1 **frontend**

Setup Next.js Project

Install Next.js 13: npm install next@13 react@17 react-dom@17

Create a new Next.js app: npx create-next-app@13 your-app-name

Navigate into the app directory: cd your-app-name

Install Dependencies

Install Tailwind CSS: npm install tailwindcss@latest postcss@latest autoprefixer@latest

Install Axios for API requests: npm install axios

Configure Tailwind CSS

Create a tailwind.config.js file:

// tailwind.config.js

module.exports = {

mode: 'jit',

purge: ['./pages/\*\*/\*.{js,ts,jsx,tsx}', './components/\*\*/\*.{js,ts,jsx,tsx}'],

darkMode: false, // or 'media' or 'class'

theme: {

extend: {},

},

variants: {

extend: {},

},

plugins: [],

};

Create a postcss.config.js file in the root:

// postcss.config.js

module.exports = {

plugins: {

tailwindcss: {},

autoprefixer: {},

},

};

Implement App Router:

Use Next.js's built-in Link component for client-side routing:  
// pages/\_app.js

import { useEffect } from 'react';

import { useRouter } from 'next/router';

import '../styles/globals.css'; // Include Tailwind CSS styles

function MyApp({ Component, pageProps }) {

const router = useRouter();

useEffect(() => {

// Prefetch the dashboard page

router.prefetch('/dashboard');

}, []);

return <Component {...pageProps} />;

}

export default MyApp;

Create Pages

Create pages for login, register, profile, etc. using Next.js's file-based routing:

// pages/login.js

import React from 'react';

function Login() {

return <div>Login Page</div>;

}

export default Login;

Connect with API:

Use Axios to make API requests:

// Example API call to get user profile

import axios from 'axios';

async function getProfile() {

try {

const response = await axios.get('/api/profile');

return response.data;

} catch (error) {

console.error('Error fetching profile:', error);

return null;

}

}

Design and UI:

Use the provided Figma design to implement the UI using Tailwind CSS classes.

**2 backend**

User Schema:

import \* as mongoose from 'mongoose';

export const UserSchema = new mongoose.Schema({

email: { type: String, unique: true },

password: String,

// Add other fields like horoscope and zodiac here

});

export interface User extends mongoose.Document {

email: string;

password: string;

// Add other fields like horoscope and zodiac here

}

User Model:

import { Injectable } from '@nestjs/common';

import { InjectModel } from '@nestjs/mongoose';

import { Model } from 'mongoose';

import { User } from './user.schema';

@Injectable()

export class UserService {

constructor(@InjectModel('User') private readonly userModel: Model<User>) {}

async findOneByEmail(email: string): Promise<User | undefined> {

return this.userModel.findOne({ email }).exec();

}

async create(user: User): Promise<User> {

const createdUser = new this.userModel(user);

return createdUser.save();

}

}

Auth Service:

import { Injectable } from '@nestjs/common';

import { UserService } from './user.service';

import { JwtService } from '@nestjs/jwt';

import { User } from './user.schema';

@Injectable()

export class AuthService {

constructor(

private readonly userService: UserService,

private readonly jwtService: JwtService,

) {}

async validateUser(email: string, password: string): Promise<any> {

const user = await this.userService.findOneByEmail(email);

if (user && user.password === password) {

const { password, ...result } = user;

return result;

}

return null;

}

async login(user: User) {

const payload = { email: user.email, sub: user.\_id };

return {

access\_token: this.jwtService.sign(payload),

};

}

}

Auth Controller:

import { Controller, Post, Body } from '@nestjs/common';

import { AuthService } from './auth.service';

import { User } from './user.schema';

@Controller('auth')

export class AuthController {

constructor(private readonly authService: AuthService) {}

@Post('login')

async login(@Body() user: User) {

return this.authService.login(user);

}

}