UI UX Project Ideas With Source Code

Here are the latest UI UX Project Ideas With Source Code:

1. One-Handed Mobile Menu

A slide-out menu that can be used with one hand on mobile devices. It has big, easy-to-tap buttons and smooth animations.
Benefits: It makes apps more accessible for users with limited hand mobility or multitasking.

Source Code

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>One-Handed Mobile Menu</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 0;

        }

        .menu-toggle {

            position: fixed;

            bottom: 20px;

            right: 20px;

            background: #007bff;

            color: white;

            border: none;
```

```
    border-radius: 50%;
    width: 60px;
    height: 60px;
    font-size: 24px;
    cursor: pointer;
}
.menu {
    position: fixed;
    bottom: 0;
    right: -250px;
    width: 250px;
    height: 100%;
    background: #f8f9fa;
    transition: right 0.3s ease-in-out;
}
.menu.open {
    right: 0;
}
.menu ul {
    list-style-type: none;
    padding: 0;
    margin: 0;
}
.menu li {
    padding: 15px;
    border-bottom: 1px solid #dee2e6;
```

```
        }
        .menu a {
            color: #343a40;
            text-decoration: none;
            font-size: 18px;
        }
    </style>
</head>
<body>
    <button class="menu-toggle">☰</button>
    <nav class="menu">
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">Profile</a></li>
            <li><a href="#">Settings</a></li>
            <li><a href="#">Help</a></li>
            <li><a href="#">Logout</a></li>
        </ul>
    </nav>

    <script>
        const menuToggle = document.querySelector('.menu-toggle');
        const menu = document.querySelector('.menu');

        menuToggle.addEventListener('click', () => {
            menu.classList.toggle('open');
```

```
    });
  </script>
</body>
</html>
```

2. Color Blind Friendly Palette Picker

A tool that helps designers pick colors that work well for color-blind users. It shows how the colors look to different types of color blindness.
Benefits: Improves app usability for color-blind users and raises awareness about inclusive design.

Source Code

```
import React, { useState } from 'react';
import { HexColorPicker } from 'react-colorful';


const ColorBlindSimulation = ({ color }) => {
 const simulateColorBlindness = (hex, type) => {
  // This is a simplified simulation. For a real project, use a more accurate algorithm.
  const r = parseInt(hex.slice(1, 3), 16);
  const g = parseInt(hex.slice(3, 5), 16);
  const b = parseInt(hex.slice(5, 7), 16);

  switch (type) {
   case 'protanopia':
    return `rgb(${0.567 * r + 0.433 * g}, ${0.558 * r + 0.442 * g}, ${0.242 * r + 0.758 * b})`;
   case 'deuteranopia':
    return `rgb(${0.625 * r + 0.375 * g}, ${0.7 * r + 0.3 * g}, ${0.3 * r + 0.7 * b})`;
   case 'tritanopia':
    return `rgb(${0.95 * r + 0.05 * g}, ${0.433 * r + 0.567 * g}, ${0.475 * r + 0.525 * g})`;
```

```
    default:
      return hex;
  }
};

return (
  <div className="flex gap-4">
    <div className="flex flex-col items-center">
      <div className="w-20 h-20 rounded" style={{ backgroundColor: color }}></div>
      <span>Normal</span>
    </div>
    <div className="flex flex-col items-center">
      <div className="w-20 h-20 rounded" style={{ backgroundColor: simulateColorBlindness(color, 'protanopia') }}></div>
      <span>Protanopia</span>
    </div>
    <div className="flex flex-col items-center">
      <div className="w-20 h-20 rounded" style={{ backgroundColor: simulateColorBlindness(color, 'deuteranopia') }}></div>
      <span>Deuteranopia</span>
    </div>
    <div className="flex flex-col items-center">
      <div className="w-20 h-20 rounded" style={{ backgroundColor: simulateColorBlindness(color, 'tritanopia') }}></div>
      <span>Tritanopia</span>
    </div>
  </div>
);
```

```
};

const ColorBlindPalettePicker = () => {
  const [color, setColor] = useState('#000000');

  return (
    <div className="p-4">
      <h1 className="text-2xl font-bold mb-4">Color Blind Friendly Palette Picker</h1>
      <div className="flex gap-8">
        <HexColorPicker color={color} onChange={setColor} />
        <ColorBlindSimulation color={color} />
      </div>
    </div>
  );
};

export default ColorBlindPalettePicker;
```

3. Voice-Controlled Smart Home Dashboard

A dashboard for controlling smart home devices using voice commands. It shows device status and lets users turn things on or off by speaking.
Benefits: It makes smart home control easier for everyone, especially those with mobility issues.

Source Code

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

const VoiceControlledDashboard = () => {
  const [devices, setDevices] = useState({
```

```
  livingRoomLight: false,
  kitchenLight: false,
  thermostat: 70,
  tvPower: false
 });

 const [listening, setListening] = useState(false);

 useEffect(() => {
  if ('webkitSpeechRecognition' in window) {
   const recognition = new window.webkitSpeechRecognition();
   recognition.continuous = true;
   recognition.interimResults = true;

   recognition.onresult = (event) => {
                    const transcript = event.results[event.results.length - 1][0].transcript.toLowerCase();
    handleVoiceCommand(transcript);
   };

   if (listening) {
    recognition.start();
   } else {
    recognition.stop();
   }

   return () => {
```

```
      recognition.stop();
    };
  }
}, [listening]);

const handleVoiceCommand = (command) => {
  if (command.includes('turn on living room light')) {
    setDevices(prev => ({ ...prev, livingRoomLight: true }));
  } else if (command.includes('turn off living room light')) {
    setDevices(prev => ({ ...prev, livingRoomLight: false }));
  } else if (command.includes('turn on kitchen light')) {
    setDevices(prev => ({ ...prev, kitchenLight: true }));
  } else if (command.includes('turn off kitchen light')) {
    setDevices(prev => ({ ...prev, kitchenLight: false }));
  } else if (command.includes('set thermostat to')) {
    const temp = parseInt(command.split('to')[1]);
    if (!isNaN(temp)) {
      setDevices(prev => ({ ...prev, thermostat: temp }));
    }
  } else if (command.includes('turn on tv')) {
    setDevices(prev => ({ ...prev, tvPower: true }));
  } else if (command.includes('turn off tv')) {
    setDevices(prev => ({ ...prev, tvPower: false }));
  }
};
```

```
  return (
    <div className="p-4">
      <h1 className="text-2xl font-bold mb-4">Smart Home Dashboard</h1>
      <button
        onClick={() => setListening(!listening)}
        className={`mb-4 px-4 py-2 rounded ${listening ? 'bg-red-500' : 'bg-green-500'} text-white`}
      >
        {listening ? 'Stop Listening' : 'Start Listening'}
      </button>
      <div className="grid grid-cols-2 gap-4">
        <div className="p-4 border rounded">
          <h2 className="text-lg font-semibold">Living Room Light</h2>
          <p>{devices.livingRoomLight ? 'On' : 'Off'}</p>
        </div>
        <div className="p-4 border rounded">
          <h2 className="text-lg font-semibold">Kitchen Light</h2>
          <p>{devices.kitchenLight ? 'On' : 'Off'}</p>
        </div>
        <div className="p-4 border rounded">
          <h2 className="text-lg font-semibold">Thermostat</h2>
          <p>{devices.thermostat}°F</p>
        </div>
        <div className="p-4 border rounded">
          <h2 className="text-lg font-semibold">TV</h2>
          <p>{devices.tvPower ? 'On' : 'Off'}</p>
        </div>
```

```
      </div>
    </div>
  );
};

export default VoiceControlledDashboard;
```

4. Emoji Mood Tracker

An app that lets users track their mood using emojis. It shows mood trends over time with fun, colorful charts.
Benefits: Makes mood tracking more fun and accessible, helping users understand their emotional patterns.

Source Code

```
import React, { useState, useEffect } from 'react';
import { LineChart, Line, XAxis, YAxis, Tooltip, ResponsiveContainer } from 'recharts';

const emojis = ['😢', '😕', '😐', '🙂', '😄'];

const EmojiMoodTracker = () => {
  const [moodData, setMoodData] = useState([]);
  const [currentMood, setCurrentMood] = useState(null);

  useEffect(() => {
    const storedMoodData = localStorage.getItem('moodData');
    if (storedMoodData) {
      setMoodData(JSON.parse(storedMoodData));
    }
```

```
  }, []);

  useEffect(() => {
    localStorage.setItem('moodData', JSON.stringify(moodData));
  }, [moodData]);

  const handleMoodSelect = (index) => {
    const newMoodEntry = {
      date: new Date().toISOString().split('T')[0],
      mood: index
    };
    setMoodData([...moodData, newMoodEntry]);
    setCurrentMood(index);
  };

  return (
    <div className="p-4">
      <h1 className="text-2xl font-bold mb-4">Emoji Mood Tracker</h1>
      <div className="mb-4">
        <h2 className="text-lg font-semibold mb-2">How are you feeling today?</h2>
        <div className="flex gap-4">
          {emojis.map((emoji, index) => (
            <button
              key={index}
              onClick={() => handleMoodSelect(index)}
              className={`text-4xl ${currentMood === index ? 'border-2 border-blue-500' :
''}`}
```

```
          >
            {emoji}
          </button>
        ))}
      </div>
    </div>
    <div className="h-64">
      <ResponsiveContainer width="100%" height="100%">
        <LineChart data={moodData}>
          <XAxis dataKey="date" />
                <YAxis domain={[0, 4]} ticks={[0, 1, 2, 3, 4]} tickFormatter={(value) =>
emojis[value]} />
          <Tooltip
            formatter={(value) => emojis[value]}
            labelFormatter={(label) => `Date: ${label}`}
          />
          <Line type="monotone" dataKey="mood" stroke="#8884d8" />
        </LineChart>
      </ResponsiveContainer>
    </div>
  </div>
 );
};

export default EmojiMoodTracker;
```

5. Gesture-Based Music Player

A music player that uses hand gestures to control playback. Wave to skip tracks, make a fist to pause, and spread your fingers to adjust volume.
Benefits: Allows hands-free music control, great for cooking or working out.

Source Code

```
const startVideo = () => {
  handTrack.startVideo(videoRef.current).then(function (status) {
    if (status) {
      runDetection();
    }
  });
};

const runDetection = () => {
  modelRef.current.detect(videoRef.current).then((predictions) => {
    if (predictions.length > 0) {
      handleGesture(predictions[0]);
    }
    requestAnimationFrame(runDetection);
  });
};

const handleGesture = (prediction) => {
  if (prediction.label === 'open') {
    // Spread fingers to adjust volume
    const newVolume = Math.min(100, volume + 5);
    setVolume(newVolume);
```

```
  } else if (prediction.label === 'closed') {
    // Make a fist to pause/play
    setIsPlaying(!isPlaying);
  } else if (prediction.label === 'point') {
    // Point to skip track
    const nextTrack = (currentTrack + 1) % tracks.length;
    setCurrentTrack(nextTrack);
  }
};

return (
  <div className="p-4">
    <h1 className="text-2xl font-bold mb-4">Gesture-Based Music Player</h1>
    <div className="mb-4">
      <h2 className="text-lg font-semibold">{tracks[currentTrack].name}</h2>
      <p>{tracks[currentTrack].artist}</p>
    </div>
    <div className="mb-4">
      <button
        onClick={() => setIsPlaying(!isPlaying)}
        className="px-4 py-2 bg-blue-500 text-white rounded"
      >
        {isPlaying ? 'Pause' : 'Play'}
      </button>
    </div>
    <div className="mb-4">
```

```
            <label className="block text-sm font-medium text-gray-700">Volume: {volume}%</label>
        <input
          type="range"
          min="0"
          max="100"
          value={volume}
          onChange={(e) => setVolume(parseInt(e.target.value))}
          className="mt-1 block w-full"
        />
      </div>
      <video ref={videoRef} className="hidden" />
    </div>
  );
};

export default GestureMusicPlayer;
```

6. Dyslexia-Friendly Reader

A web app that makes text easier to read for people with dyslexia. It lets users change fonts, spacing, and colors.
Benefits: Improves reading experience for dyslexic users and raises awareness about accessibility.

Source Code

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


const DyslexiaFriendlyReader = () => {
```

```
const [text, setText] = useState("Enter your text here...");
const [font, setFont] = useState("Arial");
const [fontSize, setFontSize] = useState(16);
const [lineSpacing, setLineSpacing] = useState(1.5);
const [backgroundColor, setBackgroundColor] = useState("#ffffff");
const [textColor, setTextColor] = useState("#000000");

const fonts = ["Arial", "OpenDyslexic", "Comic Sans MS", "Verdana"];
const colors = ["#ffffff", "#f0f0f0", "#ffe5b4", "#e0ffff"];

return (
  <div className="p-4" style={{ backgroundColor }}>
    <h1 className="text-2xl font-bold mb-4">Dyslexia-Friendly Reader</h1>
    <div className="mb-4">
      <textarea
        value={text}
        onChange={(e) => setText(e.target.value)}
        className="w-full h-40 p-2 border rounded"
        style={{
          fontFamily: font,
          fontSize: `${fontSize}px`,
          lineHeight: lineSpacing,
          backgroundColor,
          color: textColor,
        }}
      />
```

```
</div>
<div className="grid grid-cols-2 gap-4">
  <div>
    <label className="block text-sm font-medium text-gray-700">Font</label>
    <select
      value={font}
      onChange={(e) => setFont(e.target.value)}
      className="mt-1 block w-full rounded-md border-gray-300 shadow-sm"
    >
      {fonts.map((f) => (
        <option key={f} value={f}>{f}</option>
      ))}
    </select>
  </div>
  <div>
    <label className="block text-sm font-medium text-gray-700">Font Size</label>
    <input
      type="range"
      min="12"
      max="24"
      value={fontSize}
      onChange={(e) => setFontSize(parseInt(e.target.value))}
      className="mt-1 block w-full"
    />
  </div>
  <div>
```

```
                <label className="block text-sm font-medium text-gray-700">Line Spacing</label>
        <input
          type="range"
          min="1"
          max="3"
          step="0.1"
          value={lineSpacing}
          onChange={(e) => setLineSpacing(parseFloat(e.target.value))}
          className="mt-1 block w-full"
        />
      </div>
      <div>
            <label className="block text-sm font-medium text-gray-700">Background Color</label>
        <select
          value={backgroundColor}
          onChange={(e) => setBackgroundColor(e.target.value)}
          className="mt-1 block w-full rounded-md border-gray-300 shadow-sm"
        >
          {colors.map((c) => (
            <option key={c} value={c}>{c}</option>
          ))}
        </select>
      </div>
    </div>
  </div>
```

```
  );
};

export default DyslexiaFriendlyReader;
```

7. Gamified To-Do List

A to-do list app that turns tasks into a game. Users earn points and level up by finishing tasks.
Benefits: Makes task management more fun and motivating.

Source Code

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

const GamifiedTodoList = () => {
  const [tasks, setTasks] = useState([]);
  const [newTask, setNewTask] = useState('');
  const [points, setPoints] = useState(0);
  const [level, setLevel] = useState(1);

  useEffect(() => {
    const newLevel = Math.floor(points / 100) + 1;
    setLevel(newLevel);
  }, [points]);

  const addTask = () => {
    if (newTask.trim() !== '') {
      setTasks([...tasks, { id: Date.now(), text: newTask, completed: false }]);
      setNewTask('');
```

```
  }
};

const toggleTask = (id) => {
  setTasks(tasks.map(task =>
    task.id === id ? { ...task, completed: !task.completed } : task
  ));
  setPoints(points + 10);
};

const deleteTask = (id) => {
  setTasks(tasks.filter(task => task.id !== id));
};

return (
  <div className="p-4">
    <h1 className="text-2xl font-bold mb-4">Gamified To-Do List</h1>
    <div className="mb-4">
      <p className="text-lg">Level: {level}</p>
      <p className="text-lg">Points: {points}</p>
      <div className="w-full bg-gray-200 rounded-full h-2.5 dark:bg-gray-700">
          <div className="bg-blue-600 h-2.5 rounded-full" style={{width: `${points % 100}%`}}></div>
      </div>
    </div>
    <div className="mb-4 flex">
      <input
```

```
        type="text"
        value={newTask}
        onChange={(e) => setNewTask(e.target.value)}
        className="flex-grow p-2 border rounded-l"
        placeholder="Enter a new task"
      />
            <button onClick={addTask} className="px-4 py-2 bg-blue-500 text-white rounded-r">Add</button>
    </div>
    <ul>
      {tasks.map(task => (
        <li key={task.id} className="flex items-center mb-2">
          <input
            type="checkbox"
            checked={task.completed}
            onChange={() => toggleTask(task.id)}
            className="mr-2"
          />
          <span className={task.completed ? 'line-through' : ''}>{task.text}</span>
              <button onClick={() => deleteTask(task.id)} className="ml-auto px-2 py-1 bg-red-500 text-white rounded">Delete</button>
        </li>
      ))}
    </ul>
  </div>
 );
};
```

```
export default GamifiedTodoList;
```

8. AR Plant Care Guide

An app that uses your phone's camera to identify plants and show care tips in augmented reality.
Benefits: Makes plant care easier and more interactive, especially for new plant owners.
AR Plant Care Guide

Source Code

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

import * as tf from '@tensorflow/tfjs';

import * as mobilenet from '@tensorflow-models/mobilenet';

import { Camera } from 'react-camera-pro';


const ARPlantCareGuide = () => {

  const [model, setModel] = useState(null);

  const [prediction, setPrediction] = useState(null);

  const camera = useRef(null);


  useEffect(() => {

    const loadModel = async () => {

      const loadedModel = await mobilenet.load();

      setModel(loadedModel);

    };

    loadModel();

  }, []);


  const handleCapture = async () => {
```

```
  if (camera.current && model) {
    const photo = camera.current.takePhoto();
    const img = new Image();
    img.src = photo;
    img.onload = async () => {
      const tfImg = tf.browser.fromPixels(img);
      const resized = tf.image.resizeBilinear(tfImg, [224, 224]);
      const expanded = resized.expandDims(0);
      const predictions = await model.classify(expanded);
      setPrediction(predictions[0]);
    };
  }
};

const plantCareInfo = {
  'potted plant': 'Water once a week, place in indirect sunlight.',
  'flower': 'Water daily, place in direct sunlight.',
  'tree': 'Water twice a week, prune annually.',
};

return (
  <div className="p-4">
    <h1 className="text-2xl font-bold mb-4">AR Plant Care Guide</h1>
    <div className="mb-4">
      <Camera ref={camera} />
    </div>
```

```
      <button
        onClick={handleCapture}
        className="px-4 py-2 bg-green-500 text-white rounded mb-4"
      >
        Identify Plant
      </button>
      {prediction && (
        <div className="mt-4">
          <h2 className="text-xl font-semibold">Identified Plant: {prediction.className}</h2>
          <p className="mt-2">Confidence: {(prediction.probability * 100).toFixed(2)}%</p>
          <p className="mt-2">Care Instructions: {plantCareInfo[prediction.className] || 'No specific care instructions available.'}</p>
        </div>
      )}
    </div>
  );
};

export default ARPlantCareGuide;
```

9. Minimalist Pomodoro Timer

A simple, clean Pomodoro timer with soothing sounds and gentle visual cues.
Benefits: Helps users focus and take breaks without being disruptive.

Source Code

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

```
const MinimalistPomodoroTimer = () => {
  const [minutes, setMinutes] = useState(25);
  const [seconds, setSeconds] = useState(0);
  const [isActive, setIsActive] = useState(false);
  const [isBreak, setIsBreak] = useState(false);

  useEffect(() => {
    let interval = null;
    if (isActive) {
      interval = setInterval(() => {
        if (seconds > 0) {
          setSeconds(seconds - 1);
        } else if (minutes > 0) {
          setMinutes(minutes - 1);
          setSeconds(59);
        } else {
          clearInterval(interval);
          setIsActive(false);
          playSound();
          if (isBreak) {
            setMinutes(25);
            setIsBreak(false);
          } else {
            setMinutes(5);
            setIsBreak(true);
          }
```

```
      }
    }, 1000);
  } else if (!isActive && seconds !== 0) {
    clearInterval(interval);
  }
  return () => clearInterval(interval);
}, [isActive, minutes, seconds, isBreak]);

const toggleTimer = () => {
  setIsActive(!isActive);
};

const resetTimer = () => {
  setIsActive(false);
  setMinutes(25);
  setSeconds(0);
  setIsBreak(false);
};

const playSound = () => {
                                        const audio = new Audio('https://assets.mixkit.co/sfx/preview/mixkit-alarm-digital-clock-beep-989.mp3');
  audio.play();
};

return (
  <div className="flex flex-col items-center justify-center min-h-screen bg-gray-100">
```

```
<div className="text-6xl font-bold mb-8">
  {String(minutes).padStart(2, '0')}:{String(seconds).padStart(2, '0')}
</div>
<div className="space-x-4">
  <button
    onClick={toggleTimer}
    className={`px-6 py-2 rounded-full text-white ${isActive ? 'bg-red-500' : 'bg-green
```

10. Mood-Based Music Recommender

An app that suggests music based on your current mood. It uses facial recognition to guess your mood.
Benefits: Helps users discover new music that matches their emotional state.

Source Code

```
import React, { useState, useRef, useEffect } from 'react';
import * as faceapi from 'face-api.js';


const MoodMusicRecommender = () => {
  const [mood, setMood] = useState(null);
  const [recommendations, setRecommendations] = useState([]);
  const videoRef = useRef(null);


  useEffect(() => {
    const loadModels = async () => {
      await faceapi.nets.tinyFaceDetector.loadFromUri('/models');
      await faceapi.nets.faceExpressionNet.loadFromUri('/models');
      startVideo();
    };
```

```
  loadModels();
}, []);

const startVideo = () => {
  navigator.mediaDevices.getUserMedia({ video: {} })
    .then((stream) => {
      videoRef.current.srcObject = stream;
    })
    .catch((err) => console.error(err));
};

const detectMood = async () => {
        const detections = await faceapi.detectSingleFace(videoRef.current, new faceapi.TinyFaceDetectorOptions())
    .withFaceExpressions();

  if (detections) {
    const mood = Object.keys(detections.expressions).reduce((a, b) =>
      detections.expressions[a] > detections.expressions[b] ? a : b
    );
    setMood(mood);
    recommendMusic(mood);
  }
};

const recommendMusic = (mood) => {
  const moodRecommendations = {
```

```
    happy: ['Upbeat pop', 'Feel-good rock', 'Cheerful electronic'],
    sad: ['Melancholic ballads', 'Soft acoustic', 'Emotional indie'],
    angry: ['Heavy metal', 'Intense rap', 'Aggressive rock'],
    neutral: ['Ambient', 'Classical', 'Soft jazz'],
    surprised: ['Energetic EDM', 'Exciting pop', 'Lively world music'],
  };
  setRecommendations(moodRecommendations[mood] || []);
};

return (
  <div className="p-4">
    <h1 className="text-2xl font-bold mb-4">Mood-Based Music Recommender</h1>
    <div className="mb-4">
      <video ref={videoRef} autoPlay muted className="w-full" />
    </div>
    <button
      onClick={detectMood}
      className="px-4 py-2 bg-blue-500 text-white rounded mb-4"
    >
      Detect Mood
    </button>
    {mood && (
      <div className="mt-4">
        <h2 className="text-xl font-semibold">Detected Mood: {mood}</h2>
        <h3 className="text-lg font-semibold mt-2">Recommended Music:</h3>
        <ul className="list-disc pl-5">
```

```
          {recommendations.map((genre, index) => (
            <li key={index}>{genre}</li>
          ))}
        </ul>
      </div>
    )}
  </div>
 );
};

export default MoodMusicRecommender;
```

11. Interactive Recipe Builder

A drag-and-drop interface for creating recipes. Users can add ingredients and steps, then see nutrition info and cooking time.
Benefits: Makes meal planning more visual and interactive, helping users create balanced meals.

Source Code

```
import React, { useState } from 'react';
import { DragDropContext, Droppable, Draggable } from 'react-beautiful-dnd';

const InteractiveRecipeBuilder = () => {
 const [ingredients, setIngredients] = useState([]);
 const [steps, setSteps] = useState([]);
 const [newIngredient, setNewIngredient] = useState('');
 const [newStep, setNewStep] = useState('');

 const onDragEnd = (result) => {
```

```
  const { destination, source, draggableId, type } = result;

  if (!destination) {
    return;
  }

  if (
    destination.droppableId === source.droppableId &&
    destination.index === source.index
  ) {
    return;
  }

  if (type === 'ingredient') {
    const newIngredients = Array.from(ingredients);
    newIngredients.splice(source.index, 1);
    newIngredients.splice(destination.index, 0, ingredients[source.index]);
    setIngredients(newIngredients);
  } else if (type === 'step') {
    const newSteps = Array.from(steps);
    newSteps.splice(source.index, 1);
    newSteps.splice(destination.index, 0, steps[source.index]);
    setSteps(newSteps);
  }
};
```

```
const addIngredient = () => {
  if (newIngredient.trim() !== '') {
    setIngredients([...ingredients, newIngredient]);
    setNewIngredient('');
  }
};

const addStep = () => {
  if (newStep.trim() !== '') {
    setSteps([...steps, newStep]);
    setNewStep('');
  }
};

return (
  <DragDropContext onDragEnd={onDragEnd}>
    <div className="p-4">
      <h1 className="text-2xl font-bold mb-4">Interactive Recipe Builder</h1>
      <div className="grid grid-cols-2 gap-4">
        <div>
          <h2 className="text-xl font-semibold mb-2">Ingredients</h2>
          <input
            type="text"
            value={newIngredient}
            onChange={(e) => setNewIngredient(e.target.value)}
            className="w-full p-2 border rounded mb-2"
```

```
          placeholder="Add ingredient"
        />
          <button onClick={addIngredient} className="px-4 py-2 bg-blue-500 text-white
rounded mb-4">
          Add Ingredient
        </button>
        <Droppable droppableId="ingredients" type="ingredient">
          {(provided) => (
                                <ul  {...provided.droppableProps}  ref={provided.innerRef}
className="list-disc pl-5">
            {ingredients.map((ingredient, index) => (
                              <Draggable  key={index}  draggableId={`ingredient-${index}`}
index={index}>
                {(provided) => (
                  <li
                    ref={provided.innerRef}
                    {...provided.draggableProps}
                    {...provided.dragHandleProps}
                    className="mb-2"
                  >
                    {ingredient}
                  </li>
                )}
              </Draggable>
            ))}
            {provided.placeholder}
          </ul>
        )}
```

```
          </Droppable>
        </div>
        <div>
          <h2 className="text-xl font-semibold mb-2">Steps</h2>
          <textarea
            value={newStep}
            onChange={(e) => setNewStep(e.target.value)}
            className="w-full p-2 border rounded mb-2"
            placeholder="Add step"
          />
          <button onClick={addStep} className="px-4 py-2 bg-blue-500 text-white rounded mb-4">
            Add Step
          </button>
          <Droppable droppableId="steps" type="step">
            {(provided) => (
              <ol {...provided.droppableProps} ref={provided.innerRef} className="list-decimal pl-5">
                {steps.map((step, index) => (
                  <Draggable key={index} draggableId={`step-${index}`} index={index}>
                    {(provided) => (
                      <li
                        ref={provided.innerRef}
                        {...provided.draggableProps}
                        {...provided.dragHandleProps}
                        className="mb-2"
                      >
```

```
                {step}
              </li>
            )}
          </Draggable>
        ))}
        {provided.placeholder}
      </ol>
    )}
  </Droppable>
 </div>
 </div>
 </div>
 </DragDropContext>
 );
};

export default InteractiveRecipeBuilder;
```

12. Mindful Social Media Feed

A social media feed that encourages positive interactions. It hides like counts and suggests breaks after long scrolling sessions.
Benefits: Promotes healthier social media habits and reduces anxiety from constant comparisons.

Source Code

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


const MindfulSocialMediaFeed = () => {
  const [posts, setPosts] = useState([]);
```

```
const [scrollTime, setScrollTime] = useState(0);
const [showBreakReminder, setShowBreakReminder] = useState(false);

useEffect(() => {
  // Simulating fetching posts from an API
  const fetchPosts = async () => {
    const response = await fetch('https://jsonplaceholder.typicode.com/posts');
    const data = await response.json();
    setPosts(data.slice(0, 10));
  };
  fetchPosts();
}, []);

useEffect(() => {
  const timer = setInterval(() => {
    setScrollTime(prevTime => prevTime + 1);
  }, 1000);

  return () => clearInterval(timer);
}, []);

useEffect(() => {
  if (scrollTime >= 300) { // 5 minutes
    setShowBreakReminder(true);
  }
}, [scrollTime]);
```

```
  const handleLike = (postId) => {
    setPosts(posts.map(post =>
      post.id === postId ? { ...post, likes: (post.likes || 0) + 1 } : post
    ));
  };

  const handleComment = (postId, comment) => {
    setPosts(posts.map(post =>
      post.id === postId ? { ...post, comments: [...(post.comments || []), comment] } : post
    ));
  };

  const handleBreakReminder = () => {
    setShowBreakReminder(false);
    setScrollTime(0);
  };

  return (
    <div className="p-4">
      <h1 className="text-2xl font-bold mb-4">Mindful Social Media Feed</h1>
      {showBreakReminder && (
        <div className="bg-yellow-100 border-l-4 border-yellow-500 text-yellow-700 p-4 mb-4" role="alert">
          <p className="font-bold">Time for a break?</p>
          <p>You've been scrolling for 5 minutes. Maybe it's time to take a short break.</p>
```

```
                <button onClick={handleBreakReminder} className="mt-2 px-4 py-2 bg-yellow-500 text-white rounded">
          I'll take a break
        </button>
      </div>
    )}
    {posts.map(post => (
      <div key={post.id} className="mb-4 p-4 border rounded">
        <h2 className="text-xl font-semibold">{post.title}</h2>
        <p className="mt-2">{post.body}</p>
        <div className="mt-4 flex items-center">
          <button onClick={() => handleLike(post.id)} className="mr-4 text-blue-500">
            Like
          </button>
          <span className="text-gray-500">Liked by many people</span>
        </div>
        <div className="mt-2">
          <input
            type="text"
            placeholder="Add a comment..."
            onKeyPress={(e) => {
              if (e.key === 'Enter') {
                handleComment(post.id, e.target.value);
                e.target.value = '';
              }
            }}
            className="w-full p-2 border rounded"
```

```
          />
        </div>
        {post.comments && (
          <div className="mt-2">
            {post.comments.map((comment, index) => (
              <p key={index} className="text-gray-700">{comment}</p>
            ))}
          </div>
        )}
      </div>
    ))}
  </div>
 );
};

export default MindfulSocialMediaFeed;
```

14. Customizable News Dashboard

A news app that lets users arrange topics and sources in a grid layout. It uses AI to suggest related stories.
Benefits: Gives users control over their news intake and helps them discover diverse viewpoints.

Source Code

```
import React, { useState, useEffect } from 'react';
import { Responsive, WidthProvider } from 'react-grid-layout';

const ResponsiveGridLayout = WidthProvider(Responsive);

const CustomizableNewsDashboard = () => {
  const [layout, setLayout] = useState([]);
  const [topics, setTopics] = useState(['Technology', 'Politics', 'Sports', 'Entertainment']);
```

```
  const [newsTiles, setNewsTiles] = useState({});
  const [suggestedStories, setSuggestedStories] = useState([]);

  useEffect(() => {
    // Initialize layout
    const initialLayout = topics.map((topic, i) => ({
      i: topic,
      x: i % 2 * 6,
      y: Math.floor(i / 2) * 4,
      w: 6,
      h: 4,
    }));
    setLayout(initialLayout);

    // Fetch news for each topic
    topics.forEach(fetchNewsForTopic);
  }, []);

  const fetchNewsForTopic = async (topic) => {
    // Simulating API call to fetch news
    const response = await fetch(`https://newsapi.org/v2/everything?q=${topic}&apiKey=YOUR_API_KEY`);
    const data = await response.json();
    setNewsTiles(prev => ({ ...prev, [topic]: data.articles.slice(0, 5) }));
  };

  const handleLayoutChange = (newLayout) => {
    setLayout(newLayout);
  };

  const addTopic = (newTopic) => {
    setTopics([...topics, newTopic]);
    setLayout([...layout, {
      i: newTopic,
      x: (layout.length % 2) * 6,
      y: Infinity,
      w: 6,
      h: 4,
    }]);
    fetchNewsForTopic(newTopic);
  };

  const removeTopic = (topic) => {
    setTopics(topics.filter(t => t !== topic));
    setLayout(layout.filter(l => l.i !== topic));
    setNewsTiles(prev => {
      const newTiles = { ...prev };
      delete newTiles[topic];
```

```
      return newTiles;
    });
  };

  const suggestRelatedStories = (topic) => {
    // Simulating AI-powered story suggestions
    const allStories = Object.values(newsTiles).flat();
    const relatedStories = allStories
                                                    .filter(story        =>
story.title.toLowerCase().includes(topic.toLowerCase()))
      .slice(0, 3);
    setSuggestedStories(relatedStories);
  };

  return (
    <div className="p-4">
           <h1  className="text-2xl  font-bold  mb-4">Customizable  News
Dashboard</h1>
      <div className="mb-4">
        <input
          type="text"
          placeholder="Add new topic"
          onKeyPress={(e) => {
            if (e.key === 'Enter') {
              addTopic(e.target.value);
              e.target.value = '';
            }
          }}
          className="p-2 border rounded mr-2"
        />
      </div>
      <ResponsiveGridLayout
        className="layout"
        layouts={{ lg: layout }}
        breakpoints={{ lg: 1200, md: 996, sm: 768, xs: 480, xxs: 0 }}
        cols={{ lg: 12, md: 10, sm: 6, xs: 4, xxs: 2 }}
        onLayoutChange={handleLayoutChange}
      >
        {topics.map(topic => (
          <div key={topic} className="bg-white p-4 rounded shadow">
            <h2 className="text-xl font-semibold mb-2">{topic}</h2>
            <button
              onClick={() => removeTopic(topic)}
              className="px-2 py-1 bg-red-500 text-white rounded mb-2"
            >
              Remove
            </button>
            <ul>
              {newsTiles[topic]?.map((article, index) => (
                <li key={index} className="mb-2">
```

```
                    <a href={article.url} target="_blank" rel="noopener
noreferrer" className="text-blue-500 hover:underline">
                      {article.title}
                    </a>
                  </li>
                ))}
              </ul>
              <button
                onClick={() => suggestRelatedStories(topic)}
                     className="px-2 py-1 bg-green-500 text-white rounded
mt-2"
              >
                Suggest Related Stories
              </button>
            </div>
          ))}
        </ResponsiveGridLayout>
        {suggestedStories.length > 0 && (
          <div className="mt-4">
            <h2 className="text-xl font-semibold mb-2">Suggested Related
Stories</h2>
            <ul>
              {suggestedStories.map((story, index) => (
                <li key={index} className="mb-2">
                      <a href={story.url} target="_blank" rel="noopener
noreferrer" className="text-blue-500 hover:underline">
                    {story.title}
                  </a>
                </li>
              ))}
            </ul>
          </div>
        )}
      </div>
    );
};

export default CustomizableNewsDashboard;
```