RecoveryQuest

Pls use this color pallet instead:

A close up of a color

AI-generated content may be incorrect.

import React, { useState, useEffect } from 'react';

import { Trophy, Heart, Zap, Moon, Utensils, Star, CheckCircle, Circle, Medal, Target, Calendar, MapPin, Edit3, Save, X, Phone, MessageCircle, Plus, Trash2, Settings } from 'lucide-react';

const RecoveryQuestApp = () => {

// Initialize state with persistent data

const [gameData, setGameData] = useState(() => {

const saved = JSON.parse(localStorage.getItem('recoveryQuestData') || '{}');

return {

currentWeek: saved.currentWeek || 1,

completedTasks: saved.completedTasks || {},

totalXP: saved.totalXP || 0,

level: saved.level || 1,

achievements: saved.achievements || [],

startDate: saved.startDate || new Date().toISOString().split('T')[0],

lastActiveDate: saved.lastActiveDate || new Date().toISOString().split('T')[0],

streakDays: saved.streakDays || 0,

emergencyContacts: saved.emergencyContacts || [

{ name: 'Crisis Text Line', contact: 'Text HOME to 741741', type: 'text' },

{ name: 'SAMHSA', contact: '1-800-662-4357', type: 'phone' }

],

customTasks: saved.customTasks || {},

personalNotes: saved.personalNotes || {}

};

});

const [currentWeek, setCurrentWeek] = useState(gameData.currentWeek);

const [completedTasks, setCompletedTasks] = useState(gameData.completedTasks);

const [totalXP, setTotalXP] = useState(gameData.totalXP);

const [level, setLevel] = useState(gameData.level);

const [achievements, setAchievements] = useState(gameData.achievements);

const [streakDays, setStreakDays] = useState(gameData.streakDays);

const [emergencyContacts, setEmergencyContacts] = useState(gameData.emergencyContacts);

const [customTasks, setCustomTasks] = useState(gameData.customTasks);

const [personalNotes, setPersonalNotes] = useState(gameData.personalNotes);

// UI State

const [editingNote, setEditingNote] = useState(false);

const [noteText, setNoteText] = useState('');

const [showSettings, setShowSettings] = useState(false);

const [editingContacts, setEditingContacts] = useState(false);

const [newContact, setNewContact] = useState({ name: '', contact: '', type: 'phone' });

// Save data to localStorage whenever state changes

const saveGameData = (newData) => {

const updatedData = { ...gameData, ...newData };

setGameData(updatedData);

localStorage.setItem('recoveryQuestData', JSON.stringify(updatedData));

};

// Weekly quest data

const weeklyQuests = {

1: {

title: "The Beginning Journey",

subtitle: "Establishing Your Foundation",

color: "bg-gradient-to-br from-blue-500 to-purple-600",

icon: MapPin,

description: "Your body is adjusting. Small wins count big! 🌱",

dailyTasks: [

{ id: 'hydrate', text: 'Drink 6 glasses of water', xp: 10, icon: '💧' },

{ id: 'tiny\_meal', text: 'Eat something small every 3 hours', xp: 15, icon: '🥄' },

{ id: 'fresh\_air', text: 'Step outside for 10 minutes', xp: 10, icon: '🌤️' },

{ id: 'sleep\_routine', text: 'Start bedtime routine by 9 PM', xp: 15, icon: '🌙' },

{ id: 'breathe', text: 'Do 5 deep breaths when anxious', xp: 10, icon: '🫁' }

],

weeklyGoal: {

text: "Complete 25 daily tasks this week",

reward: "Unlock Week 2 + Recovery Warrior badge",

xp: 100

}

},

2: {

title: "Building Momentum",

subtitle: "Your Body is Learning",

color: "bg-gradient-to-br from-green-500 to-teal-600",

icon: Zap,

description: "Appetite might still be tricky, but you're getting stronger! 💪",

dailyTasks: [

{ id: 'smoothie', text: 'Try a protein smoothie or shake', xp: 15, icon: '🥤' },

{ id: 'walk', text: 'Take a 15-minute walk', xp: 15, icon: '🚶' },

{ id: 'comfort\_food', text: 'Eat one comfort food you enjoy', xp: 10, icon: '🍞' },

{ id: 'journal', text: 'Write 3 things you notice improving', xp: 15, icon: '' },

{ id: 'connect', text: 'Text or call someone supportive', xp: 10, icon: '📱' }

],

weeklyGoal: {

text: "Complete 30 daily tasks this week",

reward: "Unlock Week 3 + Appetite Adventurer badge",

xp: 150

}

},

3: {

title: "Finding Your Rhythm",

subtitle: "The Fog is Lifting",

color: "bg-gradient-to-br from-yellow-500 to-orange-600",

icon: Star,

description: "You might start feeling more like yourself again! ✨",

dailyTasks: [

{ id: 'real\_meal', text: 'Eat one full meal (any size)', xp: 20, icon: '🍽️' },

{ id: 'hobby', text: 'Do something you used to enjoy for 20 min', xp: 15, icon: '🎨' },

{ id: 'exercise', text: 'Try gentle exercise (yoga, stretching)', xp: 15, icon: '🧘' },

{ id: 'treat', text: 'Give yourself a small treat/reward', xp: 10, icon: '🍨' },

{ id: 'progress', text: 'Notice one improvement from Week 1', xp: 15, icon: '📈' }

],

weeklyGoal: {

text: "Complete 32 daily tasks this week",

reward: "Unlock Week 4 + Rhythm Master badge",

xp: 200

}

},

4: {

title: "Mastery Mode",

subtitle: "You're a Recovery Champion",

color: "bg-gradient-to-br from-purple-500 to-pink-600",

icon: Trophy,

description: "Look how far you've come! Time to celebrate! 🎉",

dailyTasks: [

{ id: 'celebrate', text: 'Celebrate one win from your journey', xp: 20, icon: '' },

{ id: 'help\_others', text: 'Share encouragement with someone else', xp: 20, icon: '🤝' },

{ id: 'plan\_future', text: 'Make one plan for your continued wellness', xp: 15, icon: '' },

{ id: 'full\_day', text: 'Have a day with normal appetite', xp: 25, icon: '🍽️' },

{ id: 'reflect', text: 'Write about your transformation', xp: 15, icon: '🪞' }

],

weeklyGoal: {

text: "Complete 30 daily tasks this week",

reward: "Recovery Champion Badge + Special Celebration",

xp: 300

}

}

};

const badges = [

{ id: 'warrior', name: 'Recovery Warrior', icon: '⚔️', description: 'Started your journey!' },

{ id: 'appetite', name: 'Appetite Adventurer', icon: '🍎', description: 'Working with your changing appetite' },

{ id: 'rhythm', name: 'Rhythm Master', icon: '🎵', description: 'Found your recovery rhythm' },

{ id: 'champion', name: 'Recovery Champion', icon: '🏆', description: 'Completed the 4-week quest!' }

];

const toggleTask = (taskId) => {

const today = new Date().toISOString().split('T')[0];

const weekKey = `week${currentWeek}`;

const taskKey = `${weekKey}\_${taskId}\_${today}`;

setCompletedTasks(prev => {

const newCompleted = { ...prev };

const isCompleting = !newCompleted[taskKey];

if (isCompleting) {

newCompleted[taskKey] = true;

const task = [...weeklyQuests[currentWeek].dailyTasks, ...(customTasks[currentWeek] || [])].find(t => t.id === taskId);

const newXP = totalXP + task.xp;

setTotalXP(newXP);

checkAchievements(newXP, newCompleted);

updateStreak(today);

saveGameData({

completedTasks: newCompleted,

totalXP: newXP,

lastActiveDate: today

});

} else {

delete newCompleted[taskKey];

const task = [...weeklyQuests[currentWeek].dailyTasks, ...(customTasks[currentWeek] || [])].find(t => t.id === taskId);

const newXP = totalXP - task.xp;

setTotalXP(newXP);

saveGameData({

completedTasks: newCompleted,

totalXP: newXP

});

}

return newCompleted;

});

};

const checkAchievements = (xp, tasks) => {

const newAchievements = [...achievements];

if (xp >= 100 && !achievements.includes('warrior')) {

newAchievements.push('warrior');

}

if (xp >= 300 && !achievements.includes('appetite')) {

newAchievements.push('appetite');

}

if (xp >= 600 && !achievements.includes('rhythm')) {

newAchievements.push('rhythm');

}

if (xp >= 1000 && !achievements.includes('champion')) {

newAchievements.push('champion');

}

if (newAchievements.length > achievements.length) {

setAchievements(newAchievements);

saveGameData({ achievements: newAchievements });

}

};

const updateStreak = (today) => {

const yesterday = new Date();

yesterday.setDate(yesterday.getDate() - 1);

const yesterdayStr = yesterday.toISOString().split('T')[0];

if (gameData.lastActiveDate === yesterdayStr) {

const newStreak = streakDays + 1;

setStreakDays(newStreak);

saveGameData({ streakDays: newStreak });

} else if (gameData.lastActiveDate !== today) {

setStreakDays(1);

saveGameData({ streakDays: 1 });

}

};

const getWeekProgress = (week) => {

const today = new Date().toISOString().split('T')[0];

const weekKey = `week${week}`;

const todayCompleted = Object.keys(completedTasks).filter(key =>

key.startsWith(weekKey) && key.endsWith(today) && completedTasks[key]

).length;

const totalToday = weeklyQuests[week].dailyTasks.length + (customTasks[week] || []).length;

return { completed: todayCompleted, total: totalToday };

};

const getTodaysCompletedTasks = () => {

const today = new Date().toISOString().split('T')[0];

const weekKey = `week${currentWeek}`;

const allTasks = [...weeklyQuests[currentWeek].dailyTasks, ...(customTasks[currentWeek] || [])];

return allTasks.map(task => {

const taskKey = `${weekKey}\_${task.id}\_${today}`;

return {

...task,

isCompleted: !!completedTasks[taskKey]

};

});

};

const changeWeek = (week) => {

setCurrentWeek(week);

saveGameData({ currentWeek: week });

};

const saveNote = () => {

const today = new Date().toISOString().split('T')[0];

const newNotes = { ...personalNotes, [today]: noteText };

setPersonalNotes(newNotes);

saveGameData({ personalNotes: newNotes });

setEditingNote(false);

};

const addCustomTask = () => {

const newTask = {

id: `custom\_${Date.now()}`,

text: 'New custom task',

xp: 10,

icon: '⭐',

isCustom: true

};

const newCustomTasks = {

...customTasks,

[currentWeek]: [...(customTasks[currentWeek] || []), newTask]

};

setCustomTasks(newCustomTasks);

saveGameData({ customTasks: newCustomTasks });

};

const removeCustomTask = (taskId) => {

const newCustomTasks = {

...customTasks,

[currentWeek]: (customTasks[currentWeek] || []).filter(task => task.id !== taskId)

};

setCustomTasks(newCustomTasks);

saveGameData({ customTasks: newCustomTasks });

};

const addEmergencyContact = () => {

if (newContact.name && newContact.contact) {

const newContacts = [...emergencyContacts, { ...newContact, id: Date.now() }];

setEmergencyContacts(newContacts);

saveGameData({ emergencyContacts: newContacts });

setNewContact({ name: '', contact: '', type: 'phone' });

}

};

const removeEmergencyContact = (index) => {

const newContacts = emergencyContacts.filter((\_, i) => i !== index);

setEmergencyContacts(newContacts);

saveGameData({ emergencyContacts: newContacts });

};

const currentWeekProgress = getWeekProgress(currentWeek);

const progressPercentage = (currentWeekProgress.completed / currentWeekProgress.total) \* 100;

const today = new Date().toISOString().split('T')[0];

const todayNote = personalNotes[today] || '';

useEffect(() => {

const newLevel = Math.floor(totalXP / 100) + 1;

setLevel(newLevel);

}, [totalXP]);

useEffect(() => {

if (editingNote) {

setNoteText(todayNote);

}

}, [editingNote, todayNote]);

const getCurrentWeekQuest = () => weeklyQuests[currentWeek];

const quest = getCurrentWeekQuest();

return (

<div className="max-w-md mx-auto bg-gradient-to-br from-indigo-900 via-purple-900 to-pink-900 min-h-screen text-white relative">

{/\* Settings Button \*/}

<button

onClick={() => setShowSettings(!showSettings)}

className="absolute top-4 right-4 p-2 bg-white/20 rounded-full hover:bg-white/30 transition-colors z-10"

>

<Settings className="w-5 h-5" />

</button>

{/\* Settings Panel \*/}

{showSettings && (

<div className="absolute inset-0 bg-black/50 z-20 p-4 overflow-y-auto">

<div className="bg-gray-800 rounded-xl p-6 mt-16">

<div className="flex justify-between items-center mb-4">

<h3 className="text-lg font-bold">Settings</h3>

<button onClick={() => setShowSettings(false)}>

<X className="w-6 h-6" />

</button>

</div>

{/\* Emergency Contacts Section \*/}

<div className="mb-6">

<div className="flex items-center justify-between mb-3">

<h4 className="font-semibold">Emergency Contacts</h4>

<button

onClick={() => setEditingContacts(!editingContacts)}

className="text-blue-400 hover:text-blue-300"

>

<Edit3 className="w-4 h-4" />

</button>

</div>

{editingContacts && (

<div className="mb-4 p-3 bg-gray-700 rounded-lg">

<input

type="text"

placeholder="Contact name"

value={newContact.name}

onChange={(e) => setNewContact({...newContact, name: e.target.value})}

className="w-full p-2 mb-2 bg-gray-600 rounded text-white placeholder-gray-400"

/>

<input

type="text"

placeholder="Phone number or text instructions"

value={newContact.contact}

onChange={(e) => setNewContact({...newContact, contact: e.target.value})}

className="w-full p-2 mb-2 bg-gray-600 rounded text-white placeholder-gray-400"

/>

<select

value={newContact.type}

onChange={(e) => setNewContact({...newContact, type: e.target.value})}

className="w-full p-2 mb-2 bg-gray-600 rounded text-white"

>

<option value="phone">Phone</option>

<option value="text">Text</option>

</select>

<button

onClick={addEmergencyContact}

className="w-full p-2 bg-green-600 hover:bg-green-700 rounded text-white"

>

Add Contact

</button>

</div>

)}

<div className="space-y-2">

{emergencyContacts.map((contact, index) => (

<div key={index} className="flex items-center justify-between p-3 bg-red-500/20 rounded-lg border border-red-400/30">

<div className="flex items-center flex-1">

{contact.type === 'phone' ? (

<Phone className="w-4 h-4 text-red-400 mr-2" />

) : (

<MessageCircle className="w-4 h-4 text-red-400 mr-2" />

)}

<div>

<div className="font-medium text-sm">{contact.name}</div>

<div className="text-xs opacity-80">{contact.contact}</div>

</div>

</div>

{editingContacts && index >= 2 && (

<button

onClick={() => removeEmergencyContact(index)}

className="text-red-400 hover:text-red-300 ml-2"

>

<Trash2 className="w-4 h-4" />

</button>

)}

</div>

))}

</div>

</div>

{/\* Custom Tasks Section \*/}

<div className="mb-6">

<div className="flex items-center justify-between mb-3">

<h4 className="font-semibold">Custom Tasks for Week {currentWeek}</h4>

<button

onClick={addCustomTask}

className="text-green-400 hover:text-green-300"

>

<Plus className="w-4 h-4" />

</button>

</div>

<div className="space-y-2">

{(customTasks[currentWeek] || []).map((task) => (

<div key={task.id} className="flex items-center justify-between p-2 bg-gray-700 rounded">

<span className="text-sm">{task.text}</span>

<button

onClick={() => removeCustomTask(task.id)}

className="text-red-400 hover:text-red-300"

>

<Trash2 className="w-4 h-4" />

</button>

</div>

))}

</div>

</div>

</div>

</div>

)}

{/\* Header \*/}

<div className="p-6 text-center">

<div className="flex items-center justify-center mb-4">

<Trophy className="w-8 h-8 text-yellow-400 mr-2" />

<h1 className="text-2xl font-bold">Recovery Quest</h1>

</div>

{/\* XP and Level \*/}

<div className="flex items-center justify-center space-x-4 mb-4">

<div className="text-center">

<div className="text-2xl font-bold text-yellow-400">{totalXP}</div>

<div className="text-xs opacity-80">XP</div>

</div>

<div className="text-center">

<div className="text-2xl font-bold text-blue-400">Level {level}</div>

<div className="text-xs opacity-80">Recovery Hero</div>

</div>

<div className="text-center">

<div className="text-2xl font-bold text-green-400">{streakDays}</div>

<div className="text-xs opacity-80">Day Streak</div>

</div>

</div>

{/\* Progress Bar \*/}

<div className="w-full bg-white/20 rounded-full h-3 mb-2">

<div

className="bg-gradient-to-r from-green-400 to-blue-500 h-3 rounded-full transition-all duration-500"

style={{ width: `${Math.min(progressPercentage, 100)}%` }}

></div>

</div>

<div className="text-sm opacity-80">

{currentWeekProgress.completed}/{currentWeekProgress.total} tasks completed today

</div>

{/\* Achievements Display \*/}

{achievements.length > 0 && (

<div className="flex justify-center space-x-2 mt-3">

{achievements.map(achievementId => {

const badge = badges.find(b => b.id === achievementId);

return (

<div key={achievementId} className="text-2xl" title={badge?.description}>

{badge?.icon}

</div>

);

})}

</div>

)}

</div>

{/\* Week Navigation \*/}

<div className="px-6 mb-6">

<div className="flex space-x-2 justify-center">

{[1, 2, 3, 4].map(week => (

<button

key={week}

onClick={() => changeWeek(week)}

className={`px-4 py-2 rounded-full text-sm font-medium transition-all ${

currentWeek === week

? 'bg-white text-purple-900'

: 'bg-white/20 hover:bg-white/30'

}`}

>

Week {week}

</button>

))}

</div>

</div>

{/\* Current Quest \*/}

<div className="mx-6 mb-6">

<div className={`${quest.color} rounded-2xl p-6 text-center`}>

<quest.icon className="w-12 h-12 mx-auto mb-3" />

<h2 className="text-xl font-bold mb-1">{quest.title}</h2>

<p className="text-sm opacity-90 mb-2">{quest.subtitle}</p>

<p className="text-sm">{quest.description}</p>

</div>

</div>

{/\* Personal Notes Section \*/}

<div className="mx-6 mb-6">

<div className="bg-white/10 rounded-xl p-4">

<div className="flex items-center justify-between mb-3">

<h4 className="font-bold flex items-center">

<Edit3 className="w-4 h-4 mr-2" />

Today's Notes

</h4>

{!editingNote ? (

<button

onClick={() => setEditingNote(true)}

className="text-blue-400 hover:text-blue-300"

>

<Edit3 className="w-4 h-4" />

</button>

) : (

<div className="flex space-x-2">

<button

onClick={saveNote}

className="text-green-400 hover:text-green-300"

>

<Save className="w-4 h-4" />

</button>

<button

onClick={() => setEditingNote(false)}

className="text-red-400 hover:text-red-300"

>

<X className="w-4 h-4" />

</button>

</div>

)}

</div>

{editingNote ? (

<textarea

value={noteText}

onChange={(e) => setNoteText(e.target.value)}

placeholder="How are you feeling today? What's on your mind?"

className="w-full p-3 bg-white/10 rounded-lg text-white placeholder-gray-300 resize-none"

rows="3"

/>

) : (

<p className="text-sm opacity-80 italic">

{todayNote || "Tap the edit button to add your thoughts for today..."}

</p>

)}

</div>

</div>

{/\* Daily Tasks \*/}

<div className="px-6 mb-6">

<h3 className="text-lg font-bold mb-4 flex items-center">

<Target className="w-5 h-5 mr-2" />

Today's Quests

</h3>

<div className="space-y-3">

{getTodaysCompletedTasks().map(task => {

return (

<div

key={task.id}

onClick={() => toggleTask(task.id)}

className={`flex items-center p-4 rounded-xl cursor-pointer transition-all ${

task.isCompleted

? 'bg-green-500/20 border-2 border-green-400'

: 'bg-white/10 hover:bg-white/20'

}`}

>

<div className="text-2xl mr-3">{task.icon}</div>

<div className="flex-1">

<p className={`font-medium ${task.isCompleted ? 'line-through opacity-70' : ''}`}>

{task.text}

</p>

<p className="text-sm text-yellow-400">+{task.xp} XP</p>

</div>

{task.isCompleted ? (

<CheckCircle className="w-6 h-6 text-green-400" />

) : (

<Circle className="w-6 h-6 text-white/50" />

)}

</div>

);

})}

</div>

</div>

{/\* Weekly Goal \*/}

<div className="mx-6 mb-6">

<div className="bg-gradient-to-r from-yellow-500/20 to-orange-500/20 rounded-xl p-4 border border-yellow-400/30">

<div className="flex items-center mb-2">

<Medal className="w-5 h-5 text-yellow-400 mr-2" />

<h4 className="font-bold">Weekly Challenge</h4>

</div>

<p className="text-sm mb-2">{quest.weeklyGoal.text}</p>

<p className="text-xs text-yellow-400">Reward: {quest.weeklyGoal.reward}</p>

</div>

</div>

{/\* Emergency Support \*/}

<div className="mx-6 mb-6">

<div className="bg-red-500/20 rounded-xl p-4 border border-red-400/30">

<div className="flex items-center mb-3">

<Heart className="w-5 h-5 text-red-400 mr-2" />

<h4 className="font-bold">Need Support?</h4>

</div>

<div className="space-y-2">

{emergencyContacts.map((contact, index) => (

<div key={index} className="flex items-center text-sm">

{contact.type === 'phone' ? (

<Phone className="w-4 h-4 text-red-400 mr-2" />

) : (

<MessageCircle className="w-4 h-4 text-red-400 mr-2" />

)}

<span className="font-medium mr-2">{contact.name}:</span>

<span className="opacity-90">{contact.contact}</span>

</div>

))}

</div>

</div>

</div>

{/\* Motivational Footer \*/}

<div className="text-center p-6 pb-8">

<div className="text-sm opacity-80">

🌟 "Every small step is a victory. You're doing amazing!" 🌟

</div>

</div>

</div>

);

};

export default RecoveryQuestApp;