Stripe Payment Integration Setup Guide

This guide will walk you through setting up the complete Stripe payment integration for the Autopilot web starter.

Prerequisites

- A Stripe account (sign up at https://stripe.com)
- Node.js and npm installed
- A Resend account for sending emails (sign up at https://resend.com)

Step 1: Install Dependencies

All required dependencies are already installed:
- `stripe` - Official Stripe SDK
- `resend` - Email service provider

Step 2: Configure Stripe

2.1 Get Your API Keys

1. Log in to your Stripe Dashboard (https://dashboard.stripe.com)
2. Make sure you're in **Test Mode** (toggle in top right)
3. Go to **Developers → API keys**
4. Copy your **Publishable key** (starts with `pk_test_`)
5. Copy your **Secret key** (starts with `sk_test_`)

2.2 Create Products and Prices

1. Go to **Products** in your Stripe Dashboard
2. Click **+ Add product**
3. Create three products:

Starter Plan

- **Name**: Starter Plan
- **Description**: Basic Next.js starter with essential features
- **Pricing**: One-time payment of $49.00 USD
- Copy the **Price ID** (starts with `price_`)

Professional Plan

- **Name**: Professional Plan
- **Description**: Advanced features with priority support
- **Pricing**: One-time payment of $99.00 USD
- Copy the **Price ID** (starts with `price_`)

Enterprise Plan (Optional)

- **Name**: Enterprise Plan
- **Description**: Custom solution with dedicated support

- **Pricing**: Custom (you can set a placeholder price or handle via contact form)
- Copy the **Price ID** (starts with `price_`)

2.3 Configure Webhook

1. Go to **Developers → Webhooks**
2. Click **+ Add endpoint**
3. For local development:
   - Endpoint URL: Use Stripe CLI (see below)
4. For production:
   - Endpoint URL: `https://yourdomain.com/api/webhooks/stripe`
5. Select events to listen to:
   - `checkout.session.completed`
   - `customer.subscription.created`
   - `customer.subscription.updated`
   - `customer.subscription.deleted`
   - `invoice.payment_succeeded`
   - `invoice.payment_failed`
   - `payment_intent.succeeded`
   - `payment_intent.payment_failed`
6. Copy the **Signing secret** (starts with `whsec_`)

Step 3: Configure Resend for Emails

1. Sign up at https://resend.com
2. Go to **API Keys**
3. Create a new API key
4. Copy your API key (starts with `re_`)
5. Verify your domain (optional but recommended for production)

Step 4: Set Environment Variables

Create a `.env` file in the root directory (or update existing one):

```
# App Configuration
NEXT_PUBLIC_APP_URL=http://localhost:3000

# Stripe Configuration
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_test_your_publishable_key
STRIPE_SECRET_KEY=sk_test_your_secret_key
STRIPE_WEBHOOK_SECRET=whsec_your_webhook_secret

# Stripe Price IDs
STRIPE_STARTER_PRICE_ID=price_your_starter_price_id
STRIPE_PRO_PRICE_ID=price_your_pro_price_id
STRIPE_ENTERPRISE_PRICE_ID=price_your_enterprise_price_id

# Email Configuration (Resend)
RESEND_API_KEY=re_your_resend_api_key
EMAIL_FROM=Autopilot <onboarding@yourdomain.com>
```

Step 5: Test Locally with Stripe CLI

5.1 Install Stripe CLI

macOS (Homebrew):

```
brew install stripe/stripe-cli/stripe
```

Linux:

```
wget https://github.com/stripe/stripe-cli/releases/download/v1.18.0/
stripe_1.18.0_linux_x86_64.tar.gz
tar -xvf stripe_1.18.0_linux_x86_64.tar.gz
sudo mv stripe /usr/local/bin/
```

Windows:
Download from https://github.com/stripe/stripe-cli/releases

5.2 Login to Stripe CLI

```
stripe login
```

5.3 Forward Webhooks to Local Server

```
stripe listen --forward-to localhost:3000/api/webhooks/stripe
```

This will output a webhook signing secret. Copy it and update your `.env` file:

```
STRIPE_WEBHOOK_SECRET=whsec_from_stripe_cli
```

Step 6: Run the Application

```
npm run dev
```

Visit http://localhost:3000 and test the payment flow.

Step 7: Test the Payment Flow

7.1 Test Card Numbers

Stripe provides test card numbers:

- **Success**: `4242 4242 4242 4242`
- **Decline**: `4000 0000 0000 0002`
- **3D Secure**: `4000 0025 0000 3155`

Expiry: Any future date (e.g., 12/34)
CVC: Any 3 digits (e.g., 123)
ZIP: Any 5 digits (e.g., 12345)

7.2 Test Flow

1. Navigate to the pricing section
2. Click “Get Started” on any tier
3. Fill in the checkout form with test card details
4. Complete the payment
5. Verify:
   - Redirect to success page
   - Order confirmation email sent
   - Webhook event received (check Stripe CLI output)
   - Event logged in Stripe Dashboard

Step 8: Deploy to Production

8.1 Update Environment Variables

In your production environment (Vercel, Netlify, etc.):

1. Use **live mode** API keys from Stripe (starts with `pk_live_` and `sk_live_`)
2. Update `NEXT_PUBLIC_APP_URL` to your production domain
3. Add production webhook endpoint in Stripe Dashboard
4. Update `EMAIL_FROM` with your verified domain

8.2 Vercel Deployment

```
# Add environment variables in Vercel dashboard
vercel env add STRIPE_SECRET_KEY
vercel env add NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY
vercel env add STRIPE_WEBHOOK_SECRET
vercel env add STRIPE_STARTER_PRICE_ID
vercel env add STRIPE_PRO_PRICE_ID
vercel env add RESEND_API_KEY
vercel env add EMAIL_FROM
vercel env add NEXT_PUBLIC_APP_URL

# Deploy
vercel --prod
```

8.3 Configure Production Webhook

1. In Stripe Dashboard, go to **Developers → Webhooks**
2. Add production endpoint: `https://yourdomain.com/api/webhooks/stripe`
3. Select the same events as before
4. Copy the new signing secret
5. Update `STRIPE_WEBHOOK_SECRET` in your production environment

Customer Portal (Optional)

The Customer Portal allows customers to manage their subscriptions, update payment methods, and view invoices.

Enable Customer Portal in Stripe

1. Go to **Settings → Billing → Customer Portal**

2. Enable the portal
3. Configure allowed features:
   - Cancel subscriptions
   - Update payment methods
   - View invoices
4. Customize branding and emails

Access Customer Portal

Add a button in your dashboard:

```
const handleManageSubscription = async (customerId: string) => {
  const response = await fetch('/api/create-portal-session', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      customerId,
      returnUrl: window.location.href
    }),
  });

  const { url } = await response.json();
  window.location.href = url;
};
```

Troubleshooting

Webhook Events Not Received

- Check Stripe CLI is running (`stripe listen --forward-to localhost:3000/api/webhooks/stripe`)
- Verify `STRIPE_WEBHOOK_SECRET` matches the CLI output
- Check the webhook endpoint is accessible

Payment Fails

- Ensure you’re using test mode API keys and test card numbers
- Check browser console for errors
- Verify all environment variables are set correctly

Email Not Sent

- Verify Resend API key is correct
- Check your email doesn’t land in spam
- View email logs in Resend dashboard
- Ensure `EMAIL_FROM` domain is verified for production

TypeScript Errors

If you see TypeScript errors related to Stripe API version:

```
npm install stripe@latest --save
```

API Routes Reference

Route	Method	Description
`/api/create-checkout-session`	POST	Create a Stripe Checkout session
`/api/create-portal-session`	POST	Create a Customer Portal session
`/api/webhooks/stripe`	POST	Handle Stripe webhook events
`/api/verify-session`	GET	Verify a checkout session

Next Steps

1. **Add Database Integration**: Store orders and customer data in a database (Prisma is already installed)
2. **User Authentication**: Integrate with NextAuth.js or similar
3. **Dashboard**: Build a customer dashboard to display purchases
4. **Analytics**: Track conversion rates and revenue
5. **Testing**: Write integration tests for payment flows

Security Best Practices

- ✅ Never expose `STRIPE_SECRET_KEY` in client-side code
- ✅ Always verify webhook signatures
- ✅ Use environment variables for all sensitive data
- ✅ Enable Stripe Radar for fraud protection in production
- ✅ Implement proper error handling
- ✅ Log all payment events for auditing

Support

For issues or questions:
- Stripe Documentation: https://stripe.com/docs
- Stripe Support: https://support.stripe.com
- Resend Documentation: https://resend.com/docs

Happy Coding! 🚀