```
import { NextAuthOptions } from "next-auth";
import CredentialsProvider from "next-auth/providers/credentials";
import GitHubProvider from "next-auth/providers/github";
import { prisma } from "./prisma";
import bcrypt from "bcryptjs";

export const authOptions: NextAuthOptions = {
  providers: [
    CredentialsProvider({
      name: "Credentials",
      credentials: {
        email: { label: "Email", type: "text" },
        password: { label: "Password", type: "password" }
      },
      async authorize(credentials) {
        if (!credentials?.email || !credentials.password) return null;

        const user = await prisma.user.findUnique({
          where: { email: credentials.email }
        });

        if (!user || !user.passwordHash) return null;

        const isValid = await bcrypt.compare(
          credentials.password,
          user.passwordHash
        );

        if (!isValid) return null;

        return {
          id: user.id,
          email: user.email,
          name: user.name,
          role: user.role
        };
      }
    }),
    GitHubProvider({
      clientId: process.env.GITHUB_ID || "",
      clientSecret: process.env.GITHUB_SECRET || ""
    })
  ],
```

```
  callbacks: {
    async session({ session, token }) {
      if (session.user) {
        session.user.id = token.sub!;
        session.user.role = token.role as string;
      }
      return session;
    },
    async jwt({ token, user }) {
      if (user) {
        token.role = (user as any).role;
      }
      return token;
    }
  },
  secret: process.env.NEXTAUTH_SECRET
};
```