PLOS Component-Specific Integrations & UI Design Guide

This guide outlines the specific external integrations, data sources, UI elements, and animations for each PLOS component to create a cohesive, engaging, and thematic user experience.

1. 📊 Dashboard Page

External Integrations

- **Weather API**: OpenWeatherMap API for local weather conditions
- **Calendar Integration**: Google Calendar API for upcoming events
- **News API**: NewsAPI for personalized headlines based on interests

Analysis Features

- **Daily Aggregation**: Overall wellness score calculated from all modules
- **Trend Analysis**: Week-over-week comparison of key metrics
- **AI Daily Brief**: Personalized summary of patterns across modules

UI Elements & Animations

- **3D Welcome Element**: Animated 3D text with Framer Motion + Three.js
- **Particle Background**: Subtle animated particles that respond to user interaction
- **Micro-interactions**: Cards that expand on hover with spring animations
- **Theme**: Gradient color scheme that changes based on time of day
- **Icons**: Phosphor or Lucide icons with micro-animations on hover

tsx

```jsx
// Example animated welcome component
import { motion } from 'framer-motion';
import { useTheme } from 'next-themes';
import { useState, useEffect } from 'react';

export default function WelcomeBanner() {
  const { theme } = useTheme();
  const [greeting, setGreeting] = useState('');
  const [username, setUsername] = useState('');

  // Get time-based greeting
  useEffect(() => {
    const hour = new Date().getHours();
    const greetingText = hour < 12 ? 'Good Morning' :
                         hour < 18 ? 'Good Afternoon' :
                         'Good Evening';
    setGreeting(greetingText);

    // Get user name from store or API
    // ...
  }, []);

  return (
    <motion.div
      className={`p-6 rounded-2xl ${theme === 'dark' ? 'bg-gradient-to-r from-indigo-900 to-pur
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.5, type: 'spring' }}
    >
      <motion.h1
        className="text-3xl font-bold"
        initial={{ scale: 0.9 }}
        animate={{ scale: 1 }}
        transition={{ delay: 0.2, type: 'spring' }}
      >
        {greeting}, {username} 👋
      </motion.h1>
      <motion.p
        className="mt-2 text-lg opacity-80"
        initial={{ opacity: 0 }}
        animate={{ opacity: 0.8 }}
        transition={{ delay: 0.4 }}
      >
```

```
        Here's your life at a glance
      </motion.p>
    </motion.div>
  );
}
```

2. 🏋️ Physical Health Tracker Page

External Integrations

- **Apple Health/HealthKit**: Steps, heart rate, workouts, sleep data
- **Fitbit API**: Alternative for non-Apple users
- **Google Fit API**: Another alternative for Android users
- **Strava API**: For running/cycling enthusiasts
- **Oura Ring API**: For detailed sleep metrics

Analysis Features

- **Activity Pattern Recognition**: Identifies optimal workout times
- **Heart Rate Variability Analysis**: Stress and recovery metrics
- **Sleep Quality Assessment**: Deep/REM/light sleep analysis
- **Weekly Trend Comparison**: Visual comparison with previous weeks

Personalized Recommendations

- **Workout Suggestions**: Based on previous activity and recovery status
- **Rest Day Recommendations**: Based on overtraining signals
- **Sleep Optimization Tips**: Based on sleep patterns and routines

UI Elements & Animations

- **3D Heart Model**: Interactive heart that pulses with user's average HR
- **ECG Line Animation**: Animated ECG line that mimics recent heart patterns
- **Step Counter Animation**: Walking figure with particle effects
- **Sleep Cycles**: Animated waves representing sleep cycles
- **Theme**: Blues and reds (cardiovascular theme) with anatomical illustrations
- **Icons**: Anatomically-correct heart, lungs, muscles that animate on interaction

tsx

```
// Example 3D Heart Rate Animation
import { motion, useAnimation } from 'framer-motion';
import { useEffect, useState } from 'react';
import { Heart } from 'lucide-react';

export default function HeartRateVisual({ heartRate = 70 }) {
  const controls = useAnimation();
  const [isAnimating, setIsAnimating] = useState(true);

  // Calculate animation duration based on heart rate
  const duration = 60 / heartRate;

  useEffect(() => {
    if (isAnimating) {
      controls.start({
        scale: [1, 1.2, 1],
        transition: {
          duration,
          repeat: Infinity,
          repeatType: "loop"
        }
      });
    } else {
      controls.stop();
    }
  }, [isAnimating, heartRate, controls, duration]);

  return (
    <div className="relative flex items-center justify-center h-40 w-full">
      <div className="absolute inset-0 flex items-center justify-center">
        <svg viewBox="0 0 100 100" className="w-full h-full max-w-xs">
          <motion.path
            d="M10,50 L30,50 L40,30 L50,70 L60,30 L70,70 L80,50 L90,50"
            fill="transparent"
            stroke="red"
            strokeWidth="2"
            initial={{ pathLength: 0, opacity: 0 }}
            animate={{
              pathLength: 1,
              opacity: 1,
              transition: { duration: 1.5, repeat: Infinity, repeatType: "loop" }
            }}
          />
```

```
          </svg>
        </div>

        <motion.div
          animate={controls}
          className="relative z-10"
        >
          <Heart size={64} color="red" fill="red" />
        </motion.div>

        <div className="absolute bottom-2 right-2 text-2xl font-bold text-red-600">
          {heartRate} BPM
        </div>
      </div>
    );
  }
```

3. 🧠 Mental Health Companion Page

External Integrations

- **Spotify API**: Integration for music therapy and mood playlists
- **Calm/Headspace API**: For guided meditation sessions
- **Weather API**: To correlate mood with weather patterns
- **Journal Entries**: Internal integration with journal module

Analysis Features

- **Mood Pattern Recognition**: Identifying triggers and cycles
- **Emotional Language Analysis**: NLP analysis of journal entries
- **Sentiment Trend Visualization**: Mood over time with life events correlation

Personalized Recommendations

- **Custom Meditation Suggestions**: Based on current mood and historical effectiveness
- **CBT Technique Prompts**: Cognitive behavioral therapy techniques
- **Mood-boosting Activities**: Personalized based on past effectiveness

UI Elements & Animations

- **Breathing Animation Circle**: Expands and contracts to guide breathing
- **3D Brain Visualization**: With active regions based on mood/activities

- **Mood Color Transitions**: Background subtly transitions based on mood selection
- **Theme**: Calming blues and purples with neural network patterns
- **Icons**: Abstract emotional representations that morph based on selection

tsx

```jsx
// Example Breathing Animation Component
import { motion } from 'framer-motion';
import { useState } from 'react';

export default function BreathingExercise() {
  const [isBreathing, setIsBreathing] = useState(false);
  const [breathCount, setBreathCount] = useState(0);

  const startBreathing = () => {
    setIsBreathing(true);
    setBreathCount(0);
  };

  const breathingComplete = () => {
    if (breathCount < 7) {
      setBreathCount(prev => prev + 1);
    } else {
      setIsBreathing(false);
      // Trigger completion callback or state change
    }
  };

  return (
    <div className="flex flex-col items-center justify-center p-6 bg-blue-50 dark:bg-blue-900 r
      <h3 className="text-xl font-medium mb-4">Breathing Exercise</h3>

      <div className="relative h-64 w-64 flex items-center justify-center">
        <motion.div
          className="absolute h-40 w-40 bg-blue-400 dark:bg-blue-600 rounded-full opacity-20"
          animate={isBreathing ? {
            scale: [1, 2, 2, 1, 1],
            opacity: [0.2, 0.4, 0.4, 0.2, 0.2]
          } : { scale: 1, opacity: 0.2 }}
          transition={isBreathing ? {
            duration: 8,
            repeat: Infinity,
            repeatType: "loop",
            times: [0, 0.25, 0.5, 0.75, 1],
            onComplete: breathingComplete
          } : {}}
        />

        <motion.div
```

```
      className="absolute h-32 w-32 bg-blue-500 dark:bg-blue-700 rounded-full opacity-40"
      animate={isBreathing ? {
        scale: [1, 1.7, 1.7, 1, 1],
        opacity: [0.4, 0.6, 0.6, 0.4, 0.4]
      } : { scale: 1, opacity: 0.4 }}
      transition={isBreathing ? {
        duration: 8,
        repeat: Infinity,
        repeatType: "loop",
        times: [0, 0.25, 0.5, 0.75, 1]
      } : {}}
    />

    <motion.div
      className="h-24 w-24 bg-blue-600 dark:bg-blue-800 rounded-full flex items-center just
      animate={isBreathing ? {
        scale: [1, 1.4, 1.4, 1, 1]
      } : { scale: 1 }}
      transition={isBreathing ? {
        duration: 8,
        repeat: Infinity,
        repeatType: "loop",
        times: [0, 0.25, 0.5, 0.75, 1]
      } : {}}
    >
      {isBreathing ? (
        <div className="text-center">
          <div className="text-lg">{breathCount + 1}/8</div>
          <div className="text-sm">
            {breathCount % 2 === 0 ? "Inhale..." : "Exhale..."}
          </div>
        </div>
      ) : (
        <button
          onClick={startBreathing}
          className="p-2 rounded-full"
        >
          Start
        </button>
      )}
    </motion.div>
  </div>

  {isBreathing && (
```

```
          <p className="mt-4 text-center text-blue-800 dark:text-blue-200">
            Focus on your breathing...
          </p>
        )}
      </div>
    );
  }
```

4. 🍽 Nutrition & Diet Page

External Integrations

- **MyFitnessPal API**: Food database and nutritional information
- **Cronometer API**: Detailed nutritional breakdown
- **Grocery Delivery APIs**: Instacart, Amazon Fresh for food ordering
- **Recipe APIs**: Edamam, Spoonacular for recipe suggestions
- **Food Recognition API**: Google Vision API for food image recognition

Analysis Features

- **Macronutrient Balance Analysis**: Optimal protein/carb/fat ratios
- **Micronutrient Gap Detection**: Identify missing vitamins/minerals
- **Eating Pattern Recognition**: Meal timing and frequency analysis
- **Caloric Balance Visualization**: Intake vs. expenditure

Personalized Recommendations

- **Meal Suggestions**: Based on nutritional gaps and preferences
- **Grocery Lists**: Generated from recommended meals and recipes
- **Dietary Adjustments**: Based on health goals and activity levels

UI Elements & Animations

- **3D Food Models**: Interactive 3D models of food items
- **Nutrient Flow Animation**: Animated visualization of nutrient absorption
- **Plate Builder Animation**: Interactive plate with drag-and-drop food items
- **Theme**: Vibrant food colors with organic patterns and textures
- **Icons**: Detailed food illustrations that animate when interacted with

tsx

```jsx
// Example Macronutrient Distribution Chart
import { motion } from 'framer-motion';
import { Pie } from 'recharts';
import { PieChart, Cell, ResponsiveContainer, Legend, Tooltip } from 'recharts';

export default function MacronutrientChart({ data }) {
  // Data should be in format:
  // [{name: 'Protein', value: 30, color: '#FF5733'}, {name: 'Carbs', value: 50, color: '#33FF5

  return (
    <motion.div
      className="bg-white dark:bg-gray-800 rounded-xl p-4 shadow-sm"
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.5 }}
    >
      <h3 className="text-lg font-medium mb-2">Today's Macronutrients</h3>

      <div className="h-64">
        <ResponsiveContainer width="100%" height="100%">
          <PieChart>
            <Pie
              data={data}
              cx="50%"
              cy="50%"
              innerRadius={60}
              outerRadius={80}
              paddingAngle={5}
              dataKey="value"
              animationDuration={1000}
              animationBegin={200}
            >
              {data.map((entry, index) => (
                <Cell key={`cell-${index}`} fill={entry.color} />
              ))}
            </Pie>
            <Tooltip />
            <Legend verticalAlign="bottom" height={36} />
          </PieChart>
        </ResponsiveContainer>
      </div>

      <div className="mt-4 grid grid-cols-3 gap-2">
```

```
          {data.map((macro, index) => (
            <motion.div
              key={`macro-${index}`}
              className="text-center"
              initial={{ scale: 0.9, opacity: 0 }}
              animate={{ scale: 1, opacity: 1 }}
              transition={{ delay: index * 0.1 }}
            >
              <div className="text-sm text-gray-500 dark:text-gray-400">{macro.name}</div>
              <div className="text-xl font-bold" style={{ color: macro.color }}>{macro.value}g</d
            </motion.div>
          ))}
        </div>
      </motion.div>
    );
  }
```

5. 👨‍👩‍👧 Family & Social Life Page

External Integrations

- **Google Calendar API**: Event scheduling and coordination
- **WhatsApp/Telegram API**: Family chat integration
- **Facebook/Instagram API**: Social event detection
- **Location APIs**: Family member location sharing
- **Shared Task Apps**: Todoist/Trello API for family task management

Analysis Features

- **Relationship Time Analysis**: Quality time spent with different people
- **Social Network Visualization**: Interactive social connection graph
- **Event Participation Patterns**: Identifying social engagement trends
- **Communication Frequency Analysis**: Contact patterns with important people

Personalized Recommendations

- **Social Connection Suggestions**: Who to reach out to based on contact history
- **Family Activity Ideas**: Based on past enjoyment and current weather/availability
- **Relationship Nurturing Tips**: Personalized for each relationship

UI Elements & Animations

- **Interactive Family Tree**: 3D family/friend network visualization
- **Communication Flow Animation**: Animated lines showing message exchanges
- **Memory Timeline**: Scrollable timeline with family photos and events
- **Theme**: Warm, friendly colors with connection-themed patterns
- **Icons**: Family/relationship icons that pulse or animate on interaction

tsx

```jsx
// Example Relationship Map Component
import { useState, useEffect } from 'react';
import { motion } from 'framer-motion';
import { User, Users, Heart, MessageCircle, Calendar } from 'lucide-react';

export default function RelationshipMap({ relationships }) {
  const [activeRelationship, setActiveRelationship] = useState(null);

  // Calculate positions in a circle for each relationship
  const getPosition = (index, total) => {
    const radius = 120;
    const angle = (index / total) * Math.PI * 2 - Math.PI / 2;
    return {
      x: radius * Math.cos(angle),
      y: radius * Math.sin(angle)
    };
  };

  return (
    <div className="relative h-96 w-full bg-gradient-to-br from-orange-50 to-amber-50 dark:from
      <h3 className="text-lg font-medium mb-6 text-center">Your Relationship Map</h3>

      {/* Center user */}
      <motion.div
        className="absolute left-1/2 top-1/2 -ml-10 -mt-10 h-20 w-20 rounded-full bg-blue-500 f
        initial={{ scale: 0 }}
        animate={{ scale: 1 }}
        transition={{ type: 'spring', duration: 0.8 }}
      >
        <User size={32} />
        <div className="mt-1 text-xs">You</div>
      </motion.div>

      {/* Relationship circles */}
      {relationships.map((relation, index) => {
        const pos = getPosition(index, relationships.length);

        return (
          <motion.div
            key={relation.id}
            className={`absolute h-16 w-16 rounded-full flex items-center justify-center text-w
            style={{ backgroundColor: relation.color }}
            initial={{ scale: 0, x: 0, y: 0 }}
```

```
        animate={{
          scale: 1,
          x: pos.x,
          y: pos.y,
          transition: { delay: 0.1 * index, type: 'spring' }
        }}
        onClick={() => setActiveRelationship(relation.id)}
        whileHover={{ scale: 1.1 }}
      >
        {relation.type === 'family' ? (
          <Heart size={24} fill="white" />
        ) : relation.type === 'friend' ? (
          <Users size={24} />
        ) : (
          <User size={24} />
        )}
        <div className="absolute -bottom-6 whitespace-nowrap text-xs font-medium text-gray-
          {relation.name}
        </div>
      </motion.div>
    );
  })}

  {/* Connection lines */}
  <svg className="absolute inset-0 w-full h-full pointer-events-none">
    {relationships.map((relation, index) => {
      const pos = getPosition(index, relationships.length);
      return (
        <motion.line
          key={`line-${relation.id}`}
          x1="50%"
          y1="50%"
          x2={`calc(50% + ${pos.x}px)`}
          y2={`calc(50% + ${pos.y}px)`}
          stroke={relation.color}
          strokeWidth={relation.strength * 2}
          strokeOpacity={0.5}
          strokeDasharray={relation.type === 'family' ? "0" : "5,5"}
          initial={{ pathLength: 0, opacity: 0 }}
          animate={{
            pathLength: 1,
            opacity: 0.6,
            transition: { delay: 0.1 * index, duration: 0.6 }
          }}
```

```
            />
          );
        })}
      </svg>

      {/* Details panel when relationship is selected */}
      {activeRelationship && (
        <motion.div
          className="absolute bottom-4 left-4 right-4 bg-white dark:bg-gray-800 rounded-lg p-3
          initial={{ y: 20, opacity: 0 }}
          animate={{ y: 0, opacity: 1 }}
        >
          <h4 className="font-medium">
            {relationships.find(r => r.id === activeRelationship)?.name}
          </h4>
          <div className="flex justify-between mt-2">
            <div className="flex items-center text-sm">
              <Calendar size={16} className="mr-1" />
              <span>Last: 3 days ago</span>
            </div>
            <div className="flex items-center text-sm">
              <MessageCircle size={16} className="mr-1" />
              <span>14 messages</span>
            </div>
          </div>
        </motion.div>
      )}
    </div>
  );
}
```

6. 🧭 Personal Goals & Planner Page

External Integrations

- **Google Tasks/Todoist API**: Task management integration
- **Google Calendar API**: Deadline and milestone tracking
- **Notion API**: Project management integration
- **Habit Tracker APIs**: Integration with habit-focused apps

Analysis Features

- **Goal Completion Pattern Analysis**: Identifying success factors
- **Task Management Efficiency**: Time allocation and productivity patterns
- **Progress Visualization**: Interactive goal journey mapping
- **Habit Streak Analysis**: Visualizing consistency and patterns

Personalized Recommendations

- **Goal Refinement Suggestions**: Based on progress patterns
- **Task Prioritization Help**: AI-powered importance/urgency matrix
- **Productivity Optimization Tips**: Based on past performance patterns
- **Goal Achievement Strategies**: Personalized roadmap suggestions

UI Elements & Animations

- **3D Goal Mountain**: Interactive mountain visualization for goals
- **Progress Path Animation**: Animated path showing journey to goal
- **Task Flow Visualization**: Animated kanban or flow diagram
- **Theme**: Achievement-oriented colors with journey/map patterns
- **Icons**: Compass, map, trophy icons with progress-based animations

tsx

```jsx
// Example Goal Mountain Visualization
import { motion } from 'framer-motion';
import { useState } from 'react';

export default function GoalMountainVisual({ goal, milestones = [] }) {
  const [activeIndex, setActiveIndex] = useState(null);

  // Calculate progress percentage
  const completedMilestones = milestones.filter(m => m.completed).length;
  const progressPercent = milestones.length > 0
    ? (completedMilestones / milestones.length) * 100
    : 0;

  // Create mountain path based on milestones
  const createMountainPath = () => {
    const baseWidth = 400;
    const baseHeight = 200;

    // Start at bottom left
    let path = `M0,${baseHeight} `;

    // Create a point for each milestone
    milestones.forEach((milestone, i) => {
      const x = baseWidth * ((i + 1) / (milestones.length + 1));
      const height = baseHeight - (baseHeight * ((i + 1) / (milestones.length + 1)));
      path += `L${x},${height} `;
    });

    // End at summit
    path += `L${baseWidth},0 `;

    // Complete the shape
    path += `L${baseWidth},${baseHeight} L0,${baseHeight}`;

    return path;
  };

  return (
    <div className="relative bg-gradient-to-b from-blue-100 to-blue-50 dark:from-blue-900 dark:
      <h3 className="text-lg font-medium mb-2">{goal.title}</h3>
      <p className="text-sm text-blue-600 dark:text-blue-300 mb-4">Progress: {progressPercent.t

      <div className="relative h-64">
```

```
<svg viewBox="0 0 400 200" className="w-full h-full">
  {/* Mountain background */}
  <path
    d={createMountainPath()}
    fill="url(#mountainGradient)"
    stroke="#4b5563"
    strokeWidth="1"
  />

  {/* Mountain gradient */}
  <defs>
    <linearGradient id="mountainGradient" x1="0%" y1="0%" x2="0%" y2="100%">
      <stop offset="0%" stopColor="#60a5fa" stopOpacity="0.7" />
      <stop offset="100%" stopColor="#93c5fd" stopOpacity="0.3" />
    </linearGradient>

    <linearGradient id="progressGradient" x1="0%" y1="0%" x2="100%" y2="0%">
      <stop offset="0%" stopColor="#4ade80" />
      <stop offset="100%" stopColor="#22c55e" />
    </linearGradient>
  </defs>

  {/* Progress path */}
  <motion.path
    d={`M0,200 ${milestones.slice(0, completedMilestones).map((_, i) => {
      const x = 400 * ((i + 1) / (milestones.length + 1));
      const height = 200 - (200 * ((i + 1) / (milestones.length + 1)));
      return `L${x},${height}`;
    }).join(' ')} L${400 * ((completedMilestones + 1) / (milestones.length + 1))},${200
    fill="url(#progressGradient)"
    initial={{ opacity: 0 }}
    animate={{ opacity: 0.6 }}
    transition={{ duration: 1 }}
  />

  {/* Milestone markers */}
  {milestones.map((milestone, i) => {
    const x = 400 * ((i + 1) / (milestones.length + 1));
    const y = 200 - (200 * ((i + 1) / (milestones.length + 1)));

    return (
      <g key={i} onClick={() => setActiveIndex(i)}>
        <motion.circle
          cx={x}
```

```
            cy={y}
            r={10}
            fill={milestone.completed ? "#22c55e" : "#94a3b8"}
            stroke="white"
            strokeWidth="2"
            initial={{ scale: 0 }}
            animate={{ scale: 1 }}
            transition={{ delay: i * 0.1, type: 'spring' }}
            whileHover={{ scale: 1.2 }}
            style={{ cursor: 'pointer' }}
          />
          {activeIndex === i && (
            <motion.text
              x={x}
              y={y - 20}
              textAnchor="middle"
              fill="currentColor"
              fontSize="12"
              initial={{ opacity: 0, y: 10 }}
              animate={{ opacity: 1, y: 0 }}
            >
              {milestone.title}
            </motion.text>
          )}
        </g>
      );
    })}

    {/* Summit flag */}
    <motion.g
      transform="translate(400, 0)"
      initial={{ y: -20, opacity: 0 }}
      animate={{ y: 0, opacity: 1 }}
      transition={{ delay: 0.5, type: 'spring' }}
    >
      <rect x="-2" y="0" width="4" height="20" fill="#f43f5e" />
      <polygon points="-2,0 8,5 -2,10" fill="#f43f5e" />
    </motion.g>
  </svg>

  {/* Current position climber */}
  <motion.div
    className="absolute"
    style={{
```

```
            left: `${(completedMilestones / milestones.length) * 100}%`,
            bottom: `${(completedMilestones / milestones.length) * 50}%`
          }}
          initial={{ scale: 0 }}
          animate={{ scale: 1 }}
          transition={{ delay: 0.8, type: 'spring' }}
        >
          <span role="img" aria-label="climber" className="text-2xl">🧗</span>
        </motion.div>
      </div>
    </div>
  );
}
```

7. 📓 Life Journal Page

External Integrations

- **Dropbox/Google Drive API**: Backup and sync for journal entries
- **Notion API**: For structured journal templates
- **Calendar API**: For timeline correlation
- **Weather API**: To include weather context with entries

Analysis Features

- **Emotional Theme Detection**: AI analysis of recurring themes
- **Gratitude Pattern Recognition**: Identifying positive patterns
- **Writing Style Analysis**: Personal linguistic patterns
- **Memory Association Mapping**: Connecting related entries and memories

Personalized Recommendations

- **Journaling Prompt Suggestions**: Based on emotional needs and past entries
- **Reflection Exercise Ideas**: Customized based on current life situations
- **Memory Revisitation Suggestions**: "On this day" and meaningful memory surfacing

UI Elements & Animations

- **3D Journal Book**: Animated book with turning pages
- **Ink Flow Animation**: Writing that appears as if being handwritten

- **Timeline Visualization**: Interactive calendar with emotional color coding
- **Theme**: Paper textures, handwriting fonts, ink splatter accents
- **Icons**: Quill, book, calendar icons with subtle animations

tsx

```tsx
// Example 3D Journal Book Component
import { motion, useMotionValue, useTransform } from 'framer-motion';
import { useState } from 'react';

export default function JournalBook({ entries = [] }) {
  const [currentPage, setCurrentPage] = useState(0);
  const [isOpen, setIsOpen] = useState(false);

  // For page turn effect
  const pageX = useMotionValue(0);
  const pageRotation = useTransform(pageX, [0, 100], [0, -180]);

  const openBook = () => {
    setIsOpen(true);
  };

  const turnPage = (direction) => {
    if (direction === 'next' && currentPage < entries.length - 1) {
      pageX.set(0);
      pageX.set(100);
      setTimeout(() => {
```