Setup Next Auth

1. Install Dependencies

```
npm install next-auth bcryptjs
```

```
// app/api/auth/[...nextauth]/route.ts
import NextAuth, { NextAuthOptions } from 'next-auth'
import CredentialsProvider from 'next-auth/providers/credentials'
import prisma from '@/lib/prisma'
import bcrypt from 'bcryptjs'

declare module 'next-auth' {
  interface Session {
    user: {
      id: string
      name: string
      email: string
      role: string
    }
  }

  interface User {
    id: string
    name: string
    email: string
    role: string
  }
}

declare module 'next-auth/jwt' {
  interface JWT {
    id: string
    role: string
  }
}

export const authOptions: NextAuthOptions = {
  session: {
    strategy: 'jwt',
  },
```

```
  providers: [
    CredentialsProvider({
      name: 'Credentials',
      credentials: {
        email: { label: 'Email', type: 'email' },
        password: { label: 'Password', type: 'password' },
      },
      async authorize(credentials) {
        const { email, password } = credentials as {
          email: string
          password: string
        }

        // Find user by email
        const user = await prisma.user.findUnique({ where: { email } })

        if (!user || !user.password) {
          throw new Error('Invalid credentials')
        }

        const isPasswordValid = await bcrypt.compare(password,
user.password)
        if (!isPasswordValid) {
          throw new Error('Invalid credentials')
        }

        return {
          id: user.id,
          name: user.username,
          email: user.email,
          role: user.role,
        }
      },
    }),
  ],
  callbacks: {
    async jwt({ token, user }) {
      if (user) {
        token.id = user.id
        token.role = user.role
      }
      return token
```

```
    },
    async session({ session, token }) {
      if (session.user) {
        session.user.id = token.id as string
        session.user.role = token.role as string
      }
      return session
    },
  },
  pages: {
    signIn: '/login',
  },
  secret: process.env.NEXTAUTH_SECRET,
}

const handler = NextAuth(authOptions)

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

🔐 9. Protect /dashboard

```
Use getServerSession() in Server Component:
tsx
CopyEdit
// app/dashboard/page.tsx
import { getServerSession } from 'next-auth'
import { authOptions } from '@/lib/auth'
import { redirect } from 'next/navigation'

export default async function DashboardPage() {
  const session = await getServerSession(authOptions)
  if (!session) redirect('/login')

  return <div>Welcome, {session.user?.email}</div>
}
```