Below is a minimal, production-ready refactor that splits the monolith into focused files, adds an ErrorBoundary, abortable loads, and basic validation. Drop these into `src/` as shown.

📁 Proposed structure

```
src/
  components/
    ErrorBoundary.jsx
    SetupItem.jsx
  hooks/
    useUndoRedoReducer.js
  i18n/
    index.js
  mocks/
    ingredients.js
    setups.js
  KitchenDashboard.jsx
  index.jsx
  index.css
```

---

components/ErrorBoundary.jsx

```
import { Component } from 'react';

export default class ErrorBoundary extends Component {
  constructor(props){
    super(props);
    this.state = { hasError: false };
  }
  static getDerivedStateFromError(){ return { hasError: true }; }
  componentDidCatch(error, info){
    // Log to your telemetry if available
    console.error('[ErrorBoundary]', error, info);
  }
  render(){
    if (this.state.hasError) {
      return <div role="alert" className="p-4 bg-red-50 text-red-800
rounded">Something went wrong. Please refresh or try another date.</div>;
    }
    return this.props.children;
```

```
  }
}
```

---

hooks/useUndoRedoReducer.js

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

export function useUndoRedoReducer(reducer, initialState){
  const [history, setHistory] = useState({ past: [], present: initialState,
future: [] });

  const dispatch = (action) => {
    // Replace without recording history (useful after loads)
    if (action?.type === 'replace') {
      setHistory({ past: [], present: action.payload, future: [] });
      return;
    }
    const newPresent = reducer(history.present, action);
    if (Object.is(newPresent, history.present)) return;
    setHistory(prev => ({ past: [...prev.past, prev.present], present:
newPresent, future: [] }));
    if (action.type === 'reorder' && typeof action.onReorder === 'function') {
      action.onReorder(action.from, action.to);
    }
  };

  const undo = () => setHistory(prev => {
    if (!prev.past.length) return prev;
    const previous = prev.past[prev.past.length - 1];
    const newPast = prev.past.slice(0, -1);
    return { past: newPast, present: previous, future:
[prev.present, ...prev.future] };
  });

  const redo = () => setHistory(prev => {
    if (!prev.future.length) return prev;
    const next = prev.future[0];
    const newFuture = prev.future.slice(1);
    return { past: [...prev.past, prev.present], present: next, future:
newFuture };
  });

  const replace = (next) => setHistory({ past: [], present: next, future: [] });
```

```
  return [history.present, dispatch, { undo, redo, canUndo:
history.past.length>0, canRedo: history.future.length>0, replace }];
}
```

---

i18n/index.js

```
import { useCallback } from 'react';

export const i18n = {
  en: {
    save: 'Save', print: 'Print', today: 'TODAY', loading: 'Loading...',
saving: 'Saving...',
    undo: 'Undo', redo: 'Redo', quickSettings: 'Quick Settings', title: 'Main
Title',
    add: 'Add Item', clone: 'Clone Item', remove: 'Remove Item', edit: 'Edit
Item', toggleTemp: 'Click to toggle',
    lowStock: 'Low Stock', inventory: 'Inventory', resetTally: 'Reset Tally',
addNew: 'Add New',
    moveUp: 'Move Up', moveDown: 'Move Down', moveItem: 'Moved item {from} to
position {to}',
    pan: 'Pan', temp: 'Temp', food: 'Food Item', utensil: 'Utensil',
    serviceStart: 'Service Start', serviceEnd: 'Service End',
  },
  es: {
    save: 'Guardar', print: 'Imprimir', today: 'HOY', loading: 'Cargando...',
saving: 'Guardando...',
    undo: 'Deshacer', redo: 'Rehacer', quickSettings: 'Ajustes Rápidos', title:
'Título principal',
    add: 'Agregar Ítem', clone: 'Clonar Ítem', remove: 'Eliminar Ítem', edit:
'Editar Ítem', toggleTemp: 'Clic para cambiar',
    lowStock: 'Bajo Stock', inventory: 'Inventario', resetTally: 'Restablecer
Cuenta', addNew: 'Añadir Nuevo',
    moveUp: 'Mover Arriba', moveDown: 'Mover Abajo', moveItem: 'Ítem {from}
movido a posición {to}',
    pan: 'Bandeja', temp: 'Temp', food: 'Alimento', utensil: 'Utensilio',
    serviceStart: 'Inicio Servicio', serviceEnd: 'Fin Servicio',
  },
};

export const useI18n = (locale = 'en') => {
  const dict = i18n[locale] || i18n.en;
  return useCallback((key, replacements = {}) => {
    let msg = dict[key] || i18n.en[key] || key;
    for (const [k, v] of Object.entries(replacements)) msg = msg.replace(`{${k}}
```

```
`, v);
    return msg;
  }, [locale]);
};
```

---

mocks/ingredients.js

```
export const mockIngredients = [
  { id: 1, name: 'Ground Beef', quantity: 15, unit: 'lbs', category: 'meat',
minStock: 5 },
  { id: 2, name: 'Burger Buns', quantity: 24, unit: 'pcs', category: 'bread',
minStock: 10 },
  { id: 3, name: 'Lettuce', quantity: 3, unit: 'heads', category: 'produce',
minStock: 5 },
  { id: 4, name: 'Tomatoes', quantity: 8, unit: 'lbs', category: 'produce',
minStock: 3 },
  { id: 5, name: 'Pears', quantity: 12, unit: 'pcs', category: 'fruit',
minStock: 5 },
  { id: 6, name: 'Peaches', quantity: 18, unit: 'pcs', category: 'fruit',
minStock: 5 },
];
```

---

mocks/setups.js

```
export const mockSetups = [
  { date: '2025-09-28', title: 'Breakfast', wells: [
    { temp: 'Hot', pan: 'Full Shallow', food: 'Donut Burger', utensil:
'Tongs' },
    { temp: 'Cold', pan: 'Full Deep', food: 'Pears', utensil: 'Tongs' },
    { temp: 'Hot', pan: 'Narrow Metal', food: 'Scrambled Eggs', utensil:
'Spoodle' },
  ], shotgun: 'Extra fruit, condiments' },
  { date: '2025-09-29', title: 'Lunch', wells: [
    { temp: 'Cold', pan: 'Full Deep', food: 'Lettuce Wraps', utensil: 'Tongs' },
    { temp: 'Hot', pan: 'Top Shallow', food: 'Ground Beef', utensil:
'Spoodle' },
  ], shotgun: 'Salsa, Sour Cream, Jalapeños' },
  { date: '2025-09-30', title: 'Breakfast', wells: [
    { temp: 'Hot', pan: 'Full Shallow', food: 'Waffles', utensil: 'Tongs' },
    { temp: 'Hot', pan: 'Narrow Metal', food: 'Sausage Links', utensil:
'Tongs' },
```

```
    { temp: 'Cold', pan: 'Bottom Deep', food: 'Peaches', utensil: 'Scoop' },
  ], shotgun: 'Syrup, Butter, Whipped Cream' },
];

export const panTypes = ['Full Shallow', 'Full Deep', 'Narrow Metal', 'Narrow
Plastic', 'Top Shallow', 'Top Deep', 'Bottom Shallow', 'Bottom Deep'];
export const utensils = ['Spoodle', 'Tongs', 'Ladle', 'Scoop'];
```

---

components/SetupItem.jsx

```
import React, { useEffect, useRef, useState } from 'react';
import { motion } from 'framer-motion';
import { Move, Edit3, Minus, Copy, ChevronUp, ChevronDown } from 'lucide-react';
import { useI18n } from '../i18n';
import { panTypes, utensils } from '../mocks/setups';

export default React.memo(function SetupItem({
  item, index, total, onUpdate, onRemove, onClone, onDropIngredient, onReorder,
isEditing, onToggleEdit,
}){
  const t = useI18n('en');
  const [editData, setEditData] = useState(item);
  const firstFieldRef = useRef(null);

  useEffect(() => { if (isEditing) firstFieldRef.current?.focus(); },
[isEditing]);
  useEffect(() => { setEditData(item); }, [item]);

  const handleSave = () => { onUpdate(index, editData); onToggleEdit(null); };
  const handleCancel = () => { setEditData(item); onToggleEdit(null); };

  const handleDrop = (e) => {
    e.preventDefault(); e.stopPropagation();
    if (!e.dataTransfer.types?.includes('application/x-ingredient')) return;
    try { const ing = JSON.parse(e.dataTransfer.getData('application/x-
ingredient')); onDropIngredient(index, ing); } catch {}
  };

  const getPanStyle = (pan) => [
    pan.includes('Deep') && 'border-b-4 border-gray-500',
    pan.includes('Narrow') && 'scale-x-90',
  ].filter(Boolean).join(' ');

  const handleReorder = (dir) => {
```

```
    if (dir==='up' && index>0) onReorder(index, index-1);
    if (dir==='down' && index<total-1) onReorder(index, index+1);
  };

  return (
    <motion.div

className={`bg-white border-2 border-gray-200 rounded-lg p-4 min-w-[180px] max-
w-[220px] shadow-sm transition-shadow ${getPanStyle(item.pan)} ${isEditing ?
'border-blue-500' : 'hover:shadow-md'}`}
      style={{ minHeight: '200px' }}
      onDragOver={(e)=>e.preventDefault()} onDrop={handleDrop}
      role="group" aria-label={`Well ${index+1}: ${item.food}`} tabIndex={0}
      onKeyDown={(e)=>{ if((e.ctrlKey||e.metaKey)&&e.key==='ArrowUp')
{e.preventDefault();handleReorder('up');} if((e.ctrlKey||
e.metaKey)&&e.key==='ArrowDown'){e.preventDefault();handleReorder('down');} }}
    >
      <div className="flex justify-between items-start mb-2">
        <Move className="w-4 h-4 text-gray-400 cursor-grab" aria-hidden="true" /
>
        <div className="flex space-x-1">
          <button onClick={()=>handleReorder('up')} disabled={index===0}
className="p-1 text-gray-500 hover:text-green-600 disabled:opacity-30" aria-
label={t('moveUp')} title={t('moveUp')}>
            <ChevronUp className="w-3 h-3" aria-hidden="true" />
          </button>
          <button onClick={()=>handleReorder('down')}
disabled={index===total-1} className="p-1 text-gray-500 hover:text-green-600
disabled:opacity-30" aria-label={t('moveDown')} title={t('moveDown')}>
            <ChevronDown className="w-3 h-3" aria-hidden="true" />
          </button>
          <button onClick={()=>onClone(index)} className="p-1 text-gray-500
hover:text-indigo-600" aria-label={t('clone')} title={t('clone')}>
            <Copy className="w-3 h-3" aria-hidden="true" />
          </button>
          <button onClick={()=>onToggleEdit(index)}
className="p-1 text-gray-500 hover:text-blue-600" aria-label={isEditing?'Close
editor':t('edit')} title={t('edit')}>
            <Edit3 className="w-3 h-3" aria-hidden="true" />
          </button>
          <button onClick={()=>onRemove(index)} className="p-1 text-gray-500
hover:text-red-600" aria-label={t('remove')} title={t('remove')}>
            <Minus className="w-3 h-3" aria-hidden="true" />
          </button>
        </div>
      </div>

      {isEditing ? (
```

```
        <div className="space-y-3">
          <div>
            <label className="text-xs font-medium text-gray-600 block
mb-1">{t('food')}</label>
            <input type="text" ref={firstFieldRef} value={editData.food}
onChange={(e)=>setEditData({...editData, food:e.target.value})} className="w-
full text-sm border border-gray-300 rounded px-2 py-1 focus:ring-1 focus:ring-
blue-500 focus:border-transparent" />
          </div>
          <div>
            <label className="text-xs font-medium text-gray-600 block
mb-1">{t('temp')}</label>
            <select value={editData.temp}
onChange={(e)=>setEditData({...editData, temp:e.target.value})} className="w-
full text-sm border border-gray-300 rounded px-2 py-1 focus:ring-1 focus:ring-
blue-500 focus:border-transparent">
              <option value="Hot">Hot 🔥</option>
              <option value="Cold">Cold ❄️</option>
            </select>
          </div>
          <div>
            <label className="text-xs font-medium text-gray-600 block
mb-1">{t('utensil')}</label>
            <select value={editData.utensil}
onChange={(e)=>setEditData({...editData, utensil:e.target.value})} className="w-
full text-sm border border-gray-300 rounded px-2 py-1 focus:ring-1 focus:ring-
blue-500 focus:border-transparent">
              {utensils.map(u => <option key={u} value={u}>{u}</option>)}
            </select>
          </div>
          <div>
            <label className="text-xs font-medium text-gray-600 block
mb-1">{t('pan')}</label>
            <select value={editData.pan}
onChange={(e)=>setEditData({...editData, pan:e.target.value})}
className="w-full text-sm border border-gray-300 rounded px-2 py-1 focus:ring-1
focus:ring-blue-500 focus:border-transparent">
              {panTypes.map(p => <option key={p} value={p}>{p}</option>)}
            </select>
          </div>
          <div className="flex space-x-1">
            <button onClick={handleSave} className="flex-1 bg-blue-600 text-
white text-xs py-1 px-2 rounded hover:bg-blue-700">{t('save')}</button>
            <button onClick={handleCancel} className="flex-1 bg-gray-300 text-
gray-700 text-xs py-1 px-2 rounded hover:bg-gray-400">Cancel</button>
          </div>
        </div>
      ) : (
```

```
        <div className="text-center">
          <div className="mb-3">
            <div className="text-lg font-bold text-gray-800 mb-1">{item.food}</
div>
            <div className="text-sm text-gray-600">{item.utensil}</div>
          </div>
          <div className="bg-gray-100 rounded-md p-2 mb-2">
            <div className="text-xs text-gray-500 uppercase tracking-wide
mb-1">{t('pan')}</div>
            <div className="text-sm font-medium">{item.pan}</div>
          </div>
          <div className={`rounded-md p-2 ${item.temp==='Hot'?'bg-red-50':'bg-
blue-50'}`}>
            <div className={`text-xs uppercase tracking-wide mb-1 $
{item.temp==='Hot'?'text-red-600':'text-blue-600'}`}>{t('temp')}</div>
            <div className={`text-sm font-medium ${item.temp==='Hot'?'text-
red-800':'text-blue-800'}`}>{item.temp==='Hot'?'🔥 Hot':'❄️ Cold'}</div>
            <button onClick={()=>onUpdate(index, { temp:
item.temp==='Hot'?'Cold':'Hot' })} className="text-xs mt-1 text-gray-500
hover:text-gray-900 underline transition-colors" aria-label={t('toggleTemp')}
>{t('toggleTemp')}</button>
          </div>
        </div>
      )}
    </motion.div>
  );
});
```

KitchenDashboard.jsx (refactored)

```
import React, { useCallback, useEffect, useMemo, useRef, useState } from
'react';
import { motion } from 'framer-motion';
import { Calendar, ChefHat, Package, Save, Printer, Plus, Minus, RotateCcw,
PlusCircle, Loader, Filter } from 'lucide-react';

import ErrorBoundary from './components/ErrorBoundary';
import SetupItem from './components/SetupItem';
import { useUndoRedoReducer } from './hooks/useUndoRedoReducer';
import { useI18n } from './i18n';
import { mockIngredients } from './mocks/ingredients';
import { mockSetups, panTypes, utensils } from './mocks/setups';

// util IDs
```

```
const newId = () => (globalThis.crypto?.randomUUID?.() ?? `id_$
{Math.random().toString(36).slice(2)}`);

// reduced motion
const useReducedMotion = () => {
  const [prefersReduced, setPrefersReduced] = useState(false);
  useEffect(() => {
    if (!window.matchMedia) return;
    const mq = window.matchMedia('(prefers-reduced-motion: reduce)');
    setPrefersReduced(mq.matches);
    const handler = (e) => setPrefersReduced(e.matches);
    mq.addEventListener('change', handler);
    return () => mq.removeEventListener('change', handler);
  }, []);
  return prefersReduced;
};

// wells reducer
function wellsReducer(state, action){
  switch (action.type) {
    case 'add': return [...state, { id: newId(), ...action.payload }];
    case 'update': { if (action.index<0||action.index>=state.length) return
state; const next=[...state];
next[action.index]={...next[action.index], ...action.payload}; return next; }
    case 'remove': return state.filter((_,i)=>i!==action.index);
    case 'reorder': { const {from,to}=action; if(from===to||from<0||to<0||
from>=state.length||to>=state.length) return state; const n=[...state]; const
[d]=n.splice(from,1); n.splice(to,0,d); return n; }
    case 'clone': { if (action.index<0||action.index>=state.length) return
state; return [...state, { ...state[action.index], id: newId(), food: `$
{state[action.index].food} (Copy)` }]; }
    default: return state;
  }
}

// fake API
const fetchSetupByDate = (date) => new Promise((resolve)=>{
  setTimeout(()=>{
    const s = mockSetups.find(x=>x.date===date);
    resolve(s || { date, title: 'New Meal', wells: [], shotgun: 'No additional
items yet.' });
  }, 300);
});

// validation
const validateSetup = ({ title, wells }) => {
  if (!title) return 'Title is required';
  if (!Array.isArray(wells) || wells.length === 0) return 'Add at least one
```

```
well';
  for (let i=0;i<wells.length;i++){
    const w = wells[i];
    if (!w.food) return `Well ${i+1} is missing a food item`;
    if (!['Hot','Cold'].includes(w.temp)) return `Well ${i+1} has invalid
temperature`;
  }
  return null;
};

export default function KitchenDashboard(){
  const t = useI18n('en');
  const prefersReduced = useReducedMotion();
  const motionProps = prefersReduced ? { initial:false, animate:false } : {
initial:{opacity:0, y:20}, animate:{opacity:1, y:0} };

  const [selectedDate, setSelectedDate] = useState(()=> new
Date().toLocaleDateString('en-CA'));
  const [title, setTitle] = useState('New Meal');
  const [shotgun, setShotgun] = useState('No additional items yet.');
  const [isLoading, setIsLoading] = useState(false);
  const [isSaving, setIsSaving] = useState(false);
  const [searchTerm, setSearchTerm] = useState('');
  const [editingIndex, setEditingIndex] = useState(null);

  const [wells, dispatch, history] = useUndoRedoReducer(wellsReducer, []);

  // abortable loader
  useEffect(()=>{
    const ctrl = new AbortController();
    let alive = true;
    (async ()=>{
      try{
        setIsLoading(true);
        const setup = await fetchSetupByDate(selectedDate); // attach signal in
real API
        if (!alive) return;
        history.replace(setup.wells.map(w=>({ id: newId(), ...w })));
        setTitle(setup.title);
        setShotgun(setup.shotgun);
      } catch(e){
        if (e.name !== 'AbortError') alert('Failed to load setup. Please try
again.');
      } finally {
        alive && setIsLoading(false);
      }
    })();
    return ()=>{ alive=false; ctrl.abort(); };
```

```
  }, [selectedDate]);

  const saveSetup = async () => {
    const payload = { date: selectedDate, title, wells:
wells.map(({id,...r})=>r), shotgun };
    const err = validateSetup(payload);
    if (err) return alert(err);
    try{
      setIsSaving(true);
      await new Promise(r=>setTimeout(r, 800));
      alert('Setup saved successfully!');
    } catch(e){
      alert('Error saving setup. Please retry.');
    } finally { setIsSaving(false); }
  };

  const printSetup = () => {
    const wellText = wells.map((w,i)=>`Well ${i+1}: ${w.temp} | ${w.pan} | $
{w.food} | ${w.utensil}`).join('\n');
    const html = `<!doctype html><html><head><title>${title} Setup - $
{selectedDate}</title>
      <style>body{font-family:sans-serif;padding:20px}h1{border-bottom:1px solid
#ccc;padding-bottom:10px}pre{font-family:ui-
monospace,Menlo,Consolas,monospace;padding:10px;background:#f4f4f4;border:1px
solid #eee;white-space:pre-wrap}</style>
      </head><body><h1>${title} Setup - ${selectedDate}</h1><h2>Service Line
Wells</h2><pre>${wellText}</pre><h2>Shotgun Area</h2><p>${shotgun}</p>
      <script>window.onload=()=>{window.print(); setTimeout(()=>window.close(),
300)}<\/script></body></html>`;
    const win = window.open('', 'Print', 'width=800,height=900');
    if (!win) return alert('Please allow pop-ups to print.');
    win.document.write(html); win.document.close();
  };

  // Inventory state (kept local here; undo history is for wells only)
  const [ingredients, setIngredients] = useState(mockIngredients);
  const [showAddForm, setShowAddForm] = useState(false);
  const [filter, setFilter] = useState('all');
  const [showFilters, setShowFilters] = useState(false);

  const filteredIngredients = useMemo(()=>{
    return ingredients.filter(item => {
      if (filter==='low') return item.quantity < item.minStock;
      if (filter==='all') return true;
      return item.category === filter;
    });
  }, [ingredients, filter]);
```

```
  const updateQuantity = (id, change) => setIngredients(prev => prev.map(it =>
it.id===id ? {...it, quantity: Math.max(0, it.quantity + change)} : it));
  const updateMinStock = (id, v) => setIngredients(prev => prev.map(it =>
it.id===id ? {...it, minStock: Math.max(0, v)} : it));
  const resetTally = () => { if (confirm('Reset all inventory quantities to 2x
minStock?')) setIngredients(prev => prev.map(it => ({...it, quantity:
it.minStock*2}))); };
  const handleAddNewIngredient = () => {
    const name = prompt('New item name?');
    if (!name) return;
    setIngredients(prev => [...prev, { id: Date.now(), name, unit:'pcs',
category: 'other', minStock:5, quantity:0 }]);
  };

  // DnD: Inventory -> Wells
  const handleDragStartIngredient = (e, ing) => {
    e.dataTransfer.setData('application/x-ingredient', JSON.stringify(ing));
    e.dataTransfer.effectAllowed = 'copy';
  };

  const handleDropIngredient = useCallback((index, ingredient) => {
    const utensil = (ingredient.category==='meat' ||
ingredient.category==='bread') ? 'Tongs' : 'Scoop';
    dispatch({ type: 'update', index, payload: { food: ingredient.name,
utensil } });
  }, [dispatch]);

  // Reorder helper passes SR msg through action
  const onReorder = useCallback((from, to, onAnnounce) => {
    dispatch({ type:'reorder', from, to, onReorder:onAnnounce });
  }, [dispatch]);

  // UI blocks -----------------
  const today = useMemo(()=> new Date().toLocaleDateString('en-CA'), []);
  const plannedFiltered = mockSetups.filter(s =>
s.title.toLowerCase().includes(searchTerm.toLowerCase()) ||
s.date.includes(searchTerm));

  return (
    <div className="min-h-screen bg-gray-50 p-6">
      <div className="max-w-7xl mx-auto">
        <motion.div initial={{ opacity: 0, y: -20 }} animate={{ opacity: 1, y:
0 }} className="mb-8">
          <h1 className="text-3xl font-bold text-gray-900 mb-2">Kitchen
Management Dashboard</h1>
          <p className="text-gray-600">Plan your daily meal setups and manage
inventory</p>
        </motion.div>
```

```
        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">
          {/* Left Panel */}
          <div className="md:col-span-1 space-y-4">
            <motion.div initial={{ opacity: 0, x: -20 }} animate={{ opacity: 1,
x: 0 }} className="bg-white rounded-lg shadow-sm border border-gray-200 p-4">
              <label htmlFor="datePicker" className="text-sm font-medium text-
gray-700 block mb-2">Select Date</label>
              <input id="datePicker" type="date" value={selectedDate}
onChange={(e)=>setSelectedDate(e.target.value)} className="w-full border border-
gray-300 rounded-lg px-3 py-2 focus:ring-2 focus:ring-blue-500 focus:border-
transparent" />
            </motion.div>

            <motion.div initial={{ opacity: 0, x: -20 }} animate={{ opacity: 1,
x: 0 }} className="p-4">
              <input type="text" placeholder="Search setups..."
value={searchTerm} onChange={(e)=>setSearchTerm(e.target.value)} className="w-
full border border-gray-300 rounded-lg px-3 py-2 text-sm focus:ring-2
focus:ring-blue-500 focus:border-transparent" aria-label="Search setups" />
            </motion.div>

            <motion.div initial={{ opacity: 0, x: -20 }} animate={{ opacity: 1,
x: 0 }} className="bg-white rounded-lg shadow-sm border border-gray-200 p-4">
              <div className="flex items-center mb-4"><Calendar className="w-5
h-5 text-gray-600 mr-2" /><h3 className="text-lg font-semibold text-
gray-800">Planned Days</h3></div>
              <div className="space-y-2" role="list" aria-label="Planned day
setups">
                {plannedFiltered.map(setup => {
                  const isSelected = selectedDate === setup.date; const isToday
= setup.date === today;
                  return (
                    <button key={setup.date}
onClick={()=>setSelectedDate(setup.date)} className={`w-full text-left p-3
rounded-lg border transition-colors relative ${isSelected ? 'bg-blue-50 border-
blue-200 text-blue-800' : 'bg-gray-50 border-gray-200 text-gray-700 hover:bg-
gray-100'}`} role="listitem" aria-current={isSelected ? 'date' : undefined}>
                      <div className="font-medium">{setup.title}</div><div
className="text-sm opacity-75">{setup.date}</div>
                      {isToday && <span className="absolute top-2 right-2 text-
xs font-bold text-white bg-indigo-500 px-2 py-0.5 rounded-full">{t('today')}</
span>}
                    </button>
                  );
                })}
              </div>
            </motion.div>
```

```
          </div>

          {/* Main Panel */}
          <div className="md:col-span-2 lg:col-span-2">
            <ErrorBoundary>
              <motion.div {...motionProps} className="bg-white rounded-lg
shadow-sm border border-gray-200 p-6">
                {isLoading ? (
                  <div className="flex items-center justify-center p-12 text-lg
text-gray-500" role="status" aria-live="polite">
                    <Loader className="w-6 h-6 animate-spin mr-3" />
{t('loading')}
                  </div>
                ) : (
                  <>
                    <div className="flex items-center justify-between mb-6">
                      <div className="flex items-center"><ChefHat
className="w-6 h-6 text-gray-600 mr-2" /><h2 className="text-xl font-semibold
text-gray-800">{title} Setup</h2></div>
                      <div className="flex space-x-2">
                        <button onClick={saveSetup} disabled={isSaving} aria-
busy={isSaving} aria-disabled={isSaving}
className={`flex items-center px-3 py-2 text-white rounded-lg transition-colors
text-sm ${isSaving?'bg-blue-400 cursor-not-allowed':'bg-blue-600 hover:bg-
blue-700'}`}>
                          {isSaving ? <Loader className="w-4 h-4 mr-1 animate-
spin" /> : <Save className="w-4 h-4 mr-1" />}
                          {isSaving ? t('saving') : t('save')}
                        </button>
                        <button onClick={printSetup} className="flex items-
center px-3 py-2 bg-indigo-600 text-white rounded-lg hover:bg-indigo-700
transition-colors text-sm"><Printer className="w-4 h-4 mr-1" />{t('print')}</
button>
                      </div>
                    </div>

                    <div className="mb-6">
                      <label htmlFor="titleSelect" className="text-sm font-
medium text-gray-700 block mb-2">{t('title')}</label>
                      <select id="titleSelect" value={title}
onChange={(e)=>setTitle(e.target.value)} className="w-full border border-
gray-300 rounded-lg px-3 py-2 focus:ring-2 focus:ring-blue-500 focus:border-
transparent">
                        <option>Breakfast</option><option>Lunch</
option><option>Dinner</option><option>Fruit</option><option>New Meal</option>
                      </select>
                    </div>
```

```
                    {/* Visual Builder */}
                    <VisualSetupBuilder
                      wells={wells} dispatch={dispatch} history={history}
                      shotgun={shotgun} setShotgun={setShotgun}
                      onDropIngredient={handleDropIngredient}
                      editingIndex={editingIndex}
setEditingIndex={setEditingIndex}
                    />

                    {/* Quick Settings */}
                    <div className="mb-6">
                      <div className="flex items-center justify-between
mb-4"><h3 className="text-lg font-medium text-gray-800">{t('quickSettings')}</
h3></div>
                      <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                        {wells.map((w,i)=> (
                          <div key={w.id} className="bg-gray-50 rounded-lg p-3
border border-gray-200">
                            <div className="text-sm font-medium text-gray-700
mb-2">Item {i+1}: {w.food || 'Unnamed'}</div>
                            <div className="grid grid-cols-2 gap-2 text-xs text-
gray-700">
                              <div><span role="img" aria-label="utensil">🌀</
span> <strong>Utensil:</strong> {w.utensil}</div>
                              <div><span role="img" aria-label="pan">🌲</span>
<strong>Pan:</strong> {w.pan}</div>
                              <div className="col-span-2">{w.temp==='Hot' ?
(<><span role="img" aria-label="hot">🔥</span> <strong>Hot</strong></>) :
(<><span role="img" aria-label="cold">❄️</span> <strong>Cold</strong></>)}</div>
                            </div>
                          </div>
                        ))}
                      </div>
                    </div>
                  </>
                )}
              </motion.div>
            </ErrorBoundary>
          </div>

          {/* Right Panel: Inventory */}
          <div className="md:col-span-1 lg:col-span-1">
            <ErrorBoundary>
              <motion.div initial={{ opacity: 0, x: 20 }} animate={{ opacity:
1, x: 0 }} className="bg-white rounded-lg shadow-sm border border-gray-200 p-4
h-fit">
                <div className="flex items-center justify-between mb-4">
                  <div className="flex items-center"><Package
```

```
className="w-5 h-5 text-gray-600 mr-2" /><h3 className="text-lg font-semibold
text-gray-800">Inventory</h3></div>
                  <div className="flex space-x-2">
                    <button onClick={()=>setShowFilters(s=>!s)} className={`p-1
${showFilters?'text-blue-600':'text-gray-500 hover:text-blue-600'}`}
title="Filter Inventory"><Filter className="w-4 h-4" /></button>
                    <button onClick={resetTally} className="p-1 text-gray-500
hover:text-blue-600" title="Reset Tally" aria-label="Reset Tally"><RotateCcw
className="w-4 h-4" /></button>
                    <button onClick={()=>setShowAddForm(s=>!s)} className={`p-1
${showAddForm?'text-blue-600':'text-gray-500 hover:text-blue-600'}`} title="Add
New" aria-label="Add New"><PlusCircle className="w-4 h-4" /></button>
                  </div>
                </div>

                {showFilters && (
                  <div className="p-3 mb-4 border border-gray-200 rounded-lg bg-
gray-50 space-y-2">
                    <div className="flex space-x-2 text-sm">
                      {['all','low','meat','produce'].map(f => (
                        <button key={f} onClick={()=>setFilter(f)}
className={`px-3 py-1 rounded-full font-medium ${filter===f?'bg-blue-600 text-
white':'bg-white text-gray-700 hover:bg-gray-100'}`}>
                          {f==='all'?'All': f==='low'?'Low Stock' :
f[0].toUpperCase()+f.slice(1)}
                        </button>
                      ))}
                    </div>
                  </div>
                )}

                {showAddForm && (
                  <div className="p-3 mb-4 border border-blue-200 rounded-lg bg-
blue-50 space-y-2">
                    <button onClick={handleAddNewIngredient} className="w-full
bg-blue-600 text-white text-sm py-1 rounded hover:bg-blue-700">Add</button>
                  </div>
                )}

                <div className="space-y-3 max-h-[500px] overflow-y-auto">
                  {filteredIngredients.map(item => {
                    const isLow = item.quantity < item.minStock;
                    return (
                      <div key={item.id} className={`flex items-center justify-
between p-2 rounded-lg transition-all ${isLow?'bg-red-50 border border-red-300
shadow-sm':'bg-gray-50'}`} draggable
onDragStart={(e)=>handleDragStartIngredient(e, item)} title={isLow?`Low Stock:
Below ${item.minStock} ${item.unit}`:''} aria-label={`${item.name} — $
```

```
{item.quantity} ${item.unit}${isLow?', Low Stock':''}`}>
                        <div className="flex-1 min-w-0">
                          <div className="flex items-center justify-between
mb-1">
                            <span className={`font-medium text-sm truncate $
{isLow?'text-red-800':'text-gray-800'}`}>{item.name}{isLow && <span
className="ml-1">🦗</span>}</span>
                            <span className={`px-2 py-1 rounded-full text-xs
font-medium ${item.category==='meat'?'bg-red-100 text-red-800':
item.category==='produce'?'bg-green-100 text-green-800':
item.category==='fruit'?'bg-orange-100 text-orange-800':
item.category==='bread'?'bg-yellow-100 text-yellow-800':'bg-gray-100 text-
gray-800'}`}>{item.category}</span>
                          </div>
                          <div className="text-xs text-
gray-600">{item.quantity} {item.unit}</div>
                        </div>
                        <div className="flex items-center ml-3 space-x-2">
                          <input type="number" min={0} value={item.minStock}
onChange={(e)=>updateMinStock(item.id, Number(e.target.value))} className="w-10
text-xs text-center border rounded" title="Min Stock Threshold" />
                          <button onClick={()=>updateQuantity(item.id, -1)}
className="p-1 text-red-600 hover:bg-red-100 rounded disabled:opacity-50"
disabled={item.quantity===0} aria-label={`Decrease ${item.name} quantity`}>−</
button>
                          <span
className="mx-2 font-medium text-sm min-w-[30px] text-center" aria-
live="polite">{item.quantity}</span>
                          <button onClick={()=>updateQuantity(item.id, 1)}
className="p-1 text-green-600 hover:bg-green-100 rounded" aria-label={`Increase
${item.name} quantity`}>+</button>
                        </div>
                      </div>
                    );
                  })}
                </div>
              </motion.div>
            </ErrorBoundary>
          </div>
        </div>
      </div>
    </div>
  );
}

function VisualSetupBuilder({ wells, dispatch, history, shotgun, setShotgun,
onDropIngredient, editingIndex, setEditingIndex }){
  const t = useI18n('en');
```

```
  const prefersReduced = useReducedMotion();
  const motionProps = prefersReduced ? { initial:false, animate:false } : {
initial:{opacity:0, y:20}, animate:{opacity:1, y:0} };
  const dragTypeWell = 'application/x-well-index';
  const [srMsg, setSrMsg] = useState('');

  const handleDragStart = (e, index) => { e.dataTransfer.setData(dragTypeWell,
String(index)); e.dataTransfer.effectAllowed = 'move'; };
  const handleDragOver = (e) => e.preventDefault();
  const handleDrop = (e, dropIndex) => {
    e.preventDefault();
    if (!e.dataTransfer.types?.includes(dragTypeWell)) return;
    const dragIndex = parseInt(e.dataTransfer.getData(dragTypeWell), 10);
    if (Number.isNaN(dragIndex) || dragIndex === dropIndex) return;
    dispatch({ type:'reorder', from: dragIndex, to: dropIndex, onReorder:
(f,t)=>setSrMsg(t('moveItem',{from:f+1,to:t+1})) });
  };
  const updateWell = (index, payload) => dispatch({ type:'update', index,
payload });
  const removeWell = (index) => { dispatch({ type:'remove', index });
setEditingIndex(null); };
  const cloneWell = (index) => dispatch({ type:'clone', index });
  const addWell = () => dispatch({ type:'add', payload: { temp:'Hot', pan:'Full
Shallow', food:'New Item', utensil:'Spoodle' } });

  const { undo, redo, canUndo, canRedo } = history;

  return (
    <motion.div {...motionProps}
className="bg-white rounded-lg shadow-sm border border-gray-200 p-6 mb-6">
      <div className="flex items-center justify-between mb-6">
        <h3 className="text-lg font-semibold text-gray-800">Visual Service
Line</h3>
        <div className="flex space-x-2">
          <button onClick={undo} disabled={!canUndo} className="p-2 bg-gray-200
text-gray-700 rounded-lg hover:bg-gray-300 disabled:opacity-50 text-sm"
title={t('undo')} aria-label={t('undo')}>{t('undo')}</button>
          <button onClick={redo} disabled={!canRedo} className="p-2 bg-gray-200
text-gray-700 rounded-lg hover:bg-gray-300 disabled:opacity-50 text-sm"
title={t('redo')} aria-label={t('redo')}>{t('redo')}</button>
          <button onClick={addWell} className="flex items-center px-3 py-2 bg-
green-600 text-white rounded-lg hover:bg-green-700 text-sm">+ {t('add')}</
button>
        </div>
      </div>

      <div className="sr-only" aria-live="polite" aria-atomic="true">{srMsg}</
div>
```

```
      <div className="mb-6" role="list" aria-label="Service line wells">
        <div className="flex items-center mb-3">
          <div className="w-full h-0.5 bg-gray-300 relative">
            <div className="absolute left-0 -top-2 text-xs text-
gray-500">{t('serviceStart')}</div>
            <div className="absolute right-0 -top-2 text-xs text-
gray-500">{t('serviceEnd')}</div>
          </div>
        </div>
        <div className="flex gap-4 overflow-x-auto pb-4">
          {wells.map((well, index) => (
            <motion.div key={well.id} draggable
onDragStart={(e)=>handleDragStart(e,index)} onDragOver={handleDragOver}
onDrop={(e)=>handleDrop(e,index)} className="flex-shrink-0">
              <SetupItem item={well} index={index} total={wells.length}
onUpdate={updateWell} onRemove={removeWell} onClone={cloneWell}
onReorder={(f,t)=>dispatch({type:'reorder', from:f, to:t, onReorder:
(ff,tt)=>setSrMsg(t('moveItem',{from:ff+1,to:tt+1}))})}
isEditing={editingIndex===index} onToggleEdit={setEditingIndex}
onDropIngredient={onDropIngredient} />
            </motion.div>
          ))}
          {wells.length===0 && (
            <div className="flex items-center justify-center w-full h-32
border-2 border-dashed border-gray-300 rounded-lg text-gray-500">Click "Add
Item" or <span className="mx-1 font-semibold">Drag & Drop</span> an ingredient
from the right!</div>
          )}
        </div>
      </div>

      <div className="border-t border-gray-200 pt-4">
        <h4 className="font-medium text-gray-800 mb-2">Shotgun Area (Additional
Items)</h4>
        <textarea value={shotgun} onChange={(e)=>setShotgun(e.target.value)}
className="w-full border border-gray-300 rounded-lg px-3 py-2 focus:ring-2
focus:ring-blue-500 focus:border-transparent resize-none" rows={2}
placeholder="Extra fruit, condiments, backup items, etc." />
      </div>
    </motion.div>
  );
}
```

Notes

- If you use TypeScript, rename files and add types; the API stays identical.
- For popup-blocked environments, consider a `/print` route with print CSS as a progressive enhancement.
- When you wire a real API: pass `AbortController.signal` to `fetch`, map server payloads → UI shape with generated `id`s, and use `dispatch({type:'replace', payload})` or `history.replace(payload)` after load.