```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>CookBook — Step-by-step Cooking Mode</title>
  <style>
    :root{
      --bg:#f9fafc; --card:#fff; --accent:#ff7043; --muted:#6b7280; --dark:#111827;
      --radius:12px; --shadow: 0 8px 20px rgba(15,23,42,0.06);
    }
    *{box-sizing:border-box}

body{margin:0;font-family:Inter,system-ui,Arial;background:var(--bg);color:var(--dark);-webkit
-font-smoothing:antialiased}
    header{background:linear-gradient(90deg,#ff7043,#ff8a50);color:white;padding:20px
16px}
    .container{max-width:1000px;margin:18px auto;padding:0 16px}
    .top{display:flex;align-items:center;justify-content:space-between;gap:12px}
    h1{margin:0;font-size:1.4rem}
    .controls{display:flex;gap:8px;align-items:center}
    .btn{border:none;padding:9px
12px;border-radius:10px;background:var(--card);cursor:pointer;box-shadow:var(--shadow)}
    .btn.primary{background:var(--accent);color:#fff}
    .btn.danger{background:#e53e3e;color:#fff}
    .search{padding:9px;border-radius:10px;border:1px solid
rgba(0,0,0,0.06);min-width:220px}

    /* grid */

.features{display:grid;grid-template-columns:repeat(auto-fit,minmax(250px,1fr));gap:18px;ma
rgin-top:18px}

.card{background:var(--card);border-radius:var(--radius);overflow:hidden;box-shadow:var(--s
hadow);cursor:pointer;display:flex;flex-direction:column}
    .card img{width:100%;height:160px;object-fit:cover}
    .card-body{padding:12px;display:flex;flex-direction:column;gap:8px}
    .card h3{margin:0;color:var(--accent)}
    .muted{color:var(--muted)}

    /* detail */

.recipe-detail{background:var(--card);padding:18px;border-radius:var(--radius);box-shadow:v
ar(--shadow);margin-top:18px}
    .recipe-detail img{width:100%;max-height:360px;object-fit:cover;border-radius:10px}
    .meta{display:flex;gap:10px;flex-wrap:wrap;margin-top:8px}

    /* cooking mode */
```

```
    .cooking{

background:var(--card);padding:18px;border-radius:var(--radius);box-shadow:var(--shadow);
margin-top:18px;
      display:flex;flex-direction:column;gap:14px;align-items:center;
    }
    .step-card{width:100%;max-width:820px;padding:18px;border-radius:12px;border:1px
solid rgba(0,0,0,0.04);background:#fff}
    .step-header{display:flex;justify-content:space-between;align-items:center;gap:8px}
    .step-num{font-weight:700;color:var(--accent)}
    .step-text{margin-top:12px;font-size:1.05rem}

.progress-wrap{width:100%;background:#f2f4f7;border-radius:10px;height:12px;overflow:hid
den;margin-top:12px}

.progress-fill{height:100%;background:linear-gradient(90deg,var(--accent),#ff8a50);width:0%
}

.cook-controls{display:flex;gap:8px;align-items:center;justify-content:center;margin-top:12px;
flex-wrap:wrap}
    .timer{display:flex;gap:10px;align-items:center}
    .small{padding:6px 9px;font-size:0.95rem;border-radius:8px}


.all-steps{width:100%;max-width:820px;background:#fff;padding:12px;border-radius:10px;bo
rder:1px dashed rgba(0,0,0,0.04)}
    .finished{padding:18px;text-align:center}
    footer{margin:30px 0 80px;text-align:center;color:var(--muted)}
    @media(max-width:720px){
      .step-text{font-size:1rem}
    }
  </style>
</head>
<body>
  <header>
    <div class="container top">
      <div>
        <h1>CookBook — Interactive Cooking</h1>
        <div class="muted" style="font-size:0.9rem">Click a recipe, then "Start Cooking" to
follow steps one-by-one.</div>
      </div>
      <div class="controls">
        <input id="search" class="search" placeholder="Search recipes or ingredients...">
        <button id="btn-add" class="btn">Add</button>
      </div>
    </div>
  </header>
```

```
  <main class="container" id="app">
    <!-- content injected here -->
  </main>

  <footer>
    <div class="container muted">Keyboard: ← previous • → next • Space = toggle
auto-advance</div>
  </footer>

<script>
/* ---------------- Sample recipes (preloaded) ---------------- */
const sampleRecipes = [
  { id:1, title:"Classic Spaghetti Carbonara", description:"Creamy Italian pasta with pancetta
and parmesan.",
image:"https://images.unsplash.com/photo-1523983302354-2cc8cf0060d7",
ingredients:["Spaghetti","Eggs","Parmesan cheese","Pancetta","Black pepper"], steps:["Boil
pasta until al dente (8 minutes)","Cook pancetta until crisp","Whisk eggs & parmesan
together","Drain pasta and toss with pancetta, then remove from heat","Quickly mix in
egg-cheese mixture to form creamy sauce","Serve immediately with extra parmesan"] },

  { id:2, title:"Japanese Sushi Rolls", description:"Fresh homemade sushi with salmon and
avocado.", image:"https://images.unsplash.com/photo-1546069901-ba9599a7e63c",
ingredients:["Sushi rice","Nori sheets","Salmon","Avocado","Soy sauce"], steps:["Cook rice
and season with rice vinegar","Place nori on bamboo mat","Spread rice thinly on nori","Add
salmon and avocado slices","Roll tightly and slice into 6-8 pieces","Serve with soy sauce and
wasabi"] },

  { id:3, title:"Fluffy Pancakes", description:"Golden, fluffy pancakes perfect for breakfast.",
image:"https://images.unsplash.com/photo-1587735202440-51b6d1f96f2d",
ingredients:["Flour","Eggs","Milk","Sugar","Butter","Maple syrup"], steps:["Mix flour, sugar,
baking powder","Whisk eggs and milk together","Combine wet & dry ingredients to make
batter","Heat pan and add butter","Pour batter to make pancakes, flip when bubbles form
(2-3 min)","Serve stacked with butter and maple syrup"] },

  { id:4, title:"Indian Chicken Curry", description:"Spicy chicken curry with rich tomato gravy.",
image:"https://images.unsplash.com/photo-1604908177729-9aa01a0ebd4c",
ingredients:["Chicken","Onions","Tomatoes","Curry spices","Garlic","Ginger"], steps:["Heat oil
and sauté onions until golden","Add ginger-garlic paste and spices","Add chicken and brown
on all sides","Pour in tomatoes and cook until soft","Simmer for 25-30 minutes until chicken
is tender","Garnish with coriander and serve with rice"] },

  { id:5, title:"Greek Salad", description:"Refreshing salad with feta, cucumber, and olives.",
image:"https://images.unsplash.com/photo-1568605114967-8130f3a36994",
ingredients:["Cucumber","Tomatoes","Olives","Feta cheese","Olive oil","Oregano"],
steps:["Chop cucumber and tomatoes","Mix in bowl with olives and crumbled feta","Drizzle
olive oil and sprinkle oregano","Toss gently and serve chilled"] },
```

```
  { id:6, title:"Mexican Tacos", description:"Crispy tacos with beef, cheese, and salsa.", image:"https://images.unsplash.com/photo-1601924582975-7aa7c5d2abf9", ingredients:["Tortillas","Ground beef","Cheese","Lettuce","Salsa"], steps:["Cook beef with taco seasoning","Warm tortillas on pan","Assemble with beef, cheese, lettuce and salsa","Serve with lime wedges"] },

  { id:7, title:"Thai Green Curry", description:"Aromatic curry with coconut milk and vegetables.", image:"https://images.unsplash.com/photo-1604908177039-1a9c6b6b4e93", ingredients:["Green curry paste","Coconut milk","Chicken","Vegetables","Basil"], steps:["Fry curry paste in oil briefly","Add coconut milk and bring to simmer","Add chicken and vegetables and cook through","Stir in basil leaves and serve with rice"] },

  { id:8, title:"French Croissants", description:"Buttery, flaky croissants baked to perfection.", image:"https://images.unsplash.com/photo-1523986371872-9d3ba2e2f642", ingredients:["Flour","Butter","Yeast","Milk","Sugar"], steps:["Prepare dough and chill","Layer butter and fold dough (lamination)","Roll & shape croissants","Proof for 2 hours","Bake at 200°C for 12-15 minutes until golden"] },

  { id:9, title:"Middle Eastern Hummus", description:"Creamy chickpea dip with olive oil and tahini.", image:"https://images.unsplash.com/photo-1600891964599-f61ba0e24092", ingredients:["Chickpeas","Tahini","Olive oil","Garlic","Lemon juice"], steps:["Blend chickpeas until smooth","Add tahini, lemon juice and garlic","Adjust seasoning and texture","Serve drizzled with olive oil and paprika"] },

  { id:10, title:"Chinese Fried Rice", description:"Savory stir-fried rice with egg and vegetables.", image:"https://images.unsplash.com/photo-1617196038435-3bdf3c4d0574", ingredients:["Rice","Eggs","Carrots","Peas","Soy sauce","Spring onions"], steps:["Use day-old rice for best texture","Scramble eggs and set aside","Stir-fry vegetables","Add rice and soy sauce, toss well","Add eggs back and garnish with spring onions"] },

  // Indian unique additions
  { id:11, title:"Paneer Tikka Masala", description:"Grilled paneer cubes in creamy spiced tomato gravy.", image:"https://images.unsplash.com/photo-1628294896516-6e384c2f2c53", ingredients:["Paneer","Yogurt","Tomatoes","Onions","Spices","Cream"], steps:["Marinate paneer in spiced yogurt for 30 minutes","Grill or sear paneer until charred edges","Sauté onions then add tomatoes to make gravy","Add spices and cream to adjust consistency","Toss paneer into gravy and simmer 5 minutes","Garnish with kasuri methi and serve"] },

  { id:12, title:"Masala Dosa", description:"Crispy South Indian crepe with spiced potato filling.", image:"https://images.unsplash.com/photo-1628595351029-8271d7a12ab1", ingredients:["Dosa batter","Potatoes","Onions","Green chilies","Mustard seeds","Curry leaves"], steps:["Prepare potato masala with onions and spices","Heat tava and pour dosa batter thinly","Spread and cook until crisp","Place potato masala and fold dosa","Serve with coconut chutney and sambar"] },

  { id:13, title:"Rajma Chawal", description:"Comforting kidney bean curry with rice.", image:"https://images.unsplash.com/photo-1638194597495-95e399a097e2",
```

```
ingredients:["Kidney beans","Onions","Tomatoes","Garlic","Spices","Rice"], steps:["Soak and pressure-cook kidney beans until soft","Fry onions and tomatoes to make masala","Add spices and cooked beans, simmer 15-20 min","Cook rice separately","Serve hot with a drizzle of ghee"] },

  { id:14, title:"Mango Shrikhand", description:"Sweet mango yogurt dessert with saffron.", image:"https://images.unsplash.com/photo-1626083044797-0a2b0f9a0e75", ingredients:["Hung curd","Mango pulp","Sugar","Saffron","Cardamom"], steps:["Whisk hung curd until smooth","Fold in mango pulp and sugar","Add saffron infused in warm milk and cardamom","Chill for 2 hours and serve"] },

  { id:15, title:"Baingan Bharta", description:"Smoky mashed eggplant curry with onions and spices.", image:"https://images.unsplash.com/photo-1638195099651-2b0ef9f43a2e", ingredients:["Eggplant","Onions","Tomatoes","Garlic","Ginger","Spices"], steps:["Roast eggplant over flame until skin chars","Peel and mash the pulp","Sauté onions, add tomatoes and spices","Add mashed eggplant and cook 8-10 minutes","Garnish with coriander and serve with roti"] },

  // My own original Indo-Mexican recipe (added by assistant)
  { id:16, title:"Masala Paneer Tacos (My Recipe)", description:"Soft tortillas filled with spiced paneer, crunchy pickled onions and mint-coriander chutney — Indian street food meets tacos.", image:"https://images.unsplash.com/photo-1601050690597-df7bfc7b6b12", ingredients:["Paneer (200g)","Tortillas (4)","Onion (1, thinly sliced)","Lemon","Mint-coriander chutney","Chaat masala","Turmeric, chili, garam masala","Oil"], steps:["Cube paneer and marinate with turmeric, chili powder and salt for 10 minutes","Quickly pan-fry paneer until golden (3-4 min)","Pickle onions with a squeeze of lemon and pinch of salt","Warm tortillas on a skillet","Assemble: spread chutney, place paneer, top with pickled onions and sprinkle chaat masala","Serve immediately while warm"] }
];

/* ---------- LocalStorage "DB" ---------- */
function load() {
  let saved = JSON.parse(localStorage.getItem("cookbook_recipes") || "[]");
  if (!saved.length) {
    saved = sampleRecipes;
    localStorage.setItem("cookbook_recipes", JSON.stringify(saved));
  }
  return saved;
}
function save(list) { localStorage.setItem("cookbook_recipes", JSON.stringify(list)); }

/* ---------- App State ---------- */
let recipes = load();
let currentView = "list"; // list | detail | form | cooking
let activeId = null;

/* cooking state */
let cookingIndex = 0;
```

```
let cookingTimer = null; // interval id
let cookingTimerRemaining = 0;
let autoAdvanceInterval = null;
let autoAdvanceOn = false;
let autoAdvanceDelay = 8; // seconds

/* ---------- DOM ---------- */
const app = document.getElementById("app");
const searchInput = document.getElementById("search");
const btnAdd = document.getElementById("btn-add");
btnAdd.addEventListener("click", ()=>openForm());

searchInput.addEventListener("input", ()=>renderList());

/* ---------- Router & Render ---------- */
function router() {
  if (currentView === "list") renderList();
  else if (currentView === "detail") renderDetail(activeId);
  else if (currentView === "form") renderForm(activeId);
  else if (currentView === "cooking") renderCooking(activeId, cookingIndex);
}

/* ---------- Helpers ---------- */
function findRecipe(id){ return recipes.find(r=>r.id===id); }
function newId(){ return Date.now(); }
function detectDuration(text){
  // find hours/minutes/seconds, return seconds (small heuristic)
  const hour = text.match(/(\d+)\s*(h|hr|hrs|hour|hours)/i);
  if(hour) return parseInt(hour[1],10)*3600;
  const min = text.match(/(\d+)\s*(m|min|mins|minutes)/i);
  if(min) return parseInt(min[1],10)*60;
  const sec = text.match(/(\d+)\s*(s|sec|secs|seconds)/i);
  if(sec) return parseInt(sec[1],10);
  return null;
}
function formatTime(s){ if(s<0) s=0; const m=Math.floor(s/60); const sec=s%60; return
`${m}:${sec.toString().padStart(2,"0")}`; }
function clearCookingIntervals(){
  if(cookingTimer){ clearInterval(cookingTimer); cookingTimer=null;
cookingTimerRemaining=0; }
  if(autoAdvanceInterval){ clearInterval(autoAdvanceInterval); autoAdvanceInterval=null;
autoAdvanceOn=false; }
}

/* ---------- List View ---------- */
function renderList(){
  currentView = "list";
  clearCookingIntervals();
```

```
  app.innerHTML = "";
  const q = (searchInput.value || "").trim().toLowerCase();

  const header = document.createElement("div");
  header.className = "top";
  app.appendChild(header);

  const grid = document.createElement("div");
  grid.className = "features";

  // filter
  const filtered = recipes.filter(r => {
    if(!q) return true;
    return r.title.toLowerCase().includes(q) ||
          r.description?.toLowerCase().includes(q) ||
          (r.ingredients || []).join(" ").toLowerCase().includes(q);
  });

  if(filtered.length === 0){
    const no = document.createElement("div");
    no.className = "muted";
    no.textContent = "No recipes found. Click Add to create one.";
    app.appendChild(no);
    return;
  }

  filtered.forEach(r => {
    const card = document.createElement("article");
    card.className = "card";
    card.innerHTML = `<img src="${r.image || 'https://via.placeholder.com/400x240'}"
alt="${r.title}">
                  <div class="card-body">
                    <h3>${r.title}</h3>
                    <div class="muted">${r.description || ''}</div>
                  </div>`;
    card.addEventListener("click", ()=> { activeId = r.id; currentView = "detail"; router(); });
    grid.appendChild(card);
  });

  app.appendChild(grid);
}

/* ---------- Detail View ---------- */
function renderDetail(id){
  currentView = "detail";
  clearCookingIntervals();
  app.innerHTML = "";
  const r = findRecipe(id);
```

```
  if(!r){ app.innerHTML = "<p>Recipe not found</p>"; return; }

  const detail = document.createElement("div");
  detail.className = "recipe-detail";
  detail.innerHTML = `
    <h2>${r.title}</h2>
    <img src="${r.image || 'https://via.placeholder.com/800x400'}" alt="${r.title}">
    <p class="muted">${r.description || ''}</p>
    <div style="display:flex;gap:8px;flex-wrap:wrap;margin-top:10px">
      <button class="btn primary" id="start-cook">Start Cooking</button>
      <button class="btn" id="edit-btn">Edit</button>
      <button class="btn danger" id="delete-btn">Delete</button>
      <button class="btn" id="back-btn">Back to list</button>
    </div>
    <hr style="margin:12px 0">
    <h4>Ingredients</h4>
    <ul>${(r.ingredients || []).map(i=>`<li>${i}</li>`).join('')}</ul>
    <h4>Steps</h4>
    <ol>${(r.steps || []).map(s=>`<li>${s}</li>`).join('')}</ol>
  `;
  app.appendChild(detail);

  document.getElementById("start-cook").addEventListener("click", ()=>{
    cookingIndex = 0; activeId = id; currentView = "cooking"; router();
  });
  document.getElementById("edit-btn").addEventListener("click", ()=>openForm(id));
  document.getElementById("delete-btn").addEventListener("click", ()=> {
    if(confirm("Delete this recipe?")) {
      recipes = recipes.filter(x=>x.id!==id);
      save(recipes);
      currentView = "list";
      activeId = null;
      router();
    }
  });
  document.getElementById("back-btn").addEventListener("click", ()=> { currentView = "list";
activeId = null; router(); });
}

/* ---------- Form (Add/Edit) ---------- */
function openForm(id=null){
  currentView = "form"; activeId = id; clearCookingIntervals();
  const r = id ? findRecipe(id) : {title:'',description:'',image:'',ingredients:[],steps:[]};
  app.innerHTML = "";
  const form = document.createElement("div");
  form.className = "recipe-form";
  form.innerHTML = `
    <h2>${id ? 'Edit' : 'Add'} Recipe</h2>
```

```
    <label>Title</label><input id="f-title" value="${r.title || ''}">
    <label>Description</label><textarea id="f-desc">${r.description || ''}</textarea>
    <label>Image URL</label><input id="f-img" value="${r.image || ''}">
    <label>Ingredients (comma separated)</label><input id="f-ing"
value="${(r.ingredients||[]).join(', ')}">
    <label>Steps (each step separated by '||' to allow commas)</label><input id="f-steps"
value="${(r.steps||[]).join(' || ')}">
    <div style="display:flex;gap:8px;margin-top:12px">
      <button class="btn primary" id="save-btn">Save</button>
      <button class="btn" id="cancel-btn">Cancel</button>
    </div>
    <div style="margin-top:8px" class="muted">Tip: Use "||" between steps when a step
contains commas.</div>
  `;
  app.appendChild(form);

  document.getElementById("cancel-btn").addEventListener("click", ()=> { currentView='list';
router(); });
  document.getElementById("save-btn").addEventListener("click", ()=>{
    const newR = {
      id: id || newId(),
      title: document.getElementById("f-title").value.trim(),
      description: document.getElementById("f-desc").value.trim(),
      image: document.getElementById("f-img").value.trim(),
      ingredients:
document.getElementById("f-ing").value.split(",").map(x=>x.trim()).filter(Boolean),
      steps:
document.getElementById("f-steps").value.split("||").map(x=>x.trim()).filter(Boolean)
    };
    if(!newR.title){ alert("Title required"); return; }
    if(id){
      recipes = recipes.map(x=>x.id===id ? newR : x);
    } else {
      recipes.push(newR);
    }
    save(recipes);
    activeId = newR.id;
    currentView = "detail";
    router();
  });
}

/* ---------- Cooking Mode ---------- */
function renderCooking(recipeId, stepIndex=0){
  currentView = "cooking";
  clearCookingIntervals();
  const r = findRecipe(recipeId);
  if(!r){ app.innerHTML = "<p>Recipe not found</p>"; return; }
```

```
  // clamp
  cookingIndex = Math.max(0, Math.min(stepIndex, (r.steps || []).length - 1));
  app.innerHTML = "";

  const wrap = document.createElement("div");
  wrap.className = "cooking";

  const header = document.createElement("div");
  header.className = "step-card";
  const total = (r.steps || []).length || 1;
  const percent = Math.round(((cookingIndex) / total) * 100);

  header.innerHTML = `
    <div class="step-header">
      <div>
        <div class="step-num">Step ${cookingIndex+1} of ${total}</div>
        <div style="font-weight:700;font-size:1.15rem;margin-top:6px">${r.title}</div>
        <div class="muted" style="font-size:0.9rem;margin-top:6px">${r.description || ''}</div>
      </div>
      <div style="text-align:right">
        <div style="font-size:0.9rem;color:var(--muted)">Progress</div>
        <div
style="font-weight:700;color:var(--accent);font-size:1.05rem">${(Math.round(((cookingIndex+
1)/total)*100))}%</div>
      </div>
    </div>

    <div class="progress-wrap" aria-hidden="true">
      <div class="progress-fill"
style="width:${Math.round(((cookingIndex+1)/total)*100)}%"></div>
    </div>

    <div class="step-text">${(r.steps && r.steps[cookingIndex]) ? r.steps[cookingIndex] : "No
step text available."}</div>
  `;

  wrap.appendChild(header);

  // timer detection from current step
  const stepText = (r.steps && r.steps[cookingIndex]) ? r.steps[cookingIndex] : "";
  const detectedSeconds = detectDuration(stepText);

  // controls area
  const controls = document.createElement("div");
  controls.className = "cook-controls";

  const prevBtn = document.createElement("button");
  prevBtn.className = "btn small";
```

```
  prevBtn.textContent = "◂ Previous";
  prevBtn.disabled = cookingIndex === 0;
  prevBtn.addEventListener("click", ()=> {
    cookingIndex = Math.max(0, cookingIndex - 1);
    renderCooking(recipeId, cookingIndex);
  });

  const nextBtn = document.createElement("button");
  nextBtn.className = "btn small";
  nextBtn.textContent = "Next ▸";
  nextBtn.disabled = cookingIndex >= total - 1;
  nextBtn.addEventListener("click", ()=> {
    cookingIndex = Math.min(total - 1, cookingIndex + 1);
    if(cookingIndex >= total) finishCooking();
    else renderCooking(recipeId, cookingIndex);
  });

  const startTimerBtn = document.createElement("button");
  startTimerBtn.className = "btn small";
  startTimerBtn.textContent = detectedSeconds ? `Start Timer
(${formatTime(detectedSeconds)})` : "Set Timer";
  startTimerBtn.addEventListener("click", async ()=>{
    // If detected, use detectedSeconds; otherwise prompt user for minutes/seconds
    let seconds = detectedSeconds;
    if(!seconds){
      const answer = prompt("Set timer length in seconds (e.g. 90) or minutes with m (e.g.
2m):", "60");
      if(!answer) return;
      const mMatch = String(answer).trim().match(/^(\d+)\s*m(in)?$/i);
      if(mMatch) seconds = parseInt(mMatch[1],10)*60;
      else seconds = parseInt(answer,10);
      if(isNaN(seconds) || seconds <= 0){ alert("Invalid number"); return; }
    }
    startTimer(seconds, () => {
      // auto-advance when timer ends
      alert("Timer finished for step " + (cookingIndex+1));
      if(cookingIndex < total - 1) {
        cookingIndex++;
        renderCooking(recipeId, cookingIndex);
      } else {
        finishCooking();
      }
    });
    renderCooking(recipeId, cookingIndex); // re-render to show timer
  });

  const autoBtn = document.createElement("button");
  autoBtn.className = "btn small";
```

```
autoBtn.textContent = autoAdvanceOn ? `Auto: On (${autoAdvanceDelay}s)` : `Auto: Off`;
autoBtn.addEventListener("click", ()=> {
  autoAdvanceOn = !autoAdvanceOn;
  if(autoAdvanceOn){
    const d = prompt("Auto-advance delay in seconds:", String(autoAdvanceDelay));
    if(d !== null){
      const n = parseInt(d,10);
      if(!isNaN(n) && n>0) autoAdvanceDelay = n;
    }
    autoBtn.textContent = `Auto: On (${autoAdvanceDelay}s)`;
    autoAdvanceInterval = setInterval(()=> {
      if(cookingIndex < total - 1){
        cookingIndex++;
        renderCooking(recipeId, cookingIndex);
      } else {
        clearInterval(autoAdvanceInterval);
        autoAdvanceInterval = null;
        autoAdvanceOn = false;
        renderCooking(recipeId, cookingIndex);
      }
    }, autoAdvanceDelay * 1000);
  } else {
    if(autoAdvanceInterval){ clearInterval(autoAdvanceInterval); autoAdvanceInterval=null; }
    autoBtn.textContent = `Auto: Off`;
  }
});

const exitBtn = document.createElement("button");
exitBtn.className = "btn";
exitBtn.textContent = "Exit Cooking";
exitBtn.addEventListener("click", ()=> {
  clearCookingIntervals();
  // return to detail view of this recipe
  currentView = "detail";
  router();
});

controls.appendChild(prevBtn);
controls.appendChild(nextBtn);
controls.appendChild(startTimerBtn);
controls.appendChild(autoBtn);
controls.appendChild(exitBtn);

wrap.appendChild(controls);

// timer display area
const timerWrap = document.createElement("div");
timerWrap.className = "timer";
```

```
  timerWrap.style.justifyContent = "center";
  if(cookingTimer){
    const txt = document.createElement("div");
    txt.textContent = "Timer: " + formatTime(cookingTimerRemaining);
    timerWrap.appendChild(txt);

    const cancelBtn = document.createElement("button");
    cancelBtn.className = "btn small";
    cancelBtn.textContent = "Cancel Timer";
    cancelBtn.addEventListener("click", ()=> {
      clearCookingIntervals();
      renderCooking(recipeId, cookingIndex);
    });
    timerWrap.appendChild(cancelBtn);
  } else {
    // if detectedSeconds present, show helpful hint
    if(detectedSeconds){
      const hint = document.createElement("div");
      hint.className = "muted";
      hint.textContent = `Tip: this step mentions a time (${formatTime(detectedSeconds)}).
Click "Start Timer" to run countdown and auto-advance when it ends.`;
      hint.style.maxWidth = "720px";
      wrap.appendChild(hint);
    } else {
      const hint = document.createElement("div");
      hint.className = "muted";
      hint.textContent = `Optional: set a timer for this step (useful for baking / simmering).`;
      wrap.appendChild(hint);
    }
  }

  // show all steps collapsed under
  const all = document.createElement("div");
  all.className = "all-steps";
  all.innerHTML = `<strong>All Steps</strong>
    <ol style="margin:8px 0">${(r.steps||[]).map((s,i)=>`<li style="${i===cookingIndex ?
'font-weight:700;color:var(--accent)' : ''}">${s}</li>`).join('')}</ol>`;
  wrap.appendChild(all);

  app.appendChild(wrap);

  // keyboard navigation
  function onKey(e){
    if(currentView !== "cooking") return;
    if(e.key === "ArrowLeft") { if(cookingIndex>0){ cookingIndex--;
renderCooking(recipeId,cookingIndex);} }
    if(e.key === "ArrowRight") { if(cookingIndex < total - 1){ cookingIndex++;
renderCooking(recipeId,cookingIndex);} }
```

```
    if(e.code === "Space") { e.preventDefault(); // toggle auto
      autoBtn.click();
    }
  }
  window.removeEventListener("keydown", window._cookKeyHandler);
  window._cookKeyHandler = onKey;
  window.addEventListener("keydown", onKey);
}

/* ---------- Timer logic ---------- */
function startTimer(seconds, onFinish){
  // clear existing timers
  if(cookingTimer) clearInterval(cookingTimer);
  cookingTimerRemaining = seconds;
  cookingTimer = setInterval(()=> {
    cookingTimerRemaining--;
    if(cookingTimerRemaining <= 0){
      clearInterval(cookingTimer);
      cookingTimer = null;
      cookingTimerRemaining = 0;
      // callback
      if(typeof onFinish === "function") onFinish();
    }
    // update UI: re-render current cooking view to show remaining time
    // find active recipe id
    if(currentView === "cooking" && activeId) renderCooking(activeId, cookingIndex);
  }, 1000);
}

/* ---------- Finish ---------- */
function finishCooking(){
  clearCookingIntervals();
  app.innerHTML = "";
  const div = document.createElement("div");
  div.className = "finished";
  div.innerHTML = `<h2>🎉 Finished!</h2><p class="muted">You've completed all
steps.</p>
    <div style="display:flex;gap:8px;justify-content:center;margin-top:12px">
      <button class="btn primary" id="view-recipe">View Recipe</button>
      <button class="btn" id="back-list">Back to list</button>
    </div>`;
  app.appendChild(div);
  document.getElementById("view-recipe").addEventListener("click", ()=> { currentView =
"detail"; router(); });
  document.getElementById("back-list").addEventListener("click", ()=> { currentView = "list";
router(); });
}
```

```
/* ---------- Init ---------- */
function init(){
  renderList();
}
init();

/* ---------- Expose for clicks (Edit/Add etc.) ---------- */
function findIdFromTitle(title){ const r = recipes.find(x=>x.title===title); return r? r.id : null; }
function openFormId(id){ openForm(id); }
function editRecipe(id){ openForm(id); }
function deleteRecipe(id){ if(confirm("Delete this recipe?")){ recipes =
recipes.filter(x=>x.id!==id); save(recipes); currentView='list'; router(); } }

window.openForm = openForm; // for potential debugging
window.renderList = renderList; // debug

/* ---------- Optional: quick method to add example recipe (not exposed) ---------- */
</script>
</body>
</html>
```