Assignment :

Create API integration product similar with <https://shipfa.st/>, the code should contain basic module for email, payment functions etc user can generate their landing page. You can check their website for more information.

Source Code Link : <https://v0.dev/chat/IaRNs9M9aS7>

Output Link : <https://dtptsbpoegoj20cu.vercel.app>

***app > actions > generateLandingPage.ts***

'use server'

export async function generateLandingPage(title: string, description: string, callToAction: string) {

const landingPageHTML = `

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>${title}</title>

<style>

body { font-family: Arial, sans-serif; text-align: center; padding: 20px; }

.cta { background: #007BFF; color: white; padding: 10px 20px; text-decoration: none; border-radius: 5px; }

</style>

</head>

<body>

<h1>${title}</h1>

<p>${description}</p>

<a href="#" class="cta">${callToAction}</a>

</body>

</html>

`

return { success: true, landingPageHTML }}

***app > api > process-payment > route.ts***

import { NextResponse } from 'next/server'

import Stripe from 'stripe'

const stripe = new Stripe(process.env.STRIPE\_SECRET\_KEY!, {

apiVersion: '2022-11-15',

})

export async function POST(request: Request) {

try {

const { amount, currency, paymentMethodId } = await request.json()

const paymentIntent = await stripe.paymentIntents.create({

amount,

currency,

payment\_method: paymentMethodId,

confirm: true,

})

return NextResponse.json({ success: true, paymentIntent })

} catch (error) {

console.error('Error processing payment:', error)

return NextResponse.json({ success: false, message: 'Failed to process payment' }, { status: 500 })

}

}

***app > api > send-email > route.ts***

import { NextResponse } from 'next/server'

import nodemailer from 'nodemailer'

const transporter = nodemailer.createTransport({

service: 'Gmail',

auth: {

user: process.env.SMTP\_USER,

pass: process.env.SMTP\_PASS,

},

})

export async function POST(request: Request) {

try {

const { to, subject, message } = await request.json()

const mailOptions = {

from: process.env.SMTP\_USER,

to,

subject,

text: message,

}

await transporter.sendMail(mailOptions)

return NextResponse.json({ success: true, message: 'Email sent successfully!' })

} catch (error) {

console.error('Error sending email:', error)

return NextResponse.json({ success: false, message: 'Failed to send email' }, { status: 500 })

}

}

***app > components > ShipfastClone.tsx***

'use client'

import { useState } from 'react'

import { generateLandingPage } from '../actions/generateLandingPage'

import { Button } from '@/components/ui/button'

import { Input } from '@/components/ui/input'

import { Textarea } from '@/components/ui/textarea'

import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card'

export default function ShipfastClone() {

const [emailResult, setEmailResult] = useState<string | null>(null)

const [paymentResult, setPaymentResult] = useState<string | null>(null)

const [landingPageResult, setLandingPageResult] = useState<string | null>(null)

const handleSendEmail = async (e: React.FormEvent<HTMLFormElement>) => {

e.preventDefault()

const formData = new FormData(e.currentTarget)

const to = formData.get('to') as string

const subject = formData.get('subject') as string

const message = formData.get('message') as string

try {

const response = await fetch('/api/send-email', {

method: 'POST',

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

body: JSON.stringify({ to, subject, message }),

})

const data = await response.json()

setEmailResult(data.success ? 'Email sent successfully!' : 'Failed to send email')

} catch (error) {

setEmailResult('Error sending email')

}

}

const handleProcessPayment = async (e: React.FormEvent<HTMLFormElement>) => {

e.preventDefault()

const formData = new FormData(e.currentTarget)

const amount = Number(formData.get('amount'))

const currency = formData.get('currency') as string

const paymentMethodId = formData.get('paymentMethodId') as string

try {

const response = await fetch('/api/process-payment', {

method: 'POST',

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

body: JSON.stringify({ amount, currency, paymentMethodId }),

})

const data = await response.json()

setPaymentResult(data.success ? 'Payment processed successfully!' : 'Failed to process payment')

} catch (error) {

setPaymentResult('Error processing payment')

}

}

const handleGenerateLandingPage = async (e: React.FormEvent<HTMLFormElement>) => {

e.preventDefault()

const formData = new FormData(e.currentTarget)

const title = formData.get('title') as string

const description = formData.get('description') as string

const callToAction = formData.get('callToAction') as string

try {

const result = await generateLandingPage(title, description, callToAction)

setLandingPageResult(result.success ? 'Landing page generated successfully!' : 'Failed to generate landing page')

} catch (error) {

setLandingPageResult('Error generating landing page')

}}

return (

<div className="grid grid-cols-1 md:grid-cols-3 gap-4">

<Card>

<CardHeader>

<CardTitle>Send Email</CardTitle>

</CardHeader>

<CardContent>

<form onSubmit={handleSendEmail} className="space-y-4">

<Input name="to" placeholder="To" required />

<Input name="subject" placeholder="Subject" required />

<Textarea name="message" placeholder="Message" required />

<Button type="submit">Send Email</Button>

</form>

{emailResult && <p className="mt-2">{emailResult}</p>}

</CardContent>

</Card>

<Card>

<CardHeader>

<CardTitle>Process Payment</CardTitle>

</CardHeader>

<CardContent>

<form onSubmit={handleProcessPayment} className="space-y-4">

<Input name="amount" type="number" placeholder="Amount" required />

<Input name="currency" placeholder="Currency" required />

<Input name="paymentMethodId" placeholder="Payment Method ID" required />

<Button type="submit">Process Payment</Button>

</form>

{paymentResult && <p className="mt-2">{paymentResult}</p>}

</CardContent>

</Card>

<Card>

<CardHeader>

<CardTitle>Generate Landing Page</CardTitle>

</CardHeader>

<CardContent>

<form onSubmit={handleGenerateLandingPage} className="space-y-4">

<Input name="title" placeholder="Title" required />

<Input name="description" placeholder="Description" required />

<Input name="callToAction" placeholder="Call to Action" required />

<Button type="submit">Generate Landing Page</Button>

</form>

{landingPageResult && <p className="mt-2">{landingPageResult}</p>}

</CardContent>

</Card>

</div>

)

}

***app > page.tsx***

import ShipfastClone from './components/ShipfastClone'

export default function Home() {

return (

<main className="container mx-auto p-4">

<h1 className="text-3xl font-bold mb-4">Shipfast Clone</h1>

<ShipfastClone />

</main>

)

}

***.env.local***

SMTP\_USER=your-email@gmail.com

SMTP\_PASS=your-email-password

STRIPE\_SECRET\_KEY=your-stripe-secret-key

***Output : (Picture Format)***

