

웹 어플리케이션 보안 (캡스톤디자인)

3. 예제 웹서비스 – ecommerce 3

중부대학교 정보보호학과
이병천 교수
sultan@joongbu.ac.kr

전체 목차

1. 강의 개요
 - 캡스톤디자인 프로젝트 추진 방법
2. Next.js
 - 프론트엔드 프로그래밍
 - 백엔드 프로그래밍
 - NextAuth
3. 예제 웹서비스 : ecommerce
4. 예제 웹서비스 : 암호 활용 Forge
5. 캡스톤디자인 프로젝트 발표

3. 예제 웹서비스 – ecommerce 3

3.17 Place Order Screen

3.18 Create Order Screen

3.19 Create Register Screen

3.20 Paypal Payment

3.21 Order History

3.22 Update User Profile

3.23 Deploy to Vercel.com



3.17 Place Order Screen

- Place Order (주문하기) 페이지
 - 주소, 지불방법, 주문내역 등을 요약해서 보여주기
 - 표시된 내용을 확인하고 Place Order 버튼을 누르면 주문내역을 DB에 저장하고 Order 페이지로 이동

pages/placeorder.js

```
import axios from 'axios'
import Image from 'next/image'
import Link from 'next/link'
import { useRouter } from 'next/router'
import Cookies from 'js-cookie'
import React, { useContext, useEffect, useState } from 'react'
import { toast } from 'react-toastify'
import CheckoutWizard from '../components/CheckoutWizard'
import Layout from '../components/Layout'
import { getError } from '../utils/error'
import { Store } from '../utils/Store'

export default function PlaceOrderScreen() {
  const { state, dispatch } = useContext(Store)
  const { cart } = state
  const { cartItems, shippingAddress, paymentMethod } = cart

  const round2 = (num) => Math.round(num * 100 + Number.EPSILON) / 100

  const itemsPrice = round2(
    cartItems.reduce((a, c) => a + c.quantity * c.price, 0)
  ) // 123.4567 => 123.46

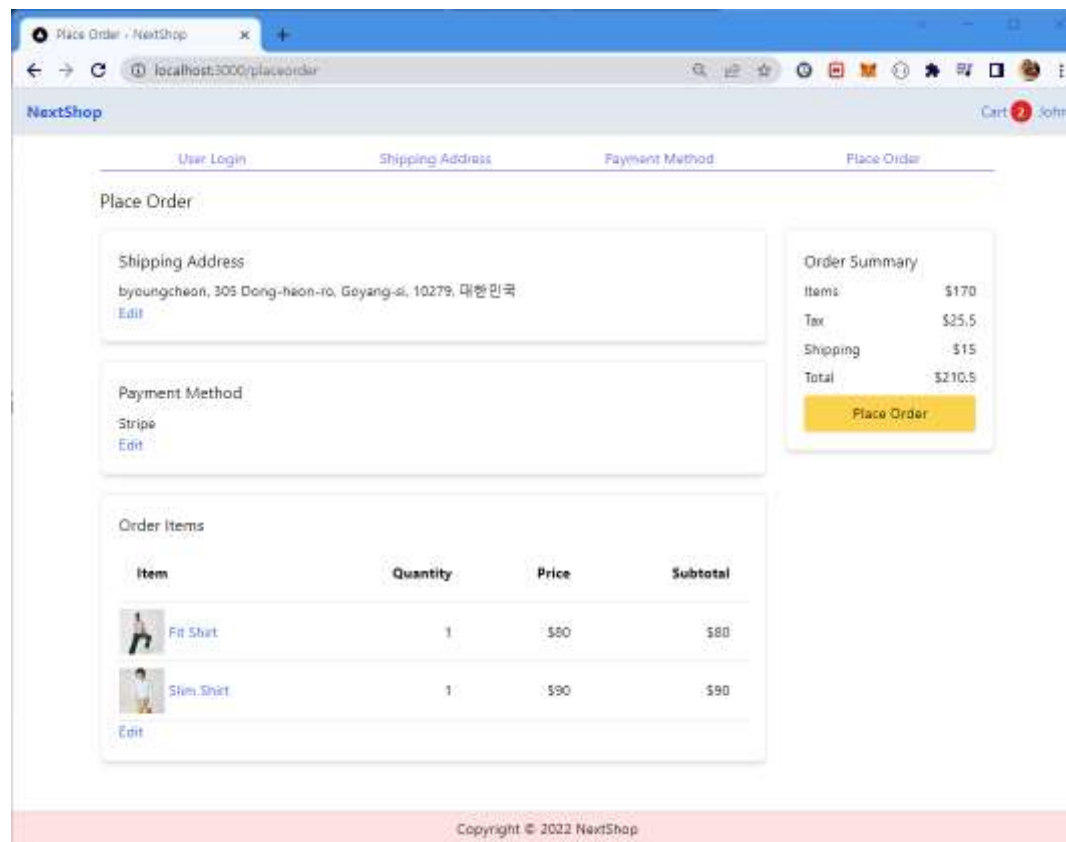
  const shippingPrice = itemsPrice > 200 ? 0 : 15
  const taxPrice = round2(itemsPrice * 0.15)
  const totalPrice = round2(itemsPrice + shippingPrice + taxPrice)

  const router = useRouter()
  useEffect(() => {
    if (!paymentMethod) {
      router.push('/payment')
    }
  }, [paymentMethod, router])

  const [loading, setLoading] = useState(false)
```

중략 - github 소스코드 참조

프런트엔드



```
import mongoose from 'mongoose'

const orderSchema = new mongoose.Schema(
  {
    user: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
    orderItems: [
      {
        name: { type: String, required: true },
        quantity: { type: Number, required: true },
        image: { type: String, required: true },
        price: { type: Number, required: true },
      },
    ],
    shippingAddress: {
      fullName: { type: String, required: true },
      address: { type: String, required: true },
      city: { type: String, required: true },
      postalCode: { type: String, required: true },
      country: { type: String, required: true },
    },
    paymentMethod: { type: String, required: true },
    itemsPrice: { type: Number, required: true },
    shippingPrice: { type: Number, required: true },
    taxPrice: { type: Number, required: true },
    totalPrice: { type: Number, required: true },
    isPaid: { type: Boolean, required: true, default: false },
    isDelivered: { type: Boolean, required: true, default: false },
    paidAt: { type: Date },
    deliveredAt: { type: Date },
  },
  {
    timestamps: true,
  }
)

const Order = mongoose.models.Order || mongoose.model('Order', orderSchema)
export default Order
```

models/Order.js

주문 데이터 모델

주문을 처리하는 데이터 모델을 정의 (Order.js)

- 주문자 (user)
- 주문 물품 (order items)
- 배달 주소 (shipping address)
- 지불방법, 금액
- 타임스탬프

백엔드

pages/api/orders/index.js

```
import { getSession } from 'next-auth/react'
import Order from '../../models/Order'
import db from '../../utils/db'

const handler = async (req, res) => {
  const session = await getSession({ req })
  if (!session) {
    return res.status(401).send('signin required')
  }

  const { user } = session
  await db.connect()
  const newOrder = new Order({
    ...req.body,
    user: user._id,
  })

  const order = await newOrder.save()
  res.status(201).send(order)
}
export default handler
```

백엔드 서버 프로그램

- 클라이언트의 요청을 받아 주문 내용을 DB에 저장함
- 주문자 아이디 정보를 추가

utils/Store.js

종락

```
case 'CART_CLEAR_ITEMS':  
  return { ...state, cart: { ...state.cart, cartItems: [] } }
```

종락

지불이 완료된 후 쇼핑카트를 초기화함

pages/payment.js

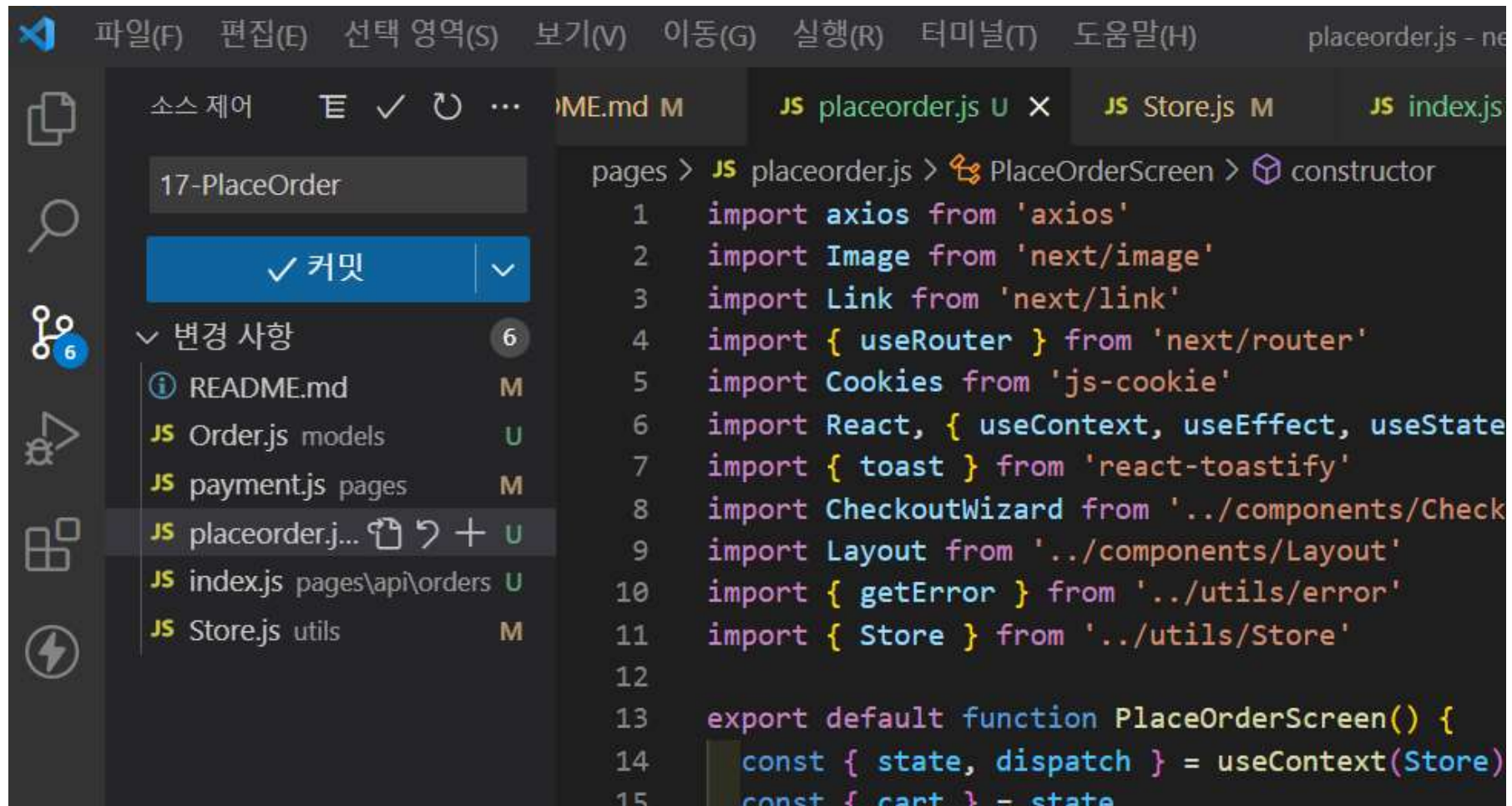
종락

```
</Layout>  
)  
}
```

PaymentScreen.auth = true

로그인된 경우에만 지불 화면 표시

변경사항 커밋, 저장



3.18 Create Order Screen

- 저장된 주문 내역을 최종 화면에 보여주면서 지불 요청
- Dynamic Routing 기능을 활용
 - 주문은 사용자의 행위에 따라 생성됨
 - 주문기록의 ID를 중심으로 처리 – dynamic routing
- 백엔드
 - pages/api/orders/[id].js
 - 클라이언트의 요청에 의해 해당 주문 내역을 DB에서 읽어와서 응답
- 프론트엔드
 - Pages/order/[id].js
 - 백엔드의 응답 내역을 화면에 표시
 - 지불버튼 표시

백엔드 Dynamic Routing

pages/api/orders/[id].js

```
// /api/orders/:id
import { getSession } from 'next-auth/react'
import Order from '../models/Order'
import db from '../utils/db'

const handler = async (req, res) => {
  const session = await getSession({ req })
  if (!session) {
    return res.status(401).send('signin required')
  }

  await db.connect()

  const order = await Order.findById(req.query.id)
  await db.disconnect()
  res.send(order)
}

export default handler
```

프론트엔드 Dynamic Routing

pages/api/orders/[id].js

```
import axios from 'axios'
import Image from 'next/image'
import Link from 'next/link'
import { useRouter } from 'next/router'
import { useEffect, useReducer } from 'react'
import Layout from '../components/Layout'
import { getError } from '../utils/error'

function reducer(state, action) {
  switch (action.type) {
    case 'FETCH_REQUEST':
      return { ...state, loading: true, error: '' }
    case 'FETCH_SUCCESS':
      return { ...state, loading: false, order: action.payload, error: '' }
    case 'FETCH_FAIL':
      return { ...state, loading: false, error: action.payload }
    default:
      state
  }
}

function OrderScreen() {
  // order/:id
  const { query } = useRouter()
  const orderId = query.id

  const [{ loading, error, order }, dispatch] = useReducer(reducer, {
    loading: true,
    order: {},
    error: '',
  })
}
```

중략 - Github 소스코드 참조

styles/globals.css

```
.alert-error {
  @apply my-3 rounded-lg bg-red-100 p-3 text-red-700;
}

.alert-success {
  @apply my-3 rounded-lg bg-green-100 p-3 text-green-700;
}
```

Github 소스코드 참조

결과 테스트

Order 63329d7b96ee3db6f1cb8c3c



Shipping Address
byoungcheon, 305 Dong-heon-ro, Goyang-si, 10279, 대한민국
Not delivered

Payment Method
PayPal
Not paid

Order Summary

Items	\$170
Tax	\$25.5
Shipping	\$15
Total	\$210.5

Order Items

Item	Quantity	Price	Subtotal
 Fit Shirt	1	\$80	\$80
 Slim Shirt	1	\$90	\$90

Copyright © 2022 NextShop

앞에서 DB에 저장한 주문데이터를
읽어와서 보여줌

nextshop.orders

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 4.85KB TOTAL DOCUMENTS: 1

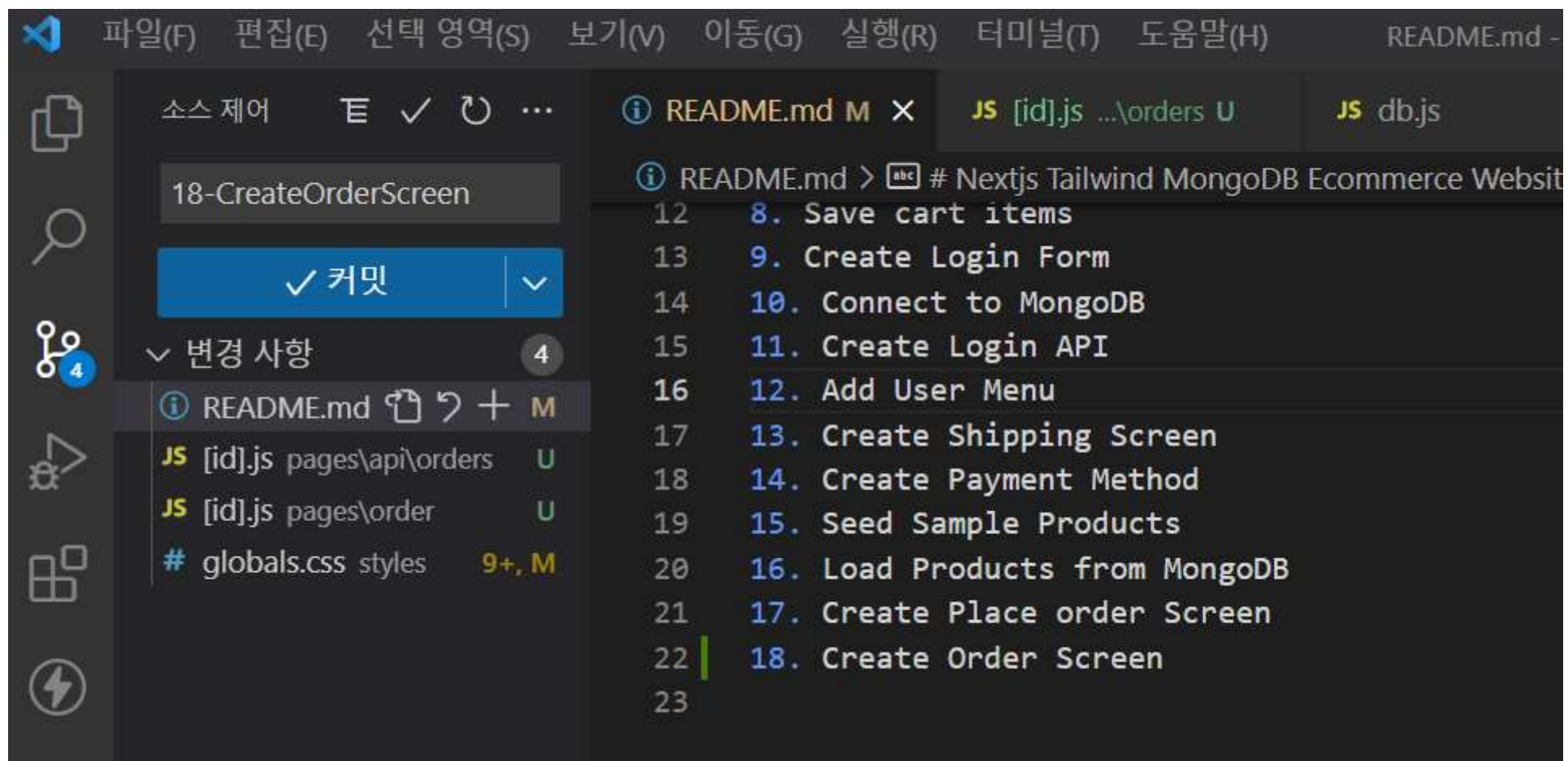
Find

Indexes

Schema Anti-Patterns 0

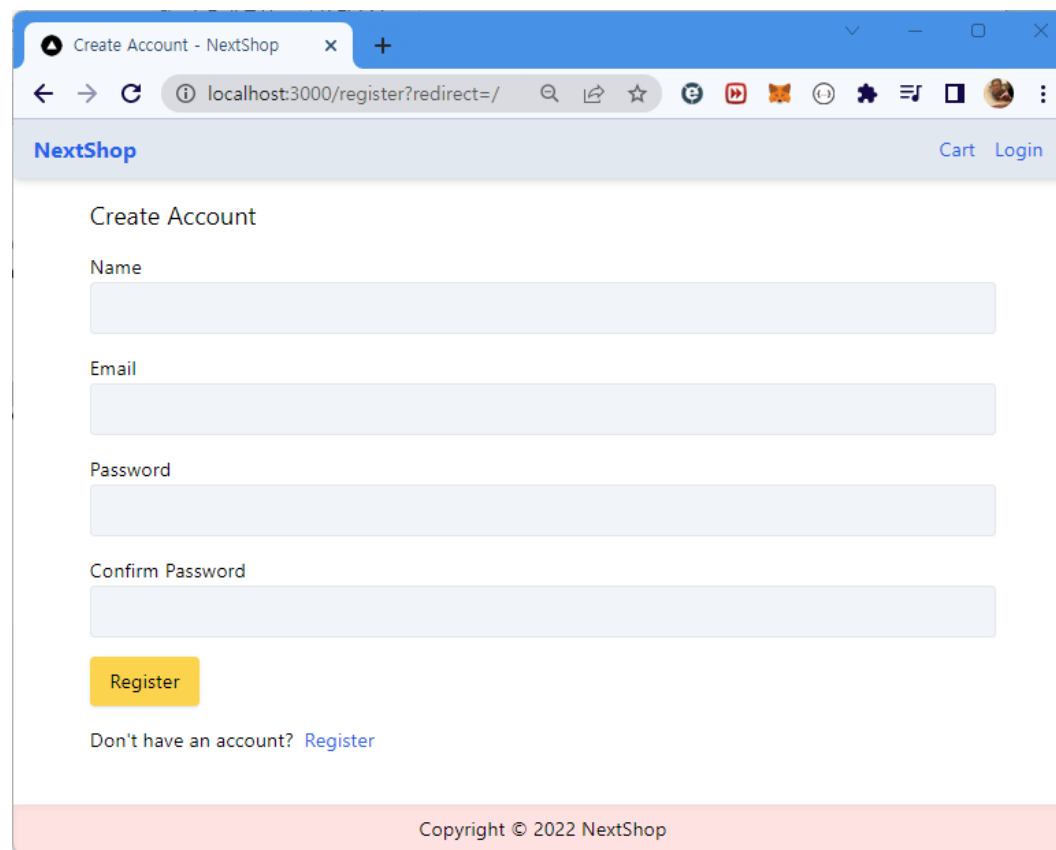
FILTER { field: 'value' }

```
{
  "_id": ObjectId('63329d7b96ee3db6f1cb8c3c'),
  "user": ObjectId('632d7390a91784c7a203f1f0'),
  "orderItems": Array
    > 0: Object
    > 1: Object
      "name": "Slim Shirt"
      "quantity": 1
      "image": "/images/shirt3.jpg"
      "price": 90
      "_id": ObjectId('632f08eebeb9b9048d703c60')
  "shippingAddress": Object
    "fullName": "byoungcheon"
    "address": "305 Dong-heon-ro"
    "city": "Goyang-si"
    "postalCode": "10279"
    "country": "대한민국"
    "paymentMethod": "PayPal"
    "itemsPrice": 170
    "shippingPrice": 15
    "taxPrice": 25.5
    "totalPrice": 210.5
    "isPaid": false
    "isDelivered": false
    "createdAt": 2022-09-27T06:51:39.118+00:00
    "updatedAt": 2022-09-27T06:51:39.118+00:00
    "__v": 0
}
```



3.19 Create Register Screen

- 사용자 등록 프론트엔드 페이지 작성
 - pages/register.js
- 사용자 등록 백엔드 API 작성
 - pages/api/auth/signup.js
- 사용자 등록, 로그인 테스트



The screenshot shows a web browser window with the title 'Create Account - NextShop'. The address bar shows 'localhost:3000/register?redirect=/' with search, share, and star icons. The page has a light blue header with 'NextShop' on the left and 'Cart Login' on the right. The main content area is titled 'Create Account' and contains four input fields: 'Name', 'Email', 'Password', and 'Confirm Password'. Below these fields is a yellow 'Register' button. At the bottom of the form, there is a link that says 'Don't have an account? Register'. The footer of the page is a light red bar with the text 'Copyright © 2022 NextShop'.

사용자 등록 프론트엔드

pages/register.js

```
import Link from 'next/link'
import React, { useEffect } from 'react'
import { signIn, useSession } from 'next-auth/react'
import { useForm } from 'react-hook-form'
import Layout from '../components/Layout'
import { getError } from '../utils/error'
import { toast } from 'react-toastify'
import { useRouter } from 'next/router'
import axios from 'axios'
```

```
export default function RegisterScreen() {
  const { data: session } = useSession()
```

```
  const router = useRouter()
  const { redirect } = router.query
```

```
  useEffect(() => {
    if (session?.user) {
      router.push(redirect || '/')
    }
  }, [router, session, redirect])
```

```
  const {
    handleSubmit,
    register,
    getValues,
    formState: { errors },
  } = useForm()
```

중략 / [github 소스코드 참조](#)

사용자 등록 백엔드

pages/api/auth/signup.js

```
import bcryptjs from 'bcryptjs'
import User from '../../models/User'
import db from '../../utils/db'
```

```
async function handler(req, res) {
  if (req.method !== 'POST') {
    return
  }
  const { name, email, password } = req.body
  if (
    !name ||
    !email ||
    !email.includes('@') ||
    !password ||
    password.trim().length < 3
  ) {
    res.status(422).json({
      message: 'Validation error',
    })
    return
  }
```

```
  await db.connect()
```

```
  const existingUser = await User.findOne({ email: email })
  if (existingUser) {
    res.status(422).json({ message: 'User exists already!' })
    await db.disconnect()
    return
  }
```

서버측 검증

패스워드해시 추가, 저장

```
const newUser = new User({
  name,
  email,
  password: bcryptjs.hashSync(password),
  isAdmin: false,
})
```

```
const user = await newUser.save()
await db.disconnect()
res.status(201).send({
  message: 'Created user!',
  _id: user._id,
  name: user.name,
  email: user.email,
  isAdmin: user.isAdmin,
})
```

```
export default handler
```

성공시 응답

기존 사용자 이메일 확인

테스트

pages/login.js

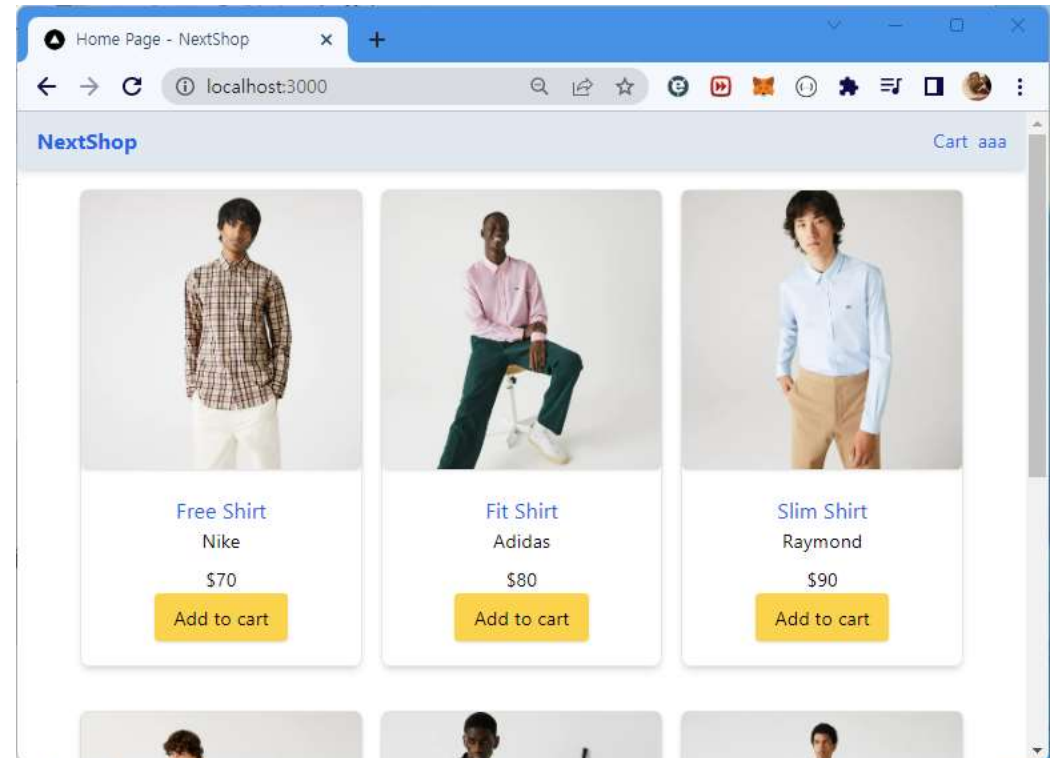
중략

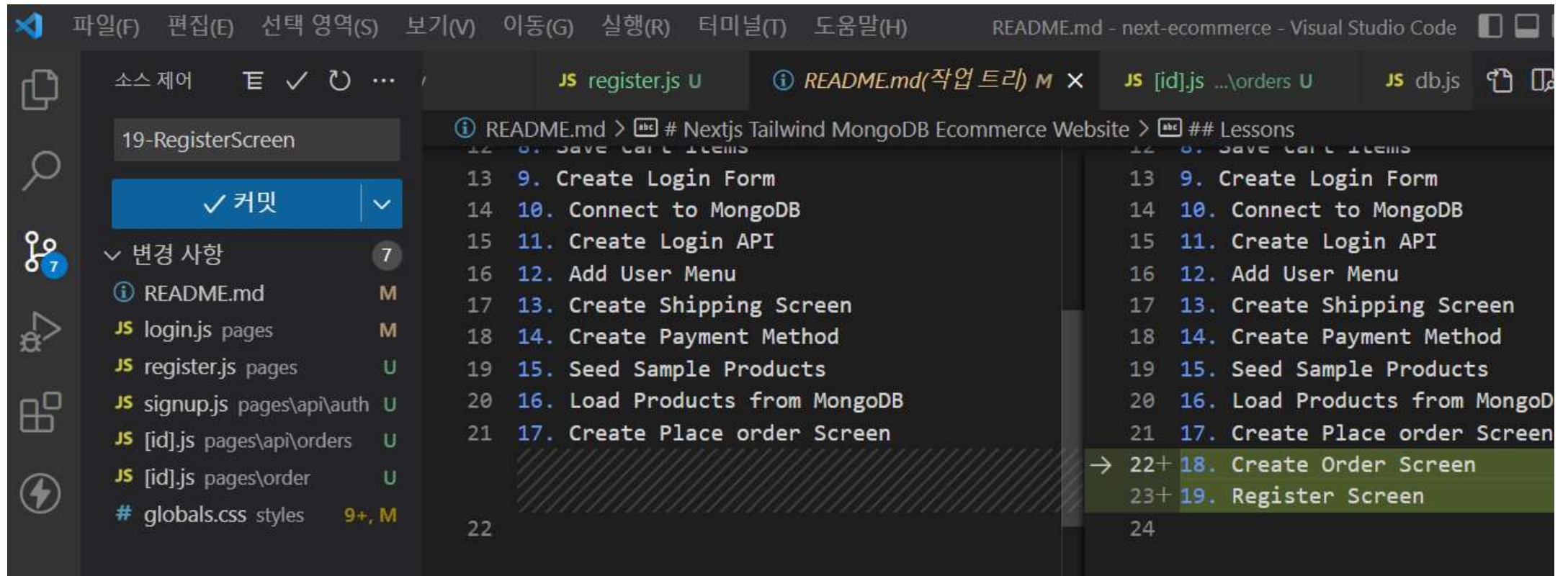
```
<div className="mb-4">  
  Don't have an account? &nbsp;  
  <Link href={` /register?redirect=${redirect || '/'}`}>Register</Link>  
</div>
```

중략

Redirect 정보를 포함한 Register 링크

새로운 아이디 등록 및 로그인





3.20 Paypal Payment

- Paypal.com에 등록, 로그인
 - <https://developer.paypal.com/home>
 - 가입하기
 - Paypal로 결제받기
 - My Apps & Credentials
 - Sandbox 환경으로 App 생성 (지불 테스트용)
- 사업자 등록이 까다로움
- 국내 지불수단 활용 필요

매일 수백만이 이용하는 PayPal과
함께하세요.

PayPal 계정을 만드세요. 가입은 무료입니다.

○ PayPal로 구매하기

주로 온라인으로 구매하는 사용자를 위한 서비스입니다.

- 일반적으로 구매는 무료입니다. 해외 거래는 환전 수수료가 적용될 수 있습니다.
- 전세계 수백만개의 온라인 매장에서 구매하면서 신용카드 포인트도 계속 적립할 수 있습니다.
- 적합한 구매는 PayPal의 구매자 보호 프로그램에 따라 보호됩니다.

● PayPal로 결제받기

주로 결제대금을 수령하는 판매자 및 비즈니스를 위한 서비스입니다.

- 개설 수수료나 월 수수료가 없습니다. 결제 금액을 수령할 때마다 **거래 수수료**만 지급하면 됩니다.
- 202개 국가의 25가지 통화로 전 세계에서 결제대금을 받을 수 있습니다.
- 적합한 판매는 PayPal의 판매자 보호 프로그램이 적용됩니다.

다음으로

Payment Method - NextShop x Applications - PayPal Developer x

developer.paypal.com/dashboard/applications/sandbox/create

Create New App

Before you create your new app, let us know what kind of solution you're looking for.

Application Details

App Name
NextShop

App Type

☒ **Merchant** - Accept payments as a merchant (seller)

☐ **Platform** - Move payments to sellers as a platform (marketplace, crowdfunding, or e-commerce platform)

Sandbox Business Account
sb-q6zoy21146357@business.exa...

As a reminder, all apps created under your account should be related to your business and the type of business it conducts.

By clicking the button below, you agree to [PayPal Developer Agreement](#) (US accounts only).

Create App

Feedback

Payment Method - NextShop x Edit applications - PayPal Developer x

developer.paypal.com/dashboard/applications/edit/58fQVdoLWNLInZ5VG...

NextShop

App display name: NextShop

SANDBOX API CREDENTIALS

Sandbox Account
sb-q6zoy21146357@business.example.com

Client ID
AWh-cKNvyTm8OWS0fPCCWolet1Su6ZT0KT4X5_OTLvDxjOKOQnVWB5N2vWFa97yGFe_GZq
xQ_V3zTns

Secret
[show](#)

SANDBOX APP SETTINGS

Return URL - Users are redirected to this URL after live transactions. Allow up to three hours for the change to take effect.[Show](#)

Feedback

Order 6332c1e8b2d730120161bf50

Shipping Address
byoungcheon, 305 Dong-heon-ro, Goyang-si, 10279, 대한민국
Not delivered

Payment Method
PayPal
Not paid

Order Summary


Items	\$80
Tax	\$12
Shipping	\$15
Total	\$107

PayPal

카드 결제 또는 신용카드

주문 PayPal

Order Items

Item	Quantity	Price	Subtotal
 Fit Shirt	1	\$80	\$80

Copyright © 2022 NextShop

PayPal 계정 로그인 - Chrome

sandbox.paypal.com/signin?intent=checkout&ctxId=xo_ctx_0A629...

PayPal로 지불하기

PayPal 계정이 있으면 무료 반품 배송, 구매 보호 등을 받을 수 있습니다.

⚠ 일부 정보가 정확하지 않습니다. 다시 시도하세요.

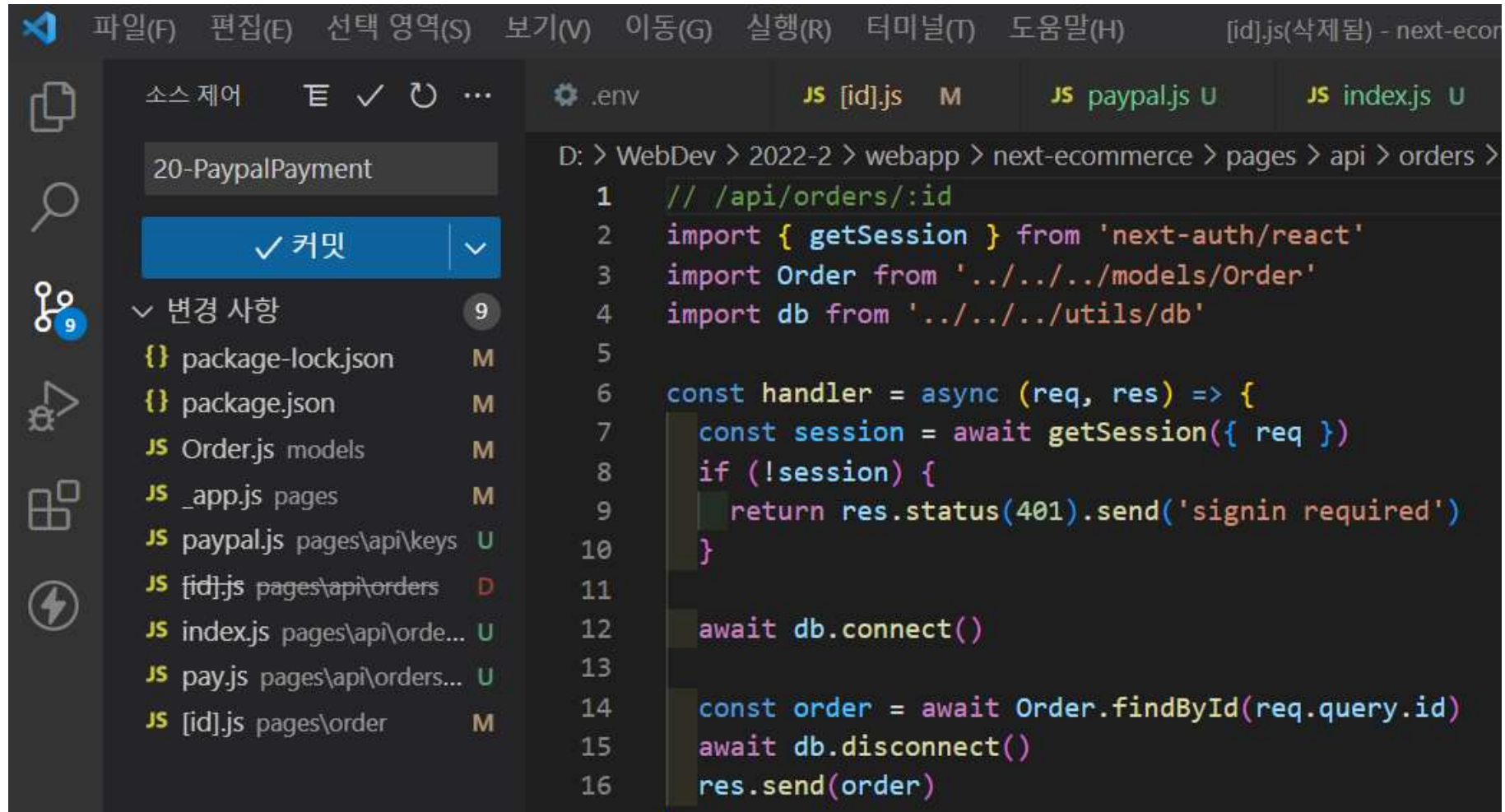
이메일 또는 휴대폰 번호
sultan6210@nate.com

비밀번호

로그인

로그인에 어려움이 있으신가요?

_____ 또는 _____



3.21 Order History

- 주문 기록 보여주기 기능
- 프론트엔드
 - Pages/order-history.js
- 백엔드
 - Pages/api/orders/history.js


```

import axios from 'axios'
import Link from 'next/link'
import React, { useEffect, useReducer } from 'react'
import Layout from '../components/Layout'
import { getError } from '../utils/error'

function reducer(state, action) {
  switch (action.type) {
    case 'FETCH_REQUEST':
      return { ...state, loading: true, error: '' }
    case 'FETCH_SUCCESS':
      return { ...state, loading: false, orders: action.payload, error: '' }
    case 'FETCH_FAIL':
      return { ...state, loading: false, error: action.payload }
    default:
      return state
  }
}

function OrderHistoryScreen() {
  const [{ loading, error, orders }, dispatch] = useReducer(reducer, {
    loading: true,
    orders: [],
    error: '',
  })

  useEffect(() => {
    const fetchOrders = async () => {
      try {
        dispatch({ type: 'FETCH_REQUEST' })
        const { data } = await axios.get('/api/orders/history')
        dispatch({ type: 'FETCH_SUCCESS', payload: data })
      } catch (err) {
        dispatch({ type: 'FETCH_FAIL', payload: getError(err) })
      }
    }
    fetchOrders()
  }, [])
}

```

```

return (
  <Layout title="Order History">
    <h1 className="mb-4 text-xl">Order History</h1>
    {loading ? (
      <div>Loading...</div>
    ) : error ? (
      <div className="alert-error">{error}</div>
    ) : (
      <div className="overflow-x-auto">
        <table className="min-w-full">
          <thead className="border-b">
            <tr>
              <th className="px-5 text-left">ID</th>
              <th className="p-5 text-left">DATE</th>
              <th className="p-5 text-left">TOTAL</th>
              <th className="p-5 text-left">PAID</th>
              <th className="p-5 text-left">DELIVERED</th>
              <th className="p-5 text-left">ACTION</th>
            </tr>
          </thead>
          <tbody>
            {orders.map((order) => (
              <tr key={order._id} className="border-b">
                <td className="p-5 ">{order._id.substring(20, 24)}</td>
                <td className="p-5 ">{order.createdAt.substring(0, 10)}</td>
                <td className="p-5 ">${order.totalPrice}</td>
                <td className="p-5 ">
                  {order.isPaid
                    ? `${order.paidAt.substring(0, 10)}`
                    : 'not paid'}
                </td>
                <td className="p-5 ">
                  {order.isDelivered
                    ? `${order.deliveredAt.substring(0, 10)}`
                    : 'not delivered'}
                </td>
                <td className="p-5 ">
                  <Link href={`/${order}/${order._id}`} passHref>
                    <a>Details</a>
                  </Link>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    )}
  </Layout>
)
}

```

```

OrderHistoryScreen.auth = true
export default OrderHistoryScreen

```

pages/order-history.js

프론트엔드

백엔드

pages/api/orders/history.js

```
import { getSession } from 'next-auth/react'
import Order from '../../models/Order'
import db from '../../utils/db'

const handler = async (req, res) => {
  const session = await getSession({ req })
  if (!session) {
    return res.status(401).send({ message: 'signin required' })
  }
  const { user } = session
  await db.connect()
  const orders = await Order.find({ user: user._id })
  await db.disconnect()
  res.send(orders)
}

export default handler
```

동작 테스트

Order History - NextShop

localhost:3000/order-history

NextShop [Cart aaa](#)

Order History

ID	DATE	TOTAL	PAID	DELIVERED	ACTION
3ba1	2022-09-27	\$187.5	not paid	not delivered	Details
bf50	2022-09-27	\$107	not paid	not delivered	Details
c205	2022-09-28	\$678.5	not paid	not delivered	Details
c21d	2022-09-28	\$299	not paid	not delivered	Details
c22d	2022-09-28	\$187.5	not paid	not delivered	Details

Copyright © 2022 NextShop

Order 6332c1e8b2d730120161bf50

NextShop [Cart aaa](#)

Order 6332c1e8b2d730120161bf50

Shipping Address

byoungcheon, 305 Dong-heon-ro, Goyang-si, 10279, 대한민국


Not delivered

Payment Method

PayPal

Not paid

Order Items

Item	Quantity	Price	Subtotal
 Fit Shirt	1	\$80	\$80

Order Summary

Items	\$80
Tax	\$12
Shipping	\$15
Total	\$107

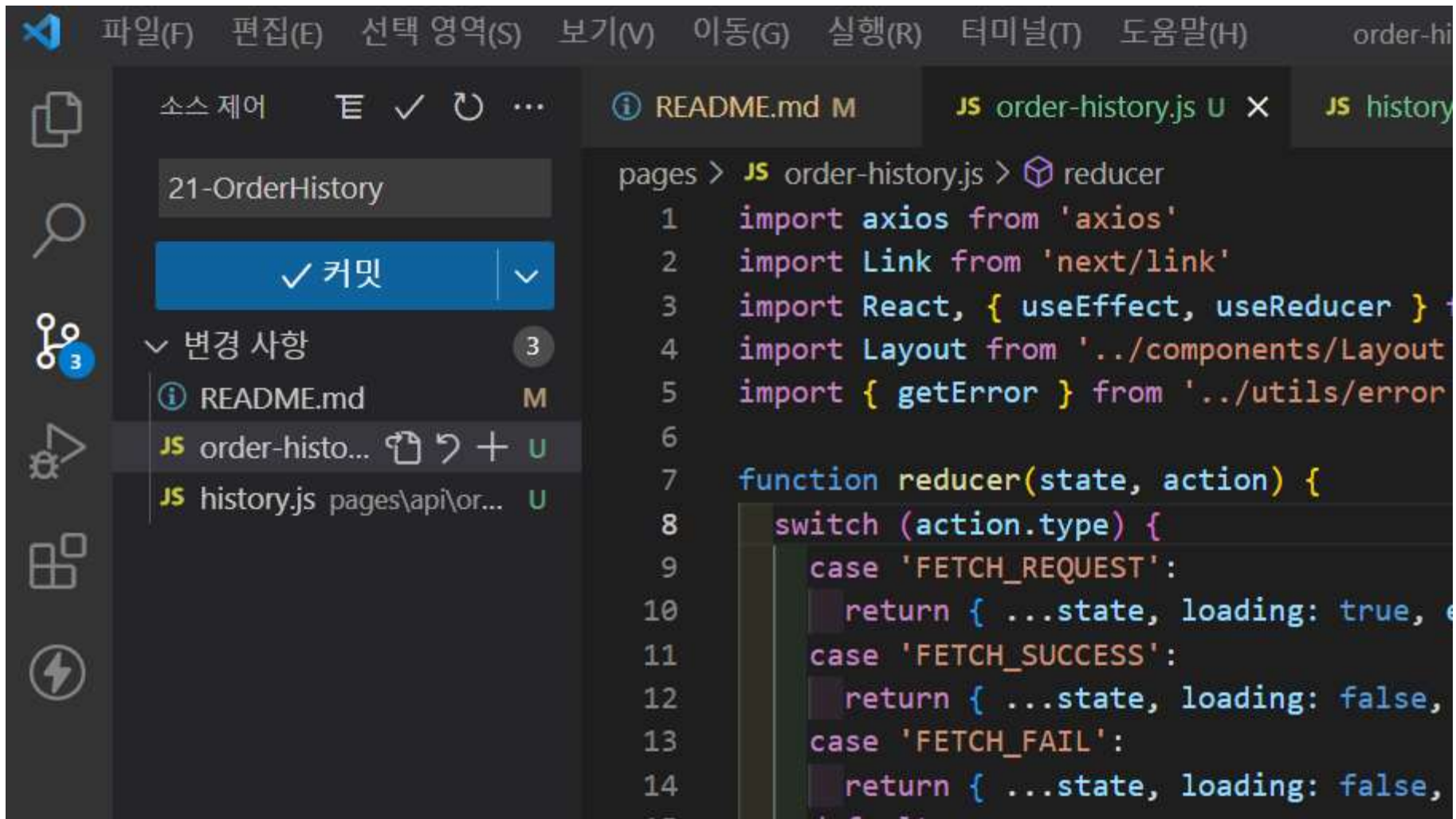
[PayPal](#)

[직불카드 또는 신용카드](#)

제공 [PayPal](#)

Copyright © 2022 NextShop

커밋, 저장



3.22 Update User Profile

- 사용자의 현재 프로필 정보를 보여줌
- 프로필 정보 수정 및 DB 업데이트
- 프론트엔드
 - Pages/profile.js
- 백엔드
 - Pages/api/auth/update.js

pages/profile.js

```
import React, { useEffect } from 'react'
import { signIn, useSession } from 'next-auth/react'
import { useForm } from 'react-hook-form'
import { toast } from 'react-toastify'
import { getError } from '../utils/error'
import axios from 'axios'
import Layout from '../components/Layout'
export default function ProfileScreen() {
  const { data: session } = useSession()
  const {
    handleSubmit,
    register,
    getValues,
    setValue,
    formState: { errors },
  } = useForm()
  useEffect(() => {
    setValue('name', session.user.name)
    setValue('email', session.user.email)
  }, [session.user, setValue])
  const submitHandler = async ({ name, email, password }) => {
    try {
      await axios.put('/api/auth/update', {
        name,
        email,
        password,
      })
      const result = await signIn('credentials', {
        redirect: false,
        email,
        password,
      })
      toast.success('Profile updated successfully')
      if (result.error) {
        toast.error(result.error)
      }
    }
  }
}
```

중략 – githun 소스코드 참조

pages/api/auth/update.js

```
import { getSession } from 'next-auth/react'
import bcryptjs from 'bcryptjs'
import User from '../../models/User'
import db from '../../utils/db'
async function handler(req, res) {
  if (req.method !== 'PUT') {
    return res.status(400).send({ message: `${req.method} not supported` })
  }
  const session = await getSession({ req })
  if (!session) {
    return res.status(401).send({ message: 'signin required' })
  }
  const { user } = session
  const { name, email, password } = req.body

  if (
    !name ||
    !email ||
    !email.includes('@') ||
    (password && password.trim().length < 5)
  ) {
    res.status(422).json({
      message: 'Validation error',
    })
    return
  }
  await db.connect()
  const toUpdateUser = await User.findById(user._id)
  toUpdateUser.name = name
  toUpdateUser.email = email
  if (password) {
    toUpdateUser.password = bcryptjs.hashSync(password)
  }
  await toUpdateUser.save()
  await db.disconnect()
  res.send({
    message: 'User updated',
  })
}
export default handler
```

Profile - NextShop

localhost:3000/pro...

NextShop

Cart aaa

Update Profile

Name

aaa

Email

aaa@aaa.com

Password

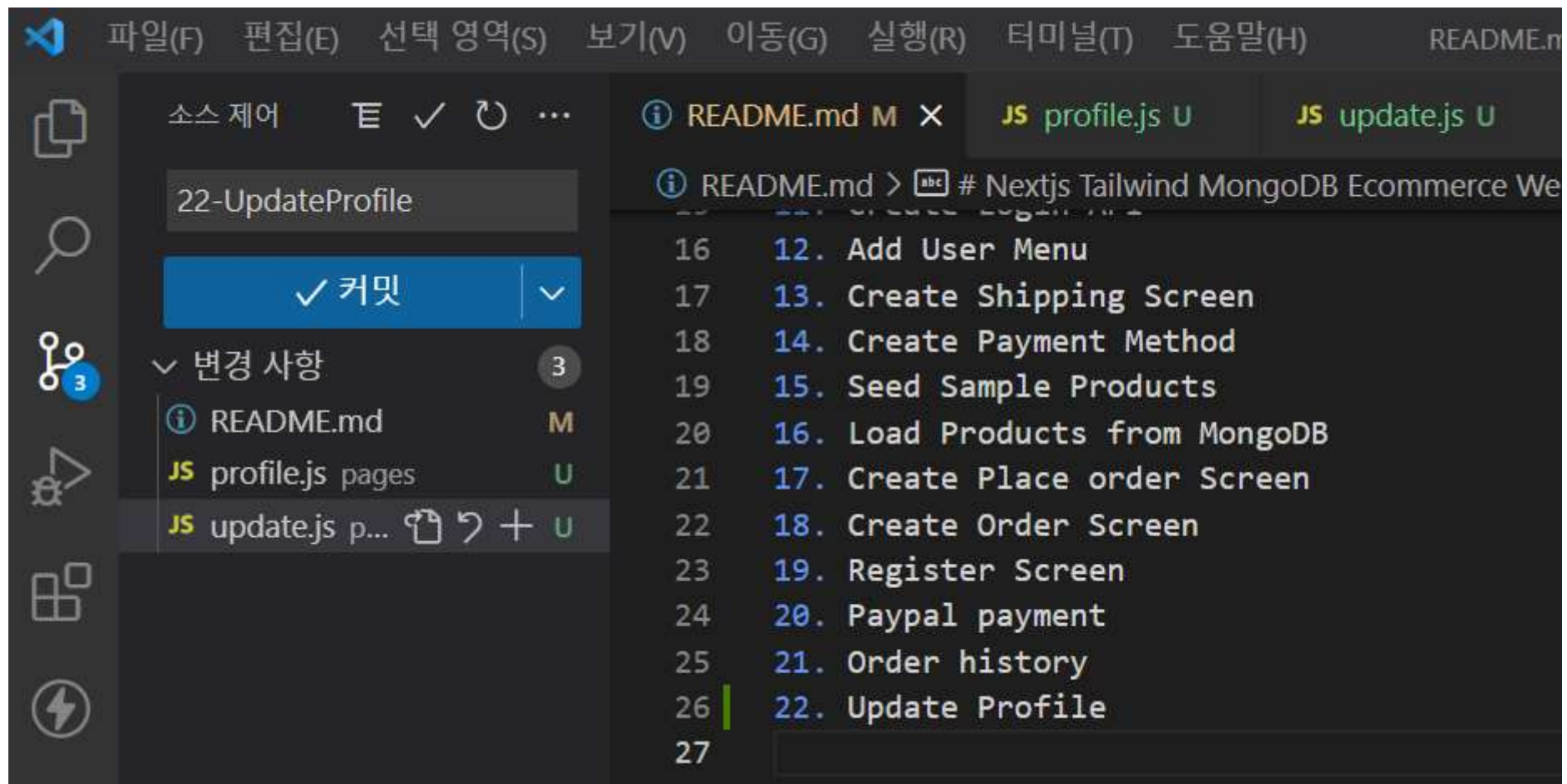
.....

Confirm Password

.....

Update Profile

Copyright © 2022 NextShop



3.23 Deploy to Vercel.com

- Lint test
 - > npm run lint
- Build
 - > npm run build
- Build version 실행
 - > npm run start

```
> next-ecommerce@0.1.0 lint
> next lint

info - Loaded env from D:\WebDev\2022-2\webapp\next-ecommerce\.env
info - SWC minify release candidate enabled. https://nextjs.link/swcmin
✓ No ESLint warnings or errors
```

Route (pages)	Size	First Load JS
λ /	1.98 kB	123 kB
/_app	0 B	92.4 kB
o /404	186 B	92.6 kB
λ /api/auth/[...nextauth]	0 B	92.4 kB
λ /api/auth/signup	0 B	92.4 kB
λ /api/auth/update	0 B	92.4 kB
λ /api/hello	0 B	92.4 kB
λ /api/keys/paypal	0 B	92.4 kB
λ /api/orders	0 B	92.4 kB
λ /api/orders/[id]	0 B	92.4 kB
chunks/framework-9b5d6ec4444c80fa.js	45.7 kB	
chunks/main-3123a443c688934f.js	30.9 kB	
chunks/pages/_app-58c0612a4447efc0.js	14.9 kB	
chunks/webpack-3433a2a2d0cf6fb6.js	819 B	
css/5bbd045c252269d4.css	3 kB	

λ (Server) server-side renders at runtime (uses `getInitialProps` or `getServerSideProps`)

o (Static) automatically rendered as static HTML (uses no initial props)

Vercel.com

- 회원가입
- 로그인
- Add New Project
- Import Git Repository
- Project Name 설정
- Environment Variables 등록
 - .env 파일의 내용을 등록
- Deploy

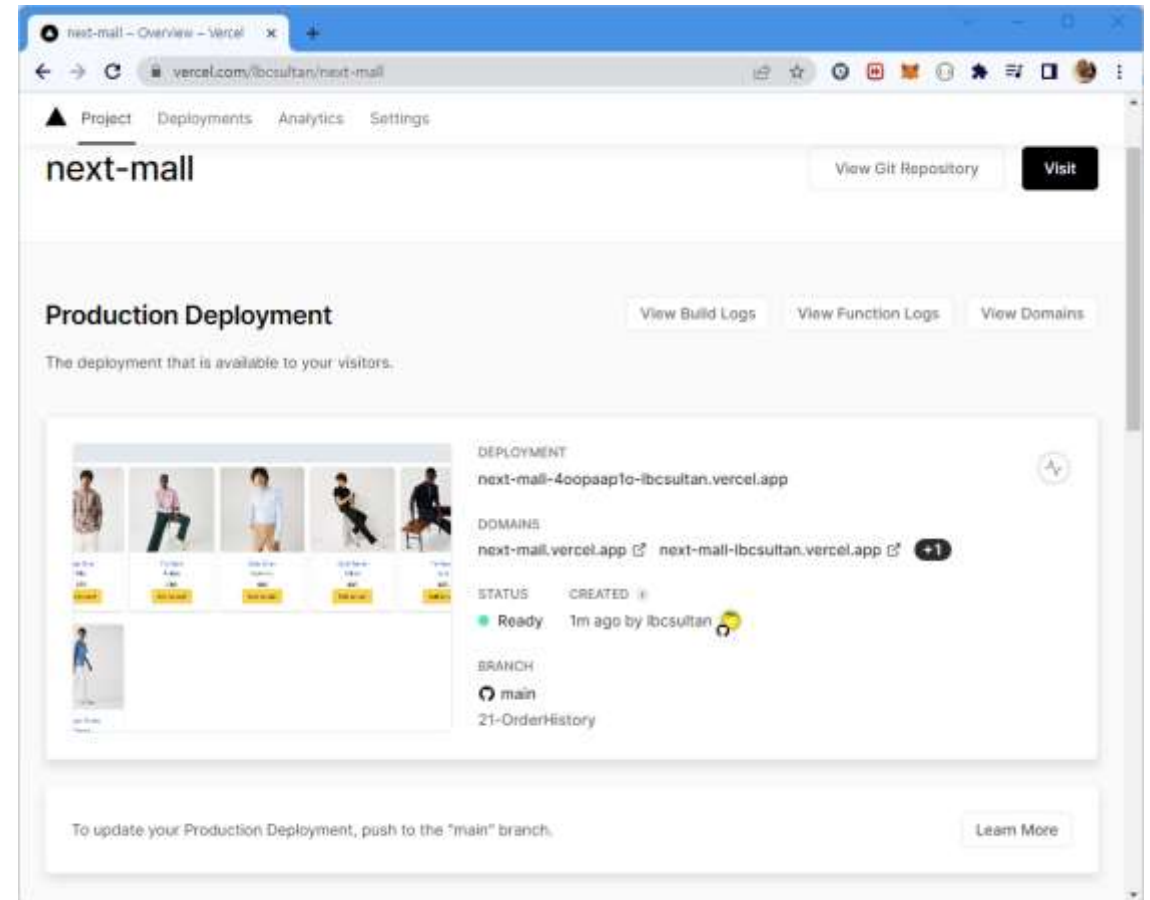
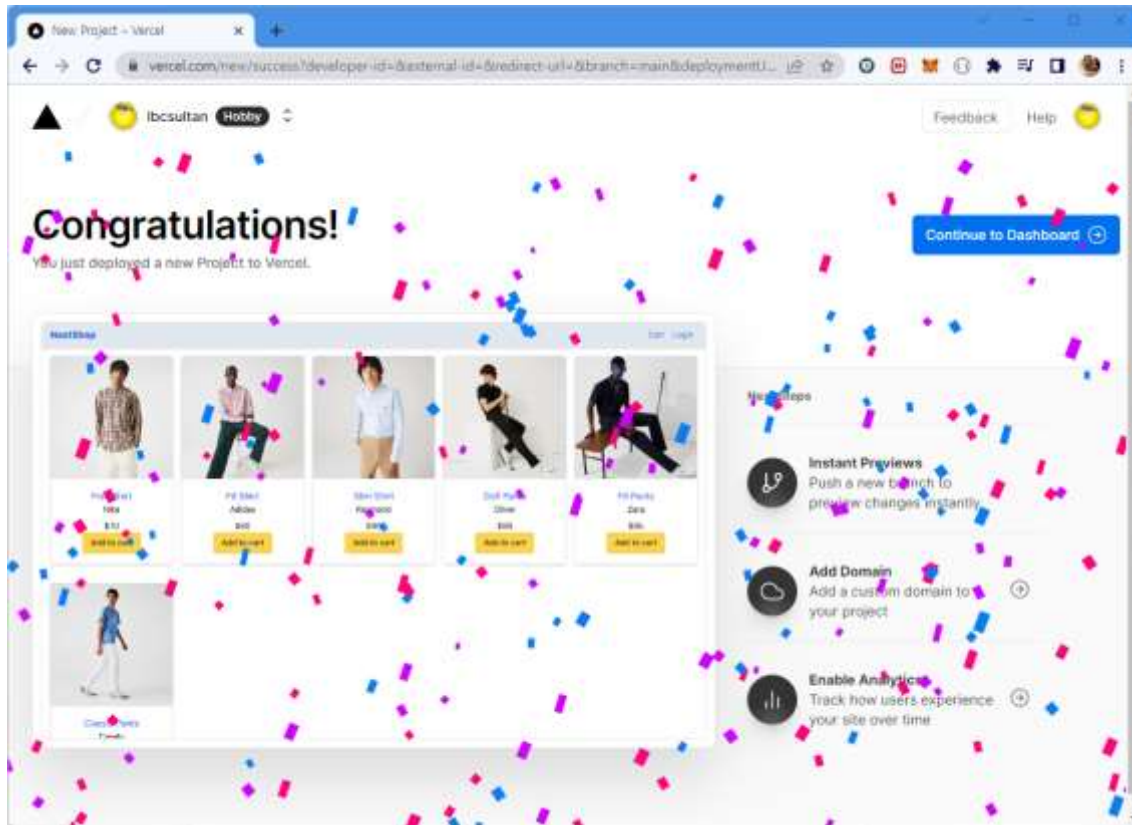
▼ Environment Variables

NAME	VALUE (WILL BE ENCRYPTED)	
<input type="text" value="EXAMPLE_NAME"/>	<input type="text" value="I9JU23NF394R6HH"/>	<input type="button" value="Add"/>

Learn more about [Environment Variables](#) ↗

NAME	VALUE	
PAYPAL_CLIENT_ID	Awh-ckNvyTm80WS0fPCCwoIet1Su6ZT0KT4X5_O...	⋮
NEXTAUTH_URL	https://next-mall.vercel.app	⋮
NEXTAUTH_SECRET	fdk1jsd1kjfsk1djf1skjdf1sk	⋮
MONGODB_URI	mongodb+srv://nextshop:nextshop@cluster...	⋮

결과



요약

- Next.js 쇼핑몰 예제 프로젝트
 - 프로젝트 시작, 레이아웃, Tailwind CSS, 컴포넌트
 - 백엔드 API
 - 상품 전시
 - 쇼핑카트
 - 주문, 지불, 히스토리
 - 사용자등록, 로그인, 프로필
 - MongoDB 활용, 데이터 모델, CRUD
 - 세션, 상태관리, Context, useState, useEffect, useReducer
 - Deploy to Vercel.com
 - 환경변수 관리