Full Stack React Ecommerce with NextJs NextAuth NextAPI

Build a massive ecommerce applications using Next.js, NextAuth, NextAPI and deploy to Vercel. Stay in touch with me Ryan Dhungel on twitter.com/@kaloraat for more..

- Full Stack React Ecommerce with NextJs NextAuth NextAPI
  - Full Authentication & Authorization using NextAuth
  - Create nextjs project
  - Using bootstrap material css
  - Create navigation
  - Nextjs API routes
  - Signup to mongodb
  - Using ENV variables
  - Connect to mongondb
  - Create user model
  - Register API route
  - Register form
  - Regisger API request
  - Login page
  - Email and password login with next auth
  - NextAuth configuration
  - Provide user session from next auth
  - Access logged in user info
  - Loading page
  - 404 Page not found
  - User dashboard
  - Protecting pages
  - Redirect back to intended page
  - What is next?

Full Authentication & Authorization using NextAuth

Create nextjs project

```
mkdir nextcom
cd nextcom
npx create-next-app@latest
// use . to create project inside nextcom
// pick typescript
// run project using
npm run dev
```

Using bootstrap material css

```
// remove all css from globals.css and page.module.css

// app/layout.js
import "./globals.css";

export const metadata = {
  title: "Create Next App",
  description: "Generated by create next app",
};

export default function RootLayout({ children }) {
  return (
    <html lang="en">
      <body>{children}</body>
    </html>
  );
}


// app/page.js
export default function Home() {
  return (
    <main>
      <div>
        <h1>Home</h1>
      </div>
    </main>
  );
}

// install bootstrap-material
// npm i bootstrap-material-design

// import in layout
import "bootstrap-material-design/dist/css/bootstrap-material-
design.min.css";

// try some bootstrap-material class names
// app/page.js
export default function Home() {
  return (
    <main>
      <div>
        <h1 className="d-flex justify-content-center align-items-center
vh-100 text-uppercase">
          Home
        </h1>
      </div>
    </main>
  );
}
```

Create navigation

```
// components/nav/TopNav.js
import Link from "next/link";

export default function TopNav() {
  return (
    <nav className="nav shadow p-2 justify-content-between mb-3">
      <Link className="nav-link" href="/">
        🛒 NEXTCOM
      </Link>

      <div className="d-flex">
        <Link className="nav-link" href="/login">
          Login
        </Link>
        <Link className="nav-link" href="/register">
          Register
        </Link>
      </div>
    </nav>
  );
}

// import in layout
import TopNav from "@/components/nav/TopNav";
// ...

export default function RootLayout({ children }) {
  return (
    <html lang="en">
      <body>
        <TopNav />
        {children}
      </body>
    </html>
  );
}
```

Nextjs API routes

```
// app/api/route.js
import { NextResponse } from "next/server";

export async function GET(req) {
  return NextResponse.json({ time: new Date().toLocaleString() });
}

// try visiting
// http://localhost:3000/api
```

Signup to mongodb

Signup to mongo atlas to get a connection string A tutorial link

Using ENV variables

Use custom `config` file along with `next.config.js` to use `env` variables so that it works perfectly once deployed to **vercel**

```
// config.js
const DB_URI =
  process.env.NODE_ENV === "production"
    ? "mongodb+srv://ryan:xxx@nextecom.xxx.mongodb.net/?
retryWrites=true&w=majority"
    : "mongodb://localhost:27017/nextecom";

module.exports = {
  DB_URI,
};

// next.config.js
const config = require("./config");

/** @type {import('next').NextConfig} */
const nextConfig = {
  env: {
    DB_URI: config.DB_URI,
  },
};

module.exports = nextConfig;
```

Connect to mongondb

```
// npm i mongoose mongoose-unique-validator

// utils/dbConnect.js
import mongoose from "mongoose";

const dbConnect = async () => {
  if (mongoose.connection.readyState >= 1) {
    return;
  }
  mongoose.connect(process.env.DB_URI);
};

export default dbConnect;
```

Create user model

```
// models/user
import mongoose from "mongoose";
import uniqueValidator from "mongoose-unique-validator";

const userSchema = new mongoose.Schema(
  {
    name: {
      type: String,
      required: [true, "Please enter your name"],
      trim: true,
      minLength: 1,
      maxLength: 20,
    },
    email: {
      type: String,
      required: true,
      index: true,
      lowercase: true,
      unique: true,
      trim: true,
      minLength: 5,
      maxLength: 20,
    },
    password: String,
    role: {
      type: String,
      default: "user",
    },
    image: String,
    resetCode: {
      data: String,
      expiresAt: {
        type: Date,
        default: () => new Date(Date.now() + 10 * 60 * 1000), // 10
minutes in milliseconds
      },
    },
  },
  { timestamps: true }
);

userSchema.plugin(uniqueValidator);

export default mongoose.models.User || mongoose.model("User", userSchema);
```

Register API route

```
// npm i bcrypt

// app/api/register/route.js
import { NextResponse } from "next/server";
import dbConnect from "@/utils/dbConnect";
import User from "@/models/user";
import bcrypt from "bcrypt";

export async function POST(req) {
  const body = await req.json();
  // console.log("body in register api => ", body);

  await dbConnect();

  try {
    const { name, email, password } = body;

    await new User({
      name,
      email,
      password: await bcrypt.hash(password, 10),
    }).save();

    return NextResponse.json({ success: "Registered Successfully" });
  } catch (err) {
    console.log(err);
    return NextResponse.json({ err: err.message }, { status: 500 });
  }
}

// try using postman
// send name, email, password as json format in re.body
```

Register form

```
// app/register/page.js
"use client";
import { set } from "mongoose";
import { useState } from "react";
import { FormEvent } from "react";

export default function Register() {
  const [name, setName] = useState("Ryan");
  const [email, setEmail] = useState("ryan@gmail.com");
  const [password, setPassword] = useState("rrrrrr");
  const [loading, setLoading] = useState(false);

  const handleSubmit = async (e: FormEvent<HTMLFormElement>) => {
    try {
      e.preventDefault();
```

```
      setLoading(true);
      console.table({ name, email, password });
    } catch (err) {
      console.log(err);
      setLoading(false);
    }
  };

  return (
    <main>
      <div className="container">
        <div className="row d-flex justify-content-center align-items-
center vh-100">
          <div className="col-lg-5 bg-light p-5 shadow">
            <h2 className="mb-4">Register</h2>

            <form onSubmit={handleSubmit}>
              <input
                type="text"
                value={name}
                onChange={(e) => setName(e.target.value)}
                className="form-control mb-4"
                placeholder="Your name"
              />
              <input
                type="email"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
                className="form-control mb-4"
                placeholder="Your email"
              />
              <input
                type="password"
                value={password}
                onChange={(e) => setPassword(e.target.value)}
                className="form-control mb-4"
                placeholder="Your password"
              />
              <button
                className="btn btn-primary btn-raised"
                disabled={loading || !name || !email || !password}
              >
                {loading ? "Please wait.." : "Submit"}
              </button>
            </form>
          </div>
        </div>
      </div>
    </main>
  );
}
```

Regisger API request

```
// config.js
const API =
  process.env.NODE_ENV === "production"
    ? "https://xxx.vercel.com/api"
    : "http://localhost:3000/api";

module.exports = {
  // ...
  API,
};

// next.config.js
const nextConfig = {
  env: {
    DB_URI: config.DB_URI,
    API: config.API,
  },
};

// npm i react-hot-toast

// layout.tsx
import { Toaster } from "react-hot-toast";
// ...
<body>
  <TopNav />
  <Toaster />
  {children}
</body>;
// ...

// app/register/page
import toast from "react-hot-toast";
import { useRouter } from "next/navigation";

// ...
const router = useRouter();

const handleSubmit = async (e: FormEvent<HTMLFormElement>) => {
  try {
    e.preventDefault();
    setLoading(true);

    const response = await fetch(`${process.env.API}/register`, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        name,
```

```
        email,
        password,
      }),
    });

    const data = await response.json();

    if (!response.ok) {
      toast.error(data.err);
      setLoading(false);
    } else {
      toast.success(data.success);
      router.push("/login");
    }
  } catch (err) {
    console.log(err);
    setLoading(false);
  }
};
```

Login page

```
// app/login/page
"use client";
import { set } from "mongoose";
import { useState } from "react";
import { FormEvent } from "react";
import toast from "react-hot-toast";
import { useRouter } from "next/navigation";

export default function Register() {
  const [email, setEmail] = useState("ryan@gmail.com");
  const [password, setPassword] = useState("rrrrrr");
  const [loading, setLoading] = useState(false);

  const router = useRouter();

  const handleSubmit = async (e: FormEvent<HTMLFormElement>) => {
    e.preventDefault();
    //
  };

  return (
    <main>
      <div className="container">
        <div className="row d-flex justify-content-center align-items-
center vh-100">
          <div className="col-lg-5 bg-light p-5 shadow">
            <h2 className="mb-4">Login</h2>

            <form onSubmit={handleSubmit}>
```

```
              <input
                type="email"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
                className="form-control mb-4"
                placeholder="Your email"
              />
              <input
                type="password"
                value={password}
                onChange={(e) => setPassword(e.target.value)}
                className="form-control mb-4"
                placeholder="Your password"
              />
              <button
                className="btn btn-primary btn-raised"
                disabled={loading || !email || !password}
              >
                {loading ? "Please wait.." : "Submit"}
              </button>
            </form>
          </div>
        </div>
      </div>
    </main>
  );
}
```

Email and password login with next auth

```
// npm i net-auth

// app/login/page
import { signIn } from "next-auth/react";

// ...
const handleSubmit = async (e: FormEvent<HTMLFormElement>) => {
  e.preventDefault();
  setLoading(true);

  const result = await signIn("credentials", {
    redirect: false,
    email,
    password,
  });

  setLoading(false);

  if (result?.error) {
    toast.error(result?.error);
  } else {
```

```
    toast.success("Login success");
    router.push("/");
  }
};
// without next-auth config, you get redirected to '/api/auth/error'
```

NextAuth configuration

```
// config.js
// name "NEXTAUTH_SECRET" is important, dont rename it
const NEXTAUTH_SECRET = "YOUR_SECRET";

// utils/authOptions.js
import CredentialsProvider from "next-auth/providers/credentials";
import User from "@/models/user";
import bcrypt from "bcrypt";
import dbConnect from "@/utils/dbConnect";

export const authOptions = {
  session: {
    strategy: "jwt",
  },
  providers: [
    CredentialsProvider({
      async authorize(credentials, req) {
        dbConnect();

        const { email, password } = credentials;

        const user = await User.findOne({ email });

        if (!user) {
          throw new Error("Invalid email or password");
        }

        // If the user has no password (i.e., they signed up via a social
network), throw an error
        if (!user?.password) {
          throw new Error("Please login via the method you used to sign
up");
        }

        const isPasswordMatched = await bcrypt.compare(
          password,
          user?.password
        );

        if (!isPasswordMatched) {
          throw new Error("Invalid email or password");
        }
```

```
        return user;
      },
    }),
  ],
  secret: process.env.NEXT_AUTH_SECRET,
  pages: {
    signIn: "/login",
  },
};

// use authOptions in [...nextauth]/route
// app/api/auth/[...nextauth]/route
import NextAuth from "next-auth";

import { authOptions } from "@/utils/authOptions";

const handler = NextAuth(authOptions);

export { handler as GET, handler as POST };
```

Provide user session from next auth

```
// SessionProvider in Layout
"use client";
import "./globals.css";
import "bootstrap-material-design/dist/css/bootstrap-material-
design.min.css";
import TopNav from "@/components/nav/TopNav";
import { Toaster } from "react-hot-toast";
import { SessionProvider } from "next-auth/react";

export default function RootLayout({ children }) {
  return (
    <html lang="en">
      <SessionProvider>
        <body>
          <TopNav />
          <Toaster />
          {/* children props/components can be server rendered */}
          {children}
        </body>
      </SessionProvider>
    </html>
  );
}
```

Access logged in user info

```
// components/nav/TopNav
import Link from "next/link";
import { useSession, signOut } from "next-auth/react";

export default function TopNav() {
  const { data, status, loading } = useSession();

  // console.log(data, status);

  return (
    <nav className="nav shadow p-2 justify-content-between mb-3">
      <Link className="nav-link" href="/">
        🛒 NEXTCOM
      </Link>

      {status === "authenticated" ? (
        <div className="d-flex">
          <Link className="nav-link" href="/dashboard/user">
            {data?.user?.name}
          </Link>
          <a
            className="nav-link pointer"
            onClick={() => signOut({ callbackUrl: "/login" })}
          >
            Logout
          </a>
        </div>
      ) : (
        <div className="d-flex">
          <Link className="nav-link" href="/login">
            Login
          </Link>
          <Link className="nav-link" href="/register">
            Register
          </Link>
        </div>
      )}
    </nav>
  );
}
```

```
// globals.css
.pointer {
  cursor: pointer;
}
```

Loading page

This is default loading page in nextjs

```
// app/loading.js
export default function Loading() {
  return (
    <div className="d-flex justify-content-center align-items-center vh-
100 text-danger">
      LOADING
    </div>
  );
}

// using session 'loading' status
// TopNav
return (
  <nav className="nav shadow p-2 justify-content-between mb-3">
    <Link className="nav-link" href="/">
      🛒 NEXTCOM
    </Link>

    {status === "authenticated" ? (
      <div className="d-flex">{/*  */}</div>
    ) : status === "loading" ? (
      <div className="d-flex">
        <a className="nav-link text-danger">Loading</a>
      </div>
    ) : (
      <div className="d-flex">{/*  */}</div>
    )}
  </nav>
);
```

404 Page not found

```
// app/not-found.js
export default function NotFound() {
  return (
    <div className="d-flex justify-content-center align-items-center vh-
100 text-danger">
      404
    </div>
  );
}
```

User dashboard

```
// app/dashboard/user/page
export default function UserDashboard() {
  return (
```

```
      <div className="container">
        <div className="row">
          <div className="col">
            <p className="lead">Dashboard</p>
            <hr />
            ...
          </div>
        </div>
      </div>
    );
}
// this page is accessible to anyone
```

Protecting pages

Protect dashboard pages

```
// middleware.js
export { default } from "next-auth/middleware";
export const config = { matcher: ["/dashboard/:path*"] };
```

Redirect back to intended page

```
// login
import { useRouter, useSearchParams } from "next/navigation";

const router = useRouter();
const searchParams = useSearchParams();
const callbackUrl = searchParams.get("callbackUrl") || "/";

// handleSubmit()
router.push(callbackUrl);
```

What is next?

Unfortunately, Udemy only allows 2 hours of content on free course. But do not worry, If you are enjoying this course, I will make sure you will have access to the rest of the course materials soon in future.

You can stay in touch with me on Twitter to learn more.