 A high-level overview of how you can use Stripe and Node.js to process payments on your website and grant access to resources after successful payment:

1. When a user selects a product and is ready to checkout, your server sends a request to Stripe to create a new Checkout Session.
2. Stripe returns a URL for the Checkout Session that redirects the user to a Stripe-hosted payment page.
3. The user enters their payment details on the payment page and completes the transaction.
4. After the transaction, Stripe sends a checkout.session.completed event to your webhook endpoint.
5. Your server handles the webhook event and uses the information in the webhook payload to update your database with details such as user info, payment ID, time, date, and product.
6. After updating your database, you can grant access to the resources for that user.

Q. Do i have to create user in strip which is going to make payment?

* No, you do not have to create a user in Stripe for a customer to make a payment. When using Stripe Checkout, customers can enter their payment information on the Stripe-hosted payment page without having a Stripe account. However, if you want to save customer information for future payments or to associate payments with a specific customer, you can create a Customer object in Stripe and associate it with the payment.

Q. How create a Customer object in Stripe using the Stripe API.

* const stripe = require('stripe')('sk\_test\_...');

const customer = await stripe.customers.create({

email: 'customer@example.com',

name: 'Customer Name',

// other customer information

});

const customerId = customer.id;

* This creates a new Customer object in Stripe with the provided information. Save this customer id in DB,

You can then associate this Customer object with payment by specifying its ID when creating a PaymentIntent or Checkout Session.

Q. how to specify cutomerId while creating checkout session?

* const session = await stripe.checkout.sessions.create({

customer: customerId,

payment\_method\_types: ['card'],

line\_items: [

{

price\_data: {

currency: 'usd',

product\_data: {

name: 'Product Name',

},

unit\_amount: 2000,

},

quantity: 1,

},

],

mode: 'payment',

success\_url: 'https://example.com/success',

cancel\_url: 'https://example.com/cancel',

});

customerId is the ID of an existing Customer object in Stripe. When creating the Checkout Session, we set the customer property to this ID to associate the session with the customer

Q. Retrieve a list of all payments for a specific customer using the Stripe API.

* const paymentIntents = await stripe.paymentIntents.list({

customer: customerId,

});

Q. How to listen to events if the user successfully completes their payment or the payment declines

* The user enters their payment details on the payment page and completes the transaction.

After the transaction, Stripe sends a checkout.session.completed event to your **webhook endpoint**

You can use the information in the webhook payload to fulfill the order and update your database with details such as transaction ID, amount, user info, time data, and product.

*// This is your Stripe CLI webhook secret for testing your endpoint locally.*

const endpointSecret = "whsec\_bc601afa4f12fc9e89e92008cb705054947ec79394701d32a85c1de92864a000";

app.post('/webhook', express.raw({type: 'application/json'}), (request, response) => {

const sig = request.headers['stripe-signature'];

let event;

try {

event = stripe.webhooks.constructEvent(request.body, sig, endpointSecret);

} catch (err) {

response.status(400).send(`Webhook Error: ${err.message}`);

return;

}

*// Handle the event*

switch (event.type) {

case 'billing\_portal.session.created':

const billingPortalSessionCreated = event.data.object;

*// Then define and call a function to handle the event billing\_portal.session.created*

break;

case 'checkout.session.async\_payment\_failed':

const checkoutSessionAsyncPaymentFailed = event.data.object;

*// Then define and call a function to handle the event checkout.session.async\_payment\_failed*

break;

case 'checkout.session.async\_payment\_succeeded':

const checkoutSessionAsyncPaymentSucceeded = event.data.object;

*// Then define and call a function to handle the event checkout.session.async\_payment\_succeeded*

break;

case 'checkout.session.completed':

const checkoutSessionCompleted = event.data.object;

*// Then define and call a function to handle the event checkout.session.completed*

break;

case 'checkout.session.expired':

const checkoutSessionExpired = event.data.object;

*// Then define and call a function to handle the event checkout.session.expired*

break;

*// ... handle other event types*

default:

console.log(`Unhandled event type ${event.type}`);

}

*// Return a 200 response to acknowledge receipt of the event*

response.send();

});

Q. Does Stripe allow different prices for the product depending on their location?

* Stripe does not have a built-in feature to set different prices for different countries. However, one can implement location-based pricing on your website or app by detecting the user’s location and displaying the appropriate price for that location. You can then create separate products or prices in Stripe for each location and charge the customer accordingly

There are several platforms that allow you to create unlimited projects for free. [Some of these platforms include **Trello**, which offers unlimited personal projects with its free plan, and **Asana**, which allows unlimited projects for up to 15 team members with its free plan**1**](https://toggl.com/blog/free-project-management-software).

CodePen that allow you to create unlimited projects. [Some popular options include Replit, CodeSandbox, JS Bin, and Glitch](https://alternativeto.net/software/codepen/)

you learn AI programming. Some popular books for beginners include “Make Your Own Neural Network” by Tariq Rashid, “Artificial Intelligence For Dummies” by John Paul Mueller and “Machine Learning For Absolute Beginners” by Oliver Theobald. Another highly recommended book is “Artificial Intelligence: A Modern Approach” by Stuart Russell and Peter Norvig.

* **Udemy**: Udemy offers a course called “Fundamentals of Building a Remote Access Tool (RAT1) in C#” that teaches basic C# programming, networking, and client-server technology by writing a Remote Access Trojan from scratch.
* **GeeksforGeeks**: GeeksforGeeks has an article called “Introduction To RAT - Remote Administration Tool” that provides an overview of what RATs are and how they work.
* **Infosec Resources**: Infosec Resources has an article called “Remote access tool” that provides information on what RATs are and how they can be used for malicious purposes.

**“How to Write Your Own Remote Access Tools in C#”**. This book was authored by a malware developer and published on Amazon in 2011

**How build**

**FIRST LOGIN USER**

**AFTER HITTING CHECKOUT ROUTE CHECK IF THE USER OBJECT ID EXISTS OR NOT**

**IF NOT**

**CREATE NEW USER OBJECT ID OF THAT USER AND STORE IT IN DB**

**IF YES**

**GET THAT USER OBJECT ID AND GENERATE CHECKOUT SESSION**

**MAKE PAYMENT AFTER SUCCESSFULL PAYMENT**

**STORE THAT PAYMENT HISTORY IN DB AND MARK USER FOR SUBSCRIBED**

**IF PAYMENT FAILES NOTHING TO WORRY ABOUT**