Razorpay Integration Guide

How to integrate the Razorpay in my website in both frontend and backend from scratch help me

To integrate Razorpay into your website from scratch, you’ll need to:

- Create a Razorpay account.
- Set up your backend to create orders securely using Razorpay's APIs.
- Set up your frontend to accept payments via the Razorpay checkout.
- Verify the payment signature on the backend for security.

🔧 Tools and Stack Assumptions

- **Backend:** Node.js with Express
- **Frontend:** Next.js (React), TailwindCSS, and Shadcn UI
- **Database:** (optional) PostgreSQL/MongoDB to store order/payment status

---

🔐 STEP 1: Razorpay Account Setup

1. Create a Razorpay account: https://dashboard.razorpay.com/signup
2. Get your **Key ID** and **Key Secret** from the dashboard.

🧠 STEP 2: Backend Setup (Node.js + Express)

1. Install Razorpay SDK

```
bash

npm install razorpay dotenv
```

2. Set Up `.env`

```
env

RAZORPAY_KEY_ID=your_key_id
RAZORPAY_KEY_SECRET=your_key_secret
```

3. Razorpay Route (`routes/razorpay.js`)

```ts
import express from 'express';
import Razorpay from 'razorpay';
import crypto from 'crypto';

const router = express.Router();

const razorpay = new Razorpay({
  key_id: process.env.RAZORPAY_KEY_ID!,
  key_secret: process.env.RAZORPAY_KEY_SECRET!,
});

// Route to create order
router.post('/create-order', async (req, res) => {
  const { amount, currency = 'INR' } = req.body;

  try {
    const options = {
      amount: amount * 100, // amount in paisa
      currency,
      receipt: `receipt_order_${Date.now()}`,
    };

    const order = await razorpay.orders.create(options);
    return res.json({ success: true, order });
  } catch (error) {
    return res.status(500).json({ success: false, error });
  }
});

// Route to verify payment
router.post('/verify-payment', (req, res) => {
  const { razorpay_order_id, razorpay_payment_id, razorpay_signature } = req.body;

  const body = razorpay_order_id + "|" + razorpay_payment_id;

  const expectedSignature = crypto
    .createHmac("sha256", process.env.RAZORPAY_KEY_SECRET!)
    .update(body.toString())
    .digest("hex");

  if (expectedSignature === razorpay_signature) {
    // Save payment status in DB here
    return res.json({ success: true, message: "Payment verified successfully" });
  } else {
    return res.status(400).json({ success: false, message: "Invalid signature" });
  }
});

export default router;
```

4. Mount in Express App

```ts
import express from 'express';
import razorpayRoutes from './routes/razorpay';
import cors from 'cors';
import dotenv from 'dotenv';

dotenv.config();
const app = express();
app.use(cors());
app.use(express.json());

app.use('/api/razorpay', razorpayRoutes);

app.listen(3001, () => console.log('Server running on http://localhost:3001'));
```

🎨 STEP 3: Frontend Integration (Next.js + Razorpay)

1. Install Razorpay Script

Razorpay recommends loading it dynamically in your component:

```
tsx

useEffect(() => {
  const script = document.createElement("script");
  script.src = "https://checkout.razorpay.com/v1/checkout.js";
  script.async = true;
  document.body.appendChild(script);
}, []);
```

2. Payment Button (e.g. `components/PaymentButton.tsx`)

```
tsx

import axios from 'axios';

const PaymentButton = ({ amount }: { amount: number }) => {
  const handlePay = async () => {
    const { data } = await axios.post('http://localhost:3001/api/razorpay/create-order', {
      amount,
    });

    const { order } = data;

    const options = {
      key: process.env.NEXT_PUBLIC_RAZORPAY_KEY_ID!,
      amount: order.amount,
      currency: order.currency,
      name: 'YourAppName',
      description: 'Course Purchase',
      order_id: order.id,
      handler: async (response: any) => {
        const verifyRes = await axios.post('http://localhost:3001/api/razorpay/verify-payment', res
        alert(verifyRes.data.message);
      },
      prefill: {
        name: "John Doe",
        email: "john@example.com",
      },
    };

    const rzp = new (window as any).Razorpay(options);
    rzp.open();
  };

  return <button onClick={handlePay} className="btn btn-primary">Pay ₹{amount}</button>;
};

export default PaymentButton;
```

3. Add Env Key in `.env.local`

```
env

NEXT_PUBLIC_RAZORPAY_KEY_ID=your_key_id
```

✅ Optional (But Important)

- Save payment/order data in your DB
- Handle failed payments and retries
- Secure backend with authentication for real users

Would you like me to generate a full working code scaffold for backend + frontend in separate folders?