```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>A Special Greeting For You</title>
    <!-- Tailwind CSS -->
    <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css"
rel="stylesheet">
    <!-- Google Fonts -->
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Playfair+Display:wght@700&display=swap"
rel="stylesheet">
    <!-- Satoshi Font (Self-hosted or alternative if not available via CDN) -->
    <style>
        @font-face {
            font-family: 'Satoshi';
            src: url('https://fonts.cdnfonts.com/s/43388/Satoshi-Regular.woff') format('woff');
            font-weight: 400;
            font-style: normal;
        }
        @font-face {
            font-family: 'Satoshi';
            src: url('https://fonts.cdnfonts.com/s/43388/Satoshi-Bold.woff') format('woff');
            font-weight: 700;
            font-style: normal;
        }
    </style>
    <!-- Three.js -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/0.158.0/three.min.js"></script>
    <!-- GSAP -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/gsap/3.9.1/gsap.min.js"></script>
    <!-- Canvas Confetti -->
    <script
src="https://cdn.jsdelivr.net/npm/canvas-confetti@1.9.2/dist/confetti.browser.min.js"></script
>
    <style>
        :root {
            --dark-bg: #0c0a09;
            --pink-light: hsla(333, 70%, 55%, .4);
            --purple-light: hsla(282, 82%, 54%, .3);
            --blue-light: hsla(210, 89%, 60%, .3);
            --gold-light: hsla(35, 90%, 60%, .3);
            --card-bg: rgba(20, 18, 17, 0.7); /* Darker semi-transparent */
            --border-color: rgba(255, 255, 255, 0.1); /* Subtle white border */
            --text-light: #e0e0e0;
```

```
        --button-gradient: linear-gradient(45deg, #FF6B8B, #FF4757);
        --progress-gradient: linear-gradient(90deg, #FF6B8B, #FF4757);
    }

    body {
        font-family: 'Satoshi', sans-serif;
        color: var(--text-light);
        background-color: var(--dark-bg);
        overflow: hidden; /* Prevent scroll for full-screen effects */
        display: flex;
        justify-content: center;
        align-items: center;
        min-height: 100vh;
        position: relative;
    }

    h1, h2 {
        font-family: 'Playfair Display', serif;
        background: linear-gradient(45deg, #f0f0f0, #ffc0cb, #f0f0f0);
        -webkit-background-clip: text;
        -webkit-text-fill-color: transparent;
    }

    /* Aurora background animation */
    .aurora-background {
        position: absolute;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        background: radial-gradient(circle at 20% 30%, var(--pink-light) 0%, transparent
25%),
                    radial-gradient(circle at 80% 70%, var(--purple-light) 0%, transparent 25%),
                    radial-gradient(circle at 10% 80%, var(--blue-light) 0%, transparent 25%),
                    radial-gradient(circle at 90% 20%, var(--gold-light) 0%, transparent 25%);
        background-size: 200% 200%;
        animation: aurora-flow 25s ease-in-out infinite alternate;
        z-index: -2;
        filter: blur(80px); /* Soften the gradient edges */
        opacity: 0.8;
    }

    @keyframes aurora-flow {
        0% { background-position: 0% 0%; }
        100% { background-position: 100% 100%; }
    }

    /* Three.js canvas for hearts */
```

```
#three-container {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    z-index: -1;
    pointer-events: none; /* Allow interaction with elements below */
}

/* Glassmorphism card style */
.glass-card {
    background-color: var(--card-bg);
    backdrop-filter: blur(20px);
    -webkit-backdrop-filter: blur(20px);
    border: 1px solid var(--border-color);
    border-radius: 1.5rem; /* More rounded */
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.4); /* Softer, larger shadow */
    padding: 2.5rem;
    max-width: 600px;
    text-align: center;
    position: relative;
    opacity: 0; /* Hidden by default, animated in by GSAP */
    transform: translateY(20px) scale(0.98); /* Initial state for animation */
}

/* Button styles */
.btn-primary {
    background: var(--button-gradient);
    color: white;
    padding: 0.8rem 2.5rem;
    border-radius: 9999px; /* Pill shape */
    font-weight: 700;
    font-size: 1.1rem;
    transition: all 0.3s cubic-bezier(0.2, 0.8, 0.2, 1);
    position: relative;
    overflow: hidden;
    display: inline-flex;
    align-items: center;
    justify-content: center;
    box-shadow: 0 0 15px rgba(255, 107, 139, 0.6); /* Soft glow */
}

.btn-primary:hover {
    transform: translateY(-3px) scale(1.02);
    box-shadow: 0 0 25px rgba(255, 107, 139, 0.9); /* Intense glow */
}
```

```
        .btn-primary:active {
            transform: translateY(1px) scale(0.98);
            box-shadow: 0 0 10px rgba(255, 107, 139, 0.5);
        }

        /* Progress Bar */
        .progress-container {
            position: fixed;
            top: 0.5rem;
            left: 50%;
            transform: translateX(-50%);
            width: 80%;
            max-width: 500px;
            height: 8px;
            background-color: rgba(255, 255, 255, 0.08); /* Glass-like track */
            border-radius: 4px;
            z-index: 100;
            backdrop-filter: blur(10px);
            -webkit-backdrop-filter: blur(10px);
            border: 1px solid rgba(255, 255, 255, 0.05);
            overflow: hidden;
        }

        .progress-bar {
            height: 100%;
            width: 0%;
            background: var(--progress-gradient);
            border-radius: 4px;
            transition: width 0.5s ease-out;
        }

        /* Icons */
        .emoji-icon {
            font-size: 4rem; /* Larger emoji */
            filter: drop-shadow(0 0 15px rgba(255, 255, 255, 0.5));
            animation: pulse-glow 2s infinite alternate;
        }

        @keyframes pulse-glow {
            0% { transform: scale(1); filter: drop-shadow(0 0 10px rgba(255, 255, 255, 0.4)); }
            100% { transform: scale(1.05); filter: drop-shadow(0 0 25px rgba(255, 255, 255,
0.8)); }
        }

        /* Bento Grid */
        .bento-grid {
            display: grid;
            grid-template-columns: repeat(2, 1fr);
```

```
    gap: 1rem;
    margin-top: 2rem;
}

.bento-item {
    background-color: rgba(255, 255, 255, 0.05);
    backdrop-filter: blur(10px);
    border-radius: 1rem;
    padding: 1.5rem;
    text-align: left;
    border: 1px solid var(--border-color);
}

.bento-item.span-2 {
    grid-column: span 2;
}

.bento-item h3 {
    font-family: 'Playfair Display', serif;
    font-size: 1.5rem;
    margin-bottom: 0.5rem;
    background: linear-gradient(45deg, #f0f0f0, #ffc0cb, #f0f0f0);
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
}

.bento-item p {
    font-size: 0.95rem;
    color: var(--text-light);
    opacity: 0.8;
}

/* Polaroid Effect */
.polaroid-container {
    position: relative;
    margin-top: 2rem;
    display: inline-block; /* To allow transform origin to be centered */
    transform-style: preserve-3d;
    perspective: 1000px; /* For subtle 3D tilt */
}

.polaroid {
    background-color: white;
    padding: 10px 10px 30px 10px; /* Thicker bottom for caption */
    border: 1px solid #ccc;
    box-shadow: 0 4px 15px rgba(0, 0, 0, 0.3);
    transform: rotateZ(-5deg) rotateY(5deg) scale(0.95); /* Initial slight tilt */
    transition: transform 0.2s ease-out;
```

```
            will-change: transform;
        }

        .polaroid img {
            display: block;
            width: 280px; /* Fixed size for polaroid image */
            height: 280px;
            object-fit: cover;
            filter: brightness(1.05) contrast(1.05); /* Slight enhancement */
        }

        .polaroid-caption {
            color: #333;
            font-family: 'Satoshi', sans-serif;
            font-size: 0.9rem;
            text-align: center;
            margin-top: 10px;
        }

        /* Responsive adjustments */
        @media (max-width: 768px) {
            .glass-card {
                padding: 1.5rem;
                margin: 0 1rem;
            }
            .emoji-icon {
                font-size: 3rem;
            }
            .btn-primary {
                padding: 0.6rem 2rem;
                font-size: 1rem;
            }
            .bento-grid {
                grid-template-columns: 1fr; /* Stack columns on small screens */
            }
            .bento-item.span-2 {
                grid-column: span 1;
            }
            .polaroid img {
                width: 200px;
                height: 200px;
            }
        }
    </style>
</head>
<body class="relative">

    <!-- Aurora Background Layer -->
```

```
    <div class="aurora-background"></div>

    <!-- Three.js Hearts Layer -->
    <div id="three-container"></div>

    <!-- Progress Bar -->
    <div class="progress-container">
        <div class="progress-bar" id="progressBar"></div>
    </div>

    <!-- Main Content Area -->
    <main class="relative z-10 p-4">
        <!-- Step 1: Welcome -->
        <section id="step-1" class="glass-card flex flex-col items-center gap-6">
            <span class="emoji-icon">🥺</span>
            <h1 class="text-5xl font-bold">Hey Beautiful,</h1>
            <p class="text-xl text-center leading-relaxed">I built a little world for you, just to bring
a smile to your face on your special day.</p>
            <button class="btn-primary" onclick="nextStep()">Let's Begin</button>
        </section>

        <!-- Step 2: Core Message -->
        <section id="step-2" class="glass-card flex flex-col items-center gap-6 hidden">
            <span class="emoji-icon">😭</span>
            <h2 class="text-4xl font-bold">sorry for i hurt you!</h2>
            <p class="text-xl text-center leading-relaxed">Another year of you making the world
brighter. Your existence is a gift, and I'm so lucky to witness it.</p>
            <button class="btn-primary" onclick="nextStep()">There's more...</button>
        </section>

        <!-- Step 3: Reasons Why (Bento Grid) -->
        <section id="step-3" class="glass-card flex flex-col items-center gap-6 hidden">
            <h2 class="text-4xl font-bold mb-4">A Few Things I Adore About You</h2>
            <div class="bento-grid w-full">
                <div class="bento-item span-2">
                    <h3>✨ Your Unmatched Kindness</h3>
                    <p>The genuine warmth you show to everyone is something truly rare and
beautiful.</p>
                </div>
                <div class="bento-item">
                    <h3>😊 That Smile</h3>
                    <p>It's a work of art.</p>
                </div>
                <div class="bento-item span-2">
                    <h3>🌟 Your Radiant Spirit</h3>
                    <p>Your passion for life is infectious. Being around you makes everything feel
more exciting and possible.</p>
                </div>
```

```
            </div>
            <button class="btn-primary mt-4" onclick="nextStep()">Remember this?</button>
        </section>

        <!-- Step 4: Shared Memory -->
        <section id="step-4" class="glass-card flex flex-col items-center gap-6 hidden">
            <h2 class="text-4xl font-bold">That One Time...</h2>
            <div class="polaroid-container" id="polaroidContainer">
                <div class="polaroid">
                    <img src="https://i.ibb.co/6Z6XgCg/crush.webp" alt="Our favorite memory">
                    <div class="polaroid-caption">Our favorite memory.</div>
                </div>
            </div>
            <p class="text-xl text-center leading-relaxed">Every moment with you feels like a scene from a movie I'd watch on repeat.</p>
            <button class="btn-primary" onclick="nextStep()">One last thing...</button>
        </section>

        <!-- Step 5: Finale -->
        <section id="step-5" class="glass-card flex flex-col items-center gap-6 hidden">
            <span class="emoji-icon">🥺</span>
            <h2 class="text-4xl font-bold">My Wish For You</h2>
            <p class="text-xl text-center leading-relaxed">May the next year bring you all the love, success, and pure happiness you so rightfully deserve. May your dreams soar higher than ever.</p>
            <p id="final-message" class="text-3xl font-bold text-center mt-6 opacity-0 translate-y-4" style="background: linear-gradient(45deg, #ffc0cb, #f0f0f0, #ff6b8b); -webkit-background-clip: text; -webkit-text-fill-color: transparent;">Sorry my bestie ❤️</p>
            <button class="btn-primary mt-4" id="celebrate-btn" onclick="celebrate()">Celebrate!</button>
        </section>
    </main>

    <script>
        // Three.js for 3D hearts
        let scene, camera, renderer, hearts = [];
        const heartGeometry = new THREE.ShapeGeometry(createHeartShape());

        function createHeartShape() {
            const x = 0, y = 0;
            const heartShape = new THREE.Shape();
            heartShape.moveTo(x + 0.5, y + 0.5);
            heartShape.bezierCurveTo(x + 0.5, y + 0.5, x + 0.4, y, x, y);
            heartShape.bezierCurveTo(x - 0.6, y, x - 0.6, y + 0.7, x - 0.6, y + 0.7);
            heartShape.bezierCurveTo(x - 0.6, y + 1.1, x - 0.3, y + 1.54, x + 0.5, y + 1.9);
            heartShape.bezierCurveTo(x + 1.3, y + 1.54, x + 1.6, y + 1.1, x + 1.6, y + 0.7);
            heartShape.bezierCurveTo(x + 1.6, y + 0.7, x + 1.6, y, x + 1.0, y);
            heartShape.bezierCurveTo(x + 0.7, y, x + 0.5, y + 0.5, x + 0.5, y + 0.5);
```

```
        return heartShape;
    }

    function initThree() {
        const container = document.getElementById('three-container');
        scene = new THREE.Scene();
        camera = new THREE.PerspectiveCamera(75, window.innerWidth /
window.innerHeight, 0.1, 1000);
        camera.position.z = 50;

        renderer = new THREE.WebGLRenderer({ antialias: true, alpha: true });
        renderer.setSize(window.innerWidth, window.innerHeight);
        renderer.setPixelRatio(window.devicePixelRatio);
        container.appendChild(renderer.domElement);

        const pointLight = new THREE.PointLight(0xffffff, 1);
        pointLight.position.set(10, 20, 30);
        scene.add(pointLight);

        const ambientLight = new THREE.AmbientLight(0x404040); // Soft white light
        scene.add(ambientLight);

        for (let i = 0; i < 25; i++) {
            const material = new THREE.MeshPhongMaterial({
                color: new THREE.Color(`hsl(${Math.random() * 60 + 330}, 80%, 70%)`), // Pink
to red hues
                specular: 0xcccccc,
                shininess: 30,
                transparent: true,
                opacity: 0.8
            });
            const heart = new THREE.Mesh(heartGeometry, material);

            heart.position.x = Math.random() * 100 - 50;
            heart.position.y = Math.random() * 100 - 50;
            heart.position.z = Math.random() * 60 - 30;

            heart.rotation.x = Math.random() * Math.PI;
            heart.rotation.y = Math.random() * Math.PI;
            heart.rotation.z = Math.random() * Math.PI;

            heart.scale.setScalar(Math.random() * 0.8 + 0.4);

            hearts.push(heart);
            scene.add(heart);
        }

        window.addEventListener('resize', onWindowResize);
```

```
    animateHearts();
}

function onWindowResize() {
    camera.aspect = window.innerWidth / window.innerHeight;
    camera.updateProjectionMatrix();
    renderer.setSize(window.innerWidth, window.innerHeight);
}

function animateHearts() {
    requestAnimationFrame(animateHearts);

    const time = Date.now() * 0.0005;

    hearts.forEach((heart, i) => {
        heart.position.y += Math.sin(time + i) * 0.03;
        heart.position.x += Math.cos(time * 0.5 + i) * 0.02;
        heart.position.z += Math.sin(time * 0.7 + i) * 0.01;

        heart.rotation.x += 0.001;
        heart.rotation.y += 0.002;

        // Loop hearts if they go too far
        if (heart.position.y > 60) heart.position.y = -60;
        if (heart.position.x > 60) heart.position.x = -60;
        if (heart.position.z > 30) heart.position.z = -30;
        if (heart.position.y < -60) heart.position.y = 60;
        if (heart.position.x < -60) heart.position.x = 60;
        if (heart.position.z < -30) heart.position.z = 30;
    });

    renderer.render(scene, camera);
}

// GSAP for step transitions and animations
let currentStep = 1;
const totalSteps = 5;
const steps = [];

document.addEventListener('DOMContentLoaded', () => {
    initThree(); // Initialize Three.js hearts

    // Populate steps array
    for (let i = 1; i <= totalSteps; i++) {
        steps.push(document.getElementById(`step-${i}`));
    }

    // Animate in the first step
```

```
        gsap.to(steps[0], { opacity: 1, y: 0, scale: 1, duration: 1, ease: "power3.out", delay:
0.5 });
        updateProgressBar();

        // Polaroid hover effect
        const polaroidContainer = document.getElementById('polaroidContainer');
        if (polaroidContainer) {
            polaroidContainer.addEventListener('mousemove', (e) => {
                const rect = polaroidContainer.getBoundingClientRect();
                const centerX = rect.left + rect.width / 2;
                const centerY = rect.top + rect.height / 2;
                const mouseX = e.clientX - centerX;
                const mouseY = e.clientY - centerY;

                const rotateY = -mouseX * 0.03; // Adjust sensitivity
                const rotateX = mouseY * 0.03; // Adjust sensitivity

                gsap.to(polaroidContainer.querySelector('.polaroid'), {
                    rotationY: rotateY,
                    rotationX: rotateX,
                    x: mouseX * 0.05,
                    y: mouseY * 0.05,
                    ease: "power1.out",
                    duration: 0.3
                });
            });

            polaroidContainer.addEventListener('mouseleave', () => {
                gsap.to(polaroidContainer.querySelector('.polaroid'), {
                    rotationY: 5, // Reset to initial tilt
                    rotationX: -5, // Reset to initial tilt
                    x: 0,
                    y: 0,
                    ease: "elastic.out(1, 0.5)",
                    duration: 1
                });
            });
        }
    });

    function nextStep() {
        if (currentStep < totalSteps) {
            gsap.to(steps[currentStep - 1], {
                opacity: 0,
                y: -20,
                scale: 0.95,
                duration: 0.6,
                ease: "power2.in",
```

```
                onComplete: () => {
                    steps[currentStep - 1].classList.add('hidden');
                    currentStep++;
                    steps[currentStep - 1].classList.remove('hidden');
                    gsap.fromTo(steps[currentStep - 1],
                        { opacity: 0, y: 20, scale: 0.98 },
                        { opacity: 1, y: 0, scale: 1, duration: 0.8, ease: "power3.out" }
                    );
                    updateProgressBar();
                }
            });
        }
    }

    function updateProgressBar() {
        const progressBar = document.getElementById('progressBar');
        const progress = (currentStep / totalSteps) * 100;
        progressBar.style.width = `${progress}%`;
    }

    function celebrate() {
        const celebrateBtn = document.getElementById('celebrate-btn');
        const finalMessage = document.getElementById('final-message');

        // Disable and fade out button
        gsap.to(celebrateBtn, { opacity: 0, y: 20, duration: 0.5, onComplete: () =>
celebrateBtn.disabled = true });

        // Fade in final message
        gsap.fromTo(finalMessage,
            { opacity: 0, y: 20 },
            { opacity: 1, y: 0, duration: 1, delay: 0.5, ease: "power3.out" }
        );

        // Confetti explosion (multi-layered)
        setTimeout(() => {
            // Initial powerful bursts
            confetti({
                particleCount: 200,
                spread: 120,
                origin: { x: 0.2, y: 1 },
                colors: ['#ffc0cb', '#ff6b8b', '#ff4757', '#ffffff', '#ffd700']
            });
            confetti({
                particleCount: 200,
                spread: 120,
                origin: { x: 0.8, y: 1 },
                colors: ['#ffc0cb', '#ff6b8b', '#ff4757', '#ffffff', '#ffd700']
```

```
    });

    // Continuous gentle shower with emojis
    const duration = 5 * 1000;
    const animationEnd = Date.now() + duration;
    const defaults = { startVelocity: 30, spread: 360, ticks: 60, zIndex: 0 };

    function randomInRange(min, max) {
      return Math.random() * (max - min) + min;
    }

    const interval = setInterval(function() {
      const timeLeft = animationEnd - Date.now();

      if (timeLeft <= 0) {
        return clearInterval(interval);
      }

      const particleCount = 50 * (timeLeft / duration);
      // since particles fall down, we can restrict the drag to a lower speed
      confetti(Object.assign({}, defaults, {
        particleCount,
        origin: { x: randomInRange(0.1, 0.9), y: Math.random() - 0.2 },
        colors: ['#ffc0cb', '#ff6b8b', '#ff4757', '#ffffff', '#ffd700'],
        shapes: ['circle', 'square']
      }));
       confetti(Object.assign({}, defaults, {
        particleCount,
        origin: { x: randomInRange(0.1, 0.9), y: Math.random() - 0.2 },
        colors: ['#ffc0cb', '#ff6b8b', '#ff4757', '#ffffff', '#ffd700'],
        shapes: ['❤️', '💖', '✨'] // Emoji confetti
      }));
    }, 250);

}, 1000); // Start confetti after message appears

// Hearts final animation
gsap.to(hearts.map(h => h.position), {
    x: (i) => hearts[i].position.x > 0 ? 100 : -100,
    y: (i) => 100 + Math.random() * 50,
    z: (i) => hearts[i].position.z > 0 ? 100 : -100,
    duration: 4,
    ease: "power2.in",
    stagger: 0.02,
    onUpdate: function() {
      // Fade out hearts during movement
      hearts.forEach(heart => {
        heart.material.opacity = Math.max(0, heart.material.opacity - 0.005);
```

```
                });
            },
            onComplete: () => {
                hearts.forEach(heart => scene.remove(heart)); // Remove hearts from scene
                hearts = []; // Clear array
            }
        });
         gsap.to(hearts.map(h => h.rotation), {
            x: (i) => hearts[i].rotation.x + Math.PI * 2,
            y: (i) => hearts[i].rotation.y + Math.PI * 2,
            z: (i) => hearts[i].rotation.z + Math.PI * 2,
            duration: 4,
            ease: "power2.in",
            stagger: 0.02,
        });
    }

  </script>
</body>
</html>
```