vouchers
- Issued vouchers
- Remaining vouchers

`GET /admin/vouchers-summary` returns:

```
[
 {
   "courseId": "JS101",
   "total": 100,
   "issued": 45,
   "remaining": 55
 },
 ...
]
```

Create a POST route: `/api/admin/vouchers/upload`

```
// routes/voucherRoutes.js
const express = require('express');
const router = express.Router();
const Voucher = require('../models/Voucher');

router.post('/upload', async (req, res) => {
```

```
  const { courseId, vouchers } = req.body;

  if (!courseId || !Array.isArray(vouchers)) {
    return res.status(400).json({ error: 'Missing courseId or voucher list' });
  }

  const voucherDocs = vouchers.map(code => ({
    courseId,
    voucherCode: code.trim(),
    used: false,
  }));

  try {
    await Voucher.insertMany(voucherDocs, { ordered: false }); // skip duplicates
    return res.status(200).json({ message: 'Vouchers uploaded successfully' });
  } catch (error) {
    if (error.code === 11000) {
      return res.status(409).json({ error: 'Some vouchers already exist (duplicates skipped)' });
    }
    return res.status(500).json({ error: 'Upload failed', details: error });
  }
});

module.exports = router;
```

must connect this route in your `server.js` or `app.js`: