Perfect — here’s a tight, copy-pasteable kit for Manus:

Design tokens + Tailwind config

Global CSS (fonts, variables, components)

Hero (Apple-style) React component (with Framer Motion)

MiniGlobe React component (realistic, bright, gold ravens, Three.js)

How to wire it up + deps to install

Everything is full-file content (delete/replace) and path-scoped.

---

0) Install deps (once)

# from your repo root

npm i framer-motion three

# If Tailwind isn’t already set up:

npm i -D tailwindcss postcss autoprefixer

npx tailwindcss init -p

---

1) tailwind.config.js (replace file)

/\*\* @type {import('tailwindcss').Config} \*/

export default {

content: [

"./index.html",

"./src/\*\*/\*.{js,jsx,ts,tsx}",

],

theme: {

extend: {

fontFamily: {

inter: ['Inter', 'ui-sans-serif', 'system-ui', 'sans-serif'],

spectral: ['Spectral', 'serif'],

},

colors: {

bg: '#0b1220',

panel: '#0f1828',

edge: '#1e2a42',

text: '#e8f0ff',

muted: '#a8b8d6',

gold: '#f6c650',

teal: '#39d7c9',

},

boxShadow: {

glow: '0 0 40px rgba(57,215,201,.25)',

},

maxWidth: {

wrap: '1200px',

},

},

},

plugins: [],

};

---

2) src/styles/globals.css (create/replace)

@tailwind base;

@tailwind components;

@tailwind utilities;

/\* Google Fonts (simple + portable) \*/

@import url('https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&family=Spectral:wght@400;500&display=swap');

/\* Root tokens (used by non-tailwind bits too) \*/

:root{

--bg:#0b1220;

--panel:#0f1828;

--edge:#1e2a42;

--text:#e8f0ff;

--muted:#a8b8d6;

--gold:#f6c650;

--teal:#39d7c9;

}

/\* Base \*/

html,body{ height:100%; background:var(--bg); color:var(--text); }

body{ font-family: Inter, ui-sans-serif, system-ui, -apple-system, "Segoe UI", Roboto, Helvetica, Arial; }

/\* Utility helpers \*/

.container-wrap{ max-width:1200px; margin:0 auto; padding:0 1.25rem; }

/\* Buttons \*/

@layer components {

.btn {

@apply inline-flex items-center justify-center rounded-xl px-4 py-2 text-[15px] border transition;

border-color: var(--edge);

background: color-mix(in srgb, var(--panel) 90%, transparent);

}

.btn:hover { box-shadow: inset 0 0 0 2px rgba(255,255,255,.06); }

.btn-gold {

@apply text-black border-0;

background: linear-gradient(#f6c650,#e5b23a);

box-shadow: 0 6px 24px rgba(246,198,80,.25);

}

.btn-ghost {

@apply text-[color:var(--text)];

background: transparent;

border-color: var(--edge);

color: #bcd0f7;

}

}

/\* Hero background helpers \*/

.hero-bg {

position: absolute; inset: 0; overflow: hidden;

}

.hero-dna {

position:absolute; inset:0;

background: url('/images/dna\_bg.png') center/cover no-repeat;

opacity: .08;

filter: saturate(.9) brightness(1.1);

}

.hero-vignette {

position:absolute; inset:0;

background: radial-gradient(60% 60% at 50% 40%, transparent 0%, rgba(0,0,0,.55) 60%, rgba(0,0,0,.75) 100%);

}

/\* Poem panel refinement \*/

.poem-backdrop {

position:absolute; inset:0;

background:url('/images/reach\_stars.png') center/cover no-repeat;

opacity:.07;

}

---

3) src/components/Hero.jsx (create/replace)

import React from "react";

import { motion, useScroll, useTransform } from "framer-motion";

import MiniGlobe from "./MiniGlobe";

export default function Hero() {

const { scrollY } = useScroll();

const parallax = useTransform(scrollY, [0, 300], [0, -6]); // tiny drift

return (

<section className="relative min-h-[88vh] overflow-hidden border-b border-white/5">

{/\* BG layers \*/}

<div className="hero-bg">

<motion.div

className="hero-dna"

style={{ y: parallax }}

aria-hidden

/>

<div className="hero-vignette" aria-hidden />

</div>

{/\* Content \*/}

<div className="container-wrap relative z-10 grid md:grid-cols-[1.1fr\_0.9fr] gap-8 items-center py-24">

<div>

<motion.h1

className="text-[clamp(40px,6vw,80px)] font-semibold tracking-tight"

initial={{ opacity:0, y:10 }} animate={{ opacity:1, y:0 }} transition={{ duration:.6 }}

>

AI + Human… <span className="text-gold">with soul.</span>

</motion.h1>

<motion.p

className="mt-3 text-muted max-w-[52ch]"

initial={{ opacity:0, y:10 }} animate={{ opacity:1, y:0 }} transition={{ duration:.6, delay:.08 }}

>

Sovereign AI agents that respect your data, privacy, and humanity.

</motion.p>

<motion.div

className="mt-6 flex gap-3"

initial={{ opacity:0, y:10 }} animate={{ opacity:1, y:0 }} transition={{ duration:.6, delay:.16 }}

>

<a href="#chat" className="btn btn-gold">Start a Demo</a>

<a href="/globe" className="btn btn-ghost">Explore the Globe</a>

</motion.div>

</div>

{/\* Mini globe teaser \*/}

<div className="relative">

<div className="rounded-2xl border border-edge/60 bg-panel/60 p-3">

<MiniGlobe height={320} />

</div>

<div className="mt-2 text-xs text-[#8fb3ff]">

Real-time sovereignty visualizer · ravens in orbit

</div>

</div>

</div>

</section>

);

}

---

4) src/components/MiniGlobe.jsx (create/replace)

What it does

Bright, realistic Earth (no muddy dark look)

Atmosphere rim light

Slow autorotation

Two gold raven sprites on distinct orbits (use your real images)

Zero heavy rings/placeholder orbits

> Assets used:

/assets/earth\_daymap.jpg, /assets/earth\_nightmap.jpg, /assets/earth\_clouds.jpg

/images/raven\_huginn.png, /images/raven\_muninn.png

import React, { useEffect, useRef } from "react";

import \* as THREE from "three";

export default function MiniGlobe({ height = 300 }) {

const mountRef = useRef(null);

useEffect(() => {

const mount = mountRef.current;

if (!mount) return;

// --- renderer

const renderer = new THREE.WebGLRenderer({ antialias: true, alpha: true });

renderer.setPixelRatio(Math.min(window.devicePixelRatio, 2));

renderer.setSize(mount.clientWidth, height);

mount.appendChild(renderer.domElement);

// --- scene + camera

const scene = new THREE.Scene();

const camera = new THREE.PerspectiveCamera(28, mount.clientWidth / height, 0.1, 100);

camera.position.set(0, 0, 7.2);

// --- lights (brighter than default so it reads on dark bg)

const key = new THREE.DirectionalLight(0xffffff, 1.35);

key.position.set(5, 3, 5);

scene.add(key);

const fill = new THREE.AmbientLight(0x88aaff, 0.5);

scene.add(fill);

const rim = new THREE.DirectionalLight(0x88d0ff, 0.8);

rim.position.set(-6, -2, -4);

scene.add(rim);

// --- textures

const loader = new THREE.TextureLoader();

const texDay = loader.load("/assets/earth\_daymap.jpg");

const texNight = loader.load("/assets/earth\_nightmap.jpg");

const texClouds = loader.load("/assets/earth\_clouds.jpg");

texDay.colorSpace = THREE.SRGBColorSpace;

texNight.colorSpace = THREE.SRGBColorSpace;

texClouds.colorSpace = THREE.SRGBColorSpace;

texClouds.wrapS = texClouds.wrapT = THREE.RepeatWrapping;

texClouds.magFilter = THREE.LinearFilter;

// --- earth (day-night blend)

const globeGroup = new THREE.Group();

scene.add(globeGroup);

const geo = new THREE.SphereGeometry(2, 64, 64);

const mat = new THREE.MeshPhongMaterial({

map: texDay,

emissiveMap: texNight,

emissive: new THREE.Color(0xffffff),

emissiveIntensity: 0.35, // night glow

shininess: 5,

specular: new THREE.Color(0x111111),

});

const earth = new THREE.Mesh(geo, mat);

globeGroup.add(earth);

// --- clouds (subtle)

const cloudGeo = new THREE.SphereGeometry(2.02, 64, 64);

const cloudMat = new THREE.MeshLambertMaterial({

map: texClouds,

transparent: true,

opacity: 0.18,

depthWrite: false,

});

const clouds = new THREE.Mesh(cloudGeo, cloudMat);

globeGroup.add(clouds);

// --- atmosphere rim (faint)

const atmoGeo = new THREE.SphereGeometry(2.1, 64, 64);

const atmoMat = new THREE.ShaderMaterial({

uniforms: {},

vertexShader: `

varying vec3 vNormal;

void main() {

vNormal = normalize(normalMatrix \* normal);

gl\_Position = projectionMatrix \* modelViewMatrix \* vec4(position,1.0);

}

`,

fragmentShader: `

varying vec3 vNormal;

void main() {

float intensity = pow(0.6 - dot(vNormal, vec3(0.0, 0.0, 1.0)), 4.0);

gl\_FragColor = vec4(0.5, 0.75, 1.0, 1.0) \* intensity;

}

`,

blending: THREE.AdditiveBlending,

side: THREE.BackSide,

transparent: true,

});

const atmo = new THREE.Mesh(atmoGeo, atmoMat);

globeGroup.add(atmo);

// --- raven sprites (real gold assets)

const ravenTex1 = loader.load("/images/raven\_huginn.png");

const ravenTex2 = loader.load("/images/raven\_muninn.png");

ravenTex1.colorSpace = ravenTex2.colorSpace = THREE.SRGBColorSpace;

function makeRaven(tex, size = 0.35) {

const mat = new THREE.SpriteMaterial({ map: tex, transparent: true, depthWrite: false });

const sprite = new THREE.Sprite(mat);

sprite.scale.set(size, size, 1);

return sprite;

}

const raven1 = makeRaven(ravenTex1, 0.38);

const raven2 = makeRaven(ravenTex2, 0.34);

scene.add(raven1, raven2);

// orbit params (different planes & speeds)

const R1 = 3.1, R2 = 2.9;

const inc1 = 22 \* Math.PI/180, inc2 = -38 \* Math.PI/180;

let t = 0;

// --- resize

const onResize = () => {

const w = mount.clientWidth;

renderer.setSize(w, height);

camera.aspect = w / height;

camera.updateProjectionMatrix();

};

const ro = new ResizeObserver(onResize);

ro.observe(mount);

// --- animate

let raf = 0;

const animate = () => {

t += 0.008; // speed

earth.rotation.y += 0.0016;

clouds.rotation.y += 0.0019;

// Raven 1 (inclined orbit)

{

const a = t \* 0.85;

const x = R1 \* Math.cos(a);

const y = R1 \* Math.sin(a) \* Math.sin(inc1);

const z = R1 \* Math.sin(a) \* Math.cos(inc1);

raven1.position.set(x, y, z);

raven1.lookAt(camera.position);

}

// Raven 2 (different plane & speed)

{

const a = t \* 1.15 + Math.PI/3;

const x = R2 \* Math.cos(a);

const y = R2 \* Math.sin(a) \* Math.sin(inc2);

const z = R2 \* Math.sin(a) \* Math.cos(inc2);

raven2.position.set(x, y, z);

raven2.lookAt(camera.position);

}

renderer.render(scene, camera);

raf = requestAnimationFrame(animate);

};

animate();

// --- cleanup

return () => {

cancelAnimationFrame(raf);

ro.disconnect();

mount.removeChild(renderer.domElement);

renderer.dispose();

[scene, texDay, texNight, texClouds, ravenTex1, ravenTex2].forEach(() => {});

};

}, [height]);

return <div ref={mountRef} style={{ width: '100%', height }} />;

}

---

5) Quick wire-up (example)

Ensure you import the CSS and use Hero somewhere on the home page.

src/App.jsx (example skeleton)

import React from "react";

import "./styles/globals.css";

import Hero from "./components/Hero";

export default function App() {

return (

<>

<Hero />

{/\* ...rest of your sections... \*/}

<section id="chat" className="container-wrap py-24">

{/\* your chat preview / demo panel \*/}

<h2 className="text-[clamp(24px,3vw,40px)] mb-4">Ask Our AI</h2>

{/\* ... \*/}

</section>

</>

);

}

src/main.jsx (Vite React default)

import React from 'react'

import { createRoot } from 'react-dom/client'

import App from './App.jsx'

createRoot(document.getElementById('root')).render(

<React.StrictMode>

<App />

</React.StrictMode>,

)

index.html should already have <div id="root"></div>.

---

Asset checklist (must exist)

/public/images/dna\_bg.png

/public/images/reach\_stars.png

/public/images/raven\_huginn.png

/public/images/raven\_muninn.png

/public/assets/earth\_daymap.jpg

/public/assets/earth\_nightmap.jpg

/public/assets/earth\_clouds.jpg

---

What Manus should verify after merging

Hero loads fast, typography matches spec, CTAs behave correctly.

Mini-globe is brighter and visibly realistic (you should see terminator rim + faint clouds), no placeholder orbits.

Both gold ravens animate in different orbits with subtle trails (the sprite solution keeps them crisp).

Lighthouse ≥ 90; reduced-motion disables parallax/animation.

No regressions on mobile.

If you also want the Poem panel and Features grid components prebuilt, say the word and I’ll drop those too.